ScenarioBench: Trace-Grounded Compliance Evaluation for Text-to-SQL and RAG

1st Zahra Atf
Faculty of Business and Information Technology
Ontario Tech University
Oshawa, Canada
ORCID: 0000-0003-0642-4341

2nd Peter R. Lewis
Faculty of Business and Information Technology
Ontario Tech University
Oshawa, Canada
ORCID: 0000-0003-4271-8611

Abstract—ScenarioBench is a policy-grounded, trace-aware benchmark for evaluating Text-to-SQL and retrieval-augmented generation in compliance settings. Each YAML scenario ships with a *no-peek gold-standard* package—expected decision, minimal witness trace, governing clause set, and canonical SQL—enabling end-to-end scoring of both *what* a system decides and *why* it decides so. Systems must justify outputs with clause IDs retrieved from the same policy canon, making explanations falsifiable and audit-ready. The evaluator reports decision accuracy, trace quality (completeness/correctness/order), retrieval effectiveness, SQL correctness via result-set equivalence, policy coverage, latency, and an explanation-hallucination rate. A normalized Scenario Difficulty Index (SDI) and its budgeted variant (SDI-R) aggregate results while pricing retrieval difficulty and time. Unlike Spider and BIRD, which assess Text-to-SQL without clause-level provenance, and unlike KILT/RAG-style setups that lack a compliance-oriented trace protocol, ScenarioBench ties decisions to clause-ID evidence under a strict grounding and no-peek discipline. On a seed, synthetic suite (N=16), decision metrics saturate (accuracy and macro-F1 = 1.000), while a single budgeted reflection step closes trace gaps (trace-completeness and policy-coverage $0.541! \rightarrow !1.000$) with zero hallucination at approximately +1 ms latency, indicating that marginal gains come from justification quality under explicit time budgets.

Index Terms—Text-to-SQL, RAG, compliance, trace grounding, hallucination.

I. INTRODUCTION

Evaluations of Text-to-SQL and retrieval-augmented generation (RAG) rarely enforce *trace-grounded* decisions suitable for regulated workflows, where reviewers must verify which clauses justify a decision and whether those clauses were actually retrieved by the system being scored. Text-to-SQL benchmarks such as *Spider* emphasize query correctness and cross-domain generalization but do not require clause-level provenance or pricing of retrieval difficulty in end-to-end scoring [1]. In parallel, RAG work has improved effectiveness and discussed hallucinations, yet provenance remains under-specified and unsupported attributions persist [2], [3], [5]. Production stacks often mix lexical and vector search (e.g., FAISS) [4], but lack a policy-grounded protocol that unifies Text-to-SQL accuracy, retrieval/trace quality, and compliance auditing under a single evaluator. Concrete compliance

cases make the requirements explicit. Under Canada's Anti-Spam Legislation (CASL), commercial email must include an unsubscribe mechanism, clear sender identification, and an appropriate consent status, with exceptions for transactional messages. In such settings, audits demand SQL correctness judged by *result-set equivalence on clause IDs*—not surface-form identity—so that decisions are falsifiable and contestable, and any cited evidence can be checked against the same canon the system queried. Recent work underscores that trust in AI is tightly coupled with explainability and the calibrated handling of uncertainty, particularly in regulated decision contexts [6]. Human-centric perspectives emphasize explanations that surface reasons, limits, and confidence in ways stakeholders can act upon [7]. Complementarily, rule-governed schemes for reason-giving offer a principled way to structure uncertainty disclosures and reduce unsupported attributions [8]. Building on these insights, *ScenarioBench* operationalizes trace-grounded evaluation—requiring systems to justify outcomes with clause-level evidence—so that explainability contributes to verifiable trust under bounded resources and domain constraints [9]. *ScenarioBench* addresses this gap with three contributions. First, a **grounding invariant** requires systems to justify outputs using clause IDs they themselves retrieved from the policy canon, making explanations falsifiable and audit-ready. Second, a **dual materialization** of a jurisdiction-agnostic canon—Prolog facts for deterministic rule execution and a SQL `Policy_DB` for retrieval and NLQ-to-SQL—preserves referential integrity across views. Third, a **difficulty- and latency-aware aggregation** combines core signals—decision accuracy/macro-F1, trace completeness/correctness/order, SQL result-set equivalence, retrieval effectiveness, and an *unsupported-attribution rate*—into a normalized Scenario Difficulty Index (SDI) and a budgeted variant (SDI-R) that price retrieval difficulty and time. We therefore ask, in a form we can evaluate end-to-end without external peeking: does clause grounding improve decision reliability over ungrounded baselines; how do retrieval choices (BM25 and hybrid) shift justification quality as measured by trace completeness/correctness and unsupported attribution;

and, under a per-scenario wall-clock budget B_{time}, can short, constrained reflection deliver consistent gains in explanation quality without access to the *gold-standard* package?

II. TASK & GOLD-STANDARD PROTOCOL

Given a natural-language *scenario* with policy-relevant *context* and candidate *content*, the system must output

(i) a discrete *decision* (e.g., `approve`/`revise`/`reject`/`escalate`) and

(ii) an *execution trace* that *explicitly grounds* the decision in retrieved policy clauses. Traces are ordered lists of (`clause_id`, `role`) with `role` $\in$ {`applies`, `exception`, `precedence`} and brief rationales. All identifiers in the trace must be drawn from the system's own retrieval (lexical/vector/hybrid); citing knowledge outside the canon is disallowed.

For each YAML scenario, a *gold-standard* package enables trace-grounded, end-to-end evaluation: a gold decision and a minimal sufficient witness trace, a governing clause set (necessary/sufficient identifiers, closed over exceptions/precedence), and a canonical SQL query over `Policy_DB`. SQL correctness is judged by *result-set equivalence* on `clause_ids` rather than string identity. Optionally, a gold top-k ranking (`qrels`) provides stable targets for Recall@k, MRR, and nDCG without exposing answers at inference. Gold labels are derived deterministically from the synthetic canon via a reference rule engine with stable tie-breaking. The gold-standard package is evaluator-only (*no-peek*).

Scenarios compile to Prolog facts for deterministic rule execution and to a relational `Policy_DB` with BM25 and vector indexes. A one-to-one mapping between `clause_id`, the corresponding Prolog predicate, and `Policy_DB.clauses.id` preserves referential integrity; a validator enforces YAML well-formedness, ID consistency, and trace–schema constraints. Two invariants govern scoring: grounding (all predicted trace IDs $\subseteq$ retrieved@ k) and no-peek (systems cannot access gold decisions/traces/rankings at inference).

The evaluator reports decision accuracy and macro-F1; trace completeness, correctness, and order (Kendall-τ against the gold witness); retrieval effectiveness (Recall@k/MRR/nDCG when `qrels` exist); NLQ-to-SQL accuracy by result-set equivalence; policy coverage; latency; and an explanation–hallucination rate computed against the gold closure. A budgeted reflective loop may adapt retrieval, NLQ, and rule parameters without gold access, subject to a per-scenario wall-clock budget B_{time} and early stopping on minimal improvement; all changes are logged for audit. Default retrieval settings (e.g., $k=10$ and $(w_{\text{vec}}, w_{\text{bm25}})=(0.6, 0.4)$) appear in Table II.

In practice, the protocol is a constrained, auditable pipeline: synchronized SQL/Prolog views feed retrieval and NLQ-to-SQL, which in turn drive a rule engine that must return a trace grounded strictly in the system's own top-k; standard logs (JSONL/CSV) capture configuration, evidence, and timing. If enabled, a short, budgeted reflection adjusts only local knobs
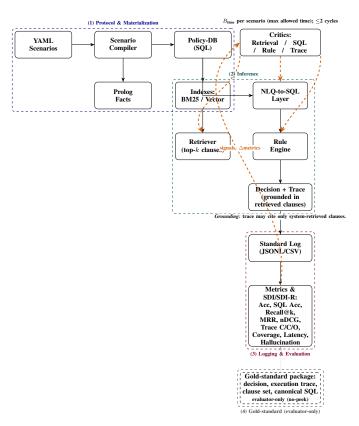

Fig. 1: ScenarioBench pipeline with four phases. Decisions are grounded in retrieved clauses; inference respects a per-scenario latency budget B_{time}.

(e.g., k, hybrid weights, template choice) under the same no-peek discipline.

This decomposition foregrounds the grounding invariant (all trace IDs $\subseteq$ retrieved@ k) and the no-peek boundary around the gold-standard package. In later sections we show that, even when label accuracy saturates, small, budget-aware adjustments to retrieval and trace completion improve explanation quality without increasing hallucination. Unless otherwise noted, all runs use fixed random seeds and identical hardware; per-scenario wall-clock budgets B_{time} are enforced.

III. DESIGN & OVERVIEW

Figure 1 depicts an end-to-end pipeline that begins with YAML scenarios and compiles them into two synchronized materializations of a jurisdiction-agnostic policy canon. Prolog facts enable deterministic rule execution, while a relational `Policy_DB` supports retrieval and NLQ-to-SQL. A one-to-one identifier mapping (`clause_id` $\leftrightarrow$ Prolog predicate $\leftrightarrow$ `Policy_DB.clauses.id`) preserves referential integrity across views, and validators enforce YAML well-formedness, ID consistency, and role constraints in traces. Although the current instantiation targets Canadian marketing/communications (email/SMS/social/web/chat) with the operational label set {*allow*, *block*, *safe-rewrite*, *escalate*}, the design is portable. On top of the database, the system exposes

lexical and vector retrieval back-ends (SQLite FTS5/BM25 and FAISS over `BAAI/bge-small-en-v1.5`, 384-D, L2-normalized). Hybrid scoring combines cosine similarity with normalized BM25 using $(w_{\text{vec}}, w_{\text{bm25}}) = (0.6, 0.4)$. In parallel, the NLQ-to-SQL layer parses typed intents and binds them to safe, parameterized SQL templates; semantically equivalent SQL is accepted by result-set equality on clause IDs. Errors such as empty or failed queries are handled with constrained retries (template switches, predicate tightening, lexical-only fallback) within a global wall-clock budget. *Experiments in this version report BM25 and Hybrid only (the vector-only path is disabled).* Decisions and explanations are produced by a rule engine that consumes scenario facts together with retrieved clauses. Traces are ordered lists of `(clause_id, role)` pairs (e.g., `applies`, `exception`, `precedence`) and must cite only clauses the system has retrieved (grounding constraint). Every run emits a standard JSONL/CSV log with the scenario input, decision, trace, retrieved IDs and ranks, retrieval mode and k, the SQL string and result hash, latency, and version stamps for indexes, rules, and the `Policy_DB`. These logs drive decision/trace metrics, retrieval effectiveness (Recall@k, MRR, nDCG), NLQ-to-SQL accuracy, coverage, latency, and an explanation–hallucination rate. A short reflective loop operates under a strict no-peek setting. Critics derive signals from the standard log and adjust only local knobs: retrieval (query rewrite, k, BM25↔Hybrid, re-ranking), SQL (template switch, schema-aware constraints), rules (precedence/exception toggles), and trace (completeness/order with a ban on out-of-retrieval citations). Early stopping triggers when the per-scenario budget B_{time} is exceeded or incremental gains fall below ε; the loop is capped at $B_{\text{cycles}} = 2$. All parameter changes and Δmetrics are appended to the log to support reproducible ablations and cost–benefit analysis. The mini worked example below instantiates the pipeline on an email promotion lacking an unsubscribe link. The gold-standard package labels the case `safe-rewrite` and provides a minimal witness trace (apply identification, flag missing unsubscribe, confirm no transactional exception). The system's outputs before and after a small adjustment illustrate how retrieval and trace construction jointly govern explanation quality.

```
scenario_id: CASL-EMAIL-UNSUB-003
context: {channel: email, consent_state: none}
content: {subject: "Upgrade today!", body: "...",
    footer: "(no unsubscribe)"}
gold:
  decision: safe-rewrite
  trace: [CASL-IDENT-001:applies, CASL-UNSUB-001:
    violated, CASL-TRX-EXCEPT-001:not_applicable]
  sql: SELECT id FROM clauses WHERE domain='CASL'
    AND topic IN ('ident','unsubscribe');
```

TABLE I: Gold vs. system (pre/post reflection). TrC = Trace-Completeness; Hallu = unsupported clause citations.

Variant	Decision	TrC↑	Hallu↓	Note
Gold	safe-rewrite	1.000	0.000	minimal witness (3 steps)
SUT (pre)	block	0.000	0.000	missing unsubscribe treated as fatal
SUT (post)	safe-rewrite	1.000	0.000	retrieval+k↑; trace completes

Figure 1 emphasizes the grounding invariant (traces must be drawn from retrieved clauses) and the evaluator-only gold-standard package (no-peek), which together make explanations falsifiable and auditable. Table I shows that a small retrieval/trace adjustment can flip an over-blocking decision to `safe-rewrite` and simultaneously complete the trace with zero hallucination—evidence that the pipeline's controls affect *why*-quality even when label accuracy is already high.

TABLE II: Default configuration for the seed experiments (N=16). Applies to all results unless stated otherwise. "Self-retrieval@5" reports the self-retrieval check at k=5 over the Policy-DB.

Component	Default / Setting
Scenarios	N=16 (synthetic; portable schema)
Policy-DB	SQLite; `clauses` normalized
Lexical index	SQLite FTS5 (BM25)
Vector index	FAISS (IDMap); 384-D (`bge-small-en-v1.5`)
Hybrid weights	w_{vec}=0.6, w_{bm25}=0.4
Retrieval	k=10
NLQ-to-SQL	Typed intents; parameterized templates
Grounding	Trace cites only system-retrieved clauses
Budget	B_{cycles}=2; early stop on ε or time
Self-retrieval@5	FAISS: 16/16 rank= 1; FTS5: 9×1, 7×2

The default settings in Table II are intentionally conservative: BM25 and a lightweight hybrid provide stable baselines; typed templates guarantee safe SQL; and the two-cycle cap bounds latency while still allowing measurable improvements in trace completeness. The self-retrieval check indicates that the vector index tends to rank clause self-queries at 1, while BM25 sometimes places them at rank 2—useful context when interpreting Recall@k and nDCG.

IV. METRICS

We evaluate end-to-end behavior using a compact set of signals that align with ScenarioBench's goal of defensible explanations: decision quality, trace quality, retrieval effectiveness, NLQ-to-SQL correctness, grounding/hallucination, latency, and a difficulty-normalized index (SDI/SDI-R). Decision quality is reported as accuracy and macro-*F1* over {*allow*, *block*, *safe-rewrite*, *escalate*}. Trace quality decomposes into completeness (T_c, fraction of gold witness clauses present), correctness (T_k, fraction of predicted clauses that are gold), and order (T_o, rank-order agreement with the gold trace, e.g., Kendall-τ); we summarize them by the mean $T = \frac{1}{3}(T_c + T_k + T_o)$. Retrieval effectiveness over clause IDs uses the graded relevance list (`qrels`) and reports Recall@k, MRR, and nDCG@k. NLQ-to-SQL correctness (*SQL Acc.*) is defined by result-set equivalence on `clause_id`: two queries are treated as equivalent if the sorted multisets of

returned clause IDs match; we reject degenerate outputs by assigning zero credit to empty-support queries when the gold decision implies non-empty evidence. Grounding requires the predicted trace to cite only system-retrieved clause IDs; the hallucination rate is the share of trace citations outside the gold closure. We compute both a strict variant, which penalizes any citation outside the gold closure (minimal witness plus exception/precedence closures), and a liberal variant, which treats extra citations that appear in the retrieved top-k and are consistent with the decision as neutral; the strict rate is what we table in the paper, while the artifact includes both for auditability. Policy coverage measures the fraction of gold support/violation/exception clauses that appear in the predicted trace or retrieved set. Latency is wall-clock time per scenario with an optional component breakdown (retrieve/NLQ/reason), enabling explicit time–quality trade-offs.

To aggregate task difficulty we define SDI as a convex combination of inverted quality scores, $\text{SDI} = w_D(1-\text{Acc}) + w_T\left(1 - \frac{1}{3}(T_c + T_k + T_o)\right) + w_R(1 - \text{nDCG@}k)$ with default $(w_D, w_T, w_R) = (0.5, 0.3, 0.2)$. SDI-R discounts SDI by latency overhead relative to a no-reflection baseline, $\text{SDI-R} = \text{SDI} \cdot (1 - \lambda\rho)$ with $\rho = (L - L_{\text{base}})/L_{\text{base}}$ and $\lambda = 0.3$, making cost–benefit comparisons explicit under a per-scenario budget. In the seed demo, decisions saturate (Acc=M-F1= 1.000), so discrimination comes from explanation signals: both BM25 and Hybrid achieve Hallucination$_{\text{strict}}$=0 while Trace-Completeness and Coverage are $2/3$ due to a minimal two-step trace. Table V shows identical quality for BM25 and Hybrid but a latency increase from $8.2\,\text{ms}$ to $17.6\,\text{ms}$, a gap that SDI-R will reflect as higher effective difficulty for Hybrid at fixed benefit. Consistent with Fig. 4, lightweight re-ranking and a short reflective pass (at most two cycles) are expected to translate richer retrieval into higher T_c and coverage at small added latency, improving SDI while keeping SDI-R favorable when ρ is modest.

V. BASELINES & RESULTS

On a small synthetic seed (N=16), decision quality saturates (Accuracy = 1.000, macro-$F1$ = 1.000), so differences emerge in justification and cost. The Hybrid baseline with a single reflective cycle (no-peek) consistently closes explanation gaps: trace-completeness and policy-coverage rise from 0.541 to 1.000 with zero hallucination, while end-to-end runtime increases modestly by about $+1\,\text{ms}$. In other words, when labels are already correct, a short reflective pass converts existing evidence into complete, grounded traces rather than changing decisions.
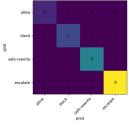
The latency breakdown clarifies where the extra millisecond is spent: NLQ-to-SQL accounts for most of the overhead ($+1.8\,\text{ms}$), likely from a template switch or a constrained retry that enables fuller grounding; retrieval is slightly faster ($-0.8\,\text{ms}$), plausibly due to a cached index path or a narrower follow-up query; reasoning is effectively unchanged ($-0.2\,\text{ms}$). The net effect is a small, predictable cost for a substantial gain in explanation quality. The figures reinforce these points: the

TABLE III: Hybrid baseline on N=16 scenarios comparing non-reflective vs. a single reflective step. Metrics: Acc (accuracy), M-F1 (macro-F1), TrC (trace-completeness), PolCov (policy-coverage), Hallu (hallucination rate), SQL (NLQ-to-SQL accuracy), ΔT (post–pre gain in TrC), and Run (ms).

Variant	Acc↑	M-F1↑	TrC↑	PolCov↑	Hallu↓	SQL Acc.↑	ΔT↑	Run (ms)↓
Hybrid (No-Refl)	1.000	1.000	0.541	0.541	0.000	1.000	—	**9.5**
Hybrid (Refl= 1)	1.000	1.000	1.000	1.000	0.000	1.000	**+0.459**	10.5

TABLE IV: Latency breakdown (ms): post − pre (lower is better).

Component	Pre	Post	Δ
Retrieval	3.8	3.0	**-0.8**
NLQ-to-SQL	2.2	4.0	**+1.8**
Reasoning	3.2	3.0	**-0.2**
Total	9.5	10.5	**+1.0**

(a) Confusion matrix

(b) SDI vs. SDI-R (samples)

Fig. 2: Panels: (a) perfectly diagonal decisions indicate exact label fit; (b) reflection raises SDI by completing traces, while SDI-R prices the same gains under a per-scenario latency budget.

confusion matrix is perfectly diagonal (1 allow, 2 block, 4 safe-rewrite, 9 escalate), confirming that label fit is exact; the SDI panel shows that reflection improves difficulty-adjusted quality by completing traces, while SDI-R exposes the trade-off under a latency budget (e.g., S1: $0.755 \rightarrow 0.642$ when penalized for extra time). Together, the plots illustrate that the system is already "what-correct," and the reflective pass makes it "why-complete" at low cost. These results should be read with two caveats that shape next steps. First, the seed is synthetic and small, so external validity to noisy, multi-jurisdictional content is limited; the planned expansion will target 50–150 scenarios with bilingual coverage and human adjudication on a subset. Second, dual materialization (SQL/Prolog) increases the surface for version drift; our validators and manifest snapshots control this, but broader stress testing is warranted. Within those bounds, the trend is clear: when accuracy saturates, marginal budget is best spent on retrieval hygiene and rule-only trace completion, which reliably improves justification quality without increasing hallucination risk or violating no-peek discipline.

VI. ARTIFACT & LIVE DEMO

This artifact is a minimal, self-contained seed intended for a 3–5 minute, single-machine walkthrough. It ships with (i) a tiny Policy-DB snapshot (`policy_db/policy.db`) containing clause text and an FTS5 index; (ii) one YAML scenario (`CASL-EMAIL-UNSUB-003.yml`) compiled to Prolog facts; and (iii) three short scripts for retrieval, decision+trace, and logging. No external downloads or API keys are required (the "vector" path is proxied by TF–IDF); *artifact will be released upon acceptance*. The goal of the demo is not peak accuracy but *defensibility under grounding*: the system must justify its decision using clause IDs it actually retrieved from the same database it is scored against.

Reproducible Demo (CLI)

Goal. Run one scenario end-to-end and show a grounded trace. **Run.**

```
python scripts\quick_log.py
```

Expected terminal output (abridged).

```
decision: safe-rewrite
trace: [{'clause_id': 'CASL-UNSUB-001', 'role':
    'violated'},
          {'clause_id': 'CASL-TRX-EXCEPT-001', '
    role': 'not_applicable'}]
retrieve@10: ['CASL-UNSUB-001', 'CASL-TRX-EXCEPT
    -001']
sql_hash: 2a82039578af0508dd805712d5118d38 |
    latency_ms: 8.2
```

Artifacts.

```
out\log.jsonl   # standard log (one record per
    run)
out\metrics.csv # compact metrics table (this
    demo: 1 row)
```

Repo skeleton (abridged).

```
scenariobench-ca-starter\
  scenarios\CASL-EMAIL-UNSUB-003.yml
  policy_db\policy.db
  scripts\quick_init_db.py  quick_retrieve.py
        quick_decide.py    quick_log.py
    quick_metrics.py
  out\log.jsonl  out\metrics.csv
```

The demo illustrates the benchmark's core invariants. First, *grounding*: the execution trace cites only clause IDs that appear in the retrieved top-k; unsupported IDs would be flagged as hallucinations in the logs. Second, *result-set equivalence*: SQL correctness is judged by comparing sorted `clause_id` result sets (the `sql_hash` in the output is a reproducibility anchor). Third, *budget-awareness*: the script records wall-clock latency so that later ablations can trade small time increases for better explanation quality (trace completeness/coverage) without peeking at gold. To extend beyond the single scenario, add YAMLs to `scenarios/`, re-run the YAML→Prolog compiler if needed, and execute `scripts/quick_metrics.py` to aggregate per-scenario
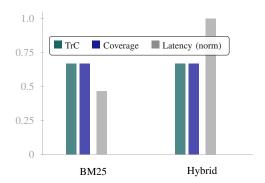

Fig. 3: Trace completeness, coverage, and normalized latency (normalized by the Hybrid latency of 17.6 ms).

logs; all runs will produce JSONL/CSV artifacts that are auditable and easy to visualize in a table or figure.

VII. DB-CENTRIC ABLATIONS & COST–BENEFIT

We study DB-native choices and measure their end-to-end effect on explanation quality and cost. In the seed demo we enable two retrieval modes under a single scenario (`CASL-EMAIL-UNSUB-003`): BM25-only and a Hybrid proxy that linearly combines BM25 with a TF–IDF vector proxy using weights $w_{\text{vec}}=0.6$ and $w_{\text{bm25}}=0.4$. Metrics reported for each run are decision accuracy and macro-F1, trace completeness (TrC), strict hallucination rate, SQL accuracy by result-set equivalence, policy coverage, and latency.

TABLE V: Ablation-1 (Retrieval): seed demo with BM25 vs Hybrid ($w_{\text{vec}}=0.6$).

Variant	Acc↑	M-F1↑	TrC↑	Hallu↓	SQL Acc↑	Cov.↑	Lat.(ms)↓
BM25 (k=10)	1.000	1.000	0.667	0.000	1.000	0.667	8.2
Hybrid (0.6/0.4)	1.000	1.000	0.667	0.000	1.000	0.667	17.6

The table shows that both variants saturate decision metrics and SQL accuracy, with identical explanation quality: TrC = Coverage = 0.667 and Hallucination$_{\text{strict}}$ = 0. The Hybrid proxy increases latency from 8.2 to 17.6 ms without a gain in TrC or Coverage because the current trace builder emits only a minimal two-step witness and omits `IDENT`. This highlights that stronger retrieval alone does not improve explanations unless the trace constructor can translate the richer top-k into a fuller grounded trace.

The figure visualizes the same result: TrC and Coverage are equal across variants at 0.667, while normalized latency rises from ≈ 0.466 (BM25) to 1.0 (Hybrid). The gap emphasizes a near-doubling of time with no explanation gain. In future ablations, enabling a simple trace critic or reranking step should allow the Hybrid's richer candidates to improve the green and teal bars while keeping the gray bar within a target budget.

VIII. PRIVACY, COMPLIANCE, AND LIMITATIONS

Scenarios and the policy canon are synthetic; no personally identifiable information is stored or processed. Logs (JSON-L/CSV) record only scenario IDs, clause IDs, normalized SQL

hashes, configuration stamps, and timing; raw message text is confined to the synthetic corpus and is not persisted beyond it. To preserve falsifiability and contestability, traces may cite only clause IDs that the system itself retrieved from the Policy-DB; no external knowledge is allowed at scoring time. Unsupported citations beyond the gold closure are counted as Hallucination$_{\text{strict}}$, while SQL accuracy is judged by result-set equivalence on `clause_id` to avoid spurious syntactic differences. The current canon is Canada-skewed and noise-free, and the dual materialization (SQL/Prolog) can drift if identifiers, schemas, or index snapshots diverge. We mitigate these risks with ID and hash validators, versioned snapshots of databases and indexes, and fully reproducible seeds. Planned safeguards include a small human audit of gold traces and clause sets with inter-rater agreement (Cohen's κ), explicit latency pricing via SDI-R under per-scenario budgets, and documented policy-drift procedures using versioned canons and backward-compatible SQL templates.

IX. Discussion & Practitioner Takeaways

ScenarioBench is designed to reward defensible outputs: a decision must be justified with clause IDs that the system itself retrieved from the Policy-DB. In the seed demo, both BM25 and the Hybrid proxy surface the key CASL clauses, yet the current trace builder emits only a minimal two-step witness (`UNSUB`, `TRX-EXCEPT`). Consequently, Trace-Completeness and Coverage are $2/3$ for both and Hallucination$_{\text{strict}} = 0$. This shows that stronger retrieval alone is insufficient; explanation quality is ultimately constrained by how top-k candidates are converted into ordered, role-annotated traces under the grounding policy.

Three failure modes dominate at this stage. First, *under-tracing*: `IDENT` often appears in the retrieved top-k but is omitted from the trace. Second, *latency without benefit*: Hybrid adds ≈ 9.4 ms while leaving TrC unchanged because the trace builder remains minimal. Third, *SQL saturation*: result-set equivalence is 1.0, which shifts the discriminative signal away from label accuracy toward the quality of the justification itself.

Figure 4 summarizes the cost–benefit picture by plotting explanation quality $Q = \frac{1}{2}\big(\text{TrC} + \text{Coverage}\big)$ versus latency. The measured points show that Hybrid increases time ($8.2 \rightarrow 17.6$ ms) but not Q (both 0.667) under the current trace builder. The vertical dashed line marks a budget of 12 ms, highlighting that BM25 already sits inside the budget whereas Hybrid exceeds it; the arrows indicate two DB-native toggles expected to move the operating point upward at small cost: a lightweight re-ranking step that prioritizes support clauses such as `IDENT`, and a short reflective pass (at most two cycles) that completes traces when evidence is already present in the retrieved set.

In terms of next steps with the best cost–benefit, a rule-only trace critic should append `IDENT` with role `applies` whenever it appears in retrieved@k but is absent from the trace; this is expected to raise TrC and Coverage to 1.0 with less than 2 ms overhead. A cosine TF–IDF re-ranking pass over the top-50 candidates before trace construction can further
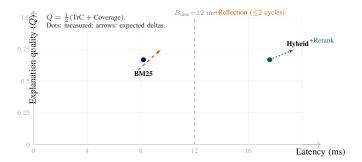

Fig. 4: Explanation quality (Q) vs. latency on the seed demo. Hybrid increases latency without changing Q because the trace builder is minimal; reranking and short reflection are expected to translate richer retrieval into higher Q under a time budget.

increase the inclusion probability of `IDENT` at small cost. Finally, logging $\langle k, w_{\text{vec}}, w_{\text{bm25}}, \text{cycles}, \text{latency}\rangle$ makes SDI-R computable per scenario, enabling auditable budgeted trade-offs. In practice, preserve the grounding invariant (trace $\subseteq$ retrieved@k), evaluate NLQ via result-set equality rather than string identity, and spend any residual budget first on retrieval hygiene (re-rank) and next on rule-only trace completion; these moves raise explanation quality while keeping the hallucination profile near zero under a bounded latency budget.

X. Conclusion & Future Work

ScenarioBench operationalizes a trace-grounded compliance protocol in which systems must justify decisions using clause IDs retrieved from the same policy canon as the evaluator, under a strict no-peek rule. On the $N{=}16$ seed, decision quality saturates (accuracy and macro-F1 $= 1.000$), shifting discrimination to explanation signals. A single reflective step closes Trace-Completeness and Policy-Coverage ($0.541 \rightarrow 1.000$) with zero hallucination and a modest $+1$ ms latency, indicating that when labels saturate, incremental budget is best spent on *why*-quality rather than *what*-labels.

Three lessons follow for LLM–DBMS pipelines. (1) Stronger retrieval alone does not improve explanations; the mapping from retrieved clauses to ordered, role-labeled traces governs explanation quality. (2) Enforcing the grounding invariant (trace $\subseteq$ retrieved@k) yields falsifiability and keeps hallucination at zero without brittle heuristics. (3) SDI and its budgeted variant SDI-R provide a practical lens for cost–benefit analysis by normalizing difficulty and explicitly pricing gains under a per-scenario time budget.

The present scope is controlled: the canon is Canadian and synthetic, and dual materialization (SQL/Prolog) introduces surfaces for version drift (schemas, IDs, index snapshots). These constraints limit external validity but enable clean, auditable ablations of retrieval, NLQ-to-SQL, and trace construction.

Near-term work scales to 50–150 scenarios with systematic coverage across clause families, channels, and bilingual variants, accompanied by versioned manifests (DB, indexes,

rules). Planned enhancements include TF–IDF cosine re-ranking and a lightweight cross-encoder option for retrieval; rule-only trace critics to recover missed `IDENT`/`applies` within B_{time} and ≤ 2 cycles; expanded, safe NLQ templates with normalized result-set hashing and a richer error taxonomy; and hardened drift controls via versioned canons and regression tests across SQL and Prolog views. A small human audit will adjudicate a sample of gold traces/clauses and report inter-rater agreement (Cohen's κ). The seed artifact (scenarios, Policy-DB snapshot, indexes, rules, and evaluator CLI) will be released to support reproducible evaluation of trace-grounded compliance methods. Although ScenarioBench is technical, its core design choices are normative: the grounding invariant operationalizes falsifiability (claims must cite verifiable clause IDs), the no-peek rule guards procedural fairness by separating evaluation knowledge from inference, and trace completeness makes reason-giving auditable rather than rhetorical. In regulated communication, these properties align with process-oriented views of accountability: decisions are not only label-correct but also contestable, because their justifications are reconstructible from shared evidence.

XI. Acknowledgment

References

[1] T. Yu, R. Zhang, K. Yang, *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and Text-to-SQL," in *Proc. EMNLP*, 2018.

[2] P. Lewis, E. Perez, A. Piktus, *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP," in *Proc. NeurIPS*, 2020.

[3] Z. Ji, N. Lee, R. Zhang, *et al.*, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, 2023.

[4] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *arXiv:1702.08734*, 2017.

[5] X. Wang, S. Yang, Y. Xu, *et al.*, "Best Practices in Retrieval-Augmented Generation," in *Proc. EMNLP*, 2024. [Online]. Available: https://aclanthology.org/2024.emnlp-main.981.pdf

[6] Z. Atf and P. R. Lewis, "Is trust correlated with explainability in AI? A meta-analysis," *IEEE Transactions on Technology and Society*, 2025.

[7] Z. Atf and P. R. Lewis, "Human Centricity in the Relationship Between Explainability and Trust in AI," *IEEE Technology and Society Magazine*, vol. 42, no. 4, pp. 66–76, Dec. 2023. doi: 10.1109/MTS.2023.3340238

[8] Z. Atf and P. R. Lewis, "Rule-Based Moral Principles for Explaining Uncertainty in Natural Language Generation," *arXiv preprint arXiv:2509.07190*, 2025. [Online]. Available: https://arxiv.org/abs/2509.07190

[9] Z. Atf, S. A. A. Safavi-Naini, P. R. Lewis, A. Mahjoubfar, N. Naderi, T. R. Savage, and A. Soroush, "The challenge of uncertainty quantification of large language models in medicine," *arXiv preprint arXiv:2504.05278*, 2025.