Democratizing Organizational Access to Federal Healthcare Data

Tria Federal
White Paper

triafed.com

Democratizing Organizational Access to Federal Healthcare Data

1. Introduction

Large troves of data are stored throughout the federal healthcare system. Much of the data is regulatory and administrative in nature. A September 2024 report estimated that approximately 25% of the government's $4 trillion annual healthcare budget is spent purely on administrative costs.

Of course, data is only as useful as the knowledge it generates, how it is applied to solve real business problems, and whether it helps achieve mission objectives.

The process of making it easier for all authorized employees to access data, regardless of technical expertise, is called democratization. Healthcare data is often organized in a tabular format in relational database management systems (RDBMS), which consist of a highly structured set of tables comprised of rows and columns (like spreadsheets) and linkages between those tables.

Traditionally, the standard mechanism to access data in an RDBMS is via standardized query language (SQL). SQL is an industry-wide standard used by virtually every organization around the world that deals with high volumes of data storage and retrieval.

However, when it comes to democratizing data access, SQL has its drawbacks. First, implementing SQL correctly requires programming expertise. Second, even with the right expertise, constructing an SQL query for non-trivial access to organizational data can be a complex process, often taking hours to build and fine-tune to get the desired results. This limits data access to the select few people in an organization with SQL expertise.

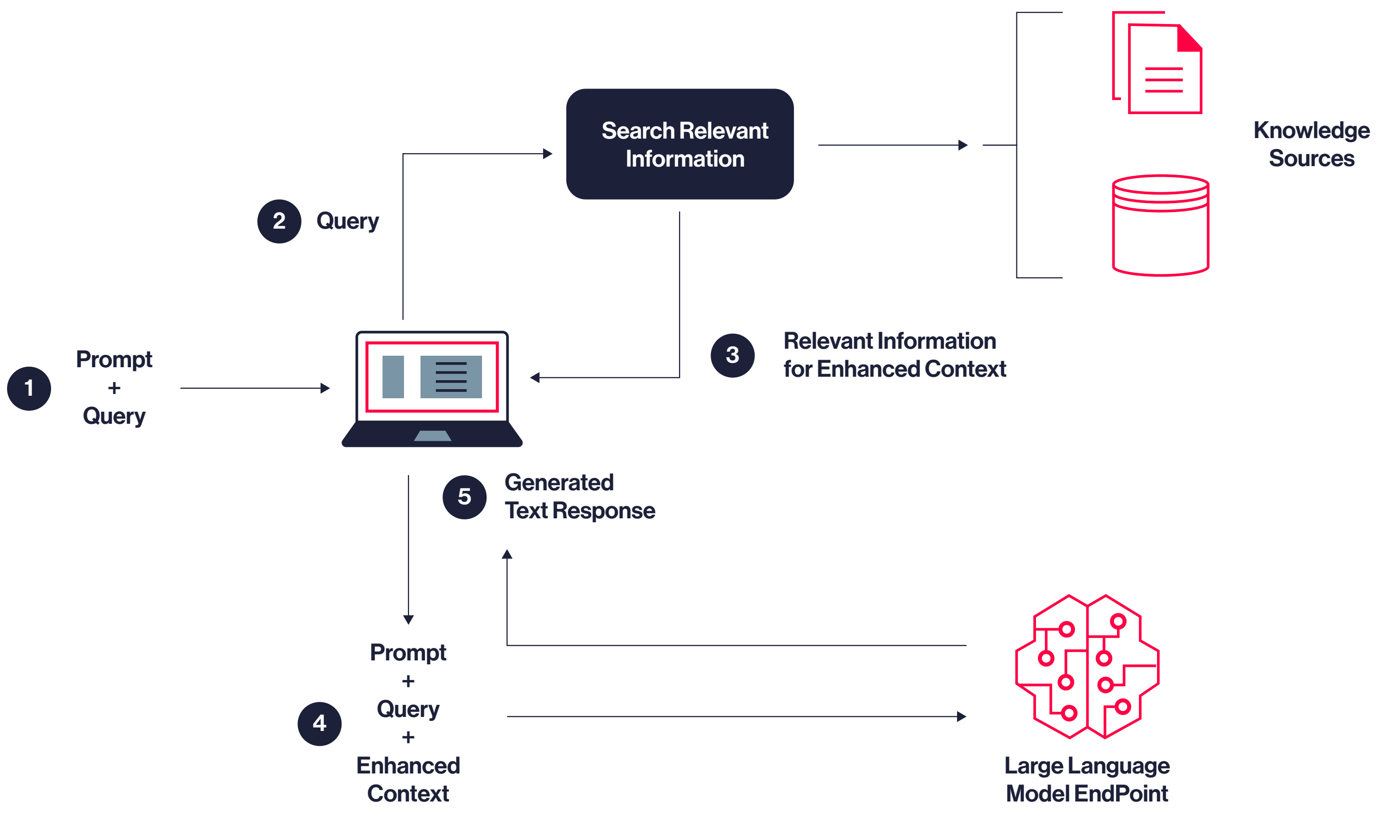
With the growing adoption and popularity of artificial intelligence and machine learning (AI/ML) solutions, we believe there is a better way for team members from any discipline to query and interact with organizational data. This white paper will discuss methods that Tria Labs is exploring to make data accessible for everyone in an organization without sacrificing quality and accuracy of the results.

2. Why Not Just Use LLMs?

Let's first explore the most obvious AI/ML approach—using publicly available large language models (LLMs). They are trained on vast amounts of text data and are extremely good at understanding and generating human-like text. Healthcare organizations generate vast amounts of complex, interrelated data from multiple sources, including electronic health records, clinical trials, insurance claims, and patient-reported outcomes. On the surface, this seems like an ideal use case for LLMs. By taking this approach, we eliminate the need for SQL altogether.

So, the question becomes, why not just feed all the tabular data into LLMs, and use a chatbot-interface to ask it about your data? The answer is because LLMs struggle to accurately query tabular data. Tria Labs recently ran trials on supply chain data in tabular format fed through the Claude Sonnet LLM via the Amazon Bedrock service, as illustrated below:

Figure A: AWS Bedrock Architecture



Search Relevant Information
Knowledge Sources
2 Query
1 Prompt + Query
3 Relevant Information for Enhanced Context
5 Generated Text Response
4 Prompt + Query + Enhanced Context
Large Language Model EndPoint

Despite being powered with newer techniques such as Retrieval Augmented Generation (RAG), we found that the LLM struggled to answer even basic questions accurately. There are various reasons for this, including:

- **Lack of structured understanding of data**
  LLMs lack an innate understanding of structured relationships between data elements. These elements have complex relationships with one another that need to be accounted for during the query process to deliver meaningful results.

- **Lack of necessary context due to ambiguity in the question**
  LLMs are trained to predict the next word in a sentence, but often lack a deeper understanding of other entities, concepts, and relationships between them to disambiguate a query. This can lead to context-related inaccuracies stemming from the original, ambiguous question. For example, if we instruct the LLM to “find all delayed shipments from the Norfolk Naval Shipyard,” is the intention to look for shipments that previously arrived late or that are now expected to arrive late? And what defines a delay? Is it a missed fixed date or a threshold period?

- **Difficulty handling complex joins and aggregations**
  To receive a meaningful answer from querying tabular data, complex join clauses, subqueries, and aggregations are often required across multiple tables. For example, even a simple query like, “show me the top 10 suppliers by product quantity sold in the last six months” might require aggregation across multiple tables with many rows of data. Pre-trained LLMs are generally not good at doing this.

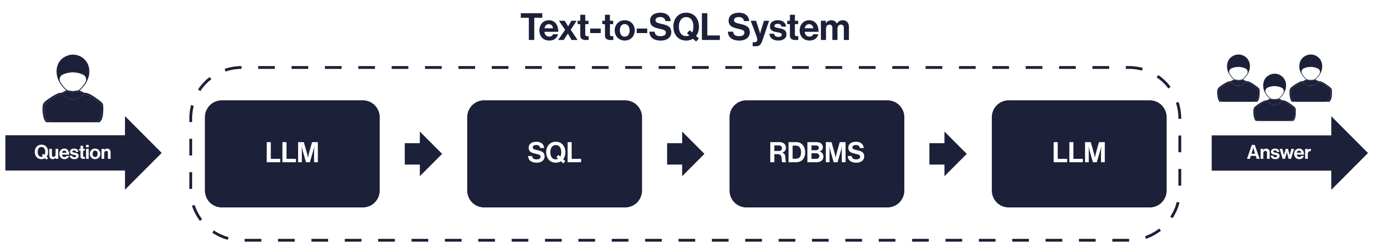
Tria Labs also tested SnapLogic’s GenAI solution with similar results. The LLM struggled to satisfactorily answer more complex questions. The main issue here was that it required more context and linkages to decipher complex relationships between tables.

3. Text-to-SQL and Limitations

Because of these issues, using natural language queries (i.e. plain English) to ask LLMs questions about tabular data is not the best way of getting quality responses. Although SQL queries are the preferred approach, they require engineering expertise and are therefore not suitable for democratizing data access. So where does that leave us?

What if we used a text-to-SQL system, which combines the AI/ML technique of natural language processing (NLP) with the ability to machine-translate human-text queries into SQL for us? This would allow any user in an organization to query their data in a natural way and have the system handle SQL generation behind the scenes to generate a response. This approach would require no programming experience to ask the question and would benefit from the power of a structured language like SQL to effectively query a relational database that consists of primarily tabular data. See the figure below for a basic notional flow of how this system works.

Figure B: Notional Diagram of Text-to-SQL Flow



While this sounds great in theory, our Tria Labs investigation found that the practical use of this approach is limited. Traditional text-to-SQL solutions are only as good as the SQL they can generate. They perform as expected with simple queries, but they struggle as the complexity increases, often failing to generate SQL queries that satisfactorily or accurately address the original question.

Tria Labs has explored querying tabular supply chain data with available commercial-off-the-shelf text-to-SQL solutions. As soon as we started asking non-trivial questions spanning multiple tables, they failed to generate acceptable SQL queries.

It's important to note here that SQL itself is perfectly capable of answering complex queries. The problem lies in the quality of the SQL generation from these solutions, which stems from many of the same factors responsible for poor LLM response performance as described above, including ambiguity in natural language; complex relationships and joins and failure to handle aggregations and groupings; and lack of domain-specific knowledge and context.

4. Graph-augmented Text-to-SQL

The root of the problem with poor Text-to-SQL performance for complex questions is a lack of contextual awareness and understanding of complex relationships between the tabular data elements. These limitations stem from an inadequate representation of the underlying data. For a system to take advantage of complex relationships and context to deliver high-quality responses, the underlying data needs to be modeled in a way that context and relationships are utilized effectively to prepare a response.

This is where using graph databases to represent the underlying relationships can provide tremendous value. Healthcare data includes a wide variety of entities—patients, diagnoses, treatments, medications, providers—and these entities are deeply interconnected.

Specifically:

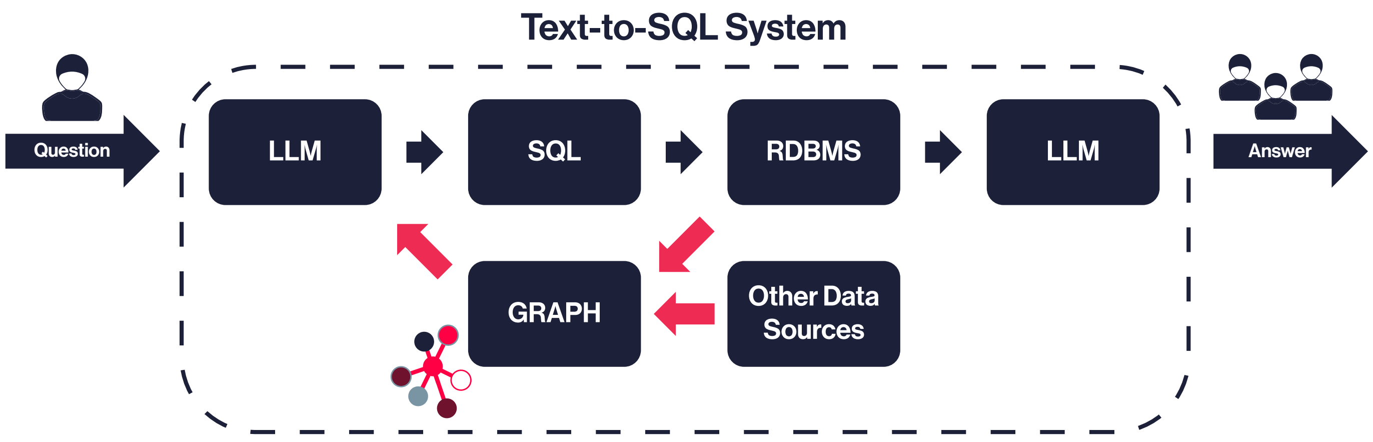
- **Patient-Provider Relationships:**
  A patient may see multiple providers over time, each for different reasons (primary care, specialists, etc.).

- **Medical History:**
  A patient's medical history is not limited to a linear set of data points but involves multiple diagnoses, treatments, test results, and outcomes.

- **Clinical Data:**
  Querying clinical data may require understanding causal relationships (e.g., how a specific treatment affects patient outcomes).

Graph databases naturally model these complex relationships, but current Text-to-SQL systems may not fully take advantage of this, leading to inaccurate query generation. Graph databases, which represent data as interconnected nodes and relationships, provide a more natural and flexible way to model healthcare data. By integrating graph databases with Text-to-SQL systems, organizations can significantly improve query accuracy and facilitate more insightful and efficient decision-making. See the below notional figure for how a system like this would look.

Figure C: Notional Diagram of Graph-enhanced Text-to-SQL Flow



Knowledge graphs model healthcare data as entities (nodes) and relationships (edges). These entities can include patients, medical conditions, medications, treatments, providers, appointments, and more. The relationships between these entities (e.g., "patient X diagnosed with condition Y," or "provider A prescribed medication B") are clearly captured in the graph. Healthcare knowledge graphs provide a semantic layer on top of the raw data, enabling Text-to-SQL systems to better understand the context and relationships between entities, leading to more accurate SQL generation.

5. How Graphs Enhance Text-to-SQL Accuracy in Healthcare

- **Capturing Complex Healthcare Relationships**
  In healthcare, relationships between entities (e.g., patients, conditions, providers) can be complex and multi-faceted. For instance, a patient may have multiple diagnoses, be prescribed various medications, and see different providers over time. Graph databases naturally model these complex relationships, allowing Text-to-SQL systems to generate more accurate and comprehensive queries that reflect these intricate connections.

- **Improved Contextual Understanding of Medical Terminology**
  Medical queries often involve domain-specific terms (e.g., "chronic conditions," "treatment plans," "medication adherence") that require a deep understanding of medical language and context. Knowledge graphs, which store structured medical knowledge, allow Text-to-SQL systems to better interpret these terms and generate SQL queries that accurately capture the user's intent. For example, a query like "find patients who have diabetes and are receiving insulin" can be more accurately interpreted when the system understands the relationship between "diabetes," "insulin," and "treatment protocols."

- **Ambiguity Resolution in Medical Queries**
  Healthcare queries can be highly contextual, with the meaning of terms shifting depending on the situation. For instance, the term "medication" may refer to different things depending on the context of the query (e.g., "medications prescribed to cancer patients" vs. "medications for pain relief"). Graph databases help resolve such ambiguities by modeling these relationships explicitly, ensuring that the Text-to-SQL system generates queries that accurately reflect the intent behind the query.

- **Domain-Specific Knowledge Integration**
  Integrating domain-specific knowledge into the Text-to-SQL system is essential for healthcare applications. Knowledge graphs can incorporate medical ontologies, clinical guidelines, and patient care protocols, which help the system understand medical terms, relationships, and best practices. This is especially useful for generating SQL queries that involve clinical decision support or research data analysis.

6. Conclusion

Data is only useful if it can be easily and readily turned into actionable information. By democratizing access to reams of data housed in federal agencies, systems with this capability will be able to empower any authorized employee to access that data and generate value for their stakeholders.

At Tria Labs, we are exploring ways to leverage emerging technologies to do just that. By modeling healthcare data with graph databases and combining it with the power of NLP, SQL, and LLMs, we can better capture intricate relationships among patients, treatments, providers, and medical histories to deliver exciting new mission-focused capabilities to our customers.

7. Resources

Administrative simplification: How to save a quarter-trillion dollars in US healthcare
https://www.mckinsey.com/industries/healthcare/our-insights/administrative-simplification-how-to-save-a-quarter-trillion-dollars-in-us-healthcare

GenAI for Aerospace: Empowering the workforce with expert knowledge on Amazon Q and Amazon Bedrock
https://aws.amazon.com/blogs/machine-learning/genai-for-aerospace-empowering-the-workforce-with-expert-knowledge-on-amazon-q-and-amazon-bedrock/

Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation
https://bolinding.github.io/papers/vldb24dailsql.pdf

Generating value from enterprise data: Best practices for Text2SQL and generative AI
https://aws.amazon.com/blogs/machine-learning/generating-value-from-enterprise-data-best-practices-for-text2sql-and-generative-ai/

Text-to-SQL Meets the Real-World
https://www.scitepress.org/Papers/2024/125552/125552.pdf

Text-to-SQL: A methodical review of challenges and models
https://journals.tubitak.gov.tr/cgi/viewcontent.cgi?article=4077&context=elektrik

Large Language Model Enhanced Text-to-SQL Generation: A Survey
https://www.researchgate.net/publication/384769411_Large_Language_Model_Enhanced_Text-to-SQL_Generation_A_Survey

State of Text2SQL 2024
https://blog.premai.io/state-of-text2sql-2024/

8. Acknowledgements

Samir Pandurangi is a Data Engineering Architect and Subject Matter Expert at Tria Labs. He focuses on prototyping and integrating emerging solutions to support mission-focused software capabilities.

Tria Federal White Paper

triafed.com