A DB-First Approach to Query Factual Information in LLMs

Mohammed Saeed
Sophia Antipolis, France
Turin, Italy

Paolo Papotti
EURECOM
Sophia Antipolis, France

Nicola De Cao
Google AI
London, UK

1 Introduction

Declarative querying is one of the main features behind the popularity of database systems. However, SQL can be executed only on structured datasets, leaving out of immediate reach information expressed as unstructured text. Several technologies have been deployed to extract structured data from unstructured text and to model such data in relations or triples (Zhang et al., 2017; Chiticariu et al., 2018). While these methods have been studied for more than 20 years, creating well-formed structured data from text is still time consuming and error prone.

While declarative querying of text is a big challenge, there has recently been progress in *question answering* (QA) over text (Rogers et al., 2023). In this setting, a question in natural language (NL) is answered by gathering information from a corpus of text documents. Large Language Models (LLMs) (Radford et al., 2018, 2019; Brown et al., 2020) can answer complex questions in a closed-book fashion (Roberts et al., 2020) (example (2) in Figure 1). While it has been shown that such models store high quality factual information (Petroni et al., 2019; Lewis et al., 2020), they are not trained to answer complex SQL queries and may fail short with such input.

We argue that query pre-trained LLMs with SQL scripts is beneficial to overcome the problem of information extraction from text and enables data applications on top of LLMs. As depicted in example (1) in Figure 1, a pre-trained LLM can act as the data storage containing the information to answer a declarative query. Our solution preserves the characteristics of SQL when executed over this new source of data: (i) queries are written in arbitrary SQL over a user defined relational schema, enabling data application to be executed on LLM's factual data; (ii) answers are *correct* and *complete* w.r.t. the information stored in the LLM. This last point requires the correct execution of the queries and does not assume that LLMs always return perfect information. While LLMs still make factual mistakes, this work shows that it is already possible to collect tuples from them with promising results. With the ongoing efforts in LLMs, there is evidence that their factuality and coverage is improving (Elazar et al., 2021; Tam et al., 2022).

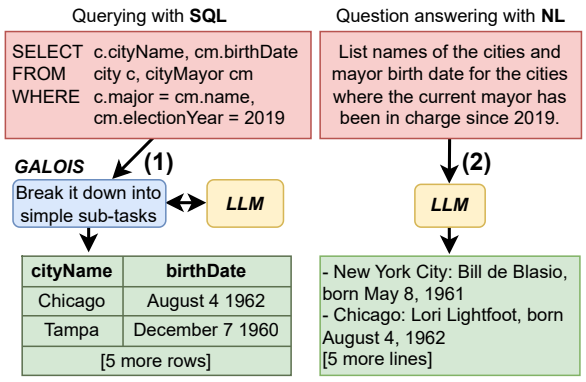

Figure 1: Querying a LLM with SQL is different from QA. GALOIS executes the query, and obtains relations, by retrieving data from a LLM (1). The corresponding QA task consumes and produces NL text (2).

Being able to SQL query LLMs is appealing, but it is not clear on which architecture a solution should pivot on. Looking at the architectures for LLMs and DBMSs, there are different paths to

2nd Table Representation Learning Workshop at NeurIPS 2023.

explore. One is *LLM-first*, where external information (including structured data) is accessed by the LLM (Borgeaud et al., 2022; Peng et al., 2023). While this approach is gaining visibility with Retrieval Augmented Generation, the limited context size in LLMs does not allow yet to execute queries that require a large number of tuples as input, such as aggregate queries over thousands of rows. The alternative path is *DB-first*, where LLMs are used as a component in a traditional DB query processing architecture, as we do in this work. Our core idea is that the query plan is a natural decomposition of the (possibly complex) process to obtain the result, in analogy with the approaches in NLP showing that breaking a complex task in a chain of thoughts is key to get the best results (Wei et al., 2022; Khattab et al., 2022). Our contributions are summarized in the following points:

(1) We introduce the problem of querying with SQL existing pre-trained LLMs. We introduce GALOIS, a DB-first prototype that executes SPJA queries under assumptions that enable a large class of data applications (code available at `https://gitlab.eurecom.fr/saeedm1/galois`).

(2) The logical query plan breaks down the complex task into simpler steps that can be handled effectively by the LLM. Physical operators in the query plan are implemented as textual prompts for LLMs. Such prompts are generated from the input schema and the logical operators.

(3) We show that GALOIS's results for 46 queries on top of popular LLMs are (i) comparable to those obtained by executing the same queries on DBMS and (ii) better than those obtained by manually rewriting the queries (and parsing the results) in NL for QA over the same LLM.

Background. Our effort is different from the problem of semantic parsing, i.e., the task of translating NL questions into SQL (Yu et al., 2018; Katsogiannis-Meimarakis and Koutrika, 2021). Our goal is also different from querying an existing relational database to answer a NL question (Herzig et al., 2020; Papicchio et al., 2023). We retrieve data from the LLM with SQL queries, with the traditional semantics and with the output expressed in the relational model, as if the query were executed on a DBMS. While some of these facts can be retrieved with QA, (i) the SQL query must be rewritten as an equivalent question in NL, which is not practical for complex scripts, (ii) the textual result must be parsed into a relation, (iii) current LLMs in some cases fail in answering complex queries expressed as NL. Indeed, QA systems are optimized for answering questions with a text, while SQL queries return results in the form of tuples, possibly with complex operations to combine intermediate values, such as aggregates, where LLMs fail short (Ribeiro et al., 2020). To overcome some of these limits, it has recently been shown that a series of intermediate reasoning steps ("chain of thought" and question decomposition (Wolfson et al., 2020)) improve LLMs' ability in complex tasks (Wei et al., 2022).

2 Design Considerations

Our goal is to execute SQL query over the data stored into LLMs. When we look at these models from a DB perspective, they have extensive coverage of facts from textual sources. However, LLMs have their shortcomings. We delve into three issues that have impacted the design of GALOIS.

1. Tuples and Keys. LLMs do not have a concept of schema or tuple, but they model existing relationships between entities ("Rome is located in Italy") or between entities and their properties ("Rome has 3M residents"). However, a query asking for city names may assume that a name identifies a city, which is not the case in reality, e.g., a Rome city in Georgia, USA. This problem is solved with keys defined with multiple attributes; a composite key defined over (name, state, country) distinguishes the Rome in Italy from the one in Georgia. In GALOIS, we assume that every relation in the query has a key and that the key is expressed with one attribute, e.g., its name.

2. Schema Ambiguity. Similarly to the issue with entities, words, including attribute labels, can have multiple meanings. These alternatives are represented differently in the LLMs. In our setting, a given attribute label in the query can be mapped to multiple "real world" attributes in the LLM, e.g., *size* for a city can refer to population or urban area (Veltri et al., 2022). In this work, we assume that meaningful labels for attributes and relations are used in the queries.

3. Factual Knowledge in LLMs. LLMs do *not know what they know*. The model returns the next token in a stream. Such token may be based on either reliable acquired knowledge, or it may be a guess. For this reason, a query result obtained LLMs is not 100% reliable. However, with

GALOIS, we experimentally demonstrate that it is possible to extract factual information from LLMs to answers SQL queries. Moreover, new models keep increasing the factuality of their answers.

3 Overview

The high-level architecture of GALOIS is presented in Figure 1. We assume that the schema (but no instances) is provided together with the query. The system processes SQL queries over data stored in a pre-trained LLM. This design enables developers to implement their data applications in a conventional manner, as the complexities of using a LLM are encapsulated within GALOIS.

We use LLMs to implement a set of specialized physical operators in a traditional query plan, as in Figure 2. As tuples are not directly available, we implement the access to the base relations (leaf nodes) with the retrieval of the key attribute values. We then retrieve the other attributes as we go across the plan. For example, if the selection operator is defined on attribute A different from the key, the corresponding implementation is a prompt that filters every key attribute based on the selection condition, e.g., "Has city *c.name* more than 1M population?", where *c.name* iterates over the set of key values. A prompt is obtained for each operator by combining a set of operator-specific prompt templates with the labels/selection conditions in the given SQL query. If a join or a projection involve an attribute that has not been collected for the tuple, this is retrieved with a special node injected right before the operation. For example, if a join involves an attribute "currentMayor", the corresponding attribute values are retrieved with a prompt that collects it for every key. Once the tuples are completed, regular operators, implemented in Python in our prototype, are executed on those, e.g., joins and aggregates.

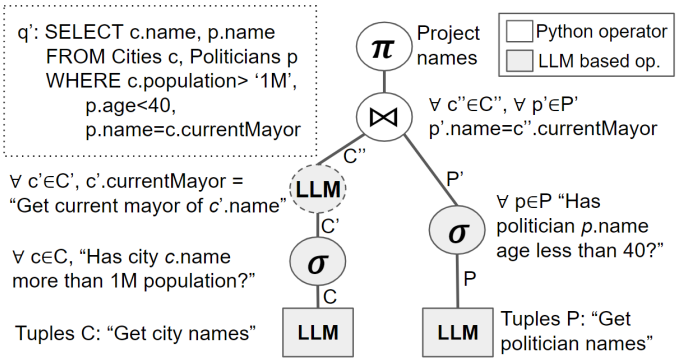

Figure 2: Logical plan for query q'. Base relations are accessed by retrieving sets of tuples (C, P) with one key attribute (*name*) from the LLM.

On one hand, the query plan acts as a chain of thought decomposition of the original task, i.e., the plan spells outs intermediate steps. On the other hand, the operators that manipulate data fill up the limitations of LLMs, e.g., in computing average values or comparing quantities (Lewkowycz et al., 2022). Together, these two features make the LLM able to execute complex queries.

Two critical steps enable the practical use of GALOIS. First, as relations can be large, we iterate with the a prompt until we stop getting new results. A second issue is the cleaning of the data from the LLM. For example, numerical data can be retrieved in different formats. We normalize every string expressing a numerical value (say, 1k) into a number (1000). The enforcing of type and domain constraints is a simple but crucial step to limit the incorrect output.

4 Experiments

GALOIS is written in Python and all LLMs have been executed locally with the exception of ChatGPT, for which we used the API. Query plans are obtained from DuckDB.

Dataset. Spider is a Text2SQL dataset with 200 databases, each with a set of SQL queries (Yu et al., 2018). For each query, it provides its paraphrase as a NL question. We focus on a subset of *46 queries for which we expect to obtain answers from an existing LLM*. More precisely, we leave out queries that are specific to the relational dataset provided by Spider and use in our evaluation only queries about generic topics, such as world geography and airports.

Setup. We test four LLMs. *Flan-T5-large* (**Flan**): T5 fine-tuned on datasets described via instructions (783M parameters). *TK-instruct-large* (**TK**): T5 with instructions and few-shot with positive

	Flan	TK	GPT-3	ChatGPT
Difference as % of R_D size	-47.4	-43.7	+1.0	-19.5

Table 1: Average difference in the cardinality of GALOIS's output relations (R_M) w.r.t. the ground truth results $|R_D|$ for the 46 Spider queries. Closer to 0 is better.

	All	Selections only	Aggregates	Joins
R_M (SQL Queries)	50	80	29	0
T_M (NL Questions)	44	71	20	8

Table 2: Cell value matches (%) between the result returned by a method and the same query executed on the ground truth data (R_D) for the 46 Spider queries. Averaged results for ChatGPT.

and negative examples (783M parameters). *InstructGPT-3* (**GPT-3**): fine-tuned GPT-3 using instructions from humans (Ouyang et al., 2022) (175B parameters). *GPT-3.5-turbo* (**ChatGPT**): chat model in the OpenAI API (175B parameters). We construct prompts appropriately for each model. For a given LLM M and a SQL query q with its Spider relation D and the corresponding NL question t, we collect three results: (a) relation R_M from GALOIS executing q over M, (b) relation R_D by executing q over D, (c) text T_M by asking t over M[1]. Only (b) uses the relations from Spider, (a) and (c) get the data from the LLM.

Evaluation. We analyze the results across two dimensions. *(1) Cardinality.* We measure to which extent GALOIS returns correct results in terms of number of tuples. As NL questions always return text paragraphs, we cannot include their results in this analysis. For GALOIS, all output relations have the expected schema, i.e., every R_M has the same schema as every R_D. However, in terms of number of tuples there are differences. We compute the ratio of the sizes as $f = \frac{|2*R_D|}{|R_D+R_M|}$, where the interval for f is [0,2] and best result occurs when $R_D == R_M$ (f=1). Table 1 reports the difference as percentage (averaged over all queries with non-empty results) with the formula 1-f. Results show that smaller models do worse and miss lots of result rows, up to 47.4% w.r.t. the size of results from the SQL execution R_D. For GPT models, almost all queries return a number of tuples close to R_D. Most differences are explainable with errors in the results of the prompts across the query plan execution, as we discuss next.

(2) Content. Second, we measure the quality of the results by comparing the content of each cell value after manually mapping tuples between R_D on one side (ground truth) and (R_M, T_M) on the other. As T_M contains NL text, we manually postprocess them to extract the values as records. We consider a numerical value in (R_M, T_M) as correct if the relative error w.r.t. R_D is less than 5%. As this analysis requires to manually verify every result, we conduct it only for one LLM. Results in Table 2 show that GALOIS executes the queries on ChatGPT with a better average accuracy in the results compared to the same queries expressed as questions in NL. We believe this is a promising result, as one can think that the results coming from the NL QA task are the upper bound for what the LLM knows. For the easiest subclass of queries, selection-only, the query approach returns correct values in 80% of the cases. Joins are the most problematic, as the equality test fails due to different formats of the same text, e.g., an attempt to join the country code "IT" with "ITA" for entity Italy.

5 Conclusion

This work shows how GALOIS integrates a LLM into a DBMS through SQL to create a novel query execution environment. This approach could lead to new applications, by unlocking information from unstructured text data, and foster collaboration between DB and NLP communities. The research leaves open many questions on logical and physical query optimization. For logical planning, future work can explore combining operators over the LLM to reduce the number of calls to the model. In the physical optimization it would be valuable to explore the automatic generation of more precise textual prompts, for example using data samples (Urban et al., 2023) or pre-defined embeddings for attribute types (Saeed and Papotti, 2022).

[1]In the full paper (Saeed et al., 2024), we report also a baseline that uses chain-of-thought reasoning; its results are better than simple prompting but lower than those from GALOIS.

References

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving Language Models by Retrieving from Trillions of Tokens. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.), Vol. 162. PMLR, 2206–2240. https://proceedings.mlr.press/v162/borgeaud22a.html

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS*. https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

Laura Chiticariu, Marina Danilevsky, Yunyao Li, Frederick Reiss, and Huaiyu Zhu. 2018. SystemT: Declarative Text Understanding for Enterprise. In *NAACL*. Association for Computational Linguistics, 76–83. https://doi.org/10.18653/v1/N18-3010

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard H. Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and Improving Consistency in Pretrained Language Models. *Trans. Assoc. Comput. Linguistics* 9 (2021), 1012–1031. https://doi.org/10.1162/tacl_a_00410

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *ACL*. ACL, 4320–4333. https://doi.org/10.18653/v1/2020.acl-main.398

George Katsogiannis-Meimarakis and Georgia Koutrika. 2021. A Deep Dive into Deep Learning Approaches for Text-to-SQL Systems. In *SIGMOD*. ACM, 2846–2851.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-Search-Predict: Composing Retrieval and Language Models for Knowledge-Intensive NLP. *arXiv preprint arXiv:2212.14024* (2022).

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*, Vol. 33. 9459–9474. https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving Quantitative Reasoning Problems with Language Models. In *NeurIPS*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv* abs/2203.02155 (2022).

Simone Papicchio, Paolo Papotti, and Luca Cagliero. 2023. QATCH: Benchmarking Table Representation Learning Models on Your Data. In *NeurIPS (Datasets and Benchmarks)*.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback. (2023). arXiv:cs.CL/2302.12813

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases?. In *EMNLP*. 2463–2473. https://doi.org/10.18653/v1/D19-1250

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

Marco Túlio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *ACL*. Association for Computational Linguistics, 4902–4912. https://doi.org/10.18653/v1/2020.acl-main.442

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model?. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 5418–5426. https://doi.org/10.18653/v1/2020.emnlp-main.437

Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2023. QA dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *Comput. Surveys* 55, 10 (2023), 1–45.

Mohammed Saeed, Nicola De Cao, and Paolo Papotti. 2024. Querying Large Language Models with SQL. In *EDBT*.

Mohammed Saeed and Paolo Papotti. 2022. You are my type! Type embeddings for pre-trained language models. In *EMNLP 2022, Conference on Empirical Methods in Natural Language Processing*, ACL (Ed.). Abu Dhabi.

Derek Tam, Anisha Mascarenhas, Shiyue Zhang, Sarah Kwan, Mohit Bansal, and Colin Raffel. 2022. Evaluating the Factual Consistency of Large Language Models Through Summarization. (2022). https://doi.org/10.48550/ARXIV.2211.08412

Matthias Urban, Duc Dat Nguyen, and Carsten Binnig. 2023. OmniscientDB: A Large Language Model-Augmented DBMS That Knows What Other DBMSs Do Not Know. In *aiDM*. ACM, 4:1–4:7. https://doi.org/10.1145/3593078.3593933

Enzo Veltri, Donatello Santoro, Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2022. Pythia: Unsupervised Generation of Ambiguous Textual Claims from Relational Data. In *SIGMOD*. ACM, 2409–2412. https://doi.org/10.1145/3514221.3520164

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *NeurIPS*.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break It Down: A Question Understanding Benchmark. *Transactions of the Association for Computational Linguistics* 8 (04 2020), 183–198. https://doi.org/10.1162/tacl_a_00309

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *EMNLP*. Association for Computational Linguistics, 3911–3921. https://doi.org/10.18653/v1/D18-1425

Ce Zhang, Christopher Ré, Michael J. Cafarella, Jaeho Shin, Feiran Wang, and Sen Wu. 2017. DeepDive: declarative knowledge base construction. *Commun. ACM* 60, 5 (2017), 93–102. https://doi.org/10.1145/3060586