A Hybrid RAG–Text2SQL Architecture for Auditable Question Answering in Brazilian Portuguese Legal-Administrative Collections

Josiel Pantaleão Cardoso Silva[1], **Sávio Salvarino Teles de Oliveira**[1]
Matheus Fares Costa Brakes[1], **João Paulo Cavalcante Presa**[1]

[1] [1]Instituto de Informática – Universidade Federal de Goiás (UFG), Goiânia-GO-Brasil

josielpantaleao@discente.ufg.br, savioteles@ufg.br
faresmatheus@gmail.com, joaopaulopresa@gmail.com

Abstract. *Question answering over Brazilian Portuguese legal-administrative collections must support semantic information needs and structured intents such as counting, filtering, aggregation, and ranking. Retrieval-augmented generation (RAG) grounds answers in text, but it does not reliably execute relational operations. Standalone Text2SQL can execute auditable queries, but lacks textual grounding and explanatory coverage. This paper presents a hybrid architecture that runs semantic retrieval and a constrained Text2SQL pipeline in parallel and synchronizes both outputs into a single answer. The structured branch includes column routing, schema restriction, normalized derived columns,* `WHERE`*-hints, read-only execution policies, and decomposition for multi-intent questions. We formalize the evaluation protocol for structured and semantic queries, define the proposed metrics, and clarify the role of the LLM-based judge used for semantic assessment. On a curated TCE-GO corpus, the hybrid architecture outperforms a RAG-only baseline on structured queries while remaining competitive on semantic ones. The contribution lies in showing how semantic retrieval and constrained SQL execution can be combined for Portuguese legal-administrative QA with explicit safety, auditability, transparency, and evaluation policies.*

1. Introduction

Brazilian legal-administrative collections combine documents, domain language, metadata, and information needs that range from open-ended questions to relational operations such as count, group by, ranking, temporal filtering, and selection by institution. In this paper, semantic understanding means the ability to retrieve and synthesize textual passages to explain an answer, justify its documentary basis, and accommodate lexical variation. By contrast, structured data operations mean deterministic execution of filters, aggregations, joins, ordering, and restrictions over relational attributes. These needs coexist in hybrid questions such as: "How many decisions of a given type were issued in 2023, and which legal grounds appear most frequently?"

RAG [Lewis et al. 2020, Gao et al. 2023] conditions generation on retrieved evidence, but it does not provide guarantees for counts, aggregations, or exact filtering. Text2SQL offers an auditable intermediate representation for structured intents, but it lacks the textual grounding required for explanation. In legal and administrative settings, correctness, auditability, and verifiability are requirements [Chalkidis et al. 2020, Chalkidis et al. 2022, Katz et al. 2023].

We propose a hybrid RAG–Text2SQL architecture for question answering in Brazilian Portuguese. The system runs two branches in parallel: a semantic branch for retrieving textual evidence, and a structured branch for generating and executing SQL over a relational database. The outputs are synchronized, prioritizing structured execution for deterministic operations and incorporating textual evidence for explanation.

The goal of this paper is system architecture and experimental protocol. We show how semantic retrieval and constrained SQL execution can be combined for Portuguese legal-administrative QA with explicit safety, auditability, and evaluation policies. We rely on derived columns, which are structured attributes generated offline from document content to support auditable SQL execution. Examples include `decision_year` and `doc_category`. During inference, the system queries their persisted values, which improves scalability and makes answers easier to audit.

The contributions of the paper are: i) a hybrid QA architecture for Brazilian Portuguese legal-administrative collections, synchronizing semantic retrieval and safe SQL execution; ii) an operational Text2SQL pipeline with column routing, schema restriction, `WHERE`-hints, derived columns, decomposition rules, and read-only validation policies; iii) a clearer experimental protocol, including explicit metric definitions, gold-standard construction for structured answers, and a constrained role for the LLM-based judge in semantic evaluation.

2. Related Work

RAG has become a dominant strategy for domain QA because it improves factuality by conditioning generation on retrieved evidence [Lewis et al. 2020, Gao et al. 2023]. Dense retrieval methods such as Sentence-BERT [Reimers and Gurevych 2019] and late-interaction models such as ColBERT [Khattab and Zaharia 2020] mitigate lexical mismatch and paraphrastic variation, while architectures such as FiD [Izacard and Grave 2021] better exploit multiple passages at generation time. In this paper, we avoid claiming that these methods solve phenomena that are specifically "prevalent in Portuguese" without direct empirical evidence. Instead, we treat the advantage of dense retrieval as a general property of specialized vocabularies and long-document corpora.

In legal NLP, domain adaptation and legal benchmarks show that general-purpose models benefit from adaptation to specialized corpora and tasks [Gururangan et al. 2020, Chalkidis et al. 2020, Chalkidis et al. 2022]. For long collections, retrieval quality and context organization remain practical bottlenecks, especially because long-context generation suffers from redundancy and the lost-in-the-middle effect [Liu et al. 2024]. This motivates our use of query expansion, fusion, reranking, and aggressive deduplication in the semantic branch.

Text2SQL research has evolved beyond raw SQL generation into questions of control, restriction, and robustness. PICARD is a representative example of constrained decoding designed to avoid invalid SQL [Scholak et al. 2021]. Other lines of work have explored retrieval-augmented Text2SQL, schema retrieval, probing-based schema selection, plan-oriented intermediate representations, and agentic decomposition strategies [Zhang et al. 2023]. Instead, we design a system with explicit guardrails for a Portuguese legal-administrative environment, where safe execution and answer traceability matter as much as raw generation quality.

A second relevant axis in Text2SQL concerns schema linking and schema selection [Pourreza and Rafiei 2023]. In realistic databases, generation quality often depends as much on identifying the relevant subset of schema elements as on decoding the SQL itself [Li et al. 2023]. This is particularly important in institutional settings, where schemas may expose many partially relevant attributes and where natural-language references are often indirect. Our column-routing stage is conceptually related to this line of work, although implemented here as an applied control layer rather than as a standalone learning contribution.

Another relevant line of work studies decomposition and tool orchestration for multi-step question answering [Khot et al. 2023, Yao et al. 2023]. Rather than forcing a single model invocation to jointly resolve filtering, aggregation, and explanation, these approaches split the problem into intermediate steps that are easier to validate and audit. Our decomposition strategy follows this general philosophy, but with a narrower focus on legal-administrative QA, where the main design objective is controlled execution and traceable outputs rather than agent autonomy.

Hybrid QA systems combine modules in different settings. Our proposal focuses on hybrid intents, auditability, and offline-normalized derived columns to make quantitative operations scalable without depending on document interpretation at query time.

3. Task Definition and Architecture

We distinguish three query types. A **semantic query** is a question whose answer depends mainly on retrieving and synthesizing textual passages, for example: “What requirements are mentioned for waiving a bidding procedure in cases of public calamity?” A **structured query** is a question whose answer depends mainly on explicit relational operations, for example: “How many decisions of type X were issued in 2023?” Finally, a **hybrid query** requires both types of treatment, for example: “How many such decisions exist in 2023, and which legal basis is most frequent among them?”

Figure 1 illustrates the macro-architecture of the proposed hybrid QA system. The processing pipeline begins when a user submits a natural language question. To handle the high lexical variability and the complex terminology of legal-administrative documents, the input first passes through a **Query expansion** module, which generates multiple semantic variations of the original question to maximize retrieval recall.

Following the expansion, the execution flow splits into two parallel streams, reflecting the dual reasoning paradigms required by the domain. On the left, the **Semantic Branch** operates over unstructured text, focusing on dense retrieval and passage reranking. Simultaneously, on the right, the **Structured Branch** handles deterministic data operations, translating the query into an executable format through column routing and safe SQL generation. The internal mechanics of each branch are detailed in Sections 3.1 and 3.2, respectively.

Finally, the intermediate results from both parallel branches converge into the **Synchronizer**. This final component acts as an LLM-mediated adjudicator. It evaluates both the retrieved documentary evidence and the deterministic tabular data, resolving potential conflicts and synthesizing the information. The output is a single, coherent **Final answer** that provides the exact quantitative or relational result alongside its supporting textual context.

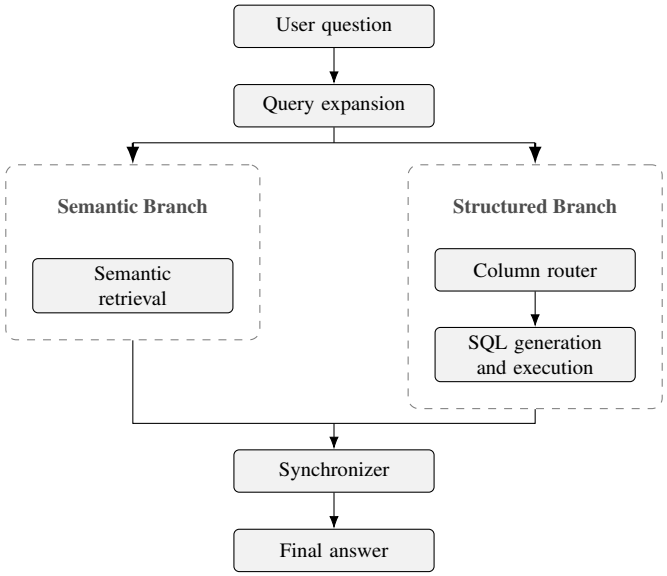

Figure 1. Overview of the proposed hybrid architecture. The system processes the question in parallel through semantic and structured branches and synchronizes both outputs into a single response.

3.1. Semantic branch

The semantic branch receives 3 to 6 variants of the original question and performs vector retrieval with top-k search for each variant. The result sets are fused, score-normalized, reranked, and deduplicated. A confidentiality filter then removes or masks restricted spans before the final context is passed downstream. The objective is to deliver a compact set of high-signal evidence passages rather than a large but redundant context. Query expansion is implemented as a controlled reformulation step: variants preserve the original intent while adding synonyms, paraphrases, and legal-administrative formulations before retrieval.

The use of multiple query variants is motivated by the high lexical and stylistic variability of legal-administrative language. The same procedural concept may be expressed through formal legal terminology, abbreviated institutional phrasing, or paraphrastic formulations closer to user language. Multi-query retrieval therefore acts as a recall-oriented mechanism, while reranking and deduplication compensate for the redundancy introduced by this expansion.

Beyond passages themselves, the semantic branch can emit lightweight contextual hints such as years, institution names, and document types mentioned with high confidence. These hints do not replace structured interpretation, but they help reduce ambiguity in the SQL branch.

3.2. Structured branch

Figure 2 presents the detailed execution flow of the Text2SQL branch. In contrast to a direct "question-to-SQL" invocation, the branch is designed as a constrained execution pipeline. The figure is not merely illustrative: it shows how query analysis, schema-oriented mapping, SQL generation, validation, execution, and response composition interact in the proposed system.

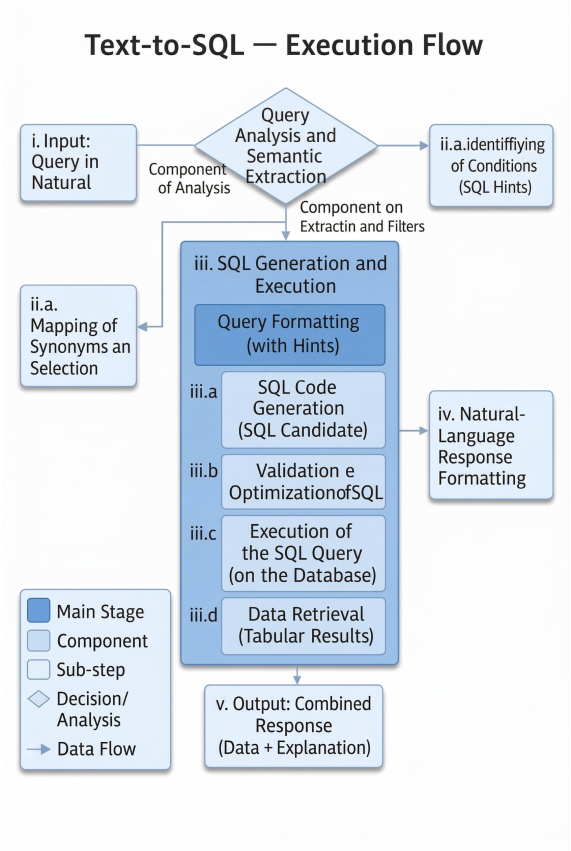

Figure 2. Detailed Text2SQL execution flow used in the structured branch. Starting from natural-language input, the pipeline passes through query analysis and semantic extraction, identifies conditions and schema mappings, generates a candidate SQL statement, validates and optimizes it, executes the query, retrieves tabular results, and returns a combined response with data and explanation.

Query Analysis and Semantic Extraction. The pipeline begins with a natural language query, which undergoes immediate intent evaluation (*Query Analysis and Semantic Extraction*). At this stage, the system identifies whether the question contains explicit or implicit signals of aggregation, ranking, exact filtering, temporal restriction, entity selection, or multi-intent composition. Based on this analysis, the system performs two parallel preparatory steps before generating any code.

First, a column router estimates which database attributes are potentially relevant for the question (*Mapping of Synonyms and Selection*). It uses domain lexicons, synonym mappings, lightweight cues from the semantic branch, and question templates to output a restricted schema subset. The metric *Acc@Column* measures whether this routed subset contains all necessary columns and avoids critical omissions. In practice, this routing step is important because legal-administrative questions often refer to database attributes only indirectly. Users may mention a procedural category, an institutional actor, or a time constraint using surface forms that do not coincide with column names or stored values. By inserting an intermediate lexical-to-schema mapping stage, the system reduces the burden on the SQL generator and makes routing errors easier to inspect.

Second, temporal expressions, document categories, institutional actors, and normalized entity mentions are converted into candidate values that can feed `WHERE` clauses (*Identifying of Conditions / SQL Hints*). These hints do not force a single SQL query, but they reduce the search space of the generator and make filter values more stable. This is especially useful when the question contains underspecified but recoverable constraints, improving both stability and auditability.

SQL Generation and Execution. Once the schema is restricted and constraints are identified, the pipeline constructs a structured prompt combining these elements (*Query Formatting with Hints*). This tailored prompt guides the LLM to formulate a draft query targeting the specific relational tables (*SQL Code Generation / SQL Candidate*). However, before any database interaction occurs, this draft goes through a strict deterministic algorithm (*Validation and Optimization of SQL*). Only `SELECT` statements are allowed. Modification keywords, unsafe subqueries, and excessive result volumes are rejected. The validator inserts a conservative `LIMIT` when needed and enforces stable ordering in ranking or aggregation scenarios. This is important in the legal domain, where the structured branch must remain auditable and non-destructive. If the candidate query passes all safety policies, the system safely runs it against the database (*Execution of the SQL Query*) and retrieves the exact relational data (*Data Retrieval / Tabular Results*).

Fallback behavior. If the structured branch cannot produce a validated query during the aforementioned validation step, the system does not emit free-form pseudo-SQL. Instead, it falls back to the semantic branch and explicitly avoids unsupported quantitative claims. This preserves usability without hiding execution failure.

Decomposition and Response Formatting. To handle complex queries, the system employs decomposition before synthesizing the final text. Multi-intent questions are decomposed when at least one of the following conditions holds: (i) the query contains more than one aggregation target; (ii) the answer requires both a quantitative result and an explanatory summary; or (iii) the explanation depends on a subset defined by a structured filter. For example, the question "How many decisions of type X were issued in 2023, and which legal grounds appear most frequently?" can be split into: (a) one SQL subquery for the count, and (b) one semantic step restricted to the documents returned by the structured filter.

Decomposition is represented internally as a small execution plan rather than as a free-form chain of prompts. Each substep has a typed role, such as quantitative extraction, subset definition, or textual summarization. Intermediate outputs remain linked, allowing the system to compose the final answer in a traceable way (*Natural-Language Response Formatting*), instead of relying on a single monolithic generation step. Finally, the system synthesizes this synchronized data to explicitly state the computed tabular value alongside the supporting textual evidence (*Output: Combined Response*).

4. Experimental Setup

This section makes the evaluation protocol more explicit than in the earlier version of the manuscript.

4.1. Implementation and infrastructure

The system is implemented in Python 3.12, with workflow orchestration in a state graph and an API layer in FastAPI. The semantic branch uses Qdrant 1.11 as vector storage, while the structured branch uses PostgreSQL 16 as relational backend. LLM-mediated steps such as query expansion, column routing, SQL generation, and final synthesis are run with a GPT-4.1-class model at temperature 0. In the semantic branch, the system generates 3 to 6 query variants, retrieves the top 4 passages per variant, and reduces the final context to at most 10 reranked passages.

4.2. Corpus and database construction

Experiments are conducted on a curated corpus from the Court of Auditors of the State of Goiás (TCE-GO). The relational database stores identifiers, metadata, and derived columns extracted offline, while the vector index stores chunked textual passages.

Derived columns were created through normalization rules and semantically assisted extraction, followed by manual validation on development samples. Their purpose is to make operations such as filtering by year, category, institution, and monetary value auditable without on-the-fly parsing during user inference. We treat them as operational schema features whose quality is verified during corpus construction.

The application of the proposed solution requires persistent document identifiers, normalized metadata, and relational attributes aligned with the expected structured operations. In particular, temporal filtering, document-type filtering, institutional filtering, counting, aggregation, and ranking depend on fields prepared before inference. Without these attributes, the Text2SQL branch would either fail validation or rely on unstable interpretation at query time.

The offline construction followed a curation protocol: candidate attributes were selected, extraction rules were iteratively refined, and the attributes were manually inspected on targeted samples. This process reduces the dependence on unstable interpretation during inference.

4.3. Evaluation sets and annotation protocol

The evaluation contains 550 structured queries (requiring counting, exact filtering, aggregation, or ranking) and 580 semantic queries (requiring textual synthesis or justification). Questions with mixed behavior are assigned according to the dominant evaluation criterion: if correctness primarily depends on a verifiable relational operation, the question belongs to the structured set; if correctness primarily depends on textual grounding, it belongs to the semantic set.

For each structured question, annotators define the required database columns and the exact target answer. To evaluate different systems under a common protocol, we standardize the formatting of these answers. If a system provides a correct value surrounded by explanatory text, a normalization step extracts the core value and applies a uniform format. Categorical answers are also mapped to a standard label. This ensures that systems are evaluated on data correctness rather than formatting variations.

For semantic questions, there is no single reference string. Instead, answers are evaluated against the retrieved document evidence using a rubric. This assessment relies

on an evaluation judge, which operates exclusively on semantic outputs. The judge is not used to create or score target values for structured queries.

4.4. Metrics

To reduce ambiguity, we define the metrics operationally.

- `Acc@Column`: For each structured question, let C^* be the minimal set of necessary columns and $\hat{C}$ be the subset returned by the router. The metric is 1 if all necessary columns are present ($C^* \subseteq \hat{C}$) and no omission prevents execution; otherwise, it is 0.
- `Exec@SQL`: Receives 1 if the generated SQL statement passes validation, executes without error, and returns a well-formed result, regardless of final answer correctness.
- `Answer@Exec`: Evaluates the correctness of the structured answer. Let y^* be the normalized gold answer and $\hat{y}$ be the normalized system output. The metric is 1 when $\hat{y} = y^*$ and 0 otherwise. For free-form baseline outputs, normalization removes surrounding text and standardizes quantities before comparison.
- `Sem@F1`: Used for semantic queries. It is a continuous score in $[0, 1]$ assigned by an LLM-based judge based on relevance, completeness, and faithfulness to retrieved evidence.

Regarding the LLM-judge protocol, the evaluator receives the question, the generated answer, and the retrieved evidence. The rubric instructs the judge to penalize unsupported claims, omitted conditions, and contradictions. The judge operates exclusively on semantic assessment and is not used for structured-value annotation. We manually validated this procedure on a sample of 50 question–answer pairs. The score functions as an approximation of answer quality under evidence, bypassing the limitations of lexical overlap.

4.5. Reproducibility constraints

The corpus contains institutional data that cannot be fully released, and some pipeline stages use proprietary models. Consequently, our reproducibility claim is mainly **methodological**: we aim to make the architecture, policies, metrics, and evaluation decisions explicit even when not all artifacts can be published in full.

5. Results

This section details the experimental evaluation of the hybrid architecture. The analysis segregates the results into structured queries, semantic queries, and execution latency. We compare the hybrid pipeline against standalone RAG and Text2SQL baselines to examine the operational differences between deterministic relational execution and vector-based text retrieval. The evaluation frames the metrics as a trade-off between execution correctness and response time for auditable question answering systems.

5.1. Structured queries

Table 1 compares the hybrid architecture with a RAG-only baseline. Since the RAG-only baseline neither generates SQL nor accesses the relational database, *Acc@Column* and *Exec@SQL* do not apply to it.

Table 1. Results on structured queries.

Metric	RAG-only	Hybrid	Δ abs.
Acc@Column	N/A	**0.84**	N/A
Exec@SQL	N/A	**0.91**	N/A
Answer@Exec	0.09	**0.98**	+0.89

The "N/A" (Not Applicable) values for the RAG-only baseline in *Acc@Column* and *Exec@SQL* are an architectural consequence of its design. A pure RAG system operates over unstructured text chunks via vector retrieval and does not map natural language to a relational schema. Consequently, it neither selects database attributes nor generates executable SQL statements, making intermediate relational metrics operationally undefined for this baseline.

The difference in the final *Answer@Exec* metric demonstrates the structural limitation of relying solely on textual retrieval for deterministic operations. To resolve questions that require counting, exact filtering, or aggregation, the RAG-only baseline retrieves localized text snippets and applies approximate linguistic inference. In contrast, the hybrid architecture converts the semantic intent into relational algebra, executes the query against the structured database, and guarantees exact computation over the target data population.

To illustrate this structural limitation in RAG-only systems, consider a question such as "How many decisions of category X were issued in 2023?". The RAG-only system retrieves passages mentioning both the document category and the year 2023. However, because vector retrieval operates on top-k passages rather than full table scans, the final answer reflects local evidence rather than the complete database population. Typical outputs either produce an inferred count based on retrieved examples or use non-deterministic wording such as "several decisions", even when the gold answer is an exact scalar. In the hybrid system, the same question is answered by validated SQL, ensuring full population coverage, and the final response cites representative evidence passages from the relevant subset.

5.2. Ablation of the Text2SQL branch

Table 2 isolates the operational contribution of each pipeline component in the structured branch. The baseline setup (**Base**) represents a direct natural-language-to-SQL generation. This configuration exposes the entire database schema to the generator without intermediate constraints, resulting in baseline performance for schema linking (*Acc@Column*), query syntax validity (*Exec@SQL*), and answer correctness (*Answer@Exec*).

Table 2. Ablation of the Text2SQL branch on structured queries.

Metric	Base	+Router	+Hints +Policies	+Decom-position
Acc@Column	0.58	0.72	0.81	**0.84**
Exec@SQL	0.73	0.86	0.95	**1.00**
Answer@Exec	0.71	0.88	0.94	**0.98**
Latency p50 (s)	18.1	23.2	27.1	**29.7**

The addition of the schema router (**+Router**) introduces a pre-processing step that restricts the database attributes exposed to the generator. By narrowing the search space to a subset of relevant columns, this component increases *Acc@Column*. Providing a focused schema allows the generator to formulate executable syntax more consistently, which reflects in the increase of *Exec@SQL* and *Answer@Exec*, at the cost of additional routing latency.

Introducing semantic extraction and strict validation (**+Hints +Policies**) alters the generation and execution phases. The hints mechanism pre-extracts entities and temporal bounds, injecting them as concrete values to resolve `WHERE` clause ambiguity. Simultaneously, the execution policies enforce deterministic constraints, rejecting unsafe operations, malformed subqueries, and unauthorized data mutations prior to database interaction. This combined step prevents the execution of invalid queries, increasing *Exec@SQL* to 0.95.

Finally, the decomposition module (**+Decomposition**) addresses multi-intent questions. Instead of overloading a single SQL statement with conflicting aggregations or mixed relational-textual targets, the pipeline divides the workload into a discrete execution plan with sequential sub-queries. This modularization yields an *Exec@SQL* of 1.00.

The increase in median latency across the columns reflects the accumulation of sequential operations—routing, extraction, validation, and multi-step execution—required to guarantee deterministic relational outputs in an auditable environment.

5.3. Semantic queries

Table 3 reports the results for queries that depend on document retrieval and textual synthesis.

Table 3. Results on semantic queries.

Metric	RAG-only	Hybrid	Δ abs.
Sem@F1	**0.82**	0.77	-0.05
Latency p50 (s)	**11.8**	19.3	+7.5

The reduction in *Sem@F1* indicates that the structured branch provides no advantage for purely semantic questions. From an architectural perspective, when a query requires only text retrieval, the parallel Text2SQL pipeline evaluates an empty or non-contributory relational state. Consequently, the final synchronization module must adjudicate between grounded textual passages and null tabular data. This merging step forces the generator into a conservative state, diluting the focus on the retrieved evidence and affecting the final completeness score.

The increase in median latency reflects the computational overhead of invoking the column router, generating unneeded SQL candidates, and running the LLM-mediated synchronizer. These results establish the operational boundaries of the architecture: the hybrid system trades peak semantic efficiency for execution guarantees on relational intents. The goal of the framework is to establish structured auditability while maintaining a functional baseline for textual reasoning, rather than claiming superiority across all query paradigms.

5.4. Consolidated comparison

Table 4 summarizes the performance and latency trade-offs across the evaluated architectures.

Table 4. Consolidated comparison across systems.

System	Answer@Exec	Sem@F1	Latency p50 (s)
RAG-only	0.09	**0.82**	**12.4**
Text2SQL-only	0.85	0.65	18.2
Hybrid	**0.98**	0.77	29.7

The isolated baselines expose the limitations of monolithic reasoning paradigms. The RAG-only system optimizes for textual grounding (*Sem@F1* of 0.82) but lacks the mechanisms for deterministic relational computation. Conversely, the Text2SQL-only system executes relational operations over the database but operates without unstructured documentary evidence. Consequently, it returns raw tabular data without the explanatory context required for legal justification, resulting in a *Sem@F1* of 0.65.

The hybrid architecture synchronizes both data representations. It elevates the relational execution metric to 0.98, surpassing the standalone Text2SQL baseline while preserving a text-grounding score of 0.77.

The accumulation of parallel evidence retrieval, SQL compilation, and LLM-mediated synchronization pushes the median latency to 29.7 seconds. In legal and administrative environments, exact quantitative answers and documented evidence take precedence over sub-second response times. Therefore, the proposed architecture positions execution latency as a functional operational cost to establish correctness and auditability over institutional collections.

6. Discussion

The experimental evaluation supports three architectural conclusions. First, structured intents in institutional question answering require deterministic execution; vector retrieval of relevant passages is functionally insufficient. Second, engineering guardrails account for the operational differences observed in structured QA. Third, synchronizing structured and semantic branches yields auditability, but introduces explicit trade-offs regarding latency and system complexity.

Real-world deployment raises social, ethical, and legal concerns, including unequal access, incomplete records, biased data, and overreliance on generated answers. The system should support human decision-making under traceability, oversight, access control, auditing, and clear communication of limitations.

Delimiting the scope of the contribution is a methodological requirement. This study is not designed to compete with general-domain Text2SQL benchmarks such as Spider or BIRD, nor does it claim universal algorithmic superiority over alternative hybrid QA architectures. The operational baseline is a Portuguese institutional environment governed by explicit safety, verifiability, and domain adaptation constraints. Consequently, the paper positions itself as an applied architecture study for auditable QA.

The results indicate that hybrid QA in institutional domains functions as a systems coordination problem rather than a pure language modeling problem. The architectural challenge is not solely to optimize embedding retrieval or SQL decoding in isolation, but to define the execution boundaries of each paradigm. System design must determine when exact computation supersedes textual synthesis, how intermediate routing signals constrain generation, and how the final output maintains documentary traceability for auditors and end users.

7. Conclusion

We presented a hybrid RAG–Text2SQL architecture for question answering over legal-administrative collections in Brazilian Portuguese. The system addresses the dual nature of institutional queries by running semantic retrieval and constrained SQL execution in parallel. The results indicate that delegating counting, exact filtering, and aggregation to a structured relational pipeline, supported by offline derived columns and read-only execution policies, ensures verifiable answers that textual retrieval alone cannot guarantee.

The synchronization of both reasoning paradigms establishes an auditable execution trace. The evaluation demonstrates that the architecture improves quantitative correctness while maintaining baseline performance on semantic questions. This design increases latency through routing, validation, and decomposition, but the trade-off is justified by the need for verifiable evidence and deterministic computation in legal-administrative contexts.

Future work will expand human evaluation of the semantic component, refine the assessment of *Sem@F1*, and provide stratified analysis across structured query subtypes. We also plan to compare the pipeline with additional hybrid QA baselines and investigate latency-reduction strategies such as early exit, caching, and optimized routing.

8. Limitations

This work is limited by its evaluation in a single institutional domain, the use of *Sem@F1* as an LLM-based proxy, partial reliance on non-public data and proprietary models, and the absence of broad comparisons with recent public Text2SQL systems or general-purpose benchmarks. These limitations bound the interpretation of the results without invalidating the proposed architecture.

Use of Generative AI. Generative AI was used only for language revision; the authors remain responsible for the manuscript.

Acknowledgments. We acknowledge the support provided by the Center of Excellence in Artificial Intelligence (CEIA), affiliated with the Federal University of Goiás (UFG), and by the Tribunal de Contas do Estado de Goiás (TCE-GO). This work was supported by the National Institute of Science and Technology (INCT) in Responsible Artificial Intelligence for Computational Linguistics and Information Treatment and Dissemination (TILD-IAR), grant number 408490/2024-1.

References

Chalkidis, I., Fergadiotis, M., Androutsopoulos, I., and Aletras, N. (2020). LEGAL-BERT: The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*.

Chalkidis, I., Jana, A., Hartung, D., Bommarito, M., Androutsopoulos, I., Katz, D. M., and Aletras, N. (2022). Lexglue: A benchmark dataset for legal language understanding in english. In *Proceedings of ACL*.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. (2020). Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of ACL*.

Izacard, G. and Grave, E. (2021). Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of EACL*.

Katz, D. M., Hartung, D., Gerlach, L., Jana, A., and II, M. J. B. (2023). Natural language processing in the legal domain. *arXiv preprint arXiv:2302.12039*.

Khattab, O. and Zaharia, M. (2020). ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of SIGIR*.

Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., and Sabharwal, A. (2023). Decomposed prompting: A modular approach for solving complex tasks. In *International Conference on Learning Representations (ICLR)*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*.

Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., Wang, B., Qin, B., Geng, R., Huo, N., et al. (2023). Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*.

Pourreza, M. and Rafiei, D. (2023). Din-sql: Decomposed in-context learning of text-to-sql with self-correction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of EMNLP-IJCNLP*.

Scholak, T., Schucher, N., and Bahdanau, D. (2021). PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of EMNLP*.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Zhang, K., Lin, X., Wang, Y., Zhang, X., Sun, F., Cen, J., Tan, H., Jiang, X., and Shen, H. (2023). Redsql: A retrieval-augmented framework for text-to-sql generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.