Comparative Evaluation of Large Language Models in the Text-to-SQL Task in a Real Domain-Specific Database

Cesar Yuri e Silva Santos[1], **Samara Monteiro Xavier**[1], **Anderson Pinheiro Cavalcanti**[1,2], **Péricles Miranda**[1], **Rafael Ferreira Mello**[1,2]

[1]Universidade Federal Rural de Pernambuco (UFRPE)

[2]CESAR School

{cesar.yuri,samara.xavier,anderson.pinheiro}@ufrpe.br

{pericles.miranda,rafael.mello}@ufrpe.br

Abstract. *Text-to-SQL aims to democratize access to data by translating natural language questions into SQL queries. However, much of the evaluation relies on standardized academic benchmarks and automated metrics that can mask semantic errors. This work presents a comparative evaluation of five proprietary LLMs (Gemini 2.5 Flash, Gemini 2.5 Pro, GPT-4o, GPT-5 mini, and GPT-5.2) on a real, domain-specific PostgreSQL database, using 22 questions in Brazilian Portuguese of varying complexity. We combined execution accuracy with a semantic correctness analysis of the queries and an error classification. The results show that no model simultaneously dominates both metrics, that all perform well on simple queries but degrade with increasing complexity.*

Resumo. *Text-to-SQL busca democratizar o acesso a dados ao traduzir perguntas em linguagem natural para consultas SQL. Contudo, grande parte das avaliações apoia-se em benchmarks acadêmicos padronizados e em métricas automáticas que podem ocultar erros semânticos. Este trabalho apresenta uma avaliação comparativa de cinco LLMs proprietários (Gemini 2.5 Flash, Gemini 2.5 Pro, GPT-4o, GPT-5 mini e GPT-5.2) sobre um banco de dados PostgreSQL real e de domínio específico, utilizando 22 perguntas em português brasileiro de complexidade variada. Combinamos a acurácia de execução com uma análise de correção semântica das consultas e uma classificação de erros. Os resultados mostram que nenhum modelo domina simultaneamente as duas métricas, que todos têm bom desempenho em consultas simples, mas degradam com o aumento da complexidade.*

1. Introdução

O volume de dados armazenados em bancos de dados relacionais cresce de forma contínua e acelerada, tornando a capacidade de consultá-los de maneira eficiente um fator crítico para organizações dos mais variados setores. Todavia, a interação com esses sistemas requer o domínio da *Structured Query Language* (SQL), uma linguagem formal cujo aprendizado representa uma barreira significativa para usuários não especializados [Shi et al. 2025]. A conversão automática de consultas em linguagem natural para SQL, também conhecida como Text-to-SQL ou NL2SQL, emerge como uma solução promissora para democratizar o acesso a dados, permitindo que qualquer usuário extraia informações de bancos de dados sem necessidade de conhecimento técnico em SQL

[Liu et al. 2025a]. A relevância prática dessa tarefa é crescente, um vez que empresas na área de negócios, plataformas logísticas e outros ambientes orientados a dados têm buscado explorar interfaces de linguagem natural como alternativa às consultas SQL manuais [Jiang et al. 2025].

A pesquisa em Text-to-SQL tem uma trajetória de décadas, iniciando-se com abordagens baseadas em regras (*rule-based*), que apresentavam sérias limitações de generalização e escalabilidade pela necessidade de *templates* manuais e fragilidade diante de variações na linguagem natural [Zhan et al. 2025]. Com o aprendizado profundo, modelos sequência-a-sequência (Seq2Seq) passaram a mapear diretamente perguntas para SQL, e a posterior introdução de modelos pré-treinados elevou ainda mais o desempenho da tarefa [Shi et al. 2025]. A ascensão dos LLMs consolidou a literatura em torno de dois paradigmas, o *prompt engineering* e o *fine-tuning* [Shi et al. 2025]. Nesse contexto, abordagens *zero-shot* e *few-shot* passaram a demonstrar desempenho competitivo sem a necessidade de atualização dos parâmetros do modelo [Dong et al. 2023], enquanto a qualidade da estratégia de *prompting* mostrou-se determinante para os resultados obtidos [Gao et al. 2023].

Apesar dos avanços significativos, o campo ainda enfrenta desafios abertos relevantes. A heterogeneidade de esquemas em cenários *cross-domain*, a ambiguidade semântica inerente à linguagem natural, a dificuldade de generalização para consultas de alta complexidade e a limitada disponibilidade de dados rotulados em domínios específicos seguem como obstáculos centrais para a evolução dos sistemas Text-to-SQL [Zhan et al. 2025]. Benchmarks como o Spider [Yu et al. 2018], que disponibilizou 10.181 questões anotadas sobre 200 bancos de dados de 138 domínios distintos, foram fundamentais ao expor a dificuldade de generalização dos modelos, com os melhores sistemas à época atingindo apenas 9,7% de acurácia em *exact-match* no conjunto de teste, evidenciando a complexidade intrínseca da tarefa. Além disso, trabalhos recentes têm demonstrado que as métricas tradicionais de avaliação podem mascarar erros semânticos relevantes, subavaliando as limitações reais dos sistemas [Liu et al. 2025b, Liu et al. 2025a].

Diante desse panorama, o presente trabalho tem como objetivo realizar uma avaliação comparativa sistemática de cinco modelos LLMs (Gemini 2.5 Flash, Gemini 2.5 Pro, GPT-4o, GPT-5.2 e GPT-5 mini) na tarefa de Text-to-SQL. Para isso, são adotadas métricas de avaliação amplamente aceitas na literatura, incluindo acurácia de execução (*Execution Accuracy*) e acurácia de correspondência exata (*Exact Match Accuracy*), de modo a oferecer uma análise imparcial e reproduzível do estado atual dos modelos disponíveis. As principais contribuições deste artigo são: (i) uma avaliação comparativa controlada entre diferentes modelos LLM na tarefa de Text-to-SQL; e (ii) uma discussão sobre as implicações práticas dos resultados para a escolha de modelos em cenários reais de acesso a dados via linguagem natural.

Para atingir os objetivos propostos e orientar a condução metodológica deste estudo, foram definidas as seguintes questões de pesquisa (RQs - Research Questions):

RQ1. Qual é o desempenho comparativo dos modelos Gemini 2.5 Flash, Gemini 2.5 Pro, GPT-4o, GPT-5 mini e GPT-5.2 na tarefa de Text-to-SQL, medido pelas métricas de acurácia de execução (EX) e correção semântica da consulta (EM)?

RQ2. Como o desempenho dos modelos varia em função da complexidade das consultas SQL, e quais modelos apresentam maior robustez diante de consultas de alta complexidade?

2. Trabalhos Relacionados

Trabalhos recentes oferecem perspectivas estruturantes sobre o campo de Text-to-SQL. [Liu et al. 2025a] mapeiam o ciclo de vida completo do Text-to-SQL, discutindo métricas como EX, EM (*Exact Match*) e VES (*Valid Efficiency Score*) e identificando desafios em aberto como soluções custo-efetivas. [Shi et al. 2025] categorizam os métodos baseados em LLMs em *prompt engineering*, que demanda menos dados, mas pode produzir resultados subótimos, e *fine-tuning*, que melhora o desempenho, mas exige maiores volumes de dados rotulados. [Fan et al. 2025] reforçam que as limitações de generalização do NL2SQL decorrem mais de deficiências nos dados do que de insuficiências arquiteturais. No plano experimental, [Kokolishvili 2026] avalia GPT-4o, DeepSeek-V3, Gemini 2.0 Flash e Claude 3.7 Sonnet em dois bancos de dados com consultas de dificuldade variada e diferentes configurações de *prompt*, reportando EX de 100% na base University e queda para até 85% na base Sportsclub.

A busca por alternativas mais acessíveis e computacionalmente eficientes impulsionou pesquisas sobre Small Language Models (SLMs). [Silva et al. 2025] demonstram que estratégias como *schema linking* e ajuste eficiente de parâmetros são particularmente relevantes em modelos compactos, uma vez que o impacto dessas técnicas aumenta à medida que o número de parâmetros do modelo diminui. Em direção semelhante, [Pei et al. 2025] propõem Feather-SQL, framework baseado em colaboração entre modelos gerais e especializados em SQL, enquanto [Dai et al. 2025] introduzem PARSQL, abordagem que utiliza raciocínio explícito e análise sintática para reduzir erros semânticos em SLMs. Em cenários de domínio específico, [Jiang et al. 2025] demonstram que estratégias de prompting e ajuste fino podem melhorar significativamente o desempenho de modelos compactos mesmo em ambientes com poucos dados.

Arquiteturas multiagentes surgiram como resposta às limitações de abordagens de agente único diante de esquemas complexos. [Wang et al. 2025] propõem o MAC-SQL, composto por um agente decompositor com raciocínio em cadeia, um agente seletor que extrai sub-bancos relevantes e um agente refinador que corrige consultas errôneas. Avaliado com GPT-4, o sistema alcançou 59,59% de EX no benchmark BIRD, enquanto com Code Llama 7B ajustado obteve 43,94%, resultado próximo ao GPT-4 sem a arquitetura multiagente, evidenciando que estratégias colaborativas podem permitir que modelos menores atinjam desempenho competitivo em tarefas Text-to-SQL. [Zhang et al. 2024] apresentam SQLfuse, implantado em produção pelo Ant Group, integrando mineração de esquema, *schema linking*, geração SQL e um módulo crítico que seleciona a consulta de maior qualidade entre candidatos gerados, com modelagem explícita de relações um-para-muitos frequentemente negligenciadas por outros sistemas.

No contexto específico do português, estudos recentes têm avaliado sistematicamente a lacuna de desempenho entre inglês e português na tarefa de Text-to-SQL, majoritariamente a partir de versões traduzidas de *benchmarks* acadêmicos. [Pedroso et al. 2025] avaliam sete LLMs de diferentes escalas e especializações, utilizando versões em português e inglês do Spider. Os autores traduzem a partição de teste

para o português com GPT-4o mini e realizam revisão manual das 2.147 questões. Os resultados indicam que modelos de maior capacidade tendem a apresentar menor disparidade de desempenho entre português e inglês, enquanto modelos especializados em código se destacam pelo desempenho e pela maior consistência na geração exclusiva de consultas SQL. [de Carvalho et al. 2025] ampliam essa análise para dezessete modelos, incluindo modelos proprietários, comparando representações de esquema textual e DDL e estratificando os resultados pelos níveis de dificuldade EASY, MEDIUM, HARD e EXTRA do Spider. [Nobre et al. 2026], por sua vez, constroem o BIRD_PT, versão em português brasileiro do BIRD obtida pela tradução das perguntas e do conhecimento externo, e demonstram que a queda de acurácia de execução ao mudar de inglês para português é sistemática entre modelos, mas de magnitude variável por família. Além disso, identificam ocorrência de *language mixing*, com a maioria dos modelos utilizando inglês em seus tokens de raciocínio mesmo diante de entradas em português.

Esses três trabalhos avaliam o desempenho em português utilizando versões traduzidas de benchmarks originalmente construídos em inglês, como Spider e BIRD. O presente trabalho se distingue dessa linha ao utilizar um banco de dados real de domínio específico e perguntas concebidas originalmente em português brasileiro, evitando, no conjunto de perguntas, possíveis artefatos decorrentes de tradução e aproximando a avaliação de um cenário operacional de uso.

Em conjunto, os trabalhos relacionados revelam que os estudos comparativos existentes avaliam subconjuntos restritos de modelos [Kokolishvili 2026], exploram estratégias de *prompting* [Gao et al. 2023, Dong et al. 2023], abordam problemas específicos como esquemas longos [Hoang et al. 2025] ou erros semânticos [Liu et al. 2025b], ou investigam a transferência inglês-português por meio de *benchmarks* traduzidos [Pedroso et al. 2025, de Carvalho et al. 2025, Nobre et al. 2026]. O presente trabalho busca reduzir essa lacuna ao realizar uma comparação abrangente entre cinco modelos representativos do estado da arte sob condições experimentais idênticas, deslocando a avaliação para um banco de dados real de domínio específico, utilizando perguntas concebidas originalmente em português brasileiro e oferecendo subsídios práticos e reproduzíveis para a tomada de decisão em cenários reais de acesso a dados via linguagem natural.

3. Metodologia

3.1. Descrição do Sistema

O sistema avaliado neste trabalho consiste em um banco de dados relacional de domínio específico. O banco de dados é implementado em PostgreSQL (v17.4) e organiza informações relativas a um sistema de gerenciamento de matrículas. Cinco tabelas principais compõem o esquema, sendo elas relacionadas ao cadastro de crianças, inscrições, registros da fila de espera, rede de ensino e ofertas de vagas por escola. As tabelas são interligadas por chaves estrangeiras, exigindo o uso de operações de JOIN para a maioria das consultas analíticas. A escolha de um domínio real e específico, em contraste com os benchmarks acadêmicos amplamente utilizados na literatura, como o Spider [Yu et al. 2018], constitui uma contribuição metodológica deste trabalho, uma vez que expõe os modelos a desafios de vocabulário técnico e semântica dependente de contexto operacional, aproximando a avaliação de cenários reais de acesso a dados via linguagem natural.

3.2. Arquitetura do Sistema

O sistema consiste em três componentes principais: o grafo de execução do agente, construído com LangGraph (v0.6.5); a ferramenta de acesso ao banco de dados; e o pipeline de avaliação, responsável por comparar os resultados gerados com o gabarito humano. O agente é implementado como um grafo de estados composto por dois nós, um de inferência, que invoca o modelo de linguagem com o histórico acumulado de mensagens, e um de execução, que processa as chamadas de ferramenta presentes na resposta. O grafo alterna entre eles até que nenhuma chamada de ferramenta seja emitida, momento em que a execução é encerrada.

O processo de *Text-to-SQL* segue o seguinte pipeline. O prompt de sistema carregava o esquema do banco, obtido automaticamente via `db.get_table_info()` do LangChain (v0.3.27) e injetado no prompt em paralelo das regras de comportamento, proibições e limite de iterações permitidas do agente. Um segundo prompt era enviado em conjunto, contendo o objetivo do sistema e as perguntas a serem respondidas. Cada chamada incluía o SQL gerado e o identificador da pergunta correspondente (`id_pergunta`). A ferramenta validava se a consulta era exclusivamente um `SELECT`, rejeitando qualquer instrução DDL ou DML, e executava a query via SQLAlchemy (v2.0.43).

3.3. Conjunto de Dados e Modelos

Foram selecionados cinco LLMs proprietários representativos do estado da arte em 2025, pertencentes a duas famílias distintas, conforme descrito na Tabela 1. A seleção abrange modelos de diferentes escalas dentro de cada família, permitindo a análise do efeito de escala sobre o desempenho na tarefa de Text-to-SQL (RQ1 e RQ2).

Tabela 1. Modelos de linguagem avaliados no experimento

Modelo	Família	Categoria	Identificador de API
Gemini 2.5 Flash	Gemini 2.5	Proprietário	`gemini-2.5-flash`
Gemini 2.5 Pro	Gemini 2.5	Proprietário	`gemini-2.5-pro`
GPT-4o	GPT-4	Proprietário	`gpt-4o`
GPT-5 mini	GPT-5	Proprietário	`gpt-5-mini`
GPT-5.2	GPT-5	Proprietário	`gpt-5.2`

Adicionalmente, foram elaboradas 22 perguntas em linguagem natural (Q1-Q22), formuladas originalmente em português do Brasil, refletindo o idioma nativo dos usuários finais do sistema avaliado. Diferentemente de estudos que avaliam Text-to-SQL em português a partir versões traduzidas de *benchmarks* originalmente construídos em inglês, como Spider ou BIRD [Pedroso et al. 2025, de Carvalho et al. 2025, Nobre et al. 2026], as perguntas deste trabalho não são traduções, mas questões concebidas diretamente em português para o domínio e o esquema do banco de dados avaliado, cobrindo um espectro amplo de complexidade de consultas SQL, desde operações simples de contagem (Q1, Q3) até consultas que envolvem múltiplos JOINs, subconsultas aninhadas, cálculos proporcionais e tratamento de valores ausentes (Q8, Q11, Q22). Todas as perguntas dizem respeito ao domínio do banco de dados descrito na Seção 3.1 e foram acompanhadas de um SQL de referência e de um resultado esperado, construídos manualmente e verificados por execução direta no banco.

Tabela 2. Conjunto de operações por perguntas do experimento

ID	Operação SQL Principal	Complexidade da Pergunta
Q1	COUNT com filtro simples	Simples
Q2	EXISTS / CASE WHEN	Média
Q3	COUNT DISTINCT	Simples
Q4	SUM com filtro	Simples
Q5	SUM com filtro	Simples
Q6	Cálculo aritmético / ROUND	Simples
Q7	COUNT + COUNT DISTINCT	Simples
Q8	Subconsulta + GROUP BY	Média
Q9	JOIN + GROUP BY + ORDER + LIMIT	Média
Q10	JOIN + SUM + ORDER + LIMIT	Média
Q11	LEFT JOIN duplo + proporção + NULLIF	Alta
Q12	GROUP BY simples	Simples
Q13	AVG	Simples
Q14	MIN + MAX	Simples
Q15	GROUP BY + ORDER + LIMIT	Simples
Q16	JOIN + GROUP BY + ORDER	Média
Q17	GROUP BY	Simples
Q18	AVG	Simples
Q19	MAX + ORDER + LIMIT	Simples
Q20	information_schema / meta-consulta	Média
Q21	SUM com condição de zero-ocupação	Simples
Q22	Subquery de exclusão + NOT IN / DISTINCT	Alta

O esquema do banco era restrito a um subconjunto de cinco tabelas relevantes ao domínio, definidas previamente por configuração. O esquema era obtido automaticamente por meio da biblioteca de integração com o banco, que consultava o catálogo do banco de dados e gerava, para cada tabela permitida, o script CREATE TABLE correspondente, com os tipos de coluna, chaves primárias e estrangeiras, além de linhas de amostra conforme configuração padrão da biblioteca. Esse texto era inserido diretamente no prompt de sistema, sem pré-processamento, edição manual ou descrição textual complementar. O prompt foi padronizado entre todos os modelos avaliados, sem ajustes específicos de formulação para cada um. A temperatura das chamadas de API foi fixada em 0.2 para todos os modelos, buscando respostas mais determinísticas e reduzindo a variabilidade entre execuções. Os demais parâmetros de geração permaneceram nos valores padrão de cada provedor.

3.4. Métricas de Avaliação

A avaliação dos modelos foi conduzida por meio de três métricas complementares que em conjunto cobrem tanto a corretude dos resultados produzidos quanto a qualidade semântica das consultas geradas e a natureza dos erros cometidos.

A métrica de *Execution Accuracy* (EX) avalia se a consulta gerada pelo modelo, quando executada no banco de dados, produz o mesmo conjunto de resultados que o SQL de referência. Trata-se de uma métrica amplamente empregada na literatura de Text-to-SQL [Liu et al. 2025a] por ser independente da formulação sintática da query. Contudo, como observado por [Liu et al. 2025b], métricas baseadas exclusivamente no resultado de execução podem não ser suficientes para revelar discrepâncias semânticas entre a intenção expressa na linguagem natural e a consulta SQL gerada. A segunda métrica utilizada foi a

correção semântica da consulta SQL gerada (chamada EM), que avalia se a consulta produzida é logicamente equivalente ao SQL de referência, ainda que empregue formulações sintáticas distintas, diferentes estratégias de junção ou agregações alternativas que expressem a mesma intenção analítica. Diferentemente de métricas estritamente estruturais, essa avaliação considera equivalentes consultas que divergem na forma, mas convergem no significado.

Diferentemente da métrica Exact Match tradicional empregada em benchmarks como Spider [Yu et al. 2018], baseada em uma comparação automática da estrutura da consulta gerada com a consulta de referência, neste estudo a avaliação foi conduzida manualmente por um único avaliador humano, que comparou cada consulta SQL gerada com a respectiva consulta de referência, considerando como correta a consulta que preservasse a mesma intenção e produzisse uma formulação logicamente equivalente.

A classificação de erros semânticos foi conduzida segundo a taxonomia NL2SQL-BUGs [Liu et al. 2025b] que organiza os erros em 9 categorias principais (Erros de Atributo, Tabela, Valor, Operador, Condição, Função, Cláusula, Subconsulta e Outros), e 31 subcategorias. Para cada consulta classificada como incorreta, foram registrados a categoria principal e a subcategoria do erro, conforme a taxonomia citada. Adicionalmente, este estudo registrou a causa raiz e a gravidade (Alta, Média ou Baixa) de cada erro. Essa camada qualitativa permite identificar padrões sistemáticos de falha por modelo e por tipo de consulta.

3.5. Procedimento Experimental

As 22 perguntas foram submetidas aos modelos descritos na Tabela 1 usando a arquitetura definida na Subseção 3.2. Cada consulta SQL obtida pelo modelo foi executada diretamente contra o banco de dados e seu resultado confrontado com um gabarito composto pelo *gold SQL* de referência e pelo resultado esperado. A comparação utilizava as métricas de *Execution Accuracy* e *Exact Match*, verificando se o resultado retornado pela *query* gerada correspondia ao resultado esperado e à consulta SQL esperada.

Cada SQL gerado foi avaliado quanto à equivalência lógico-semântica em relação ao SQL de referência. Foram registrados (i) se a consulta representava corretamente a intenção analítica expressa pela pergunta; (ii) se diferenças sintáticas eram irrelevantes para o resultado pretendido; e (iii) em caso de discrepância, a categoria, a causa raiz e a gravidade do erro segundo a taxonomia NL2SQL-BUGs. Essa análise permitiu distinguir consultas conceitualmente corretas que produziam resultados incorretos por detalhes específicos dos dados de casos em que o resultado estava correto apesar de falhas semânticas.

4. Resultados

Esta seção apresenta os resultados da avaliação comparativa dos cinco modelos no conjunto de 22 perguntas, organizados em torno das métricas de *Execution Accuracy* e de *Exact Match*, da análise por nível de complexidade das consultas e da classificação dos erros semânticos segundo a taxonomia NL2SQL-BUGs [Liu et al. 2025b]. A Tabela 3 consolida os resultados globais de cada modelo nas duas métricas avaliadas. Enquanto o Gemini 2.5 Flash obteve o maior EX (72,7%) mas o menor EM (63,6%), os modelos Gemini 2.5 Pro e GPT-5.2 apresentaram o padrão inverso, ou seja, EM de 77,3% com

EX de apenas 45,5% e 54,5%, respectivamente. GPT-4o e GPT-5 mini obtiveram EX de 54,5% e 45,5%, ambos com EM de 72,7%.

Tabela 3. Resultados gerais de Execution Accuracy (EX) e Exact Match (EM) por modelo

Modelo	EX (n)	EX (%)	EM (n)	EM (%)
Gemini 2.5 Flash	16/22	**72,7%**	14/22	63,6%
Gemini 2.5 Pro	10/22	45,5%	17/22	**77,3%**
GPT-4o	12/22	54,5%	16/22	72,7%
GPT-5 mini	10/22	45,5%	16/22	72,7%
GPT-5.2	12/22	54,5%	17/22	**77,3%**

Esse resultado evidencia que consultas avaliadas como semanticamente adequadas podem produzir resultados incorretos quando incorporam condições incompatíveis com as características específicas dos dados armazenados. Em particular, observou-se um padrão recorrente relacionado à aplicação indevida de filtros sobre a Tabela de fila de oferta. Embora a estrutura geral das consultas estivesse alinhada à intenção da pergunta, a presença desse filtro resultava na eliminação de registros válidos, reduzindo significativamente a métrica EX sem necessariamente comprometer a avaliação da correção lógica da consulta.

4.1. Desempenho por Complexidade

As 22 questões foram classificadas em três níveis de complexidade conforme os operadores SQL envolvidos, conforme detalhado na Tabela 2: 14 questões simples (operações de agregação direta, sem JOINs ou subconsultas), 6 questões de complexidade média (JOINs, subconsultas ou cálculos aritméticos) e 2 questões de alta complexidade (LEFT JOINs múltiplos com subqueries independentes, tratamento de NULLs e cálculos proporcionais). A Tabela 4 apresenta o desempenho desagregado por nível. Os dados revelam que os modelos apresentaram elevado desempenho estrutural nas 14 perguntas classificadas como simples. O Gemini 2.5 Pro obteve 100% de EM nesse subconjunto, enquanto Gemini 2.5 Flash, GPT-4o, GPT-5 mini e GPT-5.2 alcançaram 92,9%. Em termos de Execution Accuracy (EX), observou-se maior variabilidade, em que o Gemini 2.5 Flash atingiu 92,9%, ao passo que Gemini 2.5 Pro, GPT-4o, GPT-5 mini e GPT-5.2 obtiveram 50,0%, o que decorre de premissas e filtros plausíveis incorporados pelos modelos que, embora semanticamente coerentes com a intenção da consulta (EM), divergiram das características dos dados armazenados na base real, reduzindo o EX.

Tabela 4. Execution Accuracy (EX) e Exact Match (EM) por nível de complexidade

Modelo	Simples (n=14)		Média (n=6)		Alta (n=2)	
	EX	EM	EX	EM	EX	EM
Gemini 2.5 Flash	**92,9%**	**92,9%**	**50,0%**	16,7%	0,0%	0,0%
Gemini 2.5 Pro	50,0%	**100,0%**	50,0%	50,0%	0,0%	0,0%
GPT-4o	50,0%	92,9%	**66,7%**	50,0%	50,0%	0,0%
GPT-5 mini	50,0%	92,9%	50,0%	50,0%	0,0%	0,0%
GPT-5.2	50,0%	92,9%	50,0%	50,0%	**100,0%**	**50,0%**

O Gemini 2.5 Flash apresentou o maior desempenho em consultas simples, com EX e EM de 92,9% nas questões simples, mas EM e EX nulos nas questões de alta complexidade, evidenciando uma limitação na capacidade de compor consultas que envolvem

múltiplos LEFT JOINs, subconsultas independentes e tratamento explícito de valores nulos. O GPT-5.2 destacou-se nas questões de alta complexidade com EX de 100%, sendo o único modelo a resolver corretamente ambas, Q11 e Q22, embora com EM de apenas 50,0% nesse estrato, indicando que o acerto ocorreu por formulações alternativas ao SQL de referência.

4.2. Classificação dos Erros Semânticos

Para cada SQL classificado como incorreto em EM, foi realizada a categorização segundo a taxonomia NL2SQL-BUGs [Liu et al. 2025b]. Ao todo, foram registrados 30 erros de SQL distribuídos entre os cinco modelos. A Tabela 5 apresenta a frequência por categoria.

Tabela 5. Frequência de erros de SQL por categoria (NL2SQL-BUGs)

Categoria	Ocorrências	Subcategoria mais frequente
Erros de Condição	14	Condição Explícita Redundante (9)
Erros de Subconsulta	6	Subconsulta Incorreta (4)
Erros de Tabela	6	Tipo de JOIN Incorreto (3)
Erros de Função	2	Função de Agregação Incorreta (2)
Erros de Atributo	1	Atributo Incorreto (1)
Erros de Operador	1	Operador de Comparação Incorreto (1)
Total	**30**	—

Erros de Condição respondem por 46,7% do total (14/30), sendo a subcategoria Condição Explícita Redundante, responsável por 9 das 14 ocorrências. Esse erro está associado em grande parte à aplicação de um filtro booleano sobre uma coluna integralmente nula em uma das tabelas, que zera o resultado da consulta. Erros de Subconsulta e de Tabela empatam em segundo lugar com 6 ocorrências cada. Erros de Função, Atributo e Operador são marginais, com 1 a 2 ocorrências por categoria. A Tabela 6 detalha os erros por modelo.

Tabela 6. Erros de SQL por modelo segundo a taxonomia NL2SQL-BUGs

Modelo	Total	Categorias (ocorrências)
Gemini 2.5 Flash	8	Condição (4), Tabela (3), Subconsulta (1)
Gemini 2.5 Pro	5	Subconsulta (3), Tabela (1), Condição (1)
GPT-4o	6	Condição (3), Tabela (1), Atributo (1), Operador (1)
GPT-5 mini	6	Condição (3), Função (1), Subconsulta (1), Tabela (1)
GPT-5.2	5	Condição (3), Função (1), Subconsulta (1)

O Gemini 2.5 Flash concentra o maior número absoluto de erros de SQL (8), com predominância de Erros de Condição e de Tabela . Os modelos GPT-5 mini e GPT-4o registraram 6 erros cada, e Gemini 2.5 Pro e GPT-5.2 apresentaram 5 erros cada. Nota-se que o Gemini 2.5 Pro, apesar de ter o menor número de erros de SQL dentre os modelos com mais falhas, acumula o maior impacto em EX, confirmando que a qualidade do SQL não se traduz diretamente em acerto de resultado quando erros de condição afetam o conjunto de dados consultado.

Em termos de gravidade, os erros de alta gravidade predominam; 23 dos 30 erros foram classificados nessa categoria, refletindo a predominância de condições que zeram completamente os resultados ou de falhas estruturais que produzem resultados completamente divergentes do esperado (como JOINs incorretos em Q11 e Q22).

5. Conclusão e Discussão

Este trabalho apresentou uma avaliação comparativa de cinco modelos de linguagem de grande porte (Gemini 2.5 Flash, Gemini 2.5 Pro, GPT-4o, GPT-5 mini e GPT-5.2) na tarefa de Text-to-SQL sobre um banco de dados real de domínio específico, combinando acurácia de execução, correção semântica das consultas e classificação de erros segundo a taxonomia NL2SQL-BUGs. Os resultados obtidos permitem responder diretamente às duas questões de pesquisa propostas. Em relação à RQ1, observou-se que não existe um modelo dominante simultaneamente em acurácia de execução (EX) e correção semântica (EM). O Gemini 2.5 Flash apresentou o melhor desempenho em EX (72,7%), enquanto Gemini 2.5 Pro e GPT-5.2 alcançaram os maiores valores de EM (77,3%). Essa divergência evidencia que a capacidade de produzir consultas semanticamente adequadas não implica necessariamente maior acurácia nos resultados finais obtidos a partir dos dados. A diferença entre EX e EM mostrou-se particularmente relevante neste estudo. Os resultados observados reforçam as críticas recentes da literatura [Liu et al. 2025b], segundo as quais métricas isoladas podem ocultar limitações importantes dos modelos. Em diversos casos, consultas avaliadas como semanticamente corretas produziram resultados incorretos devido a hipóteses inadequadas sobre características específicas do banco de dados. Consequentemente, a utilização conjunta de EX e EM permitiu uma avaliação mais abrangente do comportamento dos modelos.

A análise da RQ2 revelou que a complexidade da consulta exerce influência significativa sobre o desempenho dos modelos. Nas questões classificadas como simples, os modelos obtiveram EM entre 92,9% e 100%, indicando capacidade consistente de interpretar corretamente operações básicas de agregação, filtragem e contagem. Entretanto, à medida que a complexidade aumentou, observou-se redução progressiva tanto em EX quanto em EM. Consultas envolvendo múltiplos JOINs, subconsultas correlacionadas, tratamento de valores nulos e cálculos proporcionais apresentaram dificuldades substancialmente maiores. Esse comportamento está alinhado com observações reportadas em benchmarks como Spider [Yu et al. 2018], nos quais a generalização para consultas complexas permanece um desafio aberto.

A classificação dos erros segundo a taxonomia NL2SQL-BUGs permitiu identificar padrões consistentes entre os modelos. Erros de Condição representaram quase metade das falhas registradas, seguidos por erros de Subconsulta e de Tabela. A predominância dessas categorias indica que os principais desafios atuais não estão relacionados à sintaxe SQL básica, mas à correta modelagem das restrições lógicas e relacionamentos implícitos no esquema. Esse resultado converge com estudos recentes que apontam que grande parte dos erros em sistemas Text-to-SQL decorre de interpretações semânticas inadequadas e não de limitações estruturais da geração SQL [Liu et al. 2025b, Fan et al. 2025].

Finalmente, os resultados sugerem que modelos mais recentes não necessariamente apresentam vantagem uniforme em todos os cenários. O GPT-5.2 destacou-se em consultas de alta complexidade, enquanto o Gemini 2.5 Flash demonstrou excelente desempenho em consultas simples e médias. Isso indica que a escolha do modelo mais adequado depende das características do domínio, da complexidade das consultas esperadas e do custo operacional associado à utilização de cada sistema. Dessa forma, a seleção de um modelo para aplicações Text-to-SQL deve considerar não apenas métricas glo-

bais de desempenho, mas também o perfil das consultas predominantes no ambiente de implantação.

6. Limitações

Este estudo apresenta limitações que delimitam o alcance de suas conclusões. A avaliação baseia-se em um único banco de dados de domínio específico e em um conjunto de 22 perguntas em português brasileiro, o que aproxima a avaliação de cenários reais de uso, mas restringe a generalização estatística dos resultados. Em particular, os estratos de maior complexidade contêm poucas questões (duas no nível alto), de modo que as diferenças observadas nesse nível devem ser interpretadas como indicativas e não conclusivas. As comparações entre modelos são reportadas em proporções absolutas, sem testes formais de significância estatística nem intervalos de confiança, razão pela qual pequenas diferenças de desempenho não devem ser tomadas como evidência de superioridade definitiva. A métrica de correção semântica apoia-se em um critério de equivalência lógica que não foi submetido à verificação de concordância entre múltiplos avaliadores independentes, com um único avaliador realizando a avaliação. Embora tenham sido adotados critérios uniformes de equivalência lógica ao longo da avaliação, a ausência de múltiplos avaliadores independentes introduz a possibilidade de subjetividade na classificação e limita a análise da confiabilidade das avaliações manuais. Por fim, foram avaliados exclusivamente modelos proprietários, acessados por APIs cujas versões podem evoluir ao longo do tempo, o que impõe cautela quanto à reprodutibilidade exata e não contempla a comparação com modelos *open-source* ou com abordagens de *fine-tuning*.

Referências

Dai, Y., Yang, H., Hao, M., and Chao, P. (2025). Parsql: Enhancing text-to-sql through sql parsing and reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 661–681.

de Carvalho, L. F. C., Júnior, P. S. d. S., and de Oliveira, H. T. A. (2025). Benchmarking large language models for text-to-sql in brazilian portuguese and english. In *Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana (STIL)*, pages 101–112. SBC.

Dong, X., Zhang, C., Ge, Y., Mao, Y., Gao, Y., Lin, J., Lou, D., et al. (2023). C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*.

Fan, Y., Weng, Q., Chen, Y., and Wang, X. S. (2025). Rethinking data in nl2sql: a survey of what we have and what we expect. *Vicinagearth*, 2(1):15.

Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., and Zhou, J. (2023). Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

Hoang, C. D. V., Vu, D. Q., Tangari, G., Vu, T., Dharmasiri, D., Li, Y.-F., Duong, L., et al. (2025). Sqlong: Enhanced nl2sql for longer contexts with llms. In *Proceedings of the 4th Table Representation Learning Workshop*, pages 47–55.

Jiang, H., Zhu, M., Zhang, Q., Zhong, G., and Li, G. (2025). Research on nl2sql in the logistics field based on large language models under resource constraints. In *Proceedings of the 2025 5th International Conference on Big Data, Artificial Intelligence and Risk Management*, pages 317–325.

Kokolishvili, A. (2026). A study of the performance of large language models in text-to-sql tasks. In *2026 5th International Informatics and Software Engineering Conference (IISEC)*, pages 315–319.

Liu, X., Shen, S., Li, B., Ma, P., Jiang, R., Zhang, Y., Fan, J., Li, G., Tang, N., and Luo, Y. (2025a). A survey of text-to-sql in the era of llms: Where are we, and where are we going? *IEEE Transactions on Knowledge and Data Engineering*, 37(10):5735–5754.

Liu, X., Shen, S., Li, B., Tang, N., and Luo, Y. (2025b). Nl2sql-bugs: A benchmark for detecting semantic errors in nl2sql translation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5662–5673.

Nobre, L., Sousa, S., Teles, S., and Soares, A. (2026). Language effects in text-to-sql across english and portuguese. In *Proceedings of the 17th International Conference on Computational Processing of Portuguese (PROPOR 2026)-Vol. 1*, pages 270–280.

Pedroso, B. C., Pereira, M. R., and Pereira, D. A. (2025). Performance evaluation of llms in the text-to-sql task in portuguese. In *Simpósio Brasileiro de Sistemas de Informação (SBSI)*, pages 260–269. SBC.

Pei, W., Xu, H., Zhao, H. H., Hou, S., Han, C., Zhang, Z., Pingyi, L., and He, B. (2025). Feather-sql: A lightweight nl2sql framework with dual-model collaboration paradigm for small language models. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 2069–2086.

Shi, L., Tang, Z., Zhang, N., Zhang, X., and Yang, Z. (2025). A survey on employing large language models for text-to-sql tasks. *ACM Computing Surveys*, 58(2):1–37.

Silva, P. H., Silva, L. O., and Silva, F. A. (2025). Towards small language models for text-to-sql. In *Brazilian Conference on Intelligent Systems*, pages 516–531. Springer.

Wang, B., Ren, C., Yang, J., Liang, X., Bai, J., Chai, L., Yan, Z., Zhang, Q.-W., Yin, D., Sun, X., et al. (2025). Mac-sql: A multi-agent collaborative framework for text-to-sql. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 540–557.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., et al. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3911–3921.

Zhan, G., Shi, G., Liu, X., and Wang, N. (2025). Natural language to sql using large language models. In *2025 21st International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 495–502.

Zhang, T., Chen, C., Liao, C., Wang, J., Zhao, X., Yu, H., Wang, J., Li, J., and Shi, W. (2024). Sqlfuse: Enhancing text-to-sql performance through comprehensive llm synergy. *arXiv preprint arXiv:2407.14568*.