2024 IEEE 40th International Conference on Data Engineering (ICDE) | 979-8-3503-1715-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDE60146.2024.00420

Chat2Query: A Zero-Shot Automatic Exploratory Data Analysis System with Large Language Models

Jun-Peng Zhu[†‡], Peng Cai[†*], Boyan Niu[‡*], Zheming Ni[‡], Kai Xu[‡*]
Jiajun Huang[‡], Jianwei Wan[‡], Shengbo Ma[‡], Bing Wang[‡], Donghui Zhang[‡], Liu Tang[‡], Qi Liu[‡]
[†]*East China Normal University,* [‡]*PingCAP*
zjp.dase@stu.ecnu.edu.cn, pcai@dase.ecnu.edu.cn, {brian.niu, nizheming, xukai}@pingcap.com
{huangjiajun, jianwei.wan, shengbo.ma, bing.wang}@pingcap.com, {zhangdonghui, tl, liuqi}@pingcap.com

***Abstract*—Data analysts often encounter two primary challenges while conducting exploratory data analysis by SQL: (1) the need to skillfully craft SQL queries, and (2) the requirement to generate suitable visualizations that enhance the interpretation of query results. The emergence of large language models (LLMs) has inaugurated a paradigm shift in text-to-SQL and data-to-chart. This paper presents Chat2Query, an LLM-empowered zero-shot automatic exploration data analysis system. Firstly, Chat2Query provides a user-friendly interface that allows users to employ natural languages to interact with the database directly. Secondly, Chat2Query offers an LLM-empowered text-to-SQL generator, SQL rewriter, SQL formatter, and data-to-chart generator. Thirdly, Chat2Query is uniquely distinguished by its underlying incorporation of the TiDB Serverless, fostering superior elasticity and scalability. This strategic integration empowers Chat2Query with the capability to seamlessly adapt to change workloads, aligning with the evolving demands of the user. We have implemented and deployed Chat2Query in the production environment, and demonstrate its usability and efficiency in three representative real-world scenarios.**

***Index Terms*—EDA, Text-to-SQL, LLMs, Prompting, Usability**

I. Introduction

Exploratory Data Analysis (EDA) [1], [2] technique, coupled with SQL, assumes a crucial role for data analysts engaging in data exploration and analysis. This technique harnesses SQL to construct queries capable of extracting vital information from databases. These queries empower users to perform tasks like data aggregation, filtration, and sorting, generating statistical insights and constructing comprehensive views and reports. EDA facilitates deeper comprehension of data, unveiling latent patterns and trends, which subsequently offer valuable insights to guide subsequent decision-making endeavors. However, EDA has two pain points for junior data analysts: (1) the need to skillfully craft SQL queries, and (2) the requirement to generate suitable visualizations that enhance the interpretation of query results.

Text-to-SQL [3], [4], [5], [6], a transformative technology that translates natural language (NL) inquiries into SQL queries, holds the potential to empower data analysts with the ability to conduct intricate data analysis using the natural languages. Existing works for text-to-SQL have some limitations.

(1) **Limited Accuracy for Out-of-Domain Queries**. The recently published Learn-to-Rank [3], [4] (LTR) approach

[*] Corresponding Author

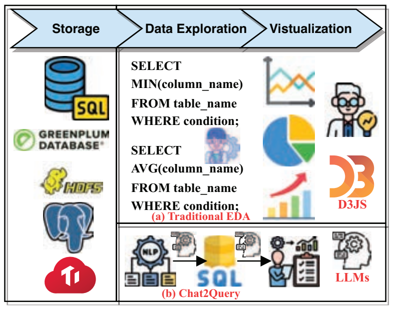


Fig. 1. A comparison illustration for traditional EDA and Chat2Query. (a) Traditional EDA encompasses data storage, data exploration via SQL, and visualization using d3js. Both SQL query and visualization creation demand expertise. (b) In contrast, Chat2Query incorporates an SQL generation engine and a chart generation engine empowered by LLMs, streamlining these processes.

assumes that the given *observed* sample queries are representative enough to cover all *unobserved* user-intended queries. In real applications, this assumption might not hold for "out-of-domain" queries, which could result in inadequate coverage of diverse query variations. On the other hand, this implies that LTR methods typically require substantial annotated data for model training. This can be challenging to acquire in specific domains such as databases world, thereby limiting the applicability of approaches.

(2) **Insufficient Domain-Specific Knowledge**. Large language models (LLMs) like ChatGPT, BLOOM, and LLaMA have undergone rapid development to enable the realization of general artificial intelligence, boasting an impressive zero-shot capabilities across diverse linguistic applications. However, those LLMs excel in the realm of general knowledge, their performance within specific knowledge domains, such as the database industry world, often results in answers that are somewhat erroneous and should not be wholly relied upon due to lacking specific training.

Figure 1 illustrates the difference of traditional EDAs (a) with our Chat2Query [1] (b). In Figure 1(a) traditional EDA, users have had to dedicate substantial efforts to manually sifting through data using SQL and visualizing data through tools like d3.js (https://d3js.org/). These limitations impeded the

[1]https://www.pingcap.com/chat2query-an-innovative-ai-powered-sql-generator-for-faster-insights/

efficient extraction of valuable insights. Simultaneously, existing EDA systems are unable to harness the elastic scalability provided by the cloud to accommodate change workloads.

To address the above mentioned limitations, we present a solution by introducing an LLM-empowered real-time exploratory data analysis system called Chat2Query as shown in Figure 1(b). Chat2Query streamlines the EDA workflow, enabling users to explore their data with minimal expertise required. Driven by user queries in natural language, Chat2Query provides a seamless transition from text to SQL and further generates charts. This innovative system offers the following distinctive features.

To begin with, Chat2Query offers a user-friendly interface that enables users to effortlessly interact with the database using natural language. Secondly, Chat2Query encompasses an LLM-empowered text-to-SQL generator, SQL rewriter, SQL formatter, and data-to-chart generator. The third key feature empowers Chat2Query to enable online real-time data exploration through the utilization of the TiDB Serverless. This integration provides Chat2Query with remarkable elasticity and scalability performance. With these capabilities at its disposal, Chat2Query showcases its value across a wide range of data analysis scenarios.

In this paper, we demonstrate Chat2Query in three representative real-world scenarios. The first is the Steam Game [7] which is a dataset within the target industry business of PingCAP. The second dataset is Fortune 500 companies [8] from 1996 to 2023, which is a common data analysis scenario at PingCAP during the Proof of Concept (POC) phase. The third is Spider [9] which is Yale semantic parsing and text-to-SQL challenge dataset.

Key Contributions. To summarize, this paper makes the following contributions:

- We develop an LLM-empowered zero-shot real-time automatic exploration data analysis system called Chat2Query.
- Chat2Query provides an easy-to-use user interface and supports an LLM-empowered zero-shot text-to-SQL generator. Meanwhile, Chat2Query also provides the LLM-empowered SQL rewriter, SQL formatter, and data-to-chart generator.
- Chat2Query facilitates online real-time data exploration by utilizing the TiDB Serverless.
- We deploy Chat2Query in three representative real-world scenarios for the system demonstration.

II. System Implementation

In this section, we first present the Chat2Query system overview, and then introduce the LLM-empowered generative engine of Chat2Query. Finally, we provide the end-to-end usability evaluation of EDA system using real-world datasets.

A. System Overview

Chat2Query is an LLM-empowered exploratory data analysis system. Figure 2 presents the system overview of our Chat2Query. Due to space limitations, we refrain from providing an exhaustive list of the prompts employed. As can be seen, the architecture of Chat2Query can be roughly divided into the following four components:

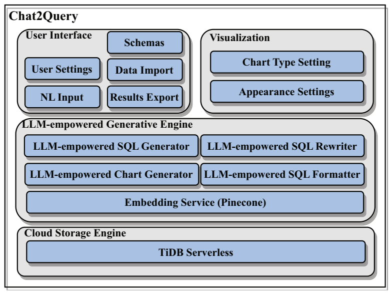

Chat2Query
User Interface
Schemas
User Settings
Data Import
NL Input
Results Export
Visualization
Chart Type Setting
Appearance Settings
LLM-empowered Generative Engine
LLM-empowered SQL Generator
LLM-empowered SQL Rewriter
LLM-empowered Chart Generator
LLM-empowered SQL Formatter
Embedding Service (Pinecone)
Cloud Storage Engine
TiDB Serverless

Fig. 2. Architecture of Chat2Query EDA System.

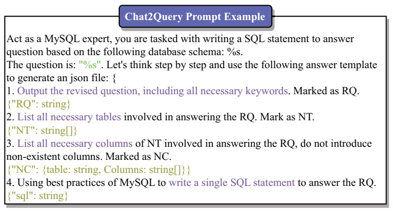
(1) **User Interface**. The user interface of Chat2Query offers a wide range of capabilities. Users have the flexibility to import their own data for exploration purposes. Additionally, it features an intuitive interface enabling users to interact with their personal data through natural language. Simultaneously, the interface supports result export in CSV format, enabling accessibility for other applications.

(2) **Visualization**. The component simplifies the process of setting recommended visualization charts suggested by Chat2Query. Users have the option to visualize specific data columns or select from a range of available chart types. It also enables users to customize the appearance of the chart. Ultimately, the generated chart can be saved to any desired location.

(3) **LLM-empowered Generative Engine**. This constitutes the core module of Chat2Query, encompassing the text-to-SQL generator, SQL rewriter, data-to-chart generator, and SQL formatter all under the guidance of LLM. Section II-B provides a comprehensive overview of these modules.

(4) **Cloud Processing Engine**. Chat2Query leverages TiDB Serverless, a fully managed and auto-scaling deployment of the TiDB database. This deployment provides the complete Hybrid Transactional/Analytical Processing (HTAP) capabilities for individuals and organizations. It enables real-time data exploration possibilities to accommodate change workloads.

B. LLM-empowered Generative Engine


Chat2Query Prompt Example

Act as a MySQL expert, you are tasked with writing a SQL statement to answer question based on the following database schema: %s.
The question is: "%s". Let's think step by step and use the following answer template to generate an json file: {
1. Output the revised question, including all necessary keywords. Marked as RQ.
{"RQ": string}
2. List all necessary tables involved in answering the RQ. Mark as NT.
{"NT": string[]}
3. List all necessary columns of NT involved in answering the RQ, do not introduce non-existent columns. Marked as NC.
{"NC": {table: string, Columns: string[]}}
4. Using best practices of MySQL to write a single SQL statement to answer the RQ.
{"sql": string}

Fig. 3. The Illustration Fragment of CoT Prompting for SQL Generator.

Text-to-SQL generator. Figure 3 illustrates a fragment of the SQL generation process for Chat2Query employing the Chain-of-Thought (CoT) [10] prompting. The text-to-SQL process includes the following steps: (1) **Revise questions and identify keywords**: Gain a comprehensive understanding of the task's objective, expected outcomes, and any pertinent conditions. Ensure a clear comprehension of the task's requirements. Recognize all pertinent keywords linked to tables and columns referenced in the clarified task. (2) **Table recall**: In this step, the tables should be listed according to their relevance to the question. Subsequently, the model needs to verify whether all tables have been taken into consideration. (3) **Column recall**: Building upon the outcomes of table recall, we proceed to extract the columns from the potential candidate tables. We devise a zero-shot prompt to facilitate the retrieval of columns. Each candidate table is then presented with a list of its corresponding columns, organized according to their relevance to the question.

SQL Rewriter. The SQL rewriter component in Chat2Query serves the purpose of enhancing the generated SQL queries to ensure accuracy, optimization, and adherence to best practices. It fine-tunes and refines the initial SQL statements produced by the natural language to the SQL generation process. This includes tasks such as identifying and correcting any ambiguous column references and ensuring the proper usage of aliases for table names. This also involves the implementation of suitable query simplification and redundant column elimination techniques. In the past, PingCAP utilized a collection of rewriting rules to govern the operation of TiDB databases, and these rules are conveyed to the LLM using a prompting template.

Data-to-Chart Generator. The data-to-chart generator within Chat2Query is responsible for producing visual representations, often in the form of charts, based on the outcomes derived from the SQL queries. These charts serve as a means of conveying insights to the user in a visually intuitive manner. Furthermore, choose a chart type from the provided options that are best suited for effectively visualizing the data generated by your SQL statement, such as: (1) pie chart; (2) line chart; (3) bar chart; (4) table. This process is facilitated through the application of zero-shot prompting, which aids in the creation of suitable charts that align with user preferences. On the other hand, the visualization component of Chat2Query empowers users with the freedom to explore charts extensively. This entails the selection of chart types and customization of their appearances, thereby allowing users to configure and visualize data in ways that best suit their analytical needs.

SQL Formatter. The SQL formatter component in Chat2Query is responsible for structuring the generated SQL queries in a standardized and readable format. It ensures that the SQL statements are properly indented, aligned, and organized, making them easier for users to comprehend and review. The SQL formatter helps enhance the clarity and consistency of the generated SQL code, contributing to better code quality and reducing the likelihood of syntax errors.

C. The End-to-End User Usability Evaluation of EDA System

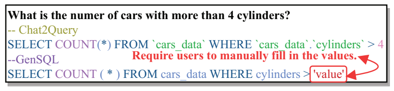
We employed the Spider dataset (200 examples, 1:2:1 for easy:medium:hard) for assessing the usability of various text-to-SQL techniques, including vanilla GPT-3.5 with prompt "Translate the user inquiry into an SQL query" and GenSQL, as a designated SQL generation engine as shown in Table I.

TABLE I
THE END-TO-END USER USABILITY EVALUATION, WHERE THE SYMBOL '-' SIGNIFIES ALMOST USABILITY, '+' INDICATES HIGH USABILITY, AND '·' DENOTES FINELY TUNED USABILITY.

Method	Easy	Medium	Hard
Vanilla GPT-3.5	-	-	-
GenSQL	·	·	-
Chat2Query	**+**	**+**	**+**

When employing vanilla GPT-3.5 as the text-to-SQL engine for our system, it falls short of producing end-to-end executable SQL statements. Users expend a substantial amount of effort on manual adjustments to the generated SQL statements. Similarly, we use text-to-SQL ranking model of GenSQL in our system. We have the following findings in Spider dataset at Section III-C:

(1) For "easy" and "medium" tasks, it necessitates additional user intervention, often requiring the specification of appropriate filtering conditions and ensuring correct column name capitalization to make the generated SQL executable (denoted as '**·**'). Figure 4 gives an example from a Spider dataset with easy difficulty. The Chat2Query generates the accurate SQL statement. In contrast, while GenSQL produces SQL with the correct formal structure, it lacks accurate filtering values, leading to reduced end-to-end user usability.

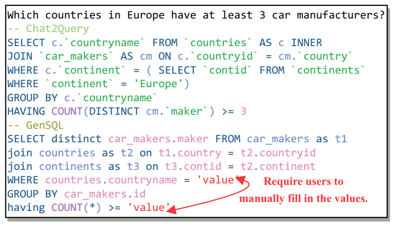


What is the numer of cars with more than 4 cylinders?

```
-- Chat2Query
SELECT COUNT(*) FROM `cars_data` WHERE `cars_data`.`cylinders` > 4
--GenSQL
SELECT COUNT ( * ) FROM cars_data WHERE cylinders > 'value'
```

Fig. 4. The Example between Chat2Query and GenSQL for Easy Difficulty.

(2) GenSQL struggles to produce accurate SQL statements for "hard" tasks, however, such as the column recall phase, often yielding entirely incorrect results (denoted as '-'). In Figure 5, we observe an instance of a Spider dataset characterized by a hard level of complexity. It is evident that the Chat2Query system produces correct SQL statements in this scenario, whereas GenSQL encounters errors such as column recall and filtering conditions. We are unable to populate the *countryname* value even.



```
Which countries in Europe have at least 3 car manufacturers?
-- Chat2Query
SELECT c.`countryname` FROM `countries` AS c INNER
JOIN `car_makers` AS cm ON c.`countryid` = cm.`country`
WHERE c.`continent` = ( SELECT `contid` FROM `continents`
WHERE `continent` = 'Europe')
GROUP BY c.`countryname`
HAVING COUNT(DISTINCT cm.`maker`) >= 3
-- GenSQL
SELECT distinct car_makers.maker FROM car_makers as t1
join countries as t2 on t1.country = t2.countryid
join continents as t3 on t3.contid = t2.continent
WHERE countries.countryname = 'value'
GROUP BY car_makers.id
having COUNT(*) >= 'value'
```

Fig. 5. The Example between Chat2Query and GenSQL for Hard Difficulty.

Despite the inability SQL generation engine of Chat2Query to consistently generate accurate SQL statements across all scenarios, it demonstrates superior end-to-end user usability (denoted as '**+**') compared to tools like GenSQL. Furthermore, we conducted experiments on real-world datasets (c.f., Steam Game and Fortune 500 datasets in Section III) by assessing the availability for SQL generation. In this context, Chat2Query outperformed GenSQL and vanilla GPT-3.5 for end-to-end user usability. We also attained a SQL generation accuracy comparable to that of GenSQL across tasks of varying difficulty levels.

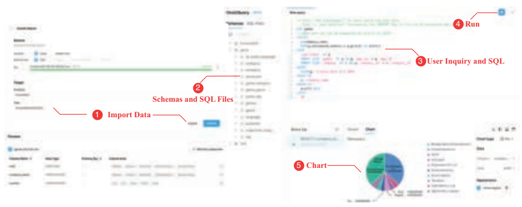


Fig. 6. Demonstration Scenarios of Steam Game Dataset.

III. Demonstration Scenarios

We present a web application to the participants and illustrate Chat2Query in three real-world scenarios, which are presented below. The Steam Game and Fortune 500 were selected for demonstration purposes, as they are frequently employed to assess the effectiveness of EDA tasks within the activities of PingCAP. Meanwhile, these datasets have accumulated a considerable number of natural language to SQL validation scenarios. The Spider is a large-scale complex and cross-domain semantic parsing and text-to-SQL dataset. These scenarios are derived from real-world situations and are readily understandable to the ICDE audience. Considering space constraints, this paper primarily concentrates on illustrating demo scenarios using the Steam Game dataset, while also briefly touching upon those involving others. A demonstration video can be found on YouTube[2].

A. Scenario 1: Steam Game Dataset

To enhance the usability for regular users, Chat2Query offers a conversational dialog interface, enabling user-system interaction. We use the natural language query "*What are the top 10 companies by profit in 2022?*". Figure 6 shows that the user interface primarily encompasses five key functionalities.

❶ **Import Data**. Users can import their personalized data and establish corresponding databases and tables according to specific needs. Users can upload a CSV file to TiDB or import data from S3. Additionally, within the *Preview* interface, users can configure primary keys, column names, data types, and other pertinent information, which is crucial to ensure precise SQL generation.

❷ **Schemas and SQL Files**. The component encapsulates the specific details of columns associated with a table imported by a legitimate user. As a service on TiDB Cloud, Chat2Query only needs to access database schema to generate SQL. *SQL Files* tab stores SQL files produced by users, enabling them to rename and delete these files as needed.

❸ **User Inquiry and SQL**. When provided with a database, users input natural language inquiry into the designated input box and await to generate SQL from the Chat2Query.

❹ **Run**. Click the *Run* button (▶) to execute the generated SQL statement and generate a chart visualization the results. The SQL statement produced by Chat2Query is sent to TiDB Serverless for execution, and the corresponding result is retrieved.

❺ **Chart**. The suggested chart type is provided, and users can customize the chart according to their preferences. In particular, users can select from a variety of chart types; at present, Chat2Query offers options such as pie, line, bar, and table. By conveniently clicking and selecting the replacement data, users can augment their insights. Lastly, users also retain the ability to customize the appearance of the chart. This flexibility empowers users to delve into and adapt the resultant outcomes to meet their specific exploration needs.

[2]https://youtu.be/BcDIj7H6mxw

B. Scenario 2: Fortune 500 Companies Dataset

In this demonstration, we have opted to utilize the queries "*What are the top 10 companies by profit in 2022?*" and "*Which 10 companies had the highest revenue increase from 2018 to 2022?*" as exemplars. For a more comprehensive understanding of features, Please refer to our video for more details.

C. Scenario 3: Spider Dataset

In this demonstration, we have opted to utilize the queries "*Which countries in Europe have at least 3 car manufacturers?*" and "*What are the average and maximum capacities for all stadiums?*" as exemplars to explore the capabilities of Chat2Query. Please refer to our video for more details.

IV. Conclusion

In this paper, we introduced Chat2Query system to provide LLM-empowered exploration data analysis. Chat2Query facilitates online real-time data exploration by utilizing the TiDB Serverless. We have implemented and deployed Chat2Query in the PingCAP production environment. We presented the implementation of Chat2Query and demonstrated on three representative datasets.

Acknowledgements

We thank the anonymous reviewers for their insightful comments and feedback. This work was supported in part by the National Natural Science Foundation of China under Grant No. U22B2020, and in part by PingCAP. Peng Cai, Boyan Niu, and Kai Xu are the corresponding authors.

References

[1] T. Milo and A. Somech, "Automating exploratory data analysis via machine learning: An overview," in *SIGMOD*, 2020, pp. 2617–2622.
[2] P. Ma, R. Ding, S. Wang, S. Han, and D. Zhang, "Xinsight: explainable data analysis through the lens of causality," *SIGMOD*, pp. 1–27, 2023.
[3] Y. Fan, T. Ren, Z. He, X. Wang, Y. Zhang, and X. Li, "GenSQL: A generative natural language interface to database systems," in *ICDE*, 2023, pp. 3603–3606.
[4] T. R. D. G. L. C. R. Z. G. C. Y. J. K. Z. X. W. Yuankai Fan, Zhenying He, "GAR: A generate-and-rank approach for natural language to sql translation," in *ICDE*, 2023, pp. 110–122.
[5] A. Liu, X. Hu, L. Wen, and P. S. Yu, "A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability," *arXiv preprint arXiv:2303.13547*, 2023.
[6] G. Katsogiannis-Meimarakis and G. Koutrika, "A survey on deep learning approaches for text-to-sql," *The VLDB Journal*, pp. 1–32, 2023.
[7] M. B. Roman, "Steam games dataset," 2022. [Online]. Available: https://www.kaggle.com/ds/2109585
[8] "Fortune 500 from 1996 to 2023," https://www.kaggle.com/datasets/rm1000/fortune-500-companies?resource=download.
[9] "Spider dataset," https://yale-lily.github.io/spider.
[10] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *NIPS*, pp. 24 824–24 837, 2022.