REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

BENCHMARK DE MODELOS DE LINGUAGEM DE CÓDIGO ABERTO PARA TEXT-TO-SQL EM DADOS ONCOLÓGICOS BRASILEIROS

BENCHMARK OF OPEN-SOURCE LANGUAGE MODELS FOR TEXT-TO-SQL ON BRAZILIAN ONCOLOGY DATA

BENCHMARK DE MODELOS DE LENGUAJE DE CÓDIGO ABIERTO PARA TEXT-TO-SQL EN DATOS ONCOLÓGICOS BRASILEÑOS

Fabrício de Carvalho Mota
Graduando em Tecnologia e Análise e Desenvolvimento de Sistemas
Instituto Federal do Piauí - IFPI, Brasil
ORCID: https://orcid.org/0009-0006-7069-0272
E-mail: fabriciomota2612@gmail.com

Wanderson de Vasconcelos Rodrigues da Silva
Doutor em Ciência da Propriedade Intelectual pela UFS
Docente do Instituto Federal do Piauí - IFPI, Brasil
ORCID: https://orcid.org/0000-0002-7631-9830
E-mail: wanderson.vasconcelos@ifpi.edu.br

Jeferson do Nascimento Soares
Mestre em Ciência da Computação pela UECE
Docente do Instituto Federal do Piauí - IFPI, Brasil
ORCID: https://orcid.org/0000-0002-0586-4786
E-mail: jeferson.soares@ifpi.edu.br

Resumo

Este trabalho apresenta um benchmark experimental para avaliar modelos de linguagem de código aberto na tarefa Text-to-SQL, utilizando perguntas em português brasileiro e dados públicos de atendimentos oncológicos no Brasil. O estudo comparou os modelos Qwen2.5-Coder 7B, DeepSeek-Coder 6.7B e Llama 3 8B, executados localmente por meio do Ollama, em quatro cenários experimentais definidos pela presença ou ausência do esquema do banco de dados no prompt e pela presença ou ausência de variações linguísticas nas perguntas. A avaliação foi baseada na métrica Execution Accuracy (EX), complementada pela análise das categorias de erro e das latências de geração e execução. Os resultados indicaram que o Qwen2.5-Coder 7B apresentou o melhor desempenho geral, com EX de 80,0%, além da menor latência média de geração. O fornecimento do esquema foi o fator de maior impacto, elevando a acurácia média dos modelos de 53,9% para 93,3%. As variações linguísticas utilizadas não degradaram o desempenho médio dos modelos, sugerindo robustez apenas ao ruído superficial controlado adotado no experimento. Conclui-se que modelos de linguagem de código aberto podem ser aplicados à geração de consultas SQL em cenários controlados, mas seu desempenho depende fortemente do contexto estrutural fornecido e da qualidade da formulação das perguntas.

Palavras-chave: Text-to-SQL; Modelos de linguagem; Banco de dados; Dados oncológicos; Avaliação por execução.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Abstract

This work presents an experimental benchmark to evaluate open-source language models in the Text-to-SQL task, using questions in Brazilian Portuguese and public data on oncology care in Brazil. The study compared Qwen2.5-Coder 7B, DeepSeek-Coder 6.7B, and Llama 3, executed locally through Ollama, across four experimental scenarios defined by the presence or absence of the database schema in the prompt and by the presence or absence of linguistic variations in the questions. The evaluation was based on the Execution Accuracy (EX) metric, complemented by an analysis of error categories and generation and execution latencies. The results showed that Qwen2.5-Coder 7B achieved the best overall performance, with an EX of 80.0%, as well as the lowest average generation latency. Providing the database schema was the most influential factor, increasing the average model accuracy from 53.9% to 93.3%. The linguistic variations used did not degrade the average model performance, suggesting robustness only to the controlled superficial noise adopted in the experiment. The findings indicate that open-source language models can be applied to SQL query generation in controlled scenarios, although their performance strongly depends on the structural context provided and on the quality of the question formulation.

Keywords: Text-to-SQL; Language models; Database; Oncology data; Execution-based evaluation.

Resumen

Este trabajo presenta un benchmark experimental para evaluar modelos de lenguaje de código abierto en la tarea Text-to-SQL, utilizando preguntas en portugués brasileño y datos públicos de atención oncológica en Brasil. El estudio comparó Qwen2.5-Coder 7B, DeepSeek-Coder 6.7B y Llama 3, ejecutados localmente a través de Ollama, en cuatro escenarios experimentales definidos por la presencia o ausencia del esquema de la base de datos en el prompt y por la presencia o ausencia de variaciones lingüísticas en las preguntas. La evaluación se basó en la métrica Execution Accuracy (EX), complementada con un análisis de las categorías de errores y las latencias de generación y ejecución. Los resultados mostraron que Qwen2.5-Coder 7B logró el mejor rendimiento general, con una EX del 80,0%, así como la latencia de generación promedio más baja. Proporcionar el esquema de la base de datos fue el factor más influyente, aumentando la precisión promedio del modelo del 53,9% al 93,3%. Las variaciones lingüísticas utilizadas no han mermado el rendimiento medio de los modelos, lo que sugiere que estos son robustos únicamente frente al ruido superficial controlado que se ha utilizado en el experimento. Los hallazgos indican que los modelos de lenguaje de código abierto se pueden aplicar a la generación de consultas SQL en escenarios controlados, aunque su rendimiento depende en gran medida del contexto estructural proporcionado y de la calidad de la formulación de las preguntas.

Palabras clave: Text-to-SQL; Modelos de lenguaje; Base de datos; Datos de oncología; Evaluación basada en la ejecución.

1. Introdução

A tarefa *Text-to-SQL* consiste em converter perguntas formuladas em linguagem natural em consultas SQL executáveis sobre bancos de dados relacionais. Essa abordagem permite que usuários não especialistas acessem dados estruturados sem a necessidade de conhecer a linguagem SQL ou a organização interna do banco.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Com o avanço dos modelos de linguagem de grande porte, tornou-se possível utilizar esses modelos para gerar consultas SQL a partir de instruções textuais. Entretanto, a tarefa ainda apresenta desafios, como a identificação correta de tabelas e colunas, o uso adequado do esquema do banco e a geração de consultas semanticamente alinhadas à pergunta. Esses desafios tornam-se mais relevantes em cenários aplicados, especialmente quando se utilizam dados reais e perguntas em idiomas diferentes do inglês.

No contexto brasileiro, bases públicas de saúde, como registros relacionados a atendimentos oncológicos, reúnem informações relevantes; contudo, sua exploração para análises estruturadas e consultas específicas pode exigir conhecimento técnico em organização e bancos de dados. Nesse cenário, modelos *Text-to-SQL* podem contribuir para tornar o acesso a essas informações mais direto, desde que sua capacidade de gerar consultas corretas seja avaliada de forma controlada. Neste estudo, o termo “dados oncológicos” refere-se ao domínio temático e estrutural do banco utilizado no *benchmark*, não à validação clínica de diagnósticos, tratamentos ou decisões assistenciais.

Diante disso, este trabalho tem como objetivo avaliar modelos de linguagem de código aberto na tarefa *Text-to-SQL*, utilizando dados públicos oncológicos brasileiros. Foram analisados os modelos Qwen2.5-Coder 7B, DeepSeek-Coder 6.7B e Llama 3 8B, executados localmente via Ollama, em quatro cenários definidos pela presença ou ausência do esquema do banco no *prompt* e pela presença ou ausência de variações linguísticas superficiais controladas nas perguntas.

A principal contribuição do estudo é a construção e análise de um *benchmark* experimental em português brasileiro, voltado a dados tabulares do domínio oncológico. Além do desempenho por *Execution Accuracy* (EX), são analisados o impacto do fornecimento do esquema, a robustez a variações linguísticas controladas, as categorias de erro e a latência de geração e execução.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta o referencial teórico; a Seção 3 descreve a metodologia; a Seção 4

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

apresenta os resultados e discussões; e a Seção 5 reúne as conclusões e trabalhos futuros

2. Referencial Teórico

Este referencial teórico apresenta os conceitos e estudos que fundamentam a tarefa *Text-to-SQL* e sua avaliação. São abordados: (i) a formulação do problema como mapeamento de linguagem natural para SQL, (ii) o uso de modelos de linguagem e estratégias de *prompting* com contexto do esquema, (iii) *benchmarks* e desafios reportados em cenários aplicados e (iv) aspectos de avaliação por execução, idioma e robustez a variações de escrita. Esses tópicos sustentam as decisões metodológicas adotadas no experimento e orientam a interpretação dos resultados.

2.1 Text-to-SQL como mapeamento de linguagem natural para SQL

A tarefa *Text-to-SQL* (também denominada *Natural Language to SQL*) consiste em converter uma pergunta em linguagem natural em uma consulta na linguagem *Structured Query Language* (SQL), executável sobre um banco de dados relacional (Yu *et al.*, 2018). Bancos relacionais organizam dados em tabelas compostas por linhas e colunas, permitindo consultas estruturadas por meio da SQL. Essa tarefa pode ser formalmente compreendida como um problema de *semantic parsing*, isto é, o mapeamento de sentenças em linguagem natural para representações formais executáveis.

Conforme definido por Yu *et al.* (2018), na prática, isso implica que o sistema deve compreender corretamente a intenção da pergunta e traduzi-la para uma consulta SQL semanticamente equivalente, respeitando a estrutura da linguagem e o esquema do banco de dados.

> A análise semântica (*semantic parsing*) é uma das tarefas mais importantes no processamento de linguagem natural (PLN). Ela requer tanto a compreensão do significado de sentenças em linguagem natural quanto o mapeamento dessas sentenças para consultas executáveis com significado, como formas lógicas, consultas SQL e código *Python*. (Yu *et al.*, 2018, p. 3911, tradução nossa)

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

A complexidade da tarefa varia conforme a estrutura da consulta requerida, podendo envolver desde operações simples de seleção e filtragem até construções mais elaboradas, como junções entre múltiplas tabelas, agregações e subconsultas aninhadas (Yu *et al.*, 2018). Além da complexidade estrutural, um dos principais desafios está no alinhamento entre a linguagem natural e o esquema do banco de dados, aspecto que se torna ainda mais crítico em cenários com múltiplas tabelas e relações (Shi *et al.*, 2024). Outro aspecto relevante é a necessidade de generalização, uma vez que os modelos devem ser capazes de lidar com consultas e esquemas não vistos durante o treinamento, característica essencial em aplicações reais (Yu *et al.*, 2018).

Por fim, é importante destacar que diferentes consultas SQL podem ser semanticamente equivalentes, produzindo o mesmo resultado quando executadas. Isso limita abordagens baseadas exclusivamente na comparação textual ou estrutural entre consultas, como a métrica *Exact Match* (EM), que verifica se a consulta gerada corresponde exatamente à consulta de referência. Como discutido por Shi *et al.* (2024): “Existem múltiplas formas de expressar um mesmo problema em SQL, de modo que a métrica EM frequentemente subestima a acurácia das previsões do modelo” (Shi *et al.*, 2024, p. 8). Por esse motivo, métricas baseadas na execução das consultas são amplamente adotadas na literatura, por aproximarem a avaliação do objetivo final da tarefa.

2.2 Modelos de linguagem, prompting e contexto do esquema

Com a popularização de Modelos de Linguagem de Grande Porte (LLMs), tornou-se comum utilizar esses modelos para gerar consultas SQL a partir de perguntas em linguagem natural (Shi *et al.*, 2024). De forma geral, LLMs são modelos treinados em grandes coleções de texto e capazes de gerar sequências linguísticas a partir de instruções, explorando padrões aprendidos durante o treinamento.

Neste trabalho, *prompting* refere-se ao fornecimento de uma instrução textual que especifica a tarefa a ser realizada, podendo incluir informações adicionais para reduzir ambiguidades. Em muitos cenários da literatura, essa

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

abordagem é utilizada sem *fine-tuning*, isto é, sem ajuste adicional dos parâmetros do modelo.

Conforme sintetizam Shi *et al.* (2024, p. 3): “A literatura atual sobre *Text-to-SQL* baseado em LLM concentra-se principalmente em duas abordagens: engenharia de *prompts* (*prompt engineering*) e ajuste fino (*fine-tuning*)”.

Um desafio recorrente nesse processo é o *schema linking*, que consiste na associação entre termos da pergunta e os elementos do esquema do banco de dados, como tabelas e colunas (Shi *et al.*, 2024; Li *et al.*, 2023). Por exemplo, uma pergunta pode mencionar município, enquanto essa informação está representada no banco por outro nome. Quando essa associação falha, o modelo pode gerar consultas incorretas, erro que atinge 41,6% dos casos em cenários práticos, conforme evidenciado por Li *et al.* (2023).

Por essa razão, muitos métodos incluem o contexto do esquema no texto de entrada fornecido ao modelo. Nessa configuração, o modelo recebe uma descrição estruturada das tabelas, colunas e relações do banco (Shi *et al.*, 2024). Essa descrição permite reduzir ambiguidades e orientar a seleção correta dos elementos, sendo especialmente relevante em bases com múltiplas tabelas e consultas que envolvem junções ou agregações (Li *et al.*, 2023).

Em síntese, a literatura indica que o desempenho em *Text-to-SQL* depende não apenas da capacidade do modelo, mas de como a tarefa e o esquema são apresentados. Neste trabalho, essa perspectiva fundamenta a adoção de dois cenários de geração: (i) com apenas a pergunta e instruções de saída e (ii) com a inclusão do contexto do esquema, permitindo analisar seu impacto na qualidade das consultas geradas.

2.3 *Benchmarks* e desafios em Text-to-SQL: SPIDER e BIRD

Para comparar métodos de *Text-to-SQL* de forma reprodutível, a literatura utiliza *benchmarks*, conjuntos padronizados de avaliação compostos por bases de dados, perguntas em linguagem natural e consultas de referência (Shi *et al.*, 2024). Esses conjuntos tornam possível avaliar modelos sob condições equivalentes e

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

observar como o desempenho varia diante de diferentes estruturas de consulta, como filtros, agregações e junções.

O Spider é um dos *benchmarks* mais utilizados na área por enfatizar consultas complexas e multi-tabela. Conforme definem Yu *et al.* (2018), seu papel central está na avaliação da capacidade de generalização, ou seja, na habilidade do modelo de lidar com consultas e esquemas não vistos no treinamento.

> Definimos uma nova tarefa de análise semântica e *ext-to-SQL* complexa e entre domínios, de modo que diferentes consultas SQL complexas e diferentes bases de dados apareçam nos conjuntos de treino e teste. Dessa forma, a tarefa exige que o modelo generalize adequadamente tanto para novas consultas SQL quanto para novos esquemas de banco de dados. (Yu *et al.,* 2018, p. 3911, tradução nossa)

Entretanto, trabalhos posteriores apontam que bons resultados nesses conjuntos nem sempre se traduzem diretamente em cenários aplicados, onde o esquema pode ser mais amplo e ambíguo (Shi *et al.*, 2024; Li *et al.*, 2023).

Nesse contexto, Li *et al.* (2023) propõem o BIRD (*BIg Bench for LaRgeScale Database Grounded Text-to-SQLs*) para reduzir a distância entre a avaliação acadêmica e o uso prático. A proposta enfatiza bancos de dados em larga escala e destaca desafios adicionais, como valores ruidosos e a necessidade de conhecimento externo para interpretar perguntas. Segundo os autores, essa mudança incorpora esquemas e perturbações mais próximas de aplicações reais.

> A maioria dos *benchmarks* predominantes, como o Spider e o WikiSQL, concentra-se no esquema do banco de dados, com poucas linhas de valores, deixando uma lacuna entre o estudo acadêmico e as aplicações do mundo real. (Li *et al*., 2023, p. 1, tradução nossa)

Essa tendência de incorporar conhecimento de domínio e cenários complexos é reforçada por Shi *et al.* (2024), que destacam a importância de protocolos reprodutíveis. Destarte, essa literatura orienta a construção do conjunto de perguntas deste trabalho e sua organização por construções de consulta, bem como o uso de referências por execução para comparar os modelos sob condições controladas.

2.4 Avaliação por execução e limitações: EX e FLEX

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Um aspecto central em *Text-to-SQL* é a definição de métricas capazes de avaliar corretamente a qualidade das consultas geradas. A comparação baseada exclusivamente na correspondência textual apresenta limitações, pois diferentes consultas SQL podem ser semanticamente equivalentes e produzir o mesmo resultado (Shi *et al.*, 2024; Kim *et al.*, 2024). Por isso, a literatura adota a avaliação por execução, comparando o resultado da consulta gerada com a *gold SQL* e seu respectivo *gold result* (Shi *et al.*, 2024).

A principal métrica dessa abordagem é a *Execution Accuracy* (EX), que mede a proporção de acertos exatos nos resultados. No entanto, trabalhos recentes demonstram limitações relevantes. Conforme discutido por Shi *et al.* (2024): "A métrica EX possui limitações significativas, podendo validar incorretamente consultas devido a estados específicos do banco de dados ou penalizar consultas corretas em função de ambiguidades". Essas falhas manifestam-se como falsos positivos (coincidências) ou falsos negativos (penalizações indevidas) (Kim *et al.*, 2024).

Fatores como perguntas ambíguas, ausência de especificações de ordenação e problemas na qualidade das anotações dos *benchmarks* comprometem a confiabilidade da EX (Shi *et al.*, 2024; Kim *et al.*, 2024). Como resposta, Kim *et al.* (2024) propõem a métrica FLEX (*False-Less EXecution*), que busca aproximar a avaliação do julgamento humano. O FLEX considera o contexto da pergunta, o esquema do banco e critérios semânticos abrangentes, permitindo identificar inconsistências não capturadas pela avaliação tradicional.

> Apresentamos o FLEX, uma abordagem que utiliza modelos de linguagem para emular uma avaliação em nível de especialista humano, considerando contexto abrangente e critérios mais sofisticados. (Kim *et al.*, 2024, p. 1, tradução nossa)

No contexto deste trabalho, a avaliação por execução constitui a base para a comparação entre modelos, sendo operacionalizada por meio da métrica *Execution Accuracy* (EX). Embora a métrica FLEX represente uma proposta recente para lidar com limitações da avaliação por execução, ela não é adotada neste estudo. Ainda assim, as limitações discutidas por essa abordagem, como falsos positivos, falsos negativos e ambiguidades na avaliação, foram consideradas no desenho do

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

protocolo experimental, especialmente na construção das perguntas canônicas e das consultas SQL de referência, com o objetivo de reduzir ambiguidades e tornar a avalição mais consistente.

2.5 Idioma e robustez a variações de escrita

Além da estrutura das consultas e do esquema do banco de dados, o idioma em que a pergunta é formulada influencia significativamente o desempenho de sistemas *Text-to-SQL*. Historicamente, a maioria dos *benchmarks* foi desenvolvida em inglês, limitando a avaliação em contextos multilíngues (Dou *et al.*, 2023; Pham, Nguyen *et al.*, 2025). Estudos recentes demonstram que essa limitação impacta diretamente os resultados; conforme observado por Pham, Nguyen *et al.* (2025), há uma diferença substancial de desempenho entre cenários monolíngues e multilíngues, evidenciando que modelos treinados majoritariamente em inglês enfrentam dificuldades com outras línguas devido a ambiguidades e menor disponibilidade de dados.

> Modelos de linguagem de última geração alcançam apenas 4% de acurácia de execução em cenários multilíngues, em comparação com aproximadamente 60% em benchmarks anteriores, evidenciando uma lacuna multilíngue substancial. (Pham, Nguyen *et al.*, 2025, p. 1, tradução nossa)

No caso do português, investigações sobre a adaptação da tarefa apontam desafios adicionais, como variações morfológicas, estrutura sintática distinta e escassez de recursos linguísticos (José; Cozman, 2021). No presente trabalho, o português brasileiro é adotado por refletir o cenário de aplicação considerado, não sendo tratado como variável de comparação entre idiomas. Outro aspecto crítico para o uso prático é a capacidade de lidar com diferentes formas de escrita, como abreviações, erros ortográficos e informalidades que, embora não alterem a intenção semântica, aumentam a complexidade do *schema linking*.

Trabalhos recentes analisam como essas perturbações na entrada, incluindo ruído linguístico e reformulações superficiais , afetam o desempenho dos sistemas, podendo comprometer tanto a geração das consultas quanto a confiabilidade da avaliação (Wretblad *et al.*, 2024; Pi *et al.*, 2022). Esses efeitos estão

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

frequentemente associados a falhas na interpretação semântica e na correspondência entre termos da pergunta e elementos do esquema. Assim, a robustez diante de variações linguísticas torna-se um fator determinante para a eficácia do modelo em situações reais.

Neste estudo, as variações são geradas de forma controlada a partir de transformações linguísticas previamente definidas, como erros de digitação, abreviações e formulações informais, mantendo constante a intenção semântica original. Essa estratégia permite avaliar o comportamento dos modelos diante de ruído superficial controlado, sem introduzir ambiguidade adicional ou alterar a consulta SQL de referência esperada. Ao manter fixos o banco de dados e o protocolo de avaliação, busca-se comparar os modelos sob condições equivalentes, simulando formas simples de variação linguística que podem ocorrer em perguntas formuladas por usuários não especialistas.

3. Metodologia

Esta seção descreve detalhadamente o delineamento metodológico adotado neste estudo, apresentando de forma sistemática como o experimento foi concebido, implementado e avaliado. O objetivo é explicitar cada etapa do processo, desde a preparação dos dados até a definição das métricas finais, de modo a garantir transparência e reprodutibilidade.

3.1 Delineamento experimental

Este estudo caracteriza-se como um experimento comparativo controlado, voltado à avaliação do desempenho de modelos de linguagem na tarefa *Text-to-SQL*. A variável independente corresponde ao modelo de linguagem avaliado, enquanto a variável dependente é o desempenho na geração de consultas SQL, mensurado pela métrica *Execution Accuracy* (EX).

O procedimento experimental consiste em três etapas fundamentais: (i) geração de consultas SQL a partir de perguntas em linguagem natural; (ii) execução das consultas em banco de dados relacional; e (iii) comparação dos resultados obtidos com resultados de referência previamente definidos.

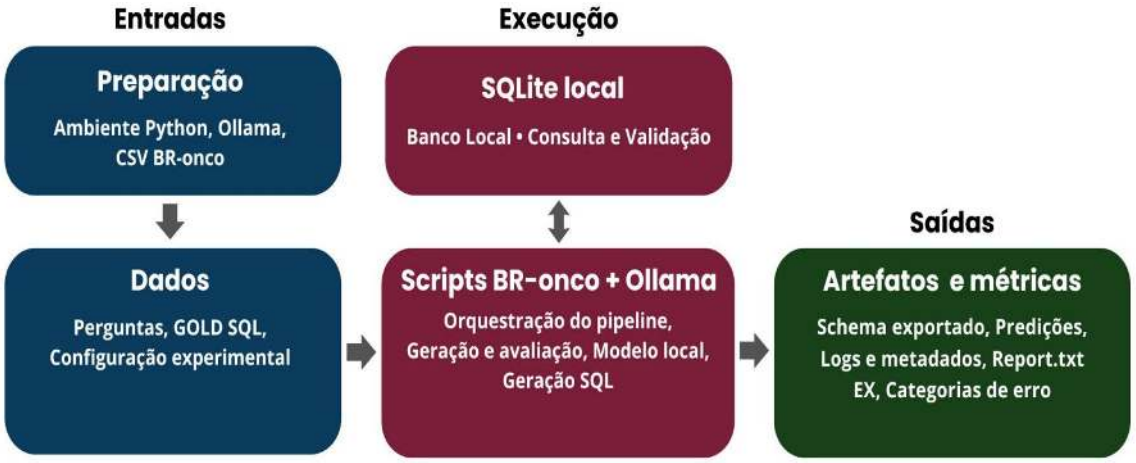
REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Para garantir a validade do experimento, foram mantidos constantes o conjunto de perguntas-base, o banco de dados, os principais parâmetros de inferência e o protocolo de avaliação. A presença ou ausência do esquema relacional no contexto de geração foi tratada como um dos fatores experimentais, permitindo comparar o comportamento dos modelos sob condições equivalentes.

A Figura 1 apresenta uma visão geral do pipeline experimental adotado, evidenciando a sequência entre preparação dos dados, definição das entradas, execução local dos modelos, construção e consulta ao banco SQLite, e geração dos artefatos utilizados na avaliação.

Figura 1 – Visão geral do pipeline experimental de geração.



Fonte: Elaborada pelos autores (2026).

O fluxo inicia-se com a preparação do ambiente, obtenção do conjunto de dados em formato CSV e definição das perguntas, consultas de referência e configuração experimental. Em seguida, os *scripts* de orquestração coordenam a construção do banco SQLite local, a geração das consultas pelos modelos executados via Ollama e a avaliação das respostas produzidas. Como resultado, são gerados artefatos intermediários, como consultas preditas, versões normalizadas e registros de execução, além dos relatórios finais contendo a EX, categorias de erro e análises comparativas por cenário e modelo.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

O experimento foi estruturado em quatro cenários, definidos pela combinação de dois fatores: a presença ou ausência do esquema do banco de dados no *prompt* e a presença ou ausência de variações linguísticas nas perguntas. Dessa forma, foram considerados os seguintes cenários: (a) sem esquema e sem ruído linguístico; (b) com esquema e sem ruído; (c) sem esquema e com ruído linguístico; e (d) com esquema e com ruído linguístico.

3.2 Fundamentação metodológica e posicionamento do estudo

Do ponto de vista metodológico, este estudo caracteriza-se como uma pesquisa aplicada, de abordagem quantitativa e delineamento experimental computacional. A pesquisa é aplicada porque busca avaliar, em um cenário específico de uso, a capacidade de modelos de linguagem de código aberto na geração de consultas SQL sobre dados públicos brasileiros do domínio oncológico. A abordagem é quantitativa porque a comparação entre os modelos é realizada por meio de métricas mensuráveis, como a EX, categorias de erro e latência.

O delineamento experimental adotado está alinhado à perspectiva de estudos empíricos em engenharia de software, nos quais variáveis são controladas para permitir comparação sistemática entre tratamentos ou alternativas avaliadas (Wohlin *et al.*, 2012). Neste trabalho, a variável independente principal corresponde ao modelo de linguagem avaliado, enquanto o desempenho na tarefa *Text-to-SQL* constitui a variável dependente. Além disso, os cenários com e sem fornecimento do esquema, bem como com e sem variações linguísticas, permitem observar o efeito de fatores experimentais relevantes sob condições padronizadas.

Em relação à literatura específica de *Text-to-SQL*, o estudo dialoga com *benchmarks* consolidados, como Spider e BIRD, ao adotar perguntas associadas a consultas SQL de referência e avaliação baseada em execução (Yu *et al.*, 2018; Li *et al.*, 2023; Shi *et al.*, 2025). Entretanto, diferentemente desses *benchmarks* amplos, o presente trabalho delimita um cenário aplicado e controlado, com banco relacional derivado de dados do domínio oncológico, formulação de perguntas em português brasileiro e execução local de modelos de código aberto via Ollama.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Essa delimitação permite analisar fatores específicos, como o impacto do esquema no *prompt* e a robustez a variações linguísticas controladas, preservando a verificação sobre os dados, perguntas, consultas de referência e protocolo de avaliação. Assim, a contribuição metodológica do estudo não está na proposição de uma nova métrica ou de um novo modelo, mas na construção de um protocolo experimental reprodutível para avaliar modelos abertos em um domínio aplicado. Essa organização é coerente com princípios de sistematização metodológica da pesquisa científica, nos quais a clareza dos procedimentos, o controle das condições e a possibilidade de replicação são elementos centrais para a validade do estudo (Marconi; Lakatos, 2003).

3.3 Ambiente computacional e configuração experimental

O experimento foi conduzido em ambiente computacional local, com orquestração das etapas por meio de *scripts* em *Python*, linguagem utilizada para automatizar o fluxo de geração, execução e avaliação das consultas. A execução local dos modelos foi realizada por meio do repositório Ollama (2025), versão 0.13.5. Foram avaliados três modelos: *qwen2.5-coder:7b*, identificado localmente pelo *ID dae161e27b0e*, com 7,6 bilhões de parâmetros, janela de contexto de 32.768 *tokens* e quantização Q4_K_M; *deepseek-coder:6.7b*, identificado pelo *ID ce298d984115*, com 7 bilhões de parâmetros, janela de contexto de 16.384 *tokens* e quantização Q4_0; e llama3:latest, correspondente ao Llama3 8B, identificado pelo *ID 365c0bd3c000*, com 8 bilhões de parâmetros, janela de contexto de 8.192 *tokens* e quantização Q4_0.

Esses identificadores correspondem aos modelos instalados localmente no ambiente experimental no momento da execução. As chamadas de geração foram realizadas com *temperature*=0.0 e *num_predict*=512. Não foram definidos explicitamente os parâmetros *top_p*, *top_k* ou *seed*, sendo mantidos os valores padrão do *runtime*/modelo. Também não foram configuradas sequências de parada adicionais pelo *pipeline* experimental.

O *timeout* de geração foi configurado por meio do parâmetro *request_timeout*, definido em 300 segundos por requisição ao Ollama. Não foram

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

realizadas novas tentativas automáticas em caso de *timeout*, pois *max_retries* foi definido como 0. O parâmetro *retry_sleep*, embora configurado com valor padrão de 5 segundos, não foi acionado nas execuções principais em razão da ausência de retentativas automáticas. Para a execução SQL, o *script* de avaliação previa o parâmetro *timeout_ms*; nas execuções principais, esse valor permaneceu em 0, indicando que o *timeout* por consulta predita ficou desabilitado. A conexão com o banco SQLite foi aberta em modo somente leitura, com *timeout*=1 na conexão.

O ambiente físico utilizado foi um notebook Samsung Galaxy Book4, executando Windows 11 Home Single Language em sistema de 64 bits, com processador 13th Gen Intel(R) Core(TM) i7-1355U, 16 GB de memória RAM e armazenamento SSD de 512 GB. A máquina também possui GPU integrada Intel(R) Iris(R) Xe Graphics. Embora o sistema operacional hospedeiro tenha sido o Windows, o pipeline experimental foi executado por meio do *Windows Subsystem for Linux* (WSL), com distribuição *Ubuntu*, ambiente no qual foram configurados os *scripts*, o banco e a execução dos modelos.

O banco de dados relacional foi implementado em SQLite, sistema baseado em arquivo local utilizado para execução das consultas SQL geradas. Para garantir a consistência, os parâmetros de geração foram mantidos constantes ao longo de todas as execuções; em particular, o grau de aleatoriedade foi fixado, reduzindo variações entre diferentes rodadas. Assim, as consultas geradas foram processadas e executadas diretamente no SQLite sob condições padronizadas para todos os modelos avaliados.

Os modelos avaliados foram selecionados com base em sua disponibilidade para execução local, natureza *open-source* e adequação a tarefas de geração de código. A literatura recente indica que modelos especializados em código apresentam desempenho superior em tarefas estruturadas, quando comparados a modelos de propósito geral, conforme apontam Guo *et al.* (2024) e Sun *et al.* (2025).

Nesse contexto, foram considerados os modelos Qwen2.5-Coder 7B, Llama 3 8B e DeepSeek-Coder 6.7B, todos executados localmente via Ollama. Os dois primeiros representam modelos especializados em geração de código, enquanto o

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Llama 3 8B foi incluído como modelo geral de instrução, permitindo contrastar arquiteturas especializadas e não especializadas na tarefa *Text-to-SQL*.

Cada execução experimental foi registrada em diretório próprio, contendo as consultas geradas, suas versões normalizadas, os registros das interações com os modelos e os relatórios de avaliação. Esses registros permitem a rastreabilidade e a verificação posterior de todo o processo experimental. A estrutura do *pipeline* experimental, incluindo *scripts*, organização dos diretórios, consultas de referência e documentação para reprodução das execuções, está disponível em repositório próprio, indicado no Apêndice A.

3.4 Fonte de dados e construção do banco relacional

O conjunto de dados utilizado neste estudo é derivado de registros públicos de atendimentos oncológicos no Brasil, disponibilizados pela plataforma DataSUS e organizados no repositório *Oncology Treatment DataSUS – Brazil* (2013–2023) (Tenorio, 2025). O arquivo original encontra-se em formato CSV (*Comma-Separated Values*) e, para viabilizar a execução de consultas SQL, o conjunto foi transformado em um banco de dados relacional.

Neste estudo, o termo “dados oncológicos” refere-se ao domínio temático e estrutural do banco de dados utilizado no *benchmark*, composto por registros tabulares relacionados ao tratamento oncológico. O objetivo do experimento não é validar clinicamente diagnósticos, tratamentos, condutas médicas ou decisões assistenciais, mas avaliar a capacidade de modelos de linguagem gerarem consultas SQL a partir de perguntas em português brasileiro sobre esse domínio de dados.

O processo de construção do banco seguiu um fluxo de ETL (*Extract*, *Transform, Load*), organizado em três etapas: extração, transformação e carga. Na etapa de extração, o arquivo original em formato CSV foi lido a partir da base *Oncology Treatment DataSUS – Brazil* (2013–2023), preservando os campos necessários à construção do banco relacional. Os dados extraídos foram inicialmente armazenados na tabela *raw_onco*, que funcionou como camada intermediária de carga, mantendo a estrutura tabular original após a leitura do

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

arquivo.

Na etapa de transformação, os campos da tabela *raw_onco* foram reorganizados para compor um modelo dimensional simplificado. Foram padronizados nomes de atributos, tipos de dados e chaves de relacionamento. A partir dos valores distintos presentes nos registros, foram derivadas as tabelas dimensão dim_sexo, dim_cid10, dim_tratamento, dim_diagnostic, dim_estadiam, dim_uf, dim_municipio, dim_cnes e dim_tempo.

A dimensão temporal foi construída a partir dos campos de ano e ano-mês presentes nos registros, permitindo representar separadamente ano, mês e identificador temporal. Já a tabela *fact_onco* foi construída para concentrar os registros centrais de atendimento, mantendo medidas e atributos analíticos, como idade e tempo até tratamento, além das chaves de ligação com as dimensões.

Na etapa de carga, as tabelas dimensão foram preenchidas antes da tabela fato, de modo a preservar a integridade referencial. Em seguida, os registros analíticos foram inseridos na tabela *fact_onco*, vinculando cada atendimento às dimensões correspondentes por meio de chaves estrangeiras. Também foram criados índices sobre campos utilizados com maior frequência nas consultas, como anomes_dia, cid10, mun_resid, sexo, tratamento_id e uf_resid, com o objetivo de favorecer a execução das consultas SQL durante o *benchmark*.

Essas verificações tiveram como objetivo confirmar que o banco SQLite estava consistente para a execução das consultas do *benchmark*. A construção do banco priorizou clareza estrutural, rastreabilidade das transformações e compatibilidade com a tarefa *Text-to-SQL*, permitindo que os modelos fossem avaliados sobre um esquema relacional explícito e reproduzível.

O quadro 1 sintetiza o processo ocorrido durante a etapa de seleção de atributos, o arquivo CSV original foi comparado com a estrutura carregada em *raw_onco*. O arquivo original continha 24 colunas, enquanto a tabela *raw_onco* foi composta por 21 colunas. Três campos do arquivo original não foram carregados no banco: uma coluna sem nome identificada no cabeçalho do CSV, CNS_PAC e DT_NASC. A coluna sem nome foi descartada por não representar um atributo analítico do domínio, sendo tratada como índice técnico do arquivo exportado.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

O campo CNS_PAC foi removido por corresponder a um identificador individual de paciente, sem utilidade para as consultas do *benchmark* e inadequado ao princípio de minimização de atributos identificadores. O campo DT_NASC também foi removido por representar uma informação temporal individual mais e redundante para os objetivos do estudo, uma vez que a idade já estava disponível no atributo IDADE.

Quadro 1 – Colunas removidas ou não propagadas no modelo analítico.

Coluna	Etapa	Decisão adotada	Justificativa
Coluna sem nome	CSV → raw_onco	Removida	Não representa atributo analítico do domínio; tratada como índice técnico de exportação
CNS_PAC	CSV → raw_onco	Removida	Identificador individual de paciente; não necessário às consultas do *benchmark*
DT_NASC	CSV → raw_onco	Removida	Informação individual sensível e redundante, pois o atributo IDADE foi preservado
ANO_DIAGN	raw_onco → fact_onco	Não propagada diretamente	Informação temporal representada por ANOMES_DIA e pela dimensão dim_tempo
ANO_TRATAM	raw_onco → fact_onco	Não propagada diretamente	Informação temporal representada por ANOMES_TRA e pela dimensão dim_tempo

Fonte: Elaborado pelos autores (2026).

Na transformação para o modelo analítico final, os campos ANO_DIAGN e ANO_TRATAM não foram propagados diretamente para *fact_onco*, pois suas informações já estavam representadas de forma mais granular nos campos ANOMES_DIA e ANOMES_TRA. Esses campos foram utilizados para compor a dimensão temporal dim_tempo, que organiza as informações em anomes, ano e mes. Dessa forma, a exclusão dos campos anuais diretos na tabela fato teve como objetivo reduzir redundância e manter uma representação temporal padronizada para as consultas SQL.

A modelagem final resultante do processo de ETL segue uma estrutura dimensional simplificada, composta por uma tabela fato (*fact_onco*) e nove tabelas dimensão. Essa organização é compatível com modelos dimensionais, nos quais a tabela fato concentra os eventos ou registros centrais do processo analisado, enquanto as dimensões armazenam atributos descritivos utilizados para

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

contextualizar, filtrar e agrupar esses registros, com o objetivo de tornar a estrutura do banco auditável e facilitar a reprodutibilidade do experimento.

A estrutura resultante permite representar os registros oncológicos em formato relacional, mantendo separadas as entidades descritivas e a tabela fato utilizada nas consultas. Essa organização favorece a avaliação da tarefa *Text-to-SQL*, pois exige que os modelos identifiquem corretamente tabelas, colunas e relações, especialmente em consultas que envolvem junções entre *fact_onco* e as dimensões.

O arquivo SQLite final, denominado, possui 213.700.608 bytes, aproximadamente 204 MiB. Esse valor corresponde ao banco após a carga dos dados, criação das tabelas dimensionais, inserção da tabela fato e definição dos índices explícitos utilizados no experimento. O Quadro 2 apresenta o esquema relacional do banco, descrevendo o papel de cada tabela, suas chaves primárias, principais colunas e relações com as demais entidades. Essa organização permite compreender como os dados foram estruturados após o processo de ETL e como a tabela fato se relaciona com as dimensões utilizadas nas consultas do *benchmark*.

Quadro 2 – Esquema relacional do banco de dados.

Tabela	Papel no banco	Chave primária	Colunas	Relações principais
raw_onco	Tabela de carga/staging	—	ANO_DIAGN, ANOMES_DIA, ANO_TRATAM, ANOMES_TRA, UF_RESID, MUN_RESID, UF_TRATAM, MUN_TRATAM, UF_DIAGN, MUN_DIAG, TRATAMENTO, DIAGNOSTIC, IDADE, SEXO, ESTADIAM, CNES_DIAG, CNES_TRAT, TEMPO_TRAT, DIAG_DETH, DT_DIAG, DT_TRAT	Não possui chaves estrangeiras; utilizada como tabela intermediária de carga
fact_onco	Tabela fato	id	id, anomes_dia, anomes_trat, uf_resid, mun_resid, uf_diagn, mun_diagn, uf_trat, mun_trat, cnes_diagn,	Referencia todas as tabelas dimensão por chaves estrangeiras

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

			cnes_trat, cid10, sexo, idade, tratamento_id, diagnostic_id, estadiam_id, tempo_trat, dt_diag, dt_trat	
dim_cid10	Dimensão de diagnóstico CID-10	cid10	cid10	Referenciada por fact_onco.cid10
dim_cnes	Dimensão de estabelecimentos CNES	cnes_id	cnes_id	Referenciada por fact_onco.cnes_diagn e fact_onco.cnes_trat
dim_diagnostic	Dimensão de diagnóstico	diagnostic_id	diagnostic_id	Referenciada por fact_onco.diagnostic_id
dim_estadiam	Dimensão de estadiamento	estadiam_id	estadiam_id	Referenciada por fact_onco.estadiam_id
dim_municipio	Dimensão de municípios	mun_id	mun_id, uf_id	Referenciada por fact_onco.mun_resid, fact_onco.mun_diagn e fact_onco.mun_trat
dim_sexo	Dimensão de sexo	sexo	sexo	Referenciada por fact_onco.sexo
dim_tempo	Dimensão temporal	anomes	anomes, ano, mes	Referenciada por fact_onco.anomes_dia e fact_onco.anomes_trat
dim_tratamento	Dimensão de tratamento	tratamento_id	tratamento_id	Referenciada por fact_onco.tratamento_id
dim_uf	Dimensão de unidades federativas	uf_id	uf_id	Referenciada por fact_onco.uf_resid, fact_onco.uf_diagn e fact_onco.uf_trat

Fonte: Elaborado pelos autores (2026).

Após a carga, foram realizadas verificações de consistência, incluindo a validação de integridade referencial e a execução de consultas simples para confirmação da coerência dos dados. A construção do banco teve como objetivo organizar os dados para avaliação experimental, priorizando clareza estrutural e compatibilidade com a tarefa *Text-to-SQL*.

Além da descrição estrutural do esquema, a Tabela 1 apresenta a caracterização quantitativa do banco SQLite utilizado no experimento. A tabela informa, para cada entidade, seu papel no banco, número de colunas, quantidade de registros carregados, chave primária e quantidade de relações declaradas por chave estrangeira. A tabela interna *sqlite_sequence*, criada automaticamente pelo SQLite em razão do uso de *AUTOINCREMENT*, não foi considerada nessa contagem por não compor o modelo relacional do experimento.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Tabela 1 – Caracterização quantitativa do banco SQLite.

Tabela	Papel	Colunas	Registros	Chave primária	Relações declaradas
raw_onco	Carga/staging	21	995.374	—	0
fact_onco	Fato	20	995.374	id	15
dim_cid10	Dimensão	1	109	cid10	0
dim_cnes	Dimensão	1	2.013	cnes_id	0
dim_diagnostic	Dimensão	1	4	diagnostic_id	0
dim_estadiam	Dimensão	1	7	estadiam_id	0
dim_municipio	Dimensão	2	5.568	mun_id	0
dim_sexo	Dimensão	1	2	sexo	0
dim_tempo	Dimensão	3	150	anomes	0
dim_tratamento	Dimensão	1	5	tratamento_id	0
dim_uf	Dimensão	1	28	uf_id	0

Fonte: Elaborada pelos autores (2026).

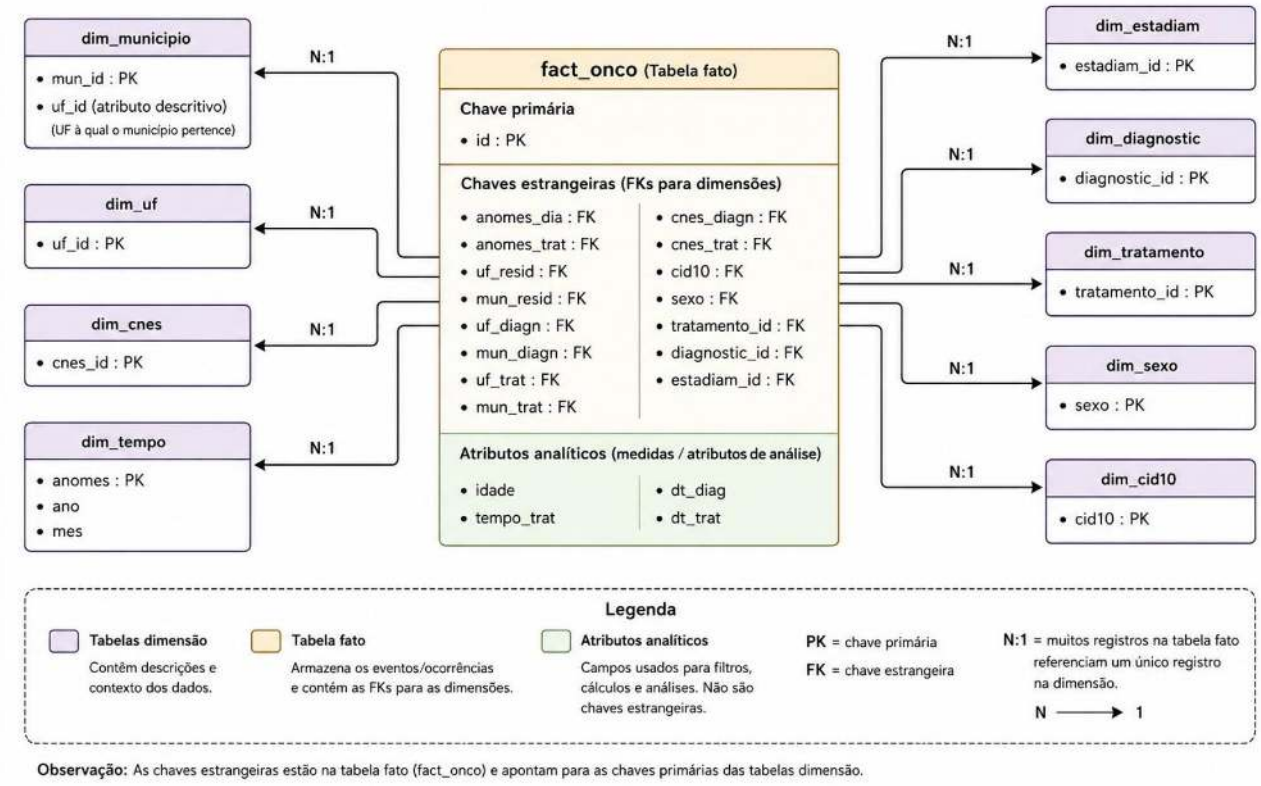
No total, o banco contém 11 tabelas de usuário, sendo uma tabela de carga (*raw_onco*), uma tabela fato (*fact_onco*) e nove tabelas dimensão. O esquema possui 53 colunas, 10 chaves primárias, 15 chaves estrangeiras declaradas e 6 índices explícitos. A tabela fato contém 995.374 registros analíticos, correspondentes aos registros utilizados nas consultas do *benchmark*. A tabela *raw_onco* possui a mesma quantidade de registros por representar a camada inicial de carga antes da reorganização relacional dos dados.

A Figura 2 apresenta esse modelo, evidenciando a posição da tabela *fact_onco* como elemento central do esquema e suas relações com as dimensões utilizadas nas consultas do *benchmark*. Essa representação complementa a descrição tabular do esquema relacional e facilita a visualização das dependências entre tabelas, chaves primárias e chaves estrangeiras.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Figura 2 – Modelo dimensional simplificado do banco relacional utilizado.



Fonte: Elaborada pelos autores (2026).

A tabela *fact_onco* concentra os registros centrais de atendimento oncológico e se conecta às dimensões por meio de chaves estrangeiras. Algumas dimensões são referenciadas por mais de um atributo da tabela fato: dim_tempo é relacionada aos campos anomes_dia e anomes_trat; dim_uf aos campos uf_resid, uf_diagn e uf_trat; dim_municipio aos campos mun_resid, mun_diagn e mun_trat; e dim_cnes aos campos cnes_diagn e cnes_trat.

As demais dimensões são associadas por chaves específicas, como cid10, sexo, tratamento_id, diagnostic_id e estadiam_id. Essa organização favorece consultas com filtros, agrupamentos e junções, características relevantes para a avaliação da tarefa *Text-to-SQL*.

3.5 Construção do conjunto de perguntas e definição das consultas de referência

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

A avaliação experimental fundamenta-se em um conjunto estruturado de perguntas em português brasileiro, cada uma vinculada a uma consulta SQL de referência. Esse conjunto constitui o conjunto avaliativo do estudo, cujas questões foram elaboradas manualmente com base na estrutura do banco relacional detalhado na seção anterior, assegurando alinhamento entre as perguntas, o esquema do banco e o domínio dos dados analisados.

O conjunto contempla diferentes padrões de consulta, incluindo agregações, filtros, agrupamentos, ordenações, limites, junções entre a tabela fato e as dimensões, além de uma subconsulta. Essa composição foi definida para avaliar a capacidade dos modelos de gerar consultas SQL executáveis em um cenário relacional controlado, mas com variações suficientes para envolver diferentes operações presentes em análises exploratórias de dados.

Cada pergunta foi redigida de forma direta e associada a uma consulta de referência (*gold SQL*), utilizada como padrão de comparação. Tais consultas foram construídas sob critérios de consistência sintática e determinismo, incorporando, quando necessário, cláusulas de ordenação explícita e o devido tratamento de valores nulos para assegurar a precisão.

Para validar o processo, todas as consultas foram executadas previamente em um banco SQLite, garantindo a ausência de erros e a coerência dos resultados. O conjunto final compreende trinta perguntas canônicas que servem de base para a geração de variações linguísticas controladas, as quais mantêm a intenção semântica original e o vínculo com as versões de referência.

Para caracterizar a composição do *benchmark*, as trinta perguntas canônicas foram classificadas de acordo com o tipo predominante de consulta SQL e com o nível de dificuldade. A classificação foi realizada a partir das consultas de referência, considerando a presença de operações como agregações, junções, filtros, ordenações, limites e subconsultas. Esse procedimento permite tornar mais transparente a complexidade do conjunto avaliativo e facilita a interpretação dos resultados por modelo e por cenário.

A Tabela 2 apresenta a distribuição das perguntas canônicas segundo o tipo predominante de consulta SQL. Essa classificação considera a operação estrutural

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

mais representativa de cada consulta de referência, distinguindo consultas baseadas apenas em agregação, consultas que combinam agregação com junção e consultas que envolvem subconsulta.

Tabela 2 – Distribuição das perguntas canônicas por tipo de consulta.

Tipo de consulta	Quantidade	Percentual
Agregação	21	70,0%
Agregação com junção	8	26,7%
Subconsulta	1	3,3%
Total	30	100,0%

Fonte: Elaborada pelos autores (2026).

Conforme apresentado na Tabela 2, o conjunto avaliativo é composto majoritariamente por consultas de agregação, que representam 70,0% das perguntas canônicas. As consultas de agregação com junção correspondem a 26,7% do conjunto, enquanto apenas uma pergunta envolve subconsulta. Esse perfil indica que o *benchmark* prioriza consultas analíticas descritivas sobre os dados, com ênfase em contagens, médias, agrupamentos e recortes temporais ou categóricos.

Além do tipo predominante de consulta, as perguntas também foram classificadas por nível de dificuldade. Essa classificação levou em consideração a combinação de operações presentes nas consultas de referência. Perguntas simples envolvem baixa composição operacional; perguntas intermediárias combinam agregações com filtros, ordenações, limites ou junções simples; e perguntas complexas envolvem maior densidade operacional, como junções associadas a filtros e ordenações, uso de limite ou presença de subconsulta. A Tabela 3 sintetiza essa distribuição.

Tabela 3 – Distribuição das perguntas canônicas por nível de dificuldade.

Dificuldade	Critério geral	Quantidade	Percentual
Simples	Agregação global sem filtros, junções, ordenações, limites ou subconsultas	1	3,3%
Intermediária	Agregações associadas a filtros, ordenações, limites ou junções simples	17	56,7%
Complexa	Combinações com múltiplas operações, junções com filtros/ordenações ou subconsulta	12	40,0%
Total	—	30	100,0%

Fonte: Elaborada pelos autores (2026).

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

A distribuição por dificuldade mostra que a maior parte do conjunto está concentrada nos níveis intermediário e complexo. Apenas uma pergunta foi classificada como simples, correspondente a uma agregação global sem filtros, junções, ordenações, limites ou subconsultas. As perguntas intermediárias representam 56,7% do conjunto, enquanto as complexas correspondem a 40,0%. Esse resultado indica que, embora o *benchmark* seja reduzido em número de perguntas, ele não se limita a consultas triviais, contemplando diferentes combinações de operações SQL relevantes para a avaliação de modelos *Text-to-SQL.*

Por fim, a Tabela 4 apresenta a frequência das principais operações SQL presentes nas consultas de referência. Diferentemente da classificação por tipo predominante, essa análise não é mutuamente exclusiva, pois uma mesma consulta pode envolver simultaneamente agregação, filtro, ordenação, junção, limite e subconsulta.

Tabela 4 – Operações SQL presentes nas consultas de referência.

Operação SQL	Quantidade de perguntas	Percentual
Agregações	30	100,0%
Filtros	25	83,3%
Ordenações	23	76,7%
Junções	8	26,7%
Limites	8	26,7%
Subconsultas	1	3,3%

Fonte: Elaborada pelos autores (2026).

A Tabela 4 mostra que todas as perguntas canônicas envolvem algum tipo de agregação, como COUNT, AVG, MIN ou MAX, refletindo o objetivo analítico do *benchmark*. Além disso, 25 perguntas incluem filtros, 23 utilizam ordenação, 8 envolvem junções entre tabelas, 8 aplicam limite de resultados e 1 envolve subconsulta. Como uma mesma pergunta pode conter mais de uma operação SQL, os percentuais dessa tabela não devem ser somados. Essa caracterização evidencia que o conjunto avaliativo, ainda que exploratório e controlado, contempla diferentes construções relevantes para a avaliação de modelos *Text-to-SQL* em um banco relacional dimensional.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

As variações linguísticas foram construídas a partir das perguntas canônicas, preservando a intenção semântica original e a mesma consulta SQL de referência. O objetivo não foi introduzir ambiguidade semântica, ruído adversarial ou reformulações capazes de alterar a resposta esperada, mas sim avaliar o comportamento dos modelos diante de ruído superficial controlado, próximo de alterações comuns em perguntas formuladas por usuários não especialistas.

Para tornar esse procedimento auditável, as variações foram classificadas em cinco categorias: abreviação lexical, simplificação sintática, erro ortográfico controlado, reformulação coloquial e substituição lexical superficial. A abreviação lexical corresponde à redução informal de termos, como “quantos” para “qtos” ou “diagnóstico” para “diag”. A simplificação sintática envolve a remoção de termos funcionais sem alteração da intenção, como a supressão de “é”, “são” ou “foi” em perguntas diretas.

O erro ortográfico controlado corresponde a pequenas alterações gráficas que não impedem a compreensão, como “pacintes” e “tratamnto”. A reformulação coloquial aproxima a pergunta de usos informais, como “pra cada” ou “só”. A substituição lexical superficial ocorre quando um termo é trocado por outro equivalente ou convencional no contexto da consulta, como “quais são os 10” por “Top 10” ou “diferente” por “dif.”. O Quadro 3 organiza as respectivas variações.

Quadro 3 – Taxonomia das variações linguísticas utilizadas.

Categoria	Descrição	Exemplos observados
Abreviação lexical	Redução informal de palavras ou expressões preservando o sentido original	Qtos, Qtd, diag, trat, estadiam, munic
Simplificação sintática	Supressão ou simplificação de termos funcionais sem alterar a intenção da pergunta	“Qual é” → “Qual”; “Quais são” → “Quais”; “No ano de 2020” → “Em 2020”
Erro ortográfico controlado	Pequenas alterações gráficas intencionais, sem gerar ambiguidade semântica relevante	pacintes, tratamnto
Reformulação coloquial	Uso de formulações mais próximas da linguagem cotidiana	Pra cada, só tempo_trat não nulo
Substituição lexical superficial	Troca por termo equivalente ou convencional, sem mudança da consulta esperada	“Quais são os 10” → Top 10; “diferente” → dif.

Fonte: Elaborado pelos autores (2026).

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

O Quadro 4 apresenta exemplos de perguntas canônicas e suas respectivas versões com ruído. Os exemplos foram selecionados para representar diferentes categorias da taxonomia adotada, preservando a mesma intenção semântica e a mesma consulta SQL de referência.

Quadro 4 – Exemplos de perguntas canônicas e versões com ruído.

Categoria predominante	Pergunta canônica	Pergunta com ruído
Abreviação lexical	Quantos registros existem no conjunto de dados?	Qtos registros existem no conjunto de dados?
Abreviação lexical	Quantos registros existem por ano de diagnóstico (dim_tempo.ano)?	Quantos registros existem por ano de diag (dim_tempo.ano)?
Erro ortográfico controlado	Qual é a idade média dos pacientes por sexo?	Qual a idade média dos pacintes por sexo?
Erro ortográfico controlado	Qual é o tempo médio até tratamento (tempo_trat) por ano de diagnóstico?	Qual o tempo médio até tratamnto (tempo_trat) por ano de diagnóstico?
Substituição lexical superficial	Quais são as 10 UFs de residência (uf_resid) com mais registros?	Top 10 UFs de residência (uf_resid) com mais registros?
Reformulação coloquial	Para cada UF de residência, qual é a proporção de casos cujo tratamento ocorreu na mesma UF?	Pra cada UF de residência, qual a proporção de casos cujo trat ocorreu na mesma UF?
Simplificação sintática	No ano de 2020, quais são os 10 CIDs (cid10) mais frequentes?	Em 2020, quais os 10 CIDs (cid10) mais frequentes?

Fonte: Elaborado pelos autores (2026).

Como indicado no Quadro 4, as variações não modificam entidades centrais, filtros, agregações ou relações esperadas nas consultas SQL. Assim, a comparação entre perguntas canônicas e perguntas com ruído avalia apenas a sensibilidade dos modelos a alterações superficiais de escrita, não sua capacidade de resolver ambiguidades semânticas, omissões de informação ou ruídos adversariais mais severos.

3.6 Geração das consultas pelos modelos

A geração das consultas SQL corresponde à etapa em que os modelos de linguagem processam perguntas em linguagem natural e produzem uma instrução SQL candidata. Esse procedimento foi conduzido exclusivamente em regime de inferência, sem ajuste fino, retreinamento ou adaptação dos parâmetros internos dos modelos ao domínio analisado. Assim, o desempenho avaliado reflete o

REMUNOM
ISSN 2178-6925
Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

comportamento dos modelos em configuração local controlada, a partir de *prompts* padronizados.

Foram utilizados dois formatos de *prompt*, correspondentes aos cenários sem esquema e com esquema. No cenário sem esquema, o *prompt* continha uma instrução inicial para geração de uma única consulta SQLite, regras estritas de saída, exemplos de consultas SQL e a pergunta em português brasileiro. As regras determinavam que a resposta deveria conter apenas a consulta SQL, sem explicações, comentários, *Markdown* ou blocos de código, além de orientar o uso de sintaxe SQLite válida, ORDER BY em consultas com LIMIT e preservação da ordem das colunas solicitadas na pergunta.

No cenário com esquema, o *prompt* manteve a mesma estrutura geral, mas acrescentou o bloco *Schema*:, preenchido com a descrição compacta do banco de dados. Esse bloco continha as tabelas, colunas e relações por chaves estrangeiras do banco. Além disso, o *prompt* com esquema incluiu regras adicionais para restringir a geração às tabelas e colunas existentes, seguir as relações declaradas entre a tabela fato e as dimensões, evitar *aliases* não definidos, utilizar corretamente os campos temporais, geográficos e categóricos, e desconsiderar valores nulos em consultas nas quais isso fosse necessário, como no cálculo de tempo até tratamento e em agregações por categoria.

Todo o conteúdo instrucional foi enviado diretamente no campo *prompt* da requisição ao Ollama. Não foi utilizado *prompt system* separado, nem campos adicionais como *system*, *template* ou *messages*. O *payload* de geração continha o nome do modelo, o *prompt* textual, *stream=false* e os parâmetros de inferência definidos em *options*, incluindo *temperature* e *num_predict*. Em todas as execuções principais, a temperatura foi fixada em 0,0 e o limite de geração foi definido em 512 *tokens*.

Após a geração, aplicou-se uma etapa de pós-processamento para extrair a consulta SQL da resposta bruta do modelo. O extrator foi implementado com expressões regulares e seguiu uma ordem fixa de decisão: inicialmente, verificava-se a existência de bloco de código *Markdown* explicitamente marcado como SQL, no formato “sql ...” ; quando esse padrão era encontrado, apenas o conteúdo

REMUNOM
ISSN 2178-6925
Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

interno do bloco era preservado. Na ausência desse padrão, o algoritmo buscava um bloco genérico de código, no formato "...". Se nenhum bloco de código fosse identificado, a resposta era mantida como texto plano, removendo-se apenas prefixos recorrentes, como "SQL:", quando presentes.

Em todos os casos, o pós-processamento também removeu espaços periféricos e ponto e vírgula final, quando existente. Em seguida, a consulta foi normalizada por meio da remoção de quebras de linha e espaços redundantes, sendo convertida para um formato padronizado em uma única linha antes da validação e execução. Esse procedimento não reescreveu semanticamente a consulta gerada, nem corrigiu nomes de tabelas, colunas, operadores ou condições; sua função foi apenas remover elementos de apresentação que poderiam impedir a execução automatizada ou dificultar a comparação experimental.

Como exemplo, uma resposta bruta como ```sql SELECT f.sexo, COUNT(*) AS n FROM fact_onco f JOIN dim_sexo s ON f.sexo = s.sexo GROUP BY f.sexo ORDER BY n DESC; ``` foi convertida para SELECT f.sexo, COUNT(*) AS n FROM fact_onco f JOIN dim_sexo s ON f.sexo = s.sexo GROUP BY f.sexo ORDER BY n DESC. Da mesma forma, uma resposta iniciada por "SQL:" teve esse prefixo removido antes da execução.

No total, foram analisadas 360 respostas geradas pelos modelos; 225 não exigiram limpeza além da padronização trivial, enquanto 135 exigiram limpeza relevante. Entre essas, 106 envolveram remoção de texto periférico e quebras de linha, 26 envolveram remoção de blocos *Markdown* associados a texto periférico e quebras de linha, e 3 envolveram apenas remoção de texto periférico. Esse registro foi utilizado para documentar a frequência de intervenções automáticas de limpeza, mantendo a avaliação baseada na consulta efetivamente extraída de cada resposta.

Cada execução registrou a pergunta, a resposta bruta produzida pelo modelo, a consulta SQL extraída, o tempo de geração e os metadados do cenário experimental, incluindo modelo, conjunto de perguntas, modo de esquema,

REMUNOM
ISSN 2178-6925
Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

temperatura e limite de geração. O procedimento foi mantido idêntico para todos os modelos dentro de cada condição avaliada, de modo a preservar a comparabilidade entre os resultados. Os *prompts* completos utilizados nos cenários sem esquema e com esquema são apresentados no Apêndice D, juntamente com os demais artefatos de reprodutibilidade do *benchmark*.

3.7 Política de segurança e validação das consultas

A execução das consultas SQL geradas pelos modelos foi realizada sob uma rigorosa política de segurança, com restrições explícitas e critérios de validação definidos previamente. Foram autorizadas apenas instruções de leitura do tipo *SELECT*, caracterizando uma abordagem estritamente *SELECT-only* para o ambiente de testes. Essa medida assegura a integridade dos dados e o isolamento do banco de dados relacional durante todo o processo de avaliação experimental.

Instruções que implicassem modificação de dados ou alterações na estrutura, como *INSERT, UPDATE, DELETE, DROP, ALTER e CREATE*, foram bloqueadas de forma mandatória pelo protocolo. Adicionalmente, restringiu-se a presença de múltiplas instruções em uma única saída, sendo permitida apenas uma instrução SQL executável por consulta. Tal restrição visa evitar ambiguidades no processamento dos resultados e garantir a precisão da métrica.

Antes da execução, cada consulta foi submetida a uma validação textual detalhada para verificar se a instrução iniciava com *SELECT* ou cláusulas compatíveis, como *WITH*, conforme as normas estabelecidas. Esse filtro preliminar também buscou identificar a presença de comandos bloqueados que pudessem ter sido gerados. Consultas que não atendiam a esses critérios de conformidade foram descartadas e não prosseguiram para a etapa de execução no banco.

As falhas identificadas foram classificadas em cinco categorias: violação de política, quando a instrução não atende às restrições de segurança definidas; erro sintático, quando a consulta é malformada; erro de execução, quando a consulta é sintaticamente válida, mas falha operacionalmente; *timeout*, quando a execução

REMUNOM
ISSN 2178-6925
Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

excede o limite estabelecido; e resultado incorreto, quando a consulta executa, mas retorna dados diferentes da referência. Essa tipologia permite uma análise mais granular das limitações dos modelos. Em suma, esta política define as condições fundamentais de validação e execução que regem o protocolo experimental e garantem a segurança do estudo.

3.8 Protocolo de avaliação

A avaliação do desempenho dos modelos foi conduzida por meio de protocolo baseado em execução. Para cada pergunta, a consulta produzida pelo modelo foi executada no banco SQLite e seu resultado foi comparado ao resultado obtido pela execução da consulta de referência correspondente no mesmo banco.

A comparação foi realizada sobre os resultados retornados após a padronização da representação dos dados, adotando-se equivalência por execução e igualdade de denotação entre os conjuntos de resultados. Além disso, foram registradas as falhas observadas conforme as categorias definidas no processo de validação, permitindo identificar a natureza exata das inconsistências em cada modelo. A métrica principal adotada foi a Execution Accuracy (EX), definida como a razão entre o número de consultas geradas corretamente e o número total de perguntas avaliadas:

$$\boldsymbol{EX = \frac{N_{corretas}}{N_{total}}}$$

Em que *Ncorretas* representa o número de consultas geradas cuja execução retornou resultado idêntico ao da consulta de referência, e *Ntotal* corresponde ao número total de perguntas avaliadas. Neste estudo, a EX foi calculada separadamente por modelo e por cenário experimental, bem como de forma agregada para cada modelo. Além da EX, foram registradas as falhas observadas conforme as categorias definidas no processo de validação: erro sintático, erro de execução e resultado incorreto.

Cada pergunta foi submetida uma única vez a cada modelo em cada cenário

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

experimental, como não foram realizadas múltiplas execuções independentes para a mesma combinação de modelo, pergunta e cenário, este estudo não estima variabilidade intra-modelo entre diferentes amostragens. A análise estatística foi conduzida sobre os acertos e erros observados nessa execução única, por meio de intervalos de confiança para EX e comparação pareada por pergunta entre modelos. Portanto, os resultados devem ser interpretados como uma avaliação exploratória controlada do comportamento dos modelos sob a configuração experimental adotada, e não como uma estimativa da variabilidade interna de cada modelo em múltiplas rodadas.

Também foram coletadas medidas temporais do experimento, incluindo latência de geração e latência de execução. A primeira corresponde ao tempo entre o envio do *prompt* e a resposta do modelo; a segunda, ao tempo de execução no banco de dados. Adicionalmente, foi adotado um limite máximo de tempo por consulta predita durante a etapa de avaliação, de modo a interromper execuções anômalas e classificá-las separadamente como *timeout*. Todas as avaliações seguiram rigorosamente o mesmo conjunto de perguntas, banco de dados e critérios de validação.

4. Resultados e Discussão

Esta seção apresenta os resultados obtidos a partir da execução do protocolo experimental descrito na metodologia. São analisados o desempenho geral dos modelos avaliados, a variação da acurácia nos quatro cenários experimentais, o impacto do fornecimento do esquema do banco de dados, a robustez diante de variações linguísticas, as categorias de erro observadas e as medidas de latência de geração e execução.

A análise é conduzida de forma comparativa entre os modelos Qwen2.5Coder 7B, DeepSeek-Coder 6.7B e Llama 3 8B, considerando a métrica principal *Execution Accuracy* (EX). Além dos valores quantitativos, são discutidos os principais padrões observados nos resultados, buscando relacionar o desempenho dos modelos às condições experimentais avaliadas.

Do ponto de vista ético e operacional, os resultados não devem ser

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

interpretados como evidência de uso clínico direto de LLMs em oncologia. As consultas geradas foram avaliadas apenas quanto à correspondência de execução em um banco relacional controlado. Em aplicações reais em saúde, modelos desse tipo exigem validação externa, auditoria, supervisão especializada, controle de privacidade e governança sobre os dados e seus usos. Assim, este estudo se limita à avaliação técnica de *Text-to-SQL* em dados tabulares de domínio oncológico, sem propor substituição de análise profissional ou apoio direto à decisão clínica.

4.1 Visão geral das execuções experimentais

Foram avaliados três modelos de linguagem em quatro cenários experimentais, totalizando 12 execuções. Cada execução considerou 30 perguntas, resultando em 360 consultas avaliadas. Os cenários foram definidos pela combinação de dois fatores: presença ou ausência do esquema do banco de dados no *prompt* e presença ou ausência de ruído linguístico nas perguntas.

Assim, o desenho experimental permitiu observar o desempenho dos modelos em quatro condições: perguntas canônicas sem esquema, perguntas canônicas com esquema, perguntas com ruído sem esquema e perguntas com ruído com esquema. Essa organização possibilita analisar tanto o desempenho geral dos modelos quanto o efeito isolado dos fatores esquema e ruído linguístico.

A métrica principal de desempenho foi a EX, complementada pela análise das categorias de erro e das medidas de latência de geração e execução. A Tabela 5 resume a configuração geral das execuções experimentais.

Tabela 5 – Configuração geral das execuções experimentais.

Modelo	Tipo	Cenários	Perguntas/cenário	Total de consultas
Qwen2.5-Coder 7B	Código	4	30	120
DeepSeek-Coder 6.7B	Código	4	30	120
Llama 3 8B	Geral	4	30	120
Total	–	4	30	360

Fonte: Elaborada pelos autores (2026).

A partir dessa configuração, as subseções seguintes apresentam a comparação do desempenho geral e por cenário, seguida da análise específica do

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

impacto do esquema, da robustez a variações linguísticas, das categorias de erro e das medidas de latência.

4.2 Desempenho geral e por cenário dos modelos

A Tabela 6 apresenta a EX obtida por cada modelo nos quatro cenários experimentais. Para melhor organização visual, os cenários são identificados como C1, C2, C3 e C4, conforme legenda apresentada abaixo da tabela. Em termos descritivos, o Qwen2.5-Coder 7B apresentou o maior valor agregado, com EX de 80,0%, seguido pelo DeepSeek-Coder 6.7B, com 71,7%, e pelo Llama 3 8B, com 69,2%.

Entretanto, a análise por cenário revela diferenças importantes. Nos cenários com fornecimento do esquema, todos os modelos alcançaram EX igual ou superior a 86,7%. Por outro lado, nos cenários sem esquema, a acurácia foi substancialmente menor, especialmente para DeepSeek-Coder 6.7B e Llama 3 8B. Esse comportamento indica que a comparação entre modelos deve considerar não apenas a EX agregada, mas também as condições de entrada fornecidas no *prompt*.

Tabela 6 – *Execution Accuracy* por modelo e cenário.

Modelo	C1	C2	C3	C4	EX geral
Qwen2.5-Coder 7B	66,7%	90,0%	66,7%	96,7%	80,0%
DeepSeek-Coder 6.7B	43,3%	93,3%	50,0%	100,0%	71,7%
Llama 3 8B	46,7%	86,7%	50,0%	93,3%	69,2%

Legenda: C1 = canônica sem esquema; C2 = canônica com esquema; C3 = ruído sem esquema; C4 = ruído com esquema.

Fonte: Elaborada pelos autores (2026).

Os resultados mostram que a especialização em código não garantiu superioridade em todos os cenários. O DeepSeek-Coder 6.7B, por exemplo, apresentou desempenho inferior ao Llama 3 8B no cenário canônico sem esquema, embora tenha alcançado os melhores resultados nos cenários com esquema. Já o Qwen2.5-Coder 7B apresentou o desempenho mais estável, mantendo 66,7% nos cenários sem esquema e atingindo 96,7% no cenário com esquema e ruído.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Esse resultado é consistente com a proposta de avaliar modelos especializados em código em uma tarefa estruturada como *Text-to-SQL*, mas também evidencia que a arquitetura do modelo não explica isoladamente o desempenho. A disponibilidade do esquema no *prompt* aparece como fator determinante para a produção de consultas mais corretas, especialmente em bancos relacionais com múltiplas tabelas e relações, conforme discutido na literatura sobre *schema linking* (Shi *et al.*, 2024; Li *et al.*, 2023). Considerando que cada cenário avaliou 30 perguntas, os valores de Execution Accuracy (EX) foram acompanhados por intervalos de confiança de 95% calculados pelo método de Wilson para proporções binomiais.

A Tabela 7 apresenta os resultados por modelo e cenário. Observa-se que a presença do esquema relacional no *prompt* elevou substancialmente a EX em todos os modelos. Nos cenários sem esquema, os intervalos de confiança são mais amplos e se sobrepõem parcialmente, refletindo a incerteza associada ao conjunto reduzido de 30 perguntas por cenário. Nos cenários com esquema, os três modelos apresentaram desempenho elevado, com EX entre 86,7% e 100,0%, embora os intervalos ainda recomendem cautela na interpretação de diferenças pontuais.

Tabela 7 – Execution Accuracy com intervalo de confiança de 95% por modelo e cenário.

Modelo	Cenário	Acertos	N	EX	IC95%
Qwen2.5-Coder 7B	Canônico sem esquema	20	30	66,7%	48,8–80,8%
Qwen2.5-Coder 7B	Canônico com esquema	27	30	90,0%	74,4–96,5%
Qwen2.5-Coder 7B	Ruído sem esquema	20	30	66,7%	48,8–80,8%
Qwen2.5-Coder 7B	Ruído com esquema	29	30	96,7%	83,3–99,4%
DeepSeek-Coder 6.7B	Canônico sem esquema	13	30	43,3%	27,4–60,8%
DeepSeek-Coder 6.7B	Canônico com esquema	28	30	93,3%	78,7–98,2%
DeepSeek-Coder 6.7B	Ruído sem esquema	15	30	50,0%	33,2–66,8%
DeepSeek-Coder 6.7B	Ruído com esquema	30	30	100,0%	88,6–100,0%
Llama 3 8B	Canônico sem esquema	14	30	46,7%	30,2–63,9%
Llama 3 8B	Canônico com esquema	26	30	86,7%	70,3–94,7%
Llama 3 8B	Ruído sem esquema	15	30	50,0%	33,2–66,8%
Llama 3 8B	Ruído com esquema	28	30	93,3%	78,7–98,2%

Fonte: Elaborada pelos autores (2026).

Como todos os modelos foram avaliados sobre as mesmas perguntas em cada cenário, as comparações entre pares de modelos foram tratadas como

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

pareadas por pergunta. Para isso, aplicou-se o teste exato de McNemar aos acertos e erros de cada par de modelos, conforme apresentado na Tabela 8.

Tabela 8 – Teste exato de McNemar entre pares de modelos por cenário.

Cenário	Modelo A	Modelo B	A correto/B errado	A errado/B correto	p-valor
Canônico sem esquema	Qwen2.5-Coder 7B	DeepSeek-Coder 6.7B	9	2	0,0654
Canônico sem esquema	Qwen2.5-Coder 7B	Llama 3 8B	8	2	0,1094
Canônico sem esquema	DeepSeek-Coder 6.7B	Llama 3 8B	5	6	1,0000
Canônico com esquema	Qwen2.5-Coder 7B	DeepSeek-Coder 6.7B	1	2	1,0000
Canônico com esquema	Qwen2.5-Coder 7B	Llama 3 8B	3	2	1,0000
Canônico com esquema	DeepSeek-Coder 6.7B	Llama 3 8B	4	2	0,6875
Ruído sem esquema	Qwen2.5-Coder 7B	DeepSeek-Coder 6.7B	6	1	0,1250
Ruído sem esquema	Qwen2.5-Coder 7B	Llama 3 8B	7	2	0,1797
Ruído sem esquema	DeepSeek-Coder 6.7B	Llama 3 8B	5	5	1,0000
Ruído com esquema	Qwen2.5-Coder 7B	DeepSeek-Coder 6.7B	0	1	1,0000
Ruído com esquema	Qwen2.5-Coder 7B	Llama 3 8B	2	1	1,0000
Ruído com esquema	DeepSeek-Coder 6.7B	Llama 3 8B	2	0	0,5000

Fonte: Elaborada pelos autores (2026).

Os resultados do teste de McNemar não indicaram diferenças estatisticamente significativas entre pares de modelos ao nível de 5%, pois nenhum p-valor foi inferior a 0,05. Os menores valores ocorreram nas comparações entre Qwen2.5-Coder 7B e DeepSeek-Coder 6.7B nos cenários sem esquema, especialmente no cenário canônico sem esquema (p=0,0654). Portanto, embora existam diferenças pontuais de EX entre modelos, os resultados pareados não sustentam, neste conjunto reduzido, uma conclusão estatisticamente robusta de superioridade geral de um modelo sobre os demais.

Além da análise agregada por cenário, a EX também foi calculada por tipo de consulta, considerando a classificação das 30 perguntas canônicas em três

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

grupos: agregação, agregação com junção e subconsulta. A Tabela 9 apresenta os resultados por modelo e cenário.

Tabela 9 – Execution Accuracy por tipo de consulta.

Modelo	Cenário	Agregação	Agregação com junção	Subconsulta
Qwen2.5-Coder 7B	Canônico sem esquema	66,7%	75,0%	0,0%
Qwen2.5-Coder 7B	Canônico com esquema	90,5%	87,5%	100,0%
Qwen2.5-Coder 7B	Ruído sem esquema	66,7%	75,0%	0,0%
Qwen2.5-Coder 7B	Ruído com esquema	95,2%	100,0%	100,0%
DeepSeek-Coder 6.7B	Canônico sem esquema	42,9%	37,5%	100,0%
DeepSeek-Coder 6.7B	Canônico com esquema	90,5%	100,0%	100,0%
DeepSeek-Coder 6.7B	Ruído sem esquema	47,6%	50,0%	100,0%
DeepSeek-Coder 6.7B	Ruído com esquema	100,0%	100,0%	100,0%
Llama 3 8B	Canônico sem esquema	52,4%	37,5%	0,0%
Llama 3 8B	Canônico com esquema	100,0%	50,0%	100,0%
Llama 3 8B	Ruído sem esquema	61,9%	25,0%	0,0%
Llama 3 8B	Ruído com esquema	100,0%	75,0%	100,0%

Fonte: Elaborada pelos autores (2026).

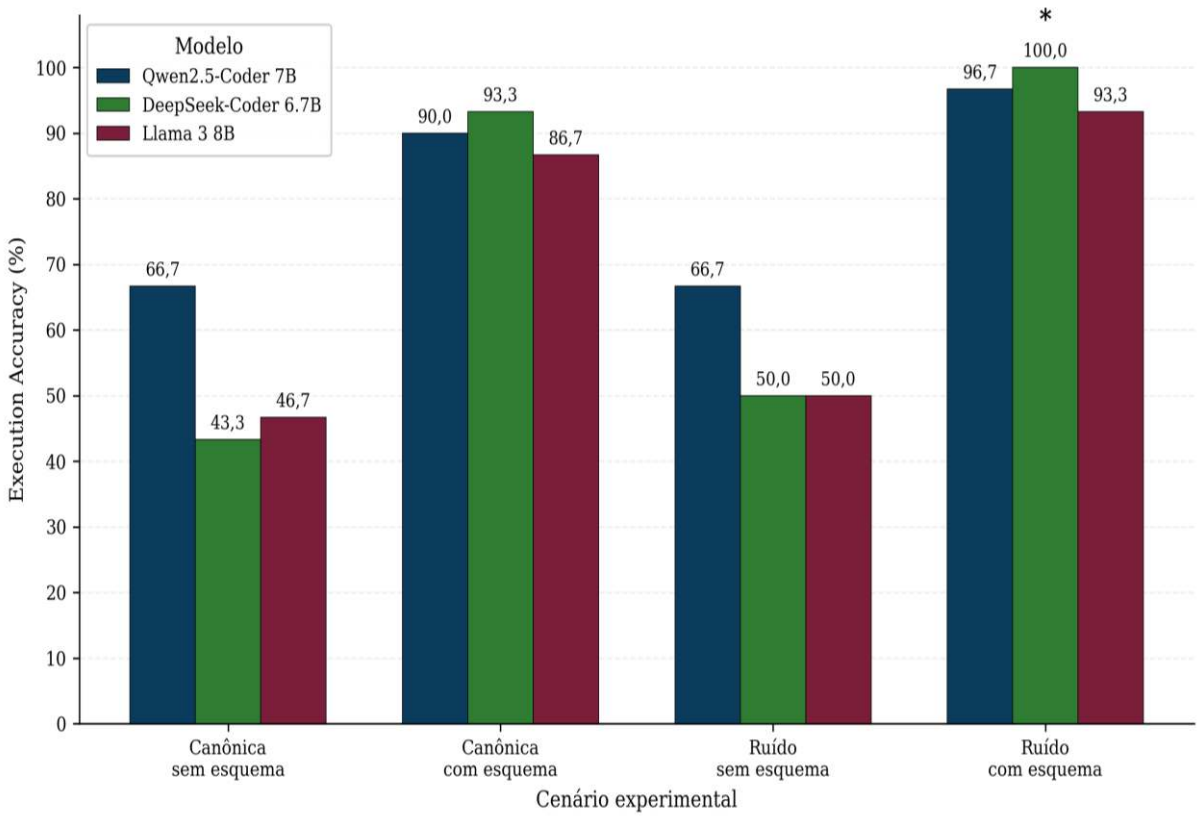
Os resultados da Tabela 9 reforçam que a presença do esquema relacional no *prompt* aumentou a acurácia em praticamente todos os tipos de consulta. O ganho é especialmente visível nas consultas de agregação com junção, que dependem da seleção correta de tabelas e relações. A categoria de subconsulta deve ser interpretada com cautela, pois é composta por apenas uma pergunta; portanto, seus percentuais representam o acerto ou erro em um único caso, não uma tendência estatística robusta.

A Figura 3 evidencia que o fornecimento do esquema foi o fator mais determinante para o desempenho dos modelos no experimento. Em ambos os conjuntos de perguntas, a transição dos cenários sem esquema para os cenários com esquema produziu aumento expressivo da EX, indicando que o acesso explícito à estrutura do banco reduziu ambiguidades relevantes para a tarefa *Text-to-SQL*.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Figura 3 – *Execution Accuracy* dos modelos nos quatro cenários experimentais.



Nota: O asterisco indica o único resultado perfeito do experimento, obtido pelo *DeepSeek-Coder* 6.7B no cenário ruído com esquema.
Fonte: Elaborada pelos autores (2026).

A figura também mostra que esse efeito não ocorreu de forma homogênea entre os modelos. O DeepSeek-Coder 6.7B apresentou maior sensibilidade à presença do esquema, saindo de desempenho baixo nos cenários sem contexto estrutural para o melhor resultado absoluto do estudo no cenário com ruído e com esquema. O Qwen2.5-Coder 7B, embora não tenha atingido o maior valor isolado, apresentou menor variação descritiva entre os cenários, especialmente por manter desempenho superior aos demais modelos nas condições sem esquema. Esse resultado sugere maior tolerância relativa à ausência de de apoio estrutural no *prompt*, dentro das condições controladas deste experimento.

4.3 Impacto do fornecimento do esquema

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

O fornecimento do esquema do banco de dados no *prompt* apresentou impacto expressivo sobre o desempenho dos modelos avaliados. Para essa análise, foram agrupados os dois cenários sem esquema (C1 e C3) e os dois cenários com esquema (C2 e C4), permitindo comparar a média de acurácia obtida quando os modelos recebiam, ou não, informações explícitas sobre tabelas, colunas e relações do banco. Para melhor organização visual, a Tabela 10 utiliza as abreviações SE para os cenários sem esquema e CE para os cenários com esquema.

A Tabela 10 mostra que todos os modelos apresentaram aumento de EX média com o fornecimento do esquema. O maior ganho foi observado no DeepSeek-Coder 6.7B, cuja EX média passou de 46,7% para 96,7%, um acréscimo de 50,0 pontos percentuais. O Llama 3 8B também apresentou ganho elevado, passando de 48,3% para 90,0%, com aumento de 41,7 pontos percentuais. O Qwen2.5-Coder 7B apresentou o menor ganho entre os três modelos, passando de 66,7% para 93,3%, mas foi o modelo com melhor desempenho médio nos cenários sem esquema.

Tabela 10 – Impacto do fornecimento do esquema na acurácia.

Modelo	SE	CE	Dif.	Var. rel.
Qwen2.5-Coder 7B	66,7%	93,3%	+26,7	+40,0%
DeepSeek-Coder 6.7B	46,7%	96,7%	+50,0	+107,2%
Llama 3 8B	48,3%	90,0%	+41,7	+86,2%

Legenda: SE = sem esquema; CE = com esquema; Dif. = diferença em pontos percentuais; Var. rel. = variação relativa.

Fonte: Elaborada pelos autores (2026).

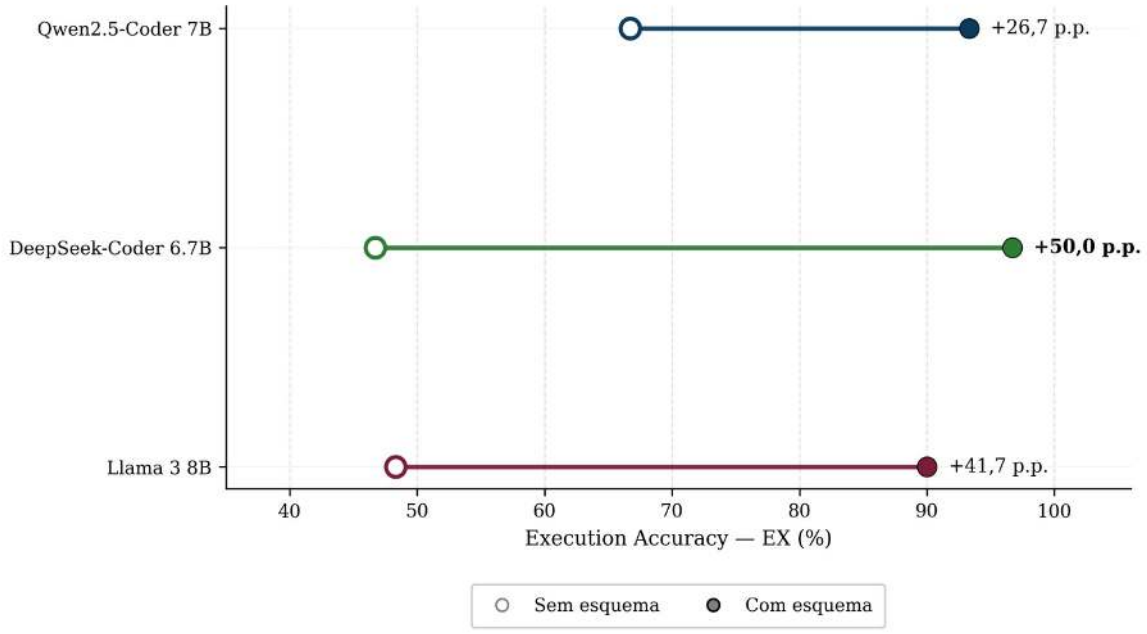
Esse resultado reforça a relevância do contexto estrutural na tarefa *Text-to-SQL*. Ao receber explicitamente o esquema, o modelo passa a operar com informações sobre as tabelas disponíveis, os nomes das colunas e as relações entre entidades, reduzindo a necessidade de inferir a estrutura do banco apenas a partir da pergunta em linguagem natural. Esse comportamento é coerente com a literatura sobre *schema linking*, que aponta o alinhamento entre linguagem natural e esquema relacional como um dos principais desafios da tarefa (Shi *et al.*, 2024; Li *et al.*, 2023).

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

A Figura 4 complementa essa análise ao representar visualmente o deslocamento da EX média entre as condições sem esquema e com esquema. O deslocamento das linhas em direção a valores mais altos indica que a inclusão do contexto estrutural beneficiou todos os modelos avaliados. A magnitude desse deslocamento, entretanto, variou entre os modelos: o DeepSeek-Coder 6.7B apresentou maior dependência do esquema para alcançar desempenho elevado, enquanto o Qwen2.5-Coder 7B partiu de um desempenho inicial mais alto mesmo sem receber a descrição formal do banco. Assim, o fornecimento do esquema não apenas aumentou a acurácia média, mas também alterou descritivamente a relação de desempenho entre os modelos nas condições avaliadas.

Figura 4 – Variação da *Execution* Accuracy com e sem fornecimento do esquema.



Nota: Os valores sem esquema correspondem à média de *canonical_c1* e *noise_c1*; os valores com esquema correspondem à média de *canonical_c2* e *noise_c2*.
Fonte: Elaborada pelos autores (2026).

Além disso, a magnitude dos deslocamentos permite observar diferentes níveis de sensibilidade dos modelos ao fornecimento do esquema. O DeepSeek-Coder 6.7B apresentou o maior ganho de EX, indicando que a disponibilização do contexto estrutural teve papel especialmente relevante para seu desempenho. O

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Qwen2.5-Coder 7B, em contraste, apresentou ganho menor, mas partiu de um desempenho inicial mais elevado nos cenários sem esquema, sugerindo maior capacidade relativa de operar quando a descrição formal do banco não é fornecida. Nesse sentido, a figura evidencia não apenas o efeito positivo do esquema, mas também diferenças na forma como cada modelo se beneficia desse recurso.

4.4 Robustez a variações linguísticas

A análise da robustez a variações linguísticas considerou a comparação entre as perguntas canônicas e suas versões com ruído. Para isso, foram agrupados os cenários sem ruído (C1 e C2) e com ruído linguístico (C3 e C4), mantendo-se os demais fatores experimentais.

A Tabela 11 mostra que as variações linguísticas não reduziram a EX média dos modelos avaliados. Ao contrário, observou-se aumento leve de desempenho em todos os casos: Qwen2.5-Coder 7B passou de 78,4% para 81,7%, DeepSeek-Coder 6.7B passou de 68,3% para 75,0%, e Llama 3 8B passou de 66,7% para 71,7%.

Tabela 11 – Impacto das variações linguísticas na acurácia.

Modelo	SR	CR	Dif.	Var. rel.
Qwen2.5-Coder 7B	78,4%	81,7%	+3,3	+4,3%
DeepSeek-Coder 6.7B	68,3%	75,0%	+6,7	+9,8%
Llama 3 8B	66,7%	71,7%	+5,0	+7,4%

Legenda: SR = sem ruído linguístico; CR = com ruído linguístico; Dif. = diferença em pontos percentuais; Var. rel. = variação relativa.

Fonte: Elaborada pelos autores (2026).

Os resultados devem ser interpretados de forma restrita ao tipo de variação empregado no experimento. As perguntas com ruído foram construídas para preservar a intenção semântica das perguntas canônicas e a mesma consulta SQL de referência. Portanto, os resultados não permitem concluir que os modelos sejam robustos a ruídos semânticos severos, perguntas ambíguas, omissão de informações essenciais ou perturbações adversariais. O achado indica apenas que, neste *benchmark* exploratório e controlado, os modelos não apresentaram degradação média diante de ruído superficial controlado.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Uma explicação plausível é que as perguntas com ruído preservaram a intenção semântica central e mantiveram termos relevantes para o mapeamento entre a pergunta e o banco de dados. Nesse sentido, o efeito das variações linguísticas foi menos determinante do que o fornecimento do esquema, que se mostrou o fator de maior impacto nos resultados. Essa leitura é coerente com estudos que discutem a influência de perturbações linguísticas em sistemas *Text-to-SQL*, nos quais alterações superficiais tendem a produzir efeitos diferentes de ruídos semânticos mais severos (Wretblad *et al*., 2024; Pi *et al*., 2022).

4.5 Análise das categorias de erro

Além da acurácia por execução, foram analisadas as categorias de erro registradas durante o processo de validação e avaliação das consultas. Essa análise permite diferenciar falhas estruturais, relacionadas à formação ou execução da consulta SQL, de falhas semânticas, nas quais a consulta é executada, mas retorna resultado diferente da referência.

A Tabela 12 apresenta a distribuição das falhas por modelo. O Qwen2.5-Coder 7B apresentou o menor número total de erros, com 24 falhas em 120 consultas. O DeepSeek-Coder 6.7B registrou 34 falhas, com maior concentração em erros de execução. O Llama 3 8B apresentou 37 falhas, sendo o modelo com maior número de resultados incorretos.

Tabela 12 – Distribuição das categorias de erro por modelo.

Modelo	VP	ES	EE	TO	RI	Total
Qwen2.5-Coder 7B	0	0	11	0	13	24
DeepSeek-Coder 6.7B	2	1	20	0	11	34
Llama 3 8B	0	0	16	0	21	37
Total	2	1	47	0	45	95

Legenda: VP = violação de política; ES = erro sintático; EE = erro de execução; TO = tempo limite; RI = resultado incorreto.

Fonte: Elaborada pelos autores (2026).

Considerando as categorias de erro apresentadas na Tabela 12, observa-se que 50 falhas foram estruturais, somando violações de política, erros sintáticos e erros de execução, enquanto 45 foram classificadas como resultado incorreto. Esse equilíbrio indica que o problema não se limita à geração de consultas SQL

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

executáveis. Mesmo quando a consulta não apresenta erro de sintaxe ou execução, ainda pode falhar no alinhamento entre a intenção da pergunta e o resultado esperado. Além da distribuição por categoria, os erros também foram analisados segundo os cenários experimentais. Essa análise permite observar em quais condições as falhas se concentraram e complementa a interpretação da Tabela 12, que apresenta os tipos de erro por modelo. A Tabela 13 mostra o total de falhas em cada cenário, considerando que cada modelo respondeu a 30 perguntas por condição experimental.

Tabela 13 – Distribuição de erros por modelo e cenário.

Modelo	C1	C2	C3	C4	Total
Qwen2.5-Coder 7B	10	3	10	1	24
DeepSeek-Coder 6.7B	17	2	15	0	34
Llama 3 8B	16	4	15	2	37
Total	**43**	**9**	**40**	**3**	**95**

Legenda: C1 = canônica sem esquema; C2 = canônica com esquema; C3 = ruído sem esquema; C4 = ruído com esquema.

Fonte: Elaborada pelos autores (2026).

A distribuição apresentada na Tabela 13 reforça que a maior parte das falhas ocorreu nos cenários sem fornecimento do esquema. Nos cenários C1 e C3, foram registradas 43 e 40 falhas, respectivamente, enquanto nos cenários com esquema, C2 e C4, os totais foram substancialmente menores, com 9 e 3 falhas. Esse padrão indica que a ausência de contexto estrutural ampliou erros de mapeamento entre a pergunta e os elementos do banco, como tabelas, colunas e relações. Também se observa que o Qwen2.5-Coder 7B apresentou menor número total de falhas, enquanto o DeepSeek-Coder 6.7B eliminou os erros no cenário C4, em que as perguntas com ruído foram acompanhadas do esquema relacional.

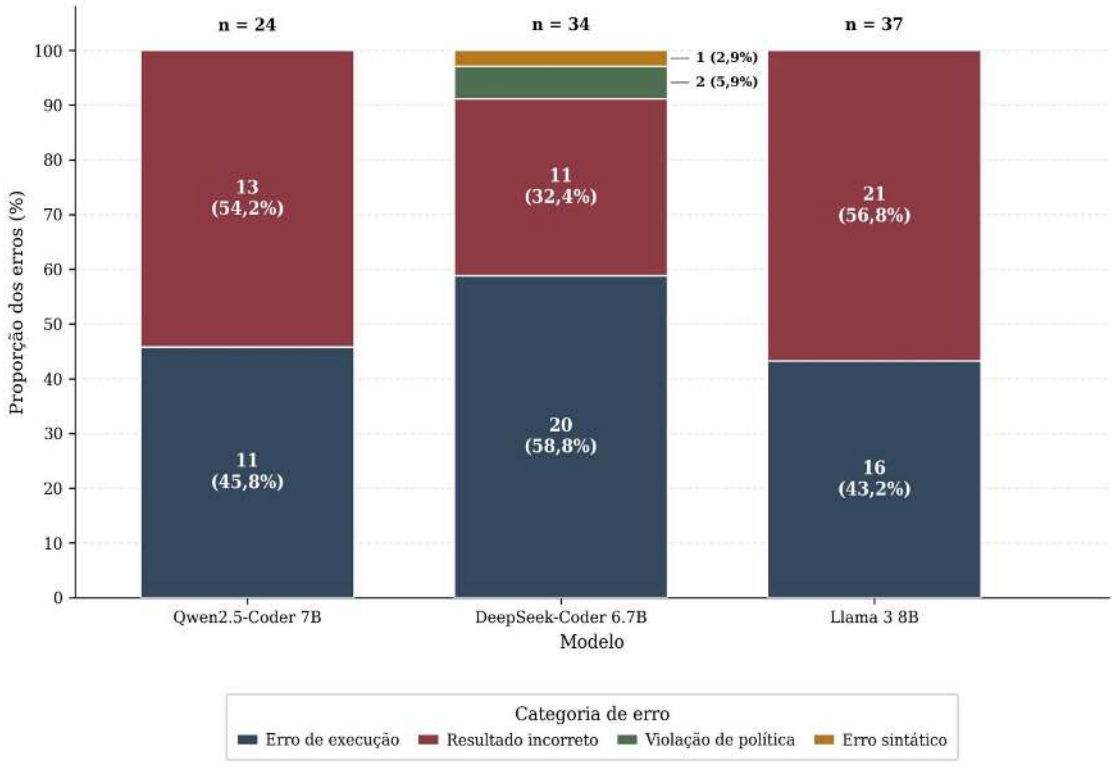
As diferenças entre os modelos também são relevantes, no DeepSeek-Coder 6.7B, predominam falhas estruturais, especialmente erros de execução, o que é compatível com seu desempenho mais baixo nos cenários sem fornecimento do esquema. No Llama 3 8B, a maior parte dos erros concentra-se em resultados incorretos, sugerindo maior incidência de consultas válidas do ponto de vista operacional, mas semanticamente desalinhadas. O Qwen2.5-Coder 7B apresentou

REMUNOM
ISSN 2178-6925
Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

o perfil mais estável, com menor volume total de falhas e distribuição mais equilibrada entre erros estruturais e semânticos.

Essa distinção é importante porque, em *Text-to-SQL*, a validade sintática da consulta não garante sua correção semântica. Uma consulta pode ser executada sem erro e ainda assim responder incorretamente à pergunta formulada, limitação também discutida na literatura sobre avaliação por execução e confiabilidade de métricas como EX (Shi *et al.*, 2024; Kim *et al.*, 2024). A Figura 5 mostra que a diferença entre os modelos não se expressa apenas pela quantidade total de falhas, mas também pelo tipo de erro predominante. O Qwen2.5-Coder 7B apresentou o menor volume total de erros, com distribuição relativamente equilibrada entre falhas estruturais e semânticas, sugerindo maior estabilidade do descritiva do conjunto avaliado.

Figura 5 – Distribuição percentual das categorias de erro por modelo.



Nota: Não foram observados *timeouts* nas execuções avaliada.
Fonte: Elaborada pelos autores (2026).

O DeepSeek-Coder 6.7B concentrou principalmente erros estruturais, sobretudo erros de execução, o que é consistente com sua maior sensibilidade à

REMUNOM
ISSN 2178-6925
Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

ausência do esquema. As duas violações de política ocorreram apenas no DeepSeek-Coder 6.7B, em cenários sem esquema. Elas não corresponderam a comandos de escrita ou modificação do banco, mas a bloqueios preventivos do validador devido ao artefato *t< ｜begin_of_sentence｜ >: 10*, classificado como *blocked_token:begin*.

Além das violações de política, foram observados erros de execução e resultados incorretos. Um exemplo de erro de execução ocorreu quando o modelo gerou consulta referenciando colunas inexistentes, como uf_residencia e uf_tratamento, em vez dos atributos efetivamente presentes no banco, uf_resid e uf_trat. Nesse caso, a consulta foi sintaticamente formada, mas falhou ao ser executada por não encontrar as colunas indicadas. Já um exemplo de resultado incorreto ocorreu na pergunta sobre o tempo mediano até tratamento, em que a resposta gerada calculou média por meio de AVG(tempo_trat), enquanto a consulta de referência implementava a mediana com ordenação, limite e deslocamento. Esse caso ilustra uma consulta executável, mas semanticamente desalinhada em relação à pergunta.

Já o Llama 3 8B apresentou predominância de resultados incorretos, indicando tendência a gerar consultas executáveis, porém semanticamente inadequadas. Esse contraste é importante porque mostra que modelos com desempenho global semelhante podem falhar por razões distintas, o que tem implicações diferentes para seu uso em aplicações reais.

4.6 Análise de latência de geração e execução

Além da acurácia, foram analisadas as medidas de latência associadas ao processo experimental. A latência de geração corresponde ao tempo necessário para que o modelo produza a resposta a partir do *prompt*, enquanto a latência de execução corresponde ao tempo utilizado para executar a consulta no banco SQLite.

A Tabela 14 apresenta as estatísticas agregadas por modelo. Observa-se que o Qwen2.5-Coder 7B apresentou a menor latência média de geração, com 12,26 segundos por consulta. O DeepSeek-Coder 6.7B apresentou tempo médio intermediário, com 29,62 segundos, enquanto o Llama 3 8B foi o modelo mais

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

lento, com média de 41,96 segundos. Os resultados indicam que o custo temporal do *pipeline* esteve concentrado principalmente na etapa de geração pela LLM. Em todos os modelos, a latência média de execução no SQLite permaneceu inferior a 0,27 segundo, indicando que a execução das consultas teve impacto reduzido no tempo total do experimento.

Tabela 14 – Latência agregada por modelo.

Modelo	GM	Gp50	Gp95	Gmax	EM
Qwen2.5-Coder 7B	12,26	8,62	19,03	188,29	0,147
DeepSeek-Coder 6.7B	29,62	20,77	68,28	477,41	0,261
Llama 3 8B	41,96	31,97	78,13	565,97	0,260

Legenda: GM = geração média (s); Gp50 = mediana da geração (s); Gp95 = percentil 95 da geração (s); Gmax = maior latência de geração (s); EM = execução média (s).

Fonte: Elaborada pelos autores (2026).

A comparação entre acurácia e latência também evidencia diferenças práticas entre os modelos. O Qwen2.5-Coder 7B combinou a maior EX geral com a menor latência média de geração, indicando melhor relação entre acurácia e custo temporal no conjunto avaliado. Por outro lado, DeepSeek-Coder 6.7B e Llama 3 8B apresentaram maior variabilidade temporal, refletida nos valores de Gp95 e Gmax.

Além da análise agregada por modelo, a latência de geração também foi analisada por cenário experimental. Essa separação permite observar o efeito do fornecimento do esquema e da presença de variações linguísticas sobre o tempo de resposta dos modelos. A Tabela 15 apresenta a média, a mediana, o percentil 95 e o valor máximo de latência para cada combinação de modelo e cenário.

Tabela 15 – Latência de geração por modelo e cenário, em segundos.

Modelo	Cenário	Média	Mediana	p95	Máximo
Qwen2.5-Coder 7B	Canônico sem esquema	8,50	6,19	14,26	59,44
Qwen2.5-Coder 7B	Canônico com esquema	14,80	9,61	18,38	146,61
Qwen2.5-Coder 7B	Ruído sem esquema	8,28	7,34	13,61	22,29
Qwen2.5-Coder 7B	Ruído com esquema	17,48	10,66	20,26	188,29
DeepSeek-Coder 6.7B	Canônico sem esquema	20,25	12,94	56,24	72,77
DeepSeek-Coder 6.7B	Canônico com esquema	23,26	23,68	34,99	44,62
DeepSeek-Coder 6.7B	Ruído sem esquema	11,90	9,49	21,16	28,74
DeepSeek-Coder 6.7B	Ruído com esquema	63,07	52,74	85,13	477,41
Llama 3 8B	Canônico sem esquema	28,77	22,41	44,78	181,64
Llama 3 8B	Canônico com esquema	41,42	38,74	67,93	70,78
Llama 3 8B	Ruído sem esquema	29,46	23,59	52,43	129,13

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Llama 3 8B	Ruído com esquema	68,18	45,98	89,53	565,97

Fonte: Elaborada pelos autores (2026).

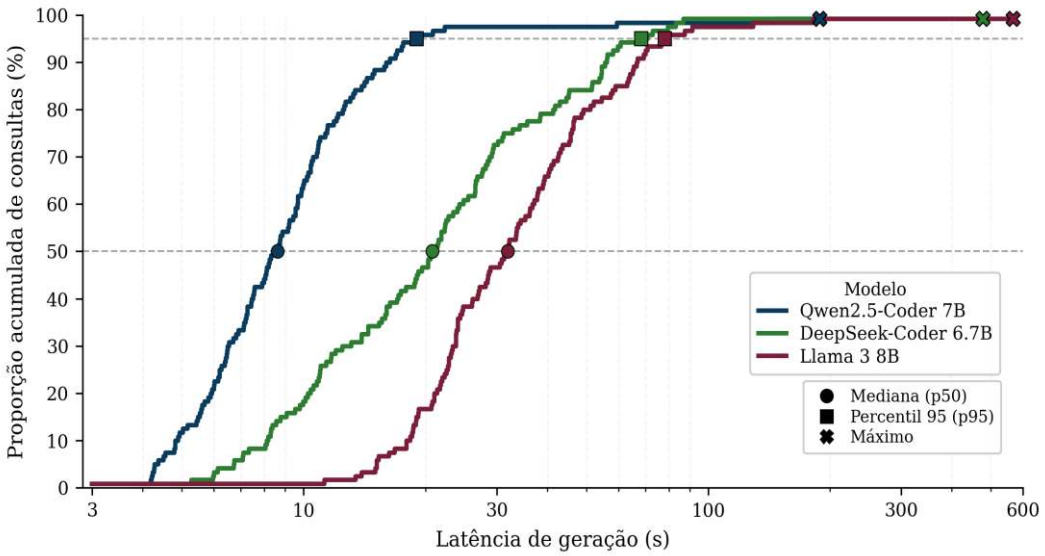
Os resultados mostram que os cenários com esquema tenderam a apresentar maior latência de geração, especialmente em ruído com esquema, que combina variação linguística e fornecimento do esquema. Esse aumento é esperado, pois o *prompt* com esquema contém maior volume de contexto textual, incluindo tabelas, colunas e relações do banco. O Qwen2.5-Coder 7B apresentou as menores medianas em todos os cenários, variando de 6,19 s a 10,66 s. O DeepSeek-Coder 6.7B e o Llama 3 8B apresentaram aumento mais expressivo nos cenários com esquema, com máximos elevados em ruído com esquema, respectivamente 477,41 s e 565,97 s. Assim, embora o fornecimento do esquema tenha aumentado a acurácia, ele também ampliou o custo temporal de geração, sobretudo nos modelos com maior latência agregada.

A latência de execução deve ser interpretada com cautela, pois é calculada apenas para consultas que chegaram à etapa de execução no banco. Assim, consultas bloqueadas na validação ou com erro sintático reduzem a quantidade de observações consideradas nessa medida. Por esse motivo, a latência de geração constitui a medida mais estável para comparação temporal entre os modelos. A Figura 6 evidencia diferenças descritivas no comportamento temporal dos modelos avaliados.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Figura 6– Curvas ECDF da latência de geração por modelo.



Nota: As curvas foram construídas a partir dos valores brutos de *latency_s* extraídos dos arquivos *log. jsonl*. O eixo horizontal está em escala logarítmica. As marcações indicam mediana, percentil 95 e valor máximo.

Fonte: Elaborada pelos Autores (2026).

A curva do Qwen2.5-Coder 7B, mais deslocada à esquerda e com subida mais rápida, indica menor latência de geração e maior concentração das respostas em tempos menores na maior parte das consultas. Isso mostra que sua maior EX agregada não esteve associada a maior tempo de resposta, mas a um comportamento temporal mais favorável no ambiente experimental avaliado.

O DeepSeek-Coder 6.7B ocupa posição intermediária, enquanto o Llama 3 8B apresenta distribuição mais deslocada à direita e maior dispersão, indicando tempos de geração mais elevados e maior variabilidade temporal. Assim, a figura reforça que a comparação entre modelos não deve se limitar à acurácia final: o custo temporal e a estabilidade da geração também constituem dimensões relevantes para avaliar sistemas *Text-to-SQL* em cenários aplicados.

As diferenças de latência observadas devem ser interpretadas considerando tanto os modelos quanto o ambiente local de execução. Os três modelos foram executados em versões quantizadas: Qwen2.5-Coder 7B em Q4_K_M, DeepSeek-Coder 6.7B em Q4_0 e Llama 3 8B em Q4_0. A quantização reduz o custo de

REMUNOM
ISSN 2178-6925
Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

memória e viabiliza a execução local de modelos maiores, mas a latência final também depende da arquitetura do modelo, do tamanho da janela de contexto, do comprimento do *prompt*, do *backend* de inferência e das características do hardware utilizado.

Nos cenários com esquema, o aumento do *prompt*, causado pela inclusão das tabelas, colunas e chaves estrangeiras do banco, tende a elevar o tempo de geração. Esse efeito foi observado nos três modelos, especialmente nos cenários com esquema e ruído, nos quais ocorreram os maiores valores máximos de latência. Assim, os resultados de latência não devem ser interpretados como propriedades absolutas dos modelos, mas como medidas obtidas sob a configuração local utilizada neste experimento. Em outros ambientes de hardware, versões de quantização ou configurações de *runtime*, os tempos podem variar.

5. Conclusões

Este trabalho estabeleceu um *benchmark* experimental para avaliar modelos de linguagem de código aberto na tarefa *Text-to-SQL*, utilizando perguntas em português brasileiro e dados públicos de atendimentos oncológicos no Brasil. Foram comparados três modelos executados localmente via Ollama, sendo eles: Qwen2.5-Coder 7B, DeepSeek-Coder 6.7B e Llama 3 8B, em quatro cenários experimentais definidos pela presença ou ausência do esquema do banco de dados e pela inclusão de variações linguísticas controladas.

Os resultados indicaram que o fornecimento do esquema relacional foi o fator de maior impacto sobre a acurácia dos modelos. Em todos os casos, a inclusão de informações sobre tabelas, colunas e relações elevou a Execution Accuracy (EX), reforçando a importância do contexto estrutural para a geração correta de consultas SQL em bancos relacionais com múltiplas tabelas. Entretanto, esse ganho também esteve associado ao aumento da latência de geração, especialmente nos cenários com esquema, evidenciando uma relação de compromisso entre desempenho preditivo e custo temporal.

Em termos descritivos, o Qwen2.5-Coder 7B apresentou a maior EX

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

agregada e as menores latências médias entre os modelos avaliados, indicando melhor equilíbrio entre acurácia e tempo de geração no ambiente experimental considerado. No entanto, as comparações pareadas não sustentaram uma conclusão estatisticamente robusta de superioridade geral de um modelo sobre os demais. Assim, as diferenças observadas devem ser interpretadas como evidências exploratórias, condicionadas ao conjunto avaliativo, aos cenários definidos e à configuração local de execução.

Quanto às variações linguísticas, os resultados sugerem que o ruído superficial controlado introduzido nas perguntas não degradou o desempenho médio dos modelos. Esse achado, porém, deve ser interpretado de forma restrita, pois as variações preservaram a intenção semântica das perguntas e não envolveram ambiguidades severas, omissão de informações essenciais, mudanças de escopo ou perturbações capazes de alterar a consulta SQL de referência esperada.

Também é importante destacar que o uso de dados oncológicos neste estudo se refere ao domínio tabular do *benchmark*, e não à validação clínica de diagnósticos, tratamentos ou decisões assistenciais. As consultas geradas foram avaliadas apenas quanto à correspondência de execução em um banco relacional controlado. Portanto, os resultados não devem ser interpretados como evidência de uso clínico direto de modelos de linguagem em oncologia.

Como limitações, destacam-se o uso de um único banco de dados, a escala reduzida do conjunto avaliativo, a realização de uma única execução por combinação de modelo, pergunta e cenário, e o foco em variações linguísticas superficiais. Como trabalhos futuros, sugere-se ampliar o conjunto de perguntas, incluir consultas com maior complexidade estrutural, explorar múltiplos domínios de dados, avaliar novas versões de modelos especializados em código e SQL, realizar múltiplas execuções por pergunta e investigar estratégias automáticas de correção iterativa de consultas.

Referências

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

DOU, L. et al. Multispider: towards benchmarking multilingual text-to-SQL semantic parsing. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 37., 2023, Washington, DC. **Proceedings** [...]. Washington, DC: AAAI Press, 2023. v. 37, n. 10, p. 12745–12753. Disponível em: https://doi.org/10.1609/aaai.v37i10.26367. Acesso em: 26 jan. 2026.

GUO, D. et al. **DeepSeek-Coder: when the large language model meets programming – the rise of code intelligence**. arXiv preprint arXiv:2401.14196, 2024. Disponível em: https://arxiv.org/abs/2401.14196. Acesso em: 26 jan. 2026.

JOSÉ, M. A.; COZMAN, F. G. mRAT-SQL+GAP: a Portuguese text-to-SQL transformer. In: BRITTO, A.; VALDIVIA DELGADO, K. (ed.). **Intelligent systems**. Cham: Springer International Publishing, 2021. p. 511–525.

KIM, H. et al. FLEX: expert-level false-less execution metric for reliable text-to-SQL benchmark. In: CONFERENCE OF THE NATIONS OF THE AMERICAS CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES, 2025, Albuquerque. **Proceedings** [...]. Albuquerque: Association for Computational Linguistics, 2025. p. 4448–4475. Disponível em: https://aclanthology.org/2025.naacl-long.228. Acesso em: 26 jan. 2026.

LI, J. et al. Can LLM already serve as a database interface? A BIg bench for large-scale database grounded text-to-SQLs. In: CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 37., 2023, New Orleans. **Proceedings**[...]. New Orleans: Curran Associates, 2023. Disponível em: https://openreview.net/forum?id=dI4wzAE6uV. Acesso em: 26 jan. 2026.

MARCONI, M. A.; LAKATOS, E. M. **Fundamentos de metodologia científica**. 5. ed. São Paulo: Atlas, 2003. ISBN 85-224-3397-6.

OLLAMA. **Ollama documentation**. 2025. Disponível em: https://github.com/ollama/ollama. Acesso em: 15 jan. 2026.

PHAM, K. T. et al. Multilingual text-to-SQL: benchmarking the limits of language models with collaborative language agents. In: BOROVICA-GAJIC, R. et al. (ed.). **Databases theory and applications**. Singapore: Springer Nature Singapore, 2026. p. 108–123. ISBN 978-981-95-6196-4. Disponível em: https://doi.org/10.1007/978-981-95-6196-4_8. Acesso em: 26 jan. 2026.

PI, X. et al. Towards robustness of text-to-SQL models against natural and realistic adversarial table perturbation. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 60., 2022, Dublin. **Proceedings** [...]. Dublin: Association for Computational Linguistics, 2022. p. 2007–2022. Disponível em: https://aclanthology.org/2022.acl-long.142/. Acesso em: 26 jan. 2026.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

SHI, L. et al. A survey on employing large language models for text-to-SQL tasks. **ACM Computing Surveys**, New York, v. 58, n. 2, p. 1–37, set. 2025. ISSN 0360-0300. Disponível em: https://doi.org/10.1145/3737873. Acesso em: 26 jan. 2026.

TENORIO, L. **Oncology treatment DataSUS – Brazil (2013–2023)**. Kaggle, 2025. Disponível em: https://www.kaggle.com/datasets/lhucastenorio/oncology-treatment-datasus-brazil-20132023. Acesso em: 26 jan. 2026.

WOHLIN, C. et al. **Experimentation in software engineering**. Berlin; Heidelberg: Springer, 2012. ISBN 978-3-642-29044-2. Disponível em: https://doi.org/10.1007/978-3-642-29044-2. Acesso em: 26 jan. 2026.

WRETBLAD, N. et al. Understanding the effects of noise in text-to-SQL: an examination of the BIRD-bench benchmark. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 62., 2024, Bangkok. **Proceedings** [...]. Bangkok: Association for Computational Linguistics, 2024. p. 356–369. Disponível em: https://aclanthology.org/2024.acl-short.34. Acesso em: 26 jan. 2026.

YU, T. et al. Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In: CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, 2018, Brussels. **Proceedings** [...]. Brussels: Association for Computational Linguistics, 2018. p. 3911–3921. Disponível em: https://aclanthology.org/D18-1425/. Acesso em: 26 jan. 2026.

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Apêndice A – Perguntas canônicas do benchmark

Este apêndice apresenta as 30 perguntas canônicas utilizadas no benchmark. Cada pergunta corresponde à formulação-base em português brasileiro, sem ruído linguístico, e está associada a uma consulta SQL de referência apresentada no Apêndice B.

ID	Pergunta canônica	Tipo de consulta	Dificuldade
br_0001	Quantos registros existem no conjunto de dados?	Agregação	Simples
br_0002	Quantos registros há por sexo?	Agregação	Intermediária
br_0003	Qual é a idade média dos pacientes por sexo?	Agregação	Intermediária
br_0004	Quais são os 10 CIDs (cid10) mais frequentes no conjunto de dados?	Agregação	Intermediária
br_0005	Quantos registros existem por ano de diagnóstico (dim_tempo.ano)?	Agregação com junção	Intermediária
br_0006	Quantos registros existem por mês de diagnóstico (dim_tempo.ano e dim_tempo.mes)?	Agregação com junção	Intermediária
br_0007	Qual é o tempo médio até tratamento (tempo_trat) por ano de diagnóstico?	Agregação com junção	Complexa
br_0008	Qual é o tempo mediano até tratamento (tempo_trat) no conjunto de dados?	Subconsulta	Complexa
br_0009	Quantos registros possuem tempo_trat preenchido (não nulo)?	Agregação	Intermediária
br_0010	Quais são as 10 UFs de residência (uf_resid) com mais registros?	Agregação	Complexa
br_0011	Quais são os 10 municípios de residência (mun_resid) com mais registros?	Agregação	Complexa
br_0012	Quais são as 10 UFs	Agregação	Complexa

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

ID	Pergunta canônica	Tipo de consulta	Dificuldade
	de tratamento (uf_trat) com mais registros?		
br_0013	Qual é o total de registros em que a UF de residência é diferente da UF de tratamento?	Agregação	Intermediária
br_0014	Para cada UF de residência, qual é a proporção de casos cujo tratamento ocorreu na mesma UF?	Agregação	Complexa
br_0015	Quantos registros existem por tipo de tratamento (tratamento_id)?	Agregação	Intermediária
br_0016	Qual é a idade média por tipo de tratamento (tratamento_id)?	Agregação	Intermediária
br_0017	Quantos registros existem por estadiamento (estadiam_id)?	Agregação	Intermediária
br_0018	Qual é o tempo médio até tratamento por estadiamento (estadiam_id) considerando apenas tempo_trat não nulo?	Agregação	Intermediária
br_0019	Quantos registros existem por diagnostic_id (tipo de diagnóstico)?	Agregação	Intermediária
br_0020	Quais são os 10 CNES de diagnóstico (cnes_diagn) com mais registros?	Agregação	Complexa
br_0021	Quais são os 10 CNES de tratamento (cnes_trat) com mais registros?	Agregação	Complexa
br_0022	Qual é a quantidade de registros por sexo e ano de diagnóstico?	Agregação com junção	Complexa
br_0023	Entre 2018 e 2023, qual foi o total de registros por ano de diagnóstico?	Agregação com junção	Complexa
br_0024	No ano de 2020, quais são os 10 CIDs (cid10)	Agregação com junção	Complexa

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

ID	Pergunta canônica	Tipo de consulta	Dificuldade
	mais frequentes?		
br_0025	No ano de 2020, qual é o total de registros por sexo?	Agregação com junção	Intermediária
br_0026	Qual é o tempo médio até tratamento por sexo no ano de 2020 (somente tempo_trat não nulo)?	Agregação com junção	Complexa
br_0027	Quantos registros possuem município de residência preenchido (mun_resid não nulo)?	Agregação	Intermediária
br_0028	Qual é a quantidade de registros cuja UF de diagnóstico é diferente da UF de residência?	Agregação	Intermediária
br_0029	Qual é a quantidade de registros cujo município de tratamento é diferente do município de residência (ambos não nulos)?	Agregação	Intermediária
br_0030	Qual é a idade mínima, máxima e média no conjunto de dados?	Agregação	Intermediária

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Apêndice B – Consultas SQL de referência

Este apêndice apresenta as consultas SQL de referência, também denominadas gold SQL, utilizadas para avaliar as respostas geradas pelos modelos. Cada consulta está associada ao mesmo identificador da pergunta canônica apresentada no Apêndice A.

br_0001

```
SELECT COUNT(*) FROM fact_onco
```

br_0002

```
SELECT sexo, COUNT(*) AS n FROM fact_onco GROUP BY sexo ORDER BY n DESC
```

br_0003

```
SELECT sexo, AVG(idade) AS idade_media FROM fact_onco WHERE idade IS NOT NULL GROUP BY sexo ORDER BY idade_media DESC
```

br_0004

```
SELECT cid10, COUNT(*) AS n FROM fact_onco GROUP BY cid10 ORDER BY n DESC LIMIT 10
```

br_0005

```
SELECT t.ano, COUNT(*) AS n FROM fact_onco f JOIN dim_tempo t ON f.anomes_dia = t.anomes GROUP BY t.ano ORDER BY t.ano
```

br_0006

```
SELECT t.ano, t.mes, COUNT(*) AS n FROM fact_onco f JOIN dim_tempo t ON f.anomes_dia = t.anomes GROUP BY t.ano, t.mes ORDER BY t.ano, t.mes
```

br_0007

```
SELECT t.ano, AVG(f.tempo_trat) AS tempo_medio FROM fact_onco f JOIN dim_tempo t ON f.anomes_dia = t.anomes WHERE f.tempo_trat IS NOT NULL GROUP BY t.ano ORDER BY t.ano
```

br_0008

```
SELECT AVG(tempo_trat) FROM (SELECT tempo_trat FROM fact_onco WHERE tempo_trat IS NOT NULL ORDER BY tempo_trat LIMIT 2 - (SELECT COUNT(*)
```

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

```
FROM fact_onco WHERE tempo_trat IS NOT NULL) % 2 OFFSET (SELECT
(COUNT(*) - 1) / 2 FROM fact_onco WHERE tempo_trat IS NOT NULL))
br_0009
SELECT COUNT(*) FROM fact_onco WHERE tempo_trat IS NOT NULL
br_0010
SELECT uf_resid, COUNT(*) AS n FROM fact_onco WHERE uf_resid IS NOT
NULL GROUP BY uf_resid ORDER BY n DESC LIMIT 10
br_0011
SELECT mun_resid, COUNT(*) AS n FROM fact_onco WHERE mun_resid IS NOT
NULL GROUP BY mun_resid ORDER BY n DESC LIMIT 10
br_0012
SELECT uf_trat, COUNT(*) AS n FROM fact_onco WHERE uf_trat IS NOT NULL
GROUP BY uf_trat ORDER BY n DESC LIMIT 10
br_0013
SELECT COUNT(*) FROM fact_onco WHERE uf_resid IS NOT NULL AND uf_trat
IS NOT NULL AND uf_resid <> uf_trat
br_0014
SELECT uf_resid, CAST(SUM(CASE WHEN uf_trat = uf_resid THEN 1 ELSE 0
END) AS REAL) / COUNT(*) AS proporcao_mesma_uf FROM fact_onco WHERE
uf_resid IS NOT NULL AND uf_trat IS NOT NULL GROUP BY uf_resid ORDER BY
uf_resid
br_0015
SELECT tratamento_id, COUNT(*) AS n FROM fact_onco WHERE tratamento_id
IS NOT NULL GROUP BY tratamento_id ORDER BY n DESC
br_0016
SELECT tratamento_id, AVG(idade) AS idade_media FROM fact_onco WHERE
tratamento_id IS NOT NULL AND idade IS NOT NULL GROUP BY tratamento_id
ORDER BY idade_media DESC
br_0017
SELECT estadiam_id, COUNT(*) AS n FROM fact_onco WHERE estadiam_id IS
NOT NULL GROUP BY estadiam_id ORDER BY n DESC
```

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

br_0018

```
SELECT estadiam_id, AVG(tempo_trat) AS tempo_medio FROM fact_onco
WHERE estadiam_id IS NOT NULL AND tempo_trat IS NOT NULL GROUP BY
estadiam_id ORDER BY estadiam_id
```

br_0019

```
SELECT diagnostic_id, COUNT(*) AS n FROM fact_onco WHERE diagnostic_id IS
NOT NULL GROUP BY diagnostic_id ORDER BY n DESC
```

br_0020

```
SELECT cnes_diagn, COUNT(*) AS n FROM fact_onco WHERE cnes_diagn IS
NOT NULL GROUP BY cnes_diagn ORDER BY n DESC LIMIT 10
```

br_0021

```
SELECT cnes_trat, COUNT(*) AS n FROM fact_onco WHERE cnes_trat IS NOT
NULL GROUP BY cnes_trat ORDER BY n DESC LIMIT 10
```

br_0022

```
SELECT f.sexo, t.ano, COUNT(*) AS n FROM fact_onco f JOIN dim_tempo t ON
f.anomes_dia = t.anomes WHERE f.sexo IS NOT NULL GROUP BY f.sexo, t.ano
ORDER BY t.ano, f.sexo
```

br_0023

```
SELECT t.ano, COUNT(*) AS n FROM fact_onco f JOIN dim_tempo t ON
f.anomes_dia = t.anomes WHERE t.ano BETWEEN 2018 AND 2023 GROUP BY
t.ano ORDER BY t.ano
```

br_0024

```
SELECT f.cid10, COUNT(*) AS n FROM fact_onco f JOIN dim_tempo t ON
f.anomes_dia = t.anomes WHERE t.ano = 2020 GROUP BY f.cid10 ORDER BY n
DESC LIMIT 10
```

br_0025

```
SELECT f.sexo, COUNT(*) AS n FROM fact_onco f JOIN dim_tempo t ON
f.anomes_dia = t.anomes WHERE t.ano = 2020 GROUP BY f.sexo ORDER BY n
DESC
```

br_0026

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

```
SELECT f.sexo, AVG(f.tempo_trat) AS tempo_medio FROM fact_onco f JOIN
dim_tempo t ON f.anomes_dia = t.anomes WHERE t.ano = 2020 AND f.tempo_trat
IS NOT NULL GROUP BY f.sexo ORDER BY tempo_medio DESC
br_0027
SELECT COUNT(*) FROM fact_onco WHERE mun_resid IS NOT NULL
br_0028
SELECT COUNT(*) FROM fact_onco WHERE uf_diagn IS NOT NULL AND
uf_resid IS NOT NULL AND uf_diagn <> uf_resid
br_0029
SELECT COUNT(*) FROM fact_onco WHERE mun_trat IS NOT NULL AND
mun_resid IS NOT NULL AND mun_trat <> mun_resid
br_0030
SELECT MIN(idade) AS idade_min, MAX(idade) AS idade_max, AVG(idade) AS
idade_media FROM fact_onco WHERE idade IS NOT NULL
```

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Apêndice C – Variações linguísticas controladas

Este apêndice apresenta as versões com ruído das perguntas canônicas. As variações foram construídas de forma controlada, preservando a intenção semântica original e mantendo a mesma consulta SQL de referência associada à pergunta canônica correspondente.

ID	Pergunta canônica	Pergunta com ruído	Tipo de variação
br_0001	Quantos registros existem no conjunto de dados?	Qtos registros existem no conjunto de dados?	Abreviação lexical
br_0002	Quantos registros há por sexo?	Qtd de registros por sexo?	Abreviação lexical; simplificação sintática
br_0003	Qual é a idade média dos pacientes por sexo?	Qual a idade média dos pacintes por sexo?	Erro ortográfico controlado; simplificação sintática
br_0004	Quais são os 10 CIDs (cid10) mais frequentes no conjunto de dados?	Quais são os 10 CIDs (cid10) mais frequentes nos dados?	Simplificação sintática
br_0005	Quantos registros existem por ano de diagnóstico (dim_tempo.ano)?	Quantos registros existem por ano de diag (dim_tempo.ano)?	Abreviação lexical
br_0006	Quantos registros existem por mês de diagnóstico (dim_tempo.ano e dim_tempo.mes)?	Qtd de registros por mês de diagnóstico (dim_tempo.ano e dim_tempo.mes)?	Abreviação lexical; simplificação sintática
br_0007	Qual é o tempo médio até tratamento (tempo_trat) por ano de diagnóstico?	Qual o tempo médio até tratamnto (tempo_trat) por ano de diagnóstico?	Erro ortográfico controlado; simplificação sintática
br_0008	Qual é o tempo mediano até tratamento (tempo_trat) no conjunto de dados?	Qual o tempo mediano até tratamnto (tempo_trat) no conjunto de dados?	Erro ortográfico controlado; simplificação sintática
br_0009	Quantos registros possuem tempo_trat preenchido (não nulo)?	Quantos registros têm tempo_trat preenchido (não nulo)?	Substituição lexical superficial
br_0010	Quais são as 10 UFs de residência (uf_resid) com mais registros?	Top 10 UFs de residência (uf_resid) com mais registros?	Reformulação coloquial; simplificação sintática
br_0011	Quais são os 10 municípios de residência (mun_resid) com mais registros?	Quais os 10 munic de residência (mun_resid) com mais registros?	Abreviação lexical; simplificação sintática
br_0012	Quais são as 10 UFs	Top 10 UFs de	Reformulação

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

ID	Pergunta canônica	Pergunta com ruído	Tipo de variação
	de tratamento (uf_trat) com mais registros?	tratamento (uf_trat) com mais registros?	coloquial; simplificação sintática
br_0013	Qual é o total de registros em que a UF de residência é diferente da UF de tratamento?	Qual o total de registros em que a UF de residência é dif da UF de tratamento?	Abreviação lexical; simplificação sintática
br_0014	Para cada UF de residência, qual é a proporção de casos cujo tratamento ocorreu na mesma UF?	Pra cada UF de residência, qual a proporção de casos cujo trat ocorreu na mesma UF?	Abreviação lexical; reformulação coloquial
br_0015	Quantos registros existem por tipo de tratamento (tratamento_id)?	Qtd de registros por tipo de tratamento (tratamento_id)?	Abreviação lexical; simplificação sintática
br_0016	Qual é a idade média por tipo de tratamento (tratamento_id)?	Qual a idade média por tipo de trat (tratamento_id)?	Abreviação lexical; simplificação sintática
br_0017	Quantos registros existem por estadiamento (estadiam_id)?	Qtd de registros por estadiam (estadiam_id)?	Abreviação lexical; simplificação sintática
br_0018	Qual é o tempo médio até tratamento por estadiamento (estadiam_id) considerando apenas tempo_trat não nulo?	Qual o tempo médio até tratamnto por estadiam (estadiam_id) considerando apenas tempo_trat não nulo?	Abreviação lexical; erro ortográfico controlado; simplificação sintática
br_0019	Quantos registros existem por diagnostic_id (tipo de diagnóstico)?	Qtd de registros por diagnostic_id (tipo de diagnóstico)?	Abreviação lexical; simplificação sintática
br_0020	Quais são os 10 CNES de diagnóstico (cnes_diagn) com mais registros?	Top 10 CNES de diagnóstico (cnes_diagn) com mais registros?	Reformulação coloquial; simplificação sintática
br_0021	Quais são os 10 CNES de tratamento (cnes_trat) com mais registros?	Top 10 CNES de tratamento (cnes_trat) com mais registros?	Reformulação coloquial; simplificação sintática
br_0022	Qual é a quantidade de registros por sexo e ano de diagnóstico?	Qtd de registros por sexo e ano de diagnóstico?	Abreviação lexical; simplificação sintática
br_0023	Entre 2018 e 2023, qual foi o total de registros por ano de diagnóstico?	Entre 2018 e 2023, qual o total de registros por ano de diagnóstico?	Simplificação sintática
br_0024	No ano de 2020, quais são os 10 CIDs (cid10)	Em 2020, quais os 10 CIDs (cid10) mais	Reformulação coloquial; simplificação

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

ID	Pergunta canônica	Pergunta com ruído	Tipo de variação
	mais frequentes?	frequentes?	sintática
br_0025	No ano de 2020, qual é o total de registros por sexo?	Em 2020, qual o total de registros por sexo?	Reformulação coloquial; simplificação sintática
br_0026	Qual é o tempo médio até tratamento por sexo no ano de 2020 (somente tempo_trat não nulo)?	Qual o tempo médio até tratamnto por sexo em 2020 (só tempo_trat não nulo)?	Erro ortográfico controlado; reformulação coloquial; simplificação sintática
br_0027	Quantos registros possuem município de residência preenchido (mun_resid não nulo)?	Qtd de registros com município de residência preenchido (mun_resid não nulo)?	Abreviação lexical; simplificação sintática
br_0028	Qual é a quantidade de registros cuja UF de diagnóstico é diferente da UF de residência?	Qtd de registros em que UF de diagnóstico é diferente da UF de residência?	Abreviação lexical; simplificação sintática
br_0029	Qual é a quantidade de registros cujo município de tratamento é diferente do município de residência (ambos não nulos)?	Qtd de registros em que município de trat é diferente do município de residência?	Abreviação lexical; simplificação sintática
br_0030	Qual é a idade mínima, máxima e média no conjunto de dados?	Qual a idade mínima, máxima e média no conjunto de dados?	Simplificação sintática

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

Apêndice D – Prompts utilizados nos experimentos

Todo o conteúdo instrucional foi enviado diretamente no campo prompt da requisição ao Ollama. Não foi utilizado system prompt separado.

D.1 Prompt dos cenários sem esquema

You are an expert data analyst. Generate ONE SQLite query.

STRICT RULES:
- Output ONLY the SQL query. No markdown, no explanation, no backticks.
- SQLite ONLY. Do NOT use unsupported functions such as MEDIAN(), PERCENTILE(), DATE_TRUNC(), ILIKE.
- Return a single executable SQL query.
- Use only valid SQLite syntax.
- Do not invent explanations or comments.
- If top-k is requested, use ORDER BY ... DESC LIMIT k.
- Always return deterministic output: include ORDER BY when LIMIT is used.
- IMPORTANT: The SELECT column order MUST match the question requirement.

EXAMPLES (follow style exactly):
Q: Quantos registros existem no conjunto de dados?
SQL: SELECT COUNT(*) FROM fact_onco

Q: Quais são os 10 CIDs mais frequentes?
SQL: SELECT cid10, COUNT(*) AS n FROM fact_onco GROUP BY cid10 ORDER BY n DESC LIMIT 10

Q: Qual o tempo médio até tratamento por ano?
SQL: SELECT t.ano, AVG(f.tempo_trat) AS tempo_medio
FROM fact_onco f
JOIN dim_tempo t ON f.anomes_dia = t.anomes
WHERE f.tempo_trat IS NOT NULL
GROUP BY t.ano

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

ORDER BY t.ano

Now answer:
Q (pt-BR): {PERGUNTA_EM_PORTUGUES}
SQL:
D.2 Prompt dos cenários com esquema
You are an expert data analyst. Generate ONE SQLite query.

STRICT RULES:
- Output ONLY the SQL query. No markdown, no explanation, no backticks.
- SQLite ONLY. Do NOT use unsupported functions such as MEDIAN(), PERCENTILE(), DATE_TRUNC(), ILIKE.
- If the question asks for median, implement median using ORDER BY/LIMIT/OFFSET (no custom functions).
- Use ONLY tables/columns that exist in the provided schema.
- NEVER reference a table/alias that is not present in FROM/JOIN.
- If you select dim_sexo.sexo, you MUST JOIN dim_sexo on fact_onco.sexo = dim_sexo.sexo.
- Follow foreign keys: joins must match the schema (fact_onco FK -> dim_* PK).
- Prefer using dim_tempo (ano, mes) by joining fact_onco.anomes_dia = dim_tempo.anomes when filtering/grouping by year/month.
- If comparing UFs/municípios, use the correct columns: uf_resid/uf_trat/uf_diagn and mun_resid/mun_trat/mun_diagn.
- When the question asks about "tempo até tratamento", use fact_onco.tempo_trat and ignore NULL values (WHERE tempo_trat IS NOT NULL).
- For TOP-K by frequency or by any metric, EXCLUDE NULL values unless explicitly requested:
- e.g., WHERE <col> IS NOT NULL
- If top-k is requested, use ORDER BY ... DESC LIMIT k.
- Always return deterministic output: include ORDER BY when LIMIT is used.
- If you use an alias (e.g., f, t, d), you MUST define it in FROM/JOIN (e.g., FROM

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

fact_onco f). Never use undefined aliases.
- Do not add unnecessary dimension joins. If a column exists in fact_onco (e.g., sexo, uf_resid, cid10), use it directly (no dim_* join).
- For ratios/proportions, exclude NULLs for every column involved (e.g., uf_resid IS NOT NULL AND uf_trat IS NOT NULL).
- When computing a ratio "by X", GROUP BY X and ORDER BY X unless the question asks for ordering by the metric.
- For ANY aggregation/grouping by a categorical column X (COUNT/AVG by X, frequency tables, distributions):
  ALWAYS exclude missing values unless explicitly requested:
  add `WHERE X IS NOT NULL` before `GROUP BY X`.
- IMPORTANT: The SELECT column order MUST match the question requirement and the examples' style
  (e.g., for "por ano, tempo médio", output: ano first, metric second).

Schema:
{ESQUEMA_COMPACTO_DO_BANCO}

EXAMPLES (follow style exactly):
Q: Quantos registros existem no conjunto de dados?
SQL: SELECT COUNT(*) FROM fact_onco

Q: Quais são os 10 CIDs (cid10) mais frequentes?
SQL: SELECT cid10, COUNT(*) AS n FROM fact_onco GROUP BY cid10 ORDER BY n DESC LIMIT 10

Q: Qual o tempo médio até tratamento por ano?
SQL: SELECT t.ano, AVG(f.tempo_trat) AS tempo_medio
    FROM fact_onco f
    JOIN dim_tempo t ON f.anomes_dia = t.anomes
    WHERE f.tempo_trat IS NOT NULL

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

```
    GROUP BY t.ano
    ORDER BY t.ano
```

Q: Qual a mediana do tempo até tratamento?

```
SQL: SELECT AVG(tempo_trat) AS mediana
    FROM (
      SELECT tempo_trat
      FROM fact_onco
      WHERE tempo_trat IS NOT NULL
      ORDER BY tempo_trat
      LIMIT 2 - (SELECT COUNT(*) FROM fact_onco WHERE tempo_trat IS NOT
NULL) % 2
      OFFSET (SELECT (COUNT(*) - 1) / 2 FROM fact_onco WHERE tempo_trat IS
NOT NULL)
    )
```

Q: Em 2020, quantos registros por sexo?

```
SQL: SELECT f.sexo, COUNT(*) AS n
    FROM fact_onco f
    JOIN dim_tempo t ON f.anomes_dia = t.anomes
    WHERE t.ano = 2020
    GROUP BY f.sexo
    ORDER BY n DESC
```

Q: Qual a proporção de casos em que a UF de tratamento é a mesma UF de residência, por UF de residência?

```
SQL: SELECT uf_resid,
    CAST(SUM(CASE WHEN uf_trat = uf_resid THEN 1 ELSE 0 END) AS REAL) /
COUNT(*) AS proporcao_mesma_uf
    FROM fact_onco
    WHERE uf_resid IS NOT NULL AND uf_trat IS NOT NULL
```

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

```
    GROUP BY uf_resid
    ORDER BY uf_resid

Q: Distribuição por estadiamento (estadiam_id), do mais frequente ao menos
frequente.
SQL: SELECT estadiam_id, COUNT(*) AS n
    FROM fact_onco
    WHERE estadiam_id IS NOT NULL
    GROUP BY estadiam_id
    ORDER BY n DESC

Now answer:
Q (pt-BR): {PERGUNTA_EM_PORTUGUES}
SQL:
D.3 Esquema compacto utilizado nos prompts com esquema
# Schema dump: br_onco_v1

TABLE dim_cid10(cid10:TEXT)

TABLE dim_cnes(cnes_id:INTEGER)

TABLE dim_diagnostic(diagnostic_id:INTEGER)

TABLE dim_estadiam(estadiam_id:INTEGER)

TABLE dim_municipio(mun_id:INTEGER, uf_id:INTEGER)

TABLE dim_sexo(sexo:TEXT)

TABLE dim_tempo(anomes:INTEGER, ano:INTEGER, mes:INTEGER)
```

REMUNOM
ISSN 2178-6925

Received: 20/05/2026 - **Accepted:** 26/06/2025
Vol: 13.14
DOI: 10.66104/zvzdrv85
Pages: 1-67

TABLE dim_tratamento(tratamento_id:INTEGER)

TABLE dim_uf(uf_id:INTEGER)

TABLE fact_onco(id:INTEGER, anomes_dia:INTEGER, anomes_trat:INTEGER, uf_resid:INTEGER, mun_resid:INTEGER, uf_diagn:INTEGER, mun_diagn:INTEGER, uf_trat:INTEGER, mun_trat:INTEGER, cnes_diagn:INTEGER, cnes_trat:INTEGER, cid10:TEXT, sexo:TEXT, idade:INTEGER, tratamento_id:INTEGER, diagnostic_id:INTEGER, estadiam_id:INTEGER, tempo_trat:INTEGER, dt_diag:TEXT, dt_trat:TEXT)

FKs: fact_onco.anomes_trat -> dim_tempo.anomes; fact_onco.anomes_dia -> dim_tempo.anomes; fact_onco.cnes_trat -> dim_cnes.cnes_id; fact_onco.cnes_diagn -> dim_cnes.cnes_id; fact_onco.mun_trat -> dim_municipio.mun_id; fact_onco.mun_diagn -> dim_municipio.mun_id; fact_onco.mun_resid -> dim_municipio.mun_id; fact_onco.uf_trat -> dim_uf.uf_id; fact_onco.uf_diagn -> dim_uf.uf_id; fact_onco.uf_resid -> dim_uf.uf_id; fact_onco.estadiam_id -> dim_estadiam.estadiam_id; fact_onco.diagnostic_id -> dim_diagnostic.diagnostic_id; fact_onco.tratamento_id -> dim_tratamento.tratamento_id; fact_onco.cid10 -> dim_cid10.cid10; fact_onco.sexo -> dim_sexo.sexo