Applications and Challenges for Large Language Models: From Data Management Perspective

Meihui Zhang†, Zhaoxuan Ji†, Zhaojing Luo†*, Yuncheng Wu‡, Chengliang Chai†
†*Beijing Institute of Technology*,‡*National University of Singapore*
{meihui_zhang, jizhaoxuan, zjluo, ccl}@bit.edu.cn wuyc@comp.nus.edu.sg

***Abstract*—Data management is indispensable for informed decision-making in the big data era. In the meantime, Large Language Models (LLMs), equipped with billions of model parameters and trained on extensive data corpora, have recently achieved record-breaking results in various real-world applications, such as machine translation, content generation, information retrieval, etc. The emergent abilities of LLMs, e.g., in-context learning and advanced reasoning ability, have great potential to revolutionize data management. In this paper, we first present some promising categories of data management applications where LLMs can be adapted, including data generation, data transformation, data integration, and data exploration. We then discuss the corresponding challenges for such adaption. Finally, we envision potential solutions to these challenges.**

***Index Terms*—Large Language Models, Data Management, Vector Database.**

I. Introduction

In the big data era, the ever-increasing volumes of data generated are indispensable for making better-informed decisions [1]–[4]. This requires proper data management that covers a series of steps, from data storage and data processing to governance of how data is used for analytics. Meanwhile, recently, we have witnessed the success of Large Language Models (LLMs) in various real-world tasks such as machine translation [5], content generation [6], information retrieval [7], etc. With hundreds of billions of model parameters and pre-trained on large-scale corpora (e.g., millions of books, articles, and web pages) [8], LLMs are showing a revolutionary ability to understand, process, and synthesize complex programming [9] and natural languages.

The rapid advancements in LLMs naturally lead to the question of their adaptability for data management applications. Given the significant strides in LLM technology, it is imperative for the data management community to thoroughly examine the intersection between LLMs and data management, uncovering new opportunities and technologies for enhancing data management practices.

Specifically, LLMs possess several distinct advantages that make them particularly suitable for data management. First, the emergent abilities distinguish LLMs from other models. LLMs have the in-context learning (ICL) ability, which can learn from a few examples and directly predict without additional training. While existing machine learning methods in the data management field often have task-specific architectures, which require computationally intensive fine-tuning or retraining the entire model for a new task, ICL is training-free and has the potential to significantly reduce the computational cost of adapting to a new data management task. Furthermore, LLMs have the advanced step-by-step reasoning ability, such as chain-of-thought [10], tree-of-thought [11], etc. LLMs can utilize the prompting mechanism to perform intermediate reasoning steps to derive the final answers, which is particularly crucial for complex data management tasks like multiple-step data processing.

Second, LLMs acquire essential generation skills via pre-training on large-scale corpora, e.g., LLMs can synthesize code from natural language. This capability can be exploited in a wide range of data management applications like Structured Query Language (SQL) generation and data transformation. For example, a data analyst may ask: "Write an SQL code to show the top 10 items that have been searched in the last month". LLMs can generate an SQL draft for the data analyst to verify and refine.

Third, the combination of LLMs and vector databases achieves revolutionary performances in multi-modal data applications, encompassing text, tables, images, videos, and more. LLMs can generate better embedding vectors for multi-modal data, while vector databases can store these vectors and accelerate the query processing with efficient indexing mechanisms [12]. This integration not only facilitates the incorporation of external knowledge into traditional relational databases but also significantly improves the management of diverse, heterogeneous multi-modal data.

In this paper, we envision that LLMs can be adapted to promising applications in different steps of data management that form a pipeline, including data generation, data transformation, data integration, and data exploration, to improve the quality of data management, as shown in Figure 1. With the *ability to synthesize* both text and code by *learning from a few examples*, LLMs are promising for *data generation* tasks such as SQL generation, training data generation, etc. Meanwhile, the *multi-step reasoning ability* like chain-of-though is useful for discovering the transformation logic or rules for different *data transformation* scenarios such as transformation for SQL, tables, and columns. For traditional difficult *data integration* tasks, including entity resolution [13], [14] and column type annotation [15], they have a high demand for understanding the semantics of the underlying tabular data [16]. With the advanced *reasoning ability*, LLMs have the potential for better

* contact author

2024 IEEE 40th International Conference on Data Engineering (ICDE) | 979-8-3503-1715-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDE60146.2024.00441

2375-026X/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDE60146.2024.00441

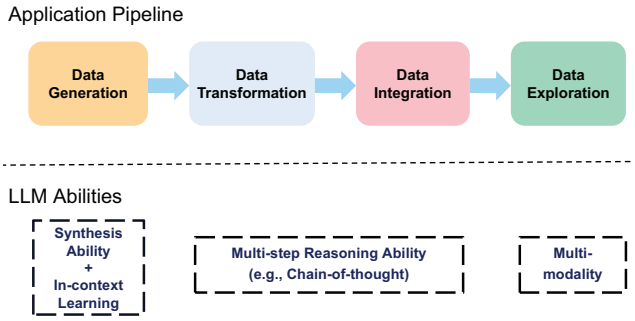

Fig. 1. Data management application pipeline that LLMs can be adapted to.

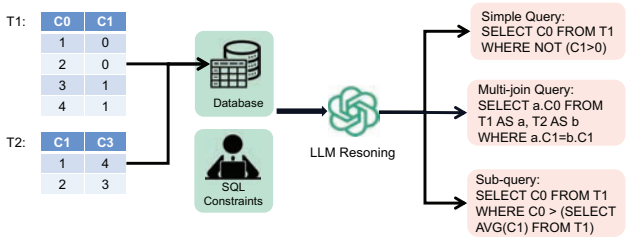

Fig. 2. SQL generation with LLMs. The table information and SQL constraints are input into LLMs. Then, LLMs reason the database tables and output multiple SQL queries that meet the constraints. For example, LLM can generate various types of SQL queries, such as simple query, multi-join query, and sub-query.

table understanding to tackle data integration problems in a more effective way. Moreover, LLMs are proven to have the extraordinary ability to *understand and process multi-modal data*. This would greatly revolutionize our approaches to multi-modal *data exploration*.

While the adaptation of LLMs to data management applications holds great promise, it also presents substantial challenges. We summarize these challenges from the database system's point of view, including *LLM prompt optimization*, *LLM query optimization*, *LLM cache optimization*, *LLM security and privacy concerns*, as well as *LLM output validation*. First, data management users may have different requirements across different domains, it is thus important to optimize the prompts to make them user-friendly and easy to use. Second, data management involves large data volumes and a variety of data types (e.g., multi-modal data), which makes it necessary to ensure cost-efficient and effective LLM queries for wider accessibility. Third, while different data management users may exploit LLMs to process similar tasks, there is great potential to reuse previous LLM queries by caching them. However, given that the LLM queries can hardly be exactly matched, which is different from that in conventional data management, it is challenging to design an efficient and effective LLM cache for data management applications. Fourth, the management of data, especially involving health or financial information, demands stringent privacy protection. This is crucial in light of privacy regulations (GDPR [17], CCPA [18]) and the risks of data exposure during the LLM training and inference stages. This triggers the need to develop privacy-preserving LLM model training and inference. Last, data management applications typically require deterministic and highly reliable solutions. However, the probabilistic nature of LLM outputs poses a challenge to their reliability. Incorporating new technologies, such as interpretable LLMs and human-in-the-loop LLM exploitation, is crucial for validating the LLM outputs for data management applications.

In this paper, we first present a number of promising data management applications where LLMs can make significant contributions. Subsequently, we discuss a series of challenges arising from adapting LLMs to data management applications. In the meantime, we envision new research directions for developing novel technologies to address these challenges.

The rest of the paper is organized as follows. In Section II, we present four categories of applications of LLMs in the data management field. In Section III, we present challenges and opportunities arising from adapting LLMs to data management applications. Finally, we conclude the paper in Section IV.

II. Applications of LLMs in Data Management

In this section, we present a series of promising applications that LLMs can be utilized in the four steps of the data management application pipeline in Figure 1.

A. LLM for Data Generation

Data management tasks usually require a large volume of data [19]. For example, to comprehensively detect the bugs of DBMS, it is important to feed the database with a huge number of SQL queries [20]. For another example, to train better AI4DB models (e.g., learning-based cardinality estimator, query optimizer and so on [21], [22]), it requires substantial training data, like $< query, execution\_time >$ pairs. However, acquiring such vast real data is challenging due to privacy issues and high collection costs (i.e., collecting the execution time or cardinality of complex queries are time-consuming). Next, we will illustrate two important data generation tasks in data management, i.e., SQL generation and training data generation, and demonstrate the application of LLMs in these tasks.

1) SQL Generation: While the research on SQL generation has been improving over the years, there are also some challenges. The first challenge is how to generate complex queries, e.g., SQL queries containing sub-queries or multi-table joins, which need a deep understanding of the underlying databases. The second challenge is how to generate SQL queries that meet certain user-defined constraints [23]. For example, it is significant to generate diverse and correctly executable SQL queries for thoroughly testing the performance of DBMS. Meanwhile, to detect the logic bugs of DBMS, we need to generate some SQL queries with semantic equivalence, which produce the same results [20].

LLMs can play a significant role in generating SQL queries by leveraging their natural language understanding and generation capabilities, as shown in Figure 2. Specifically, LLMs

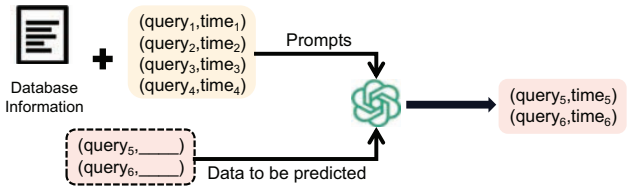

Fig. 3. Training data generation with LLMs. We feed some labeled training data (e.g., the $< query, execution\_time >$ pairs) and database information (e.g., the table schema, table statistical information, etc.) into LLMs. For the coming query (i.e., the data to be predicted), LLMs can assist in predicting its execution time.

can use their powerful reasoning ability to handle complex SQL queries, e.g., LLMs can exploit textual data such as comments and documentation to understand the databases. For the generated SQL queries, LLMs can help users identify and correct errors. They can provide suggestions for fixing syntax errors or generating semantic-equivalence SQL queries. As such, LLMs have the potential to simplify the process of SQL generation.

2) Training Data Generation: Machine Learning (ML) plays a prominent role in data management tasks [24]. For instance, ML-based methods have been proposed for cardinality estimations, query optimization, etc. Meanwhile, tabular data is usually used to train an ML model, where one column is used as the data label and the other columns as features [4], [16], [25]. As mentioned before, collecting the training data for learning-based models is time-consuming, which leads to inefficient training of these models. In addition, labels in tabular data are often missing, which is a common challenge in supervised learning tasks. Consequently, estimating the label of training data is also important in data management tasks. Previously, data programming [26] and data-driven [27] estimating methods have been proposed. They mainly emphasize the labeling accuracy, as opposed to the model generality, i.e., the trained models may not well align with human expectations in some scenarios [28]. For the new data to be predicted, we often need a small amount of labeled data to fine-tune the model, even retrain from scratch.

LLMs can be powerful tools for training data generation. For example, to generate training data for learning-based query optimization, we can input a set of queries and their corresponding execution times into the LLM and instruct it to generate additional examples with corresponding execution time, which is shown in Figure 3.

Additionally, LLMs can assist in annotation for missing fields in existing tabular data. Specifically, we can first serialize the attribute names and values into a natural language string for each row in tabular data. Then, we use prompts to feed a few labeled data (rows with complete data) to LLMs as examples in the few-shot setting. Finally, we can exploit the LLMs with powerful in-context learning (ICL) to infer the missing fields with specific prompts.

Furthermore, LLMs can generate synthetic datasets that mimic the characteristics of real-world tabular data, which can be used to supplement the training data and diversify the use scenarios for the models. After that, we can replace the tabular data with the generated datasets to train the ML model[1].

B. LLM for Data Transformation

Data transformation is a crucial step for data management tasks. It involves converting data from one format or structure into another [29], [30]. For example in healthcare applications [31]–[33], there are various sources of data, such as patients' demographics, diagnoses, lab tests, etc. However, their formats are typically inconsistent, e.g., variations in data units, abbreviations, different data types (structured or unstructured), etc., which makes data analysis challenging. If these data could be converted into a unified format (e.g., relational tables), it would make the analysis more convenient. In the following, we illustrate some promising scenarios where LLMs can be employed for data transformation.

1) Transformation for SQL: Natural Language to SQL (NL2SQL) is a technology that converts natural language queries into SQL queries, making it accessible for non-technical domain experts to analyze and manipulate data. Despite the impressive performance of current NL2SQL methods, their adaptability to real-world scenarios remains limited. The biggest challenge for NL2SQL is how to reason the complex natural language (NL) queries that contain multiple tables and involve implicit matching between the entities in the NL query and the database tables. LLMs have demonstrated remarkable complex reasoning abilities across a range of domains and tasks, and thus they have the potential for complex NL2SQL tasks. Specifically, LLMs can exploit their chain-of-thought ability to produce complex queries step by step given the natural language, i.e., to generate sub-queries first which will then be merged to get the final complex query.

As an extension of the NL2SQL application, the need for converting the natural language to a sequence of SQL queries (i.e., a transaction in DBMS) is also significant. We call this application NL2Transaction. Let us consider an example in the finance field: Alice wants to buy a laptop from Bob, they agree on a price of $1,000, and Bob needs to pay $5 to the express company as the freight. This trading process requires multiple SQL queries to complete, which is known as a transaction in the database. LLMs can help people do this transformation using their powerful step-by-step reasoning abilities to generate a sequence of SQL queries.

2) Transformation for Tables: Relational tables can be easily queried by SQL-based tools, which saves a lot of time for downstream tasks, such as data analytics [4], [34], and decision-making [35], etc. However, in real-world applications, most data does not conform to this data format. It is thus significant to transform different types of data (e.g., XML [36], JSON documents, spreadsheets in Excel) to structured tables.

We illustrate two examples in Figure 4. For transforming semi-structured data to relational tables, we need to have

[1]The real-world tabular data may have missing data or privacy issues, so the generated synthetic datasets can be considered new training datasets for ML models.

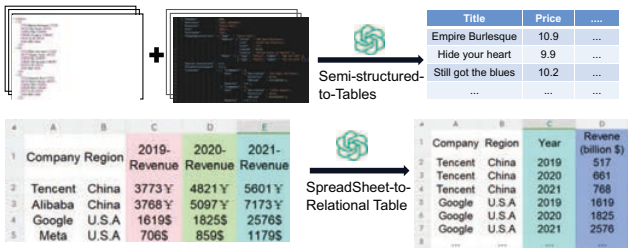

Fig. 4. Transformation for tables. There is massive semi-structured data in the wild, such as XML, JSON documents, etc. It is laborious to query from them. Meanwhile, the spreadsheets in Excel are inconvenient to query with SQL-based tools. LLMs can help transform semi-structured data into structured tables for easier queries.

a deep understanding of the semi-structured data, and then extract the schema information [37], which is difficult. Another example is transforming non-relational tables into relational forms. Tables in the wild are commonly non-relational (e.g., spreadsheet or web tables, which may contain a hierarchical structure, or redundant rows and columns) and are hard to query with SQL-based tools. Prior works [30], [38] emphasize the transformation performance, as opposed to generality. Specifically, current state-of-the-art methods, which adopt ML models, are significantly dependent on the training data, and they often assume that all transformations are composed of some pre-defined operators, which can hardly be satisfied.

LLMs will be a boost to these tasks from two aspects. The first is to transform directly. We can input the source format (e.g., XML and JSON) into LLMs, and design specific prompts to generate the destination format (i.e., relational tables). For example, we can guide LLMs to extract schema information and the corresponding values from unstructured or semi-structured data using prompts and then generate relational tables. The second approach is code synthesis. Most transformation tasks refer to generating a series of operators, e.g., transpose, pivot, explode and so on [30]. We can exploit LLMs to generate the operator sequences so that they can be used to transform other unprocessed data. In this way, we only need to call LLMs once or a few times, which consumes less cost.

3) Transformation for Table Columns: In many data management tasks, we also need to transform two table columns, which is to find the joinable columns between two tables. Note that the joinable columns here mean that the two columns can be transformed into each other by some operations. For example, one date column has the form of "Aug 14 2023", while another date column in the other table has the form of "8/14/2023". These two values refer to the same day, but they cannot be automatically merged.

A natural solution is to generate the transformation program between two columns, where the first step is to mine the patterns of the source column and the destination column. The pattern of "Aug 14 2023" can be expressed as "$< letter > \{3\} < digit > \{2\} < digit > \{4\}$". It can also be expressed as "$Aug \ < digit > \{2\}$ 2023". Obviously, the latter pattern representation has a smaller scope. Overall, it is difficult to find the accurate pattern for a specific column. With the powerful reasoning capacity, LLMs will be competent for mining column patterns to help perform transformation. To be specific, LLMs are able to find the common characteristics of each column, and then use the chain-of-thought ability to generate the transformation rule.

Meanwhile, after the data patterns (such as column patterns and row patterns) of the dataset are generated, we can utilize them to improve data quality. In today's ML systems, data is significant for performance. However, data is often refreshed. Consequently, data quality issues(e.g., data drift and schema drift) [39], [40] may arise, which causes the model to be inaccurate and need to be retrained. To validate whether the data is updated is thus important. Hence, the column patterns discovered by LLMs can help validate the data quality with much more ease.

4) Data Preparation Pipeline: Data preparation pipeline [41], [42] refers to the procedure of processing data for the best performance on downstream ML tasks. For example, in financial data analytics, data scientists aim to predict the trend of a certain stock by extracting relevant information from extensive datasets. This procedure consists of data cleaning, data integration, data extraction, etc., which can be considered as a series of data transformation operations. It is non-trivial for ML practitioners because there are various data operations that lead to a huge search space. Additionally, many domain experts may not have sufficient knowledge of data processing programming languages, such as Python and R. LLMs can be applied to data preparation pipelines in two aspects. On the one hand, LLMs can use the chain-of-thought ability and advanced reasoning abilities [11] to recommend candidate pipelines, significantly reducing the search space. On the other hand, LLMs can help non-technical experts synthesize codes for every operation in the data preparation pipeline.

C. LLM for Data Integration

Data integration is the core of the data management community [43], which includes entity resolution, schema matching, column type annotation, and data cleaning [44]–[47], etc. For example, in the context of retail businesses, various inputs from different individuals may cause issues such as inconsistencies in formatting, as well as missing information, leading retailers to draw inaccurate conclusions from the data. Data integration facilitates more effective management of inventory and revenue data for retailers. While a large number of research has been proposed to solve these problems [16], [48], data integration remains challenging because the proposed solutions are only applicable in certain scenarios.

1) Entity Resolution, Schema Matching, Column Type Annotation and Data Cleaning: These are common and significant tasks in the data management community and LLMs can be naturally adapted to address them. For instance, we can design a prompt as "Are the following entity descriptions the same real-world entity?" in the entity resolution task.

For another example, in the column type annotation task, we first give the following prompts to the LLM "Given the following column types: country, person, date, movie, sports. You need to predict the column type according to the column values. (1) USA||UK||France, this column type is country. (2) Michael Jackson||Beckham||Michael Jordan, this column type is person", and ask "Basketball||Badminton||Table Tennis, this column type is __." With the powerful natural language processing and understanding ability, LLMs offer greater opportunities for performing data integration tasks [49].

2) Table Understanding: This task refers to comprehending the tabular data [50], which includes databases, spreadsheets, and so on. Table understanding is essential for data integration applications and most state-of-the-art data integration methods need to understand tables well [48]. Recent works on table understanding mainly use pre-trained language models (PLMs) like BERT-based models to resolve it [16], [51]. However, there are still some challenges for better table understanding. Specifically, for existing methods, less attention was paid to capturing the semantic and structural information of tables well. Also, PLMs cannot do well with large tables, i.e., tables with a substantial number of rows or columns, due to the input limitations of PLMs. With LLMs, the quality of table understanding can be enhanced in the following ways.

First, while it is common to serialize the tabular data to construct the input data to the PLMs for table understanding, the serialization of prior works is usually simple (e.g., linearization by rows or by columns [51]), overlooking the semantic information of tabular data. LLMs can enhance this process by transforming each row or column of the tabular data into a natural language description, thereby capturing the table's semantic information for improved table understanding.

Second, PLMs benefit from input that includes structural information for reasoning of large volumes of tabular data. SQL queries, which inherently contain both structural and statistical data insights, can be leveraged in this context. Therefore, LLMs can be exploited to transform SQL into natural language expressions that capture both structural and statistical information of data. Subsequently, these expressions and the corresponding query results can be input to PLMs for better table understanding. For example, we can construct the SQL like "SELECT AVG(SALARY) FROM EMPLOYEE", which captures the statistical information of the table. Then, we put this SQL and the corresponding result into LLMs, which output a sentence like "the average salary of all the employees in the EMPLOYEE table is $500". Subsequently, this sentence is used as one piece of training data for PLMs.

Third, to feed the PLMs with large tables, a natural solution is to split the tables into chunks and train every chunk separately. The method of splitting these tables is critical for the model's performance. LLMs can assist in splitting big tables. Specifically, we can uniformly sample some data and give statistical table information to LLMs. Then, LLMs reason about them and provide split suggestions. Additionally, we can let LLMs recommend specific compression methods to reduce the size of tables or to choose representative tuples that will be input to PLMs.

D. LLM for Data Exploration

Data exploration is a key step in mining insights from the data [52], [53]. Nowadays, apart from structured data, a vast amount of multi-modal data are generated and need to be explored. For example, in healthcare applications [54]–[58], electronic medical records (EMR) data consists of structured data, text data, medical image data, etc. However, exploring the multi-modal data using current data management methods is not easy [59], which requires high human costs to integrate multi-modal data and lacks convenient query tools like SQL. With the abilities to process multi-modal data, LLMs can be powerful tools for data exploration. In this section, we present two potential data exploration tasks that LLMs can contribute to, namely, multi-modal data lake management and LLMs as databases.

1) Multi-Modal Data Lake Management: In real-world applications, multi-modal data is usually stored in data lakes. As a result, how to gain insights from the multi-modal data lakes is significant for multi-modal data exploration. However, current methods for exploring multi-modal data have difficulties in processing the data [59], e.g., entity resolution, data transformation, data query, etc. Moreover, SQL queries are not well supported in multi-modal data lakes, which hinders users from effectively exploring the data, such as querying, processing, and analyzing the data.

With LLMs, users can query multi-modal data with much more ease. Specifically, items of different modalities in the data lakes can be encoded in the same embedding space, by using an LLM that has been pre-trained with a huge volume of multi-modal data, e.g., relational databases, documentation, log files, knowledge graphs, etc. With this unified representation of different modalities, the query problem can be turned into the similarity search problem among embedding vectors. Different from traditional search which uses table titles and column names, the query process using LLMs can take the semantics into consideration.

2) LLM as Databases: Traditional queries over databases are on relational data, which cannot work for other modalities of data such as text and images. However, with the increasing amounts of multi-modal data, the querying and processing of these non-relational data types are becoming necessary.

One promising application of adapting LLMs for querying multi-modal data is to query LLMs as databases [60]. This will enable users to effectively query data across different modalities using SQL queries. Specifically, SQL queries can be decomposed by query optimization as in traditional databases. The decomposed sub-queries extract multi-modal information from corresponding LLMs, just like searching from tables in traditional databases.

E. Further Discussion

In previous sections, we have envisioned a series of promising data management applications where LLMs can excel. However, there are some applications that LLMs are not

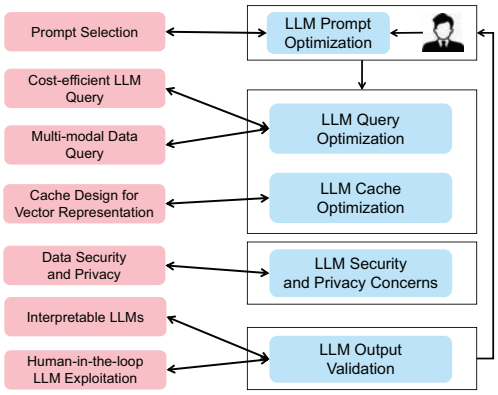

Fig. 5. An overview of the challenges and opportunities.

suitable for. The first category is numerical data understanding tasks. LLMs possess strong text reasoning capabilities, but they have poorer comprehension of numerical data. For instance, when floating-point data is inputted into LLMs as strings, it often results in loss of its inherent meanings [61]. The second category is data management applications that require low latency and high throughput, such as cardinality estimation, query optimization and transaction processing [62]. LLMs are very expensive at inference [63], which may result in poor performance on these applications.

III. Challenges and Opportunities

From the above data management applications where LLMs can be adapted, we note there are technical challenges that need to be tackled for better data management quality. From the database systems' perspective, we outline these challenges, encompassing LLM prompt optimization, LLM query optimization, LLM cache optimization, LLM security and privacy concerns, and LLM output validation. An overview of the challenges and opportunities is shown in Figure 5.

Typically, users in data management may have diverse requirements across different domains, emphasizing the importance of optimizing prompts for user-friendliness and ease of use. Furthermore, the large volume of data with diverse data types (e.g., multi-modal data) inherent in data management necessitates cost-efficient and effective LLM queries to enhance accessibility. Also, since different users may utilize LLMs for similar tasks, reusing previous LLM queries through caching presents potential benefits. However, the non-exact match nature of LLM queries poses challenges in designing an efficient and effective LLM cache. Moreover, the use of LLMs in handling sensitive, proprietary organizational data necessitates stringent data protection techniques, e.g., privacy-preserving LLM model training and inference for data management applications. Last, the high reliability requirements in data management applications pose new challenges to LLMs, which are probabilistic models in nature. The incorporation of new technologies, such as interpretable LLMs and human-in-the-loop LLM exploitation, is crucial for validating LLM outputs in the context of data management applications. In what follows, we elaborate on these key challenges and discuss potential solutions for effectively and efficiently adapting LLMs in data management.

A. LLM Prompt Optimization

In data management, different users may have different requirements; therefore, how to provide appropriate and effective prompts for users to easily select and use is important. While the design of prompts has been studied in the context of Natural Language Processing (NLP) [64], prompting in data management is more challenging as it demands a high degree of domain knowledge. In the example of utilizing LLMs to generate different SQL queries for logic bug detection in databases, we need to provide comprehensive database schema information and subsequently create sufficient SQL query examples to assist LLMs in generating more varied and precise results. Existing prompts for data management applications are mainly generated manually or based on templates, which is inefficient. Also, it is typically not effective because insufficient examples are provided. In this case, we envision that selecting appropriate historical prompts and then using them to generate new prompts automatically can be a good choice. A key point here is historical prompt selection, for which we discuss several opportunities below.

Considering prompts are typically represented as vectors, vector databases are suitable for storing historical prompts for selection. A promising research direction for better prompt selection is to develop efficient and effective indexes. While the common practice is to construct indexes for vector similar search [65], [66], the vector with the highest similarity does not necessarily indicate the optimal prompt for improving LLM performance. We may need to design an indexing method to cater to the optimal prompt. For instance, we can incorporate the performance of LLMs as a target for the learned index. Meanwhile, determining which historical prompts should be stored within a limited budget is also important. We envision that reinforcement learning algorithms can be designed to determine the most promising prompts that can increase the performance of LLMs.

B. LLM Query Optimization

Query optimization is a crucial component for enhancing performance in traditional data management applications. It is also important for LLM queries since data management typically involves large data volumes and various data types (e.g., multi-modal data). Next, we identify two key challenges in LLM query optimization for data management applications and present potential solutions to address them.

1) Cost-efficient LLM Query: The large volume of data in data management applications can be very costly for LLMs to process. For example, LLMs can facilitate the process of transforming semi-structured data to structured data as described in Section II-B. However, there may be hundreds of thousands of HTML documents in reality. Given the latest price of GPT-3.5 Turbo is $0.001/1k input tokens, and GPT4 is

TABLE I
PRELIMINARY RESULTS ON LLM CASCADE.

Model	babbage-002	gpt-3.5-turbo	gpt-4	cascade
Accuracy	27.5%	82.5%	92.5%	92.5%
API Cost	$0.0327	$0.0417	$0.8	$0.0512

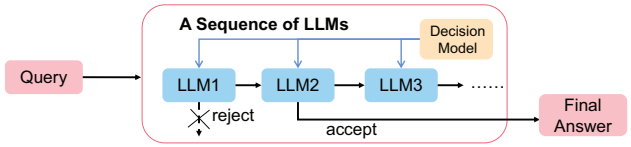

Fig. 6. The procedure of the LLM cascade. A new query is sent to a sequence of LLMs with size and cost spanning from small to large. Additionally, a decision model is required to determine whether the LLM results are acceptable.

$0.03/1k input tokens [67], reasoning these HTML documents with LLMs can pose significant costs for both individual users and organizations. Consequently, how to reduce the costs of applying LLMs to data management applications is significant. We discuss two opportunities for cost-efficient LLM queries.

LLM Cascade. As mentioned above, different LLMs have various costs. And also, they have their own strengths for various data management applications. If the queries that need domain knowledge without much deep reasoning, the performance of small LLMs (with less parameters) may be similar to, or even surpass, that of large LLMs. For example, in the healthcare data labeling [33], we need to label whether the patients have a certain disease. For diseases with obvious symptoms, such as heart disease, only a small LLM is needed for classification. Overall, using smaller LLMs that mimic the performance of large LLMs can help cost reduction. However, how to select appropriate LLMs for a given data management task is the biggest challenge. A possible solution is the LLM cascade, as shown in Figure 6. We can send a query to a sequence of LLMs. These models vary in size and cost, spanning from small to large. A decision model can be trained to determine whether a more expensive and larger LLM is needed. If the less expensive LLMs can consistently produce reliable results, there will be substantial cost savings.

Some preliminary results are shown in Table I. We first select 40 queries from HotpotQA [68] and then assess the performance of these queries on three LLMs and LLM cascade. The results show that the performance of LLMs improves as the cost (model complexity) increases, e.g., the accuracy of gpt-4 is 92.5% while babbage-002 is 27.5%. LLM cascade achieves performance similar to gpt-4 but with significantly lower costs, which demonstrates that LLM cascade is a promising solution.

Query Decomposition and Combination. In real-world applications, a proxy connected to popular LLMs such as GPT-3.5 Turbo or GPT-4 often receives multiple simultaneous queries. Many of these queries may be similar, presenting an opportunity to reduce LLM usage costs by exploiting this query similarity. For example, several users need to translate natural language queries into SQL, i.e., NL2SQL described in Section II-B. The LLM provider may encounter multiple queries simultaneously, as exemplified below:

- Q_1:"What are the names of stadiums that had concerts in 2014 or had sports meetings in 2015?";
- Q_2:"What are the names of stadiums that had the most number of concerts in 2014";
- Q_3:"Show the names of stadiums with most number of sports meetings in 2015";
- Q_4:"Show the names of stadiums that had concerts in 2014 and had sports meetings in 2015 ";
- Q_5:"Show the names of stadiums that had concerts in 2014 but did not have sports meetings in 2015".

Given the similarities among these queries, they can be decomposed into sub-queries for more efficient query handling. The detailed decomposition is shown in Figure 7. For example, Q_1 and Q_2 have a common sub-query, i.e, Q_{11} and Q_{21} are the same sub-query. Likewise, Q_3 and Q_4 also have a common sub-query. As a result, we can call LLMs only once for the common sub-query to minimize the total costs of LLMs. However, determining which (sub-)queries to input into LLMs is non-trivial. Specifically, the total costs (i.e., the total number of input tokens) of decomposed sub-queries (i.e., Q_{11}, Q_{12}, Q_{13}) is larger than the original query Q_1. That is, query decomposition may even increase the LLM costs. But, if we consider all five queries, it is cost-efficient to input the sub-queries into LLMs. Therefore, efficient algorithms need to be designed to find the set of (sub-)queries with minimum costs that can cover all the original queries.

Furthermore, query combination can also decrease the costs of LLMs. Specifically, to enhance the quality of LLM outputs, it is common to provide some examples in the prompt. For multiple queries, there may be overlapping examples in their respective prompts. To save the cost, several queries can be combined into a single query to eliminate redundant examples. In this context, we need to explore how to combine queries with similar examples, aiming to reduce the costs.

Last, query decomposition and combination can be exploited together to reduce the costs of querying LLMs for data management applications. We can first decompose complex queries into simpler ones[2]. After the query decomposition, the same or similar sub-queries can thus be found. Subsequently, we exploit query combinations for queries or sub-queries to further reduce the costs of LLMs. As an additional benefit, query decomposition and combination can improve the accuracy of LLM outputs. Specifically, query decomposition can decompose complex queries into simpler sub-queries, which are easier for LLMs to handle. Additionally, after query combination, the number of examples in the prompt will increase for each query, which can help LLMs reason the query better.

To evaluate the above vision, we craft a set of NL2SQL data inspired by the Spider dataset [69], and then use DAIL-SQL [70] to measure the performance of query decomposi-

[2]The query decomposition methods are different for different data management tasks.

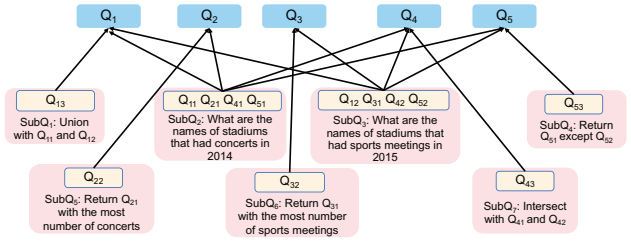

Fig. 7. An illustration of query decomposition. The queries colored blue are the original queries. The sub-queries colored yellow are the decomposed queries, e.g., Q_1 can be decomposed into Q_{11}, Q_{12}, and Q_{13}. Different queries may have the same sub-queries. For example, Q_{11} and Q_{21} are the same sub-query, so they only need to call the LLM once.

TABLE II
PRELIMINARY RESULTS ON QUERY DECOMPOSITION AND COMBINATION.

	Origin	Decomposition	Decomposition +Combination
Accuracy	79%	91%	91%
API Cost	\$0.435	\$0.289	\$0.129

tion and combination. The preliminary results are shown in Table II. We find that query decomposition can reduce costs while improving accuracy. This is because query decomposition can reuse sub-queries, thereby decreasing costs, and in the meantime, the sub-queries tend to be simpler, increasing the possibility of converting them into correct SQL. Additionally, query combination can further reduce the cost by eliminating redundant examples.

2) Multi-modal Data Query: LLMs can be exploited to manage multi-modal data because they have the extraordinary ability to understand and process the data. However, querying multi-modal data is challenging as it needs to extract relevant information from various modalities, striving to provide a reasonable result. For example, a user may ask a query that encompasses data from various modalities, including text, images, and tables. In this scenario, multi-modal data is typically stored as embedding vectors and retrieved by different similarity functions. Hence, the quality of embedding vectors and the performance of vector search are two important factors to consider for improving the effectiveness and efficiency of adapting LLMs to multi-modal data management.

For the quality of embedding vectors, we need to consider the granularity of embeddings. Specifically, for tables, an embedding can represent a table or specific rows (columns) of the table. For text, an embedding may represent a document or specific paragraphs of a certain document, and for images, an embedding can represent a single image or several similar images. Varied granularities can influence query performance differently. Consequently, we envision the necessity of choosing an appropriate granularity for different data modalities to precisely represent the data information.

Next, we should consider how to query effectively and efficiently. For multi-modal data applications, merely using embedding vectors for querying may lose important information, and similar vectors may not represent related information. For example, given the query "Could Prof. Michael Jordan play basketball", there are texts in the data lake "Michael Jordan, the greatest basketball player of all time, found the secret to success" and a table containing information about university professors, including details about Michael Jordan. The embedding vectors of the query and the texts are similar but not relevant, whereas the table is related to the query. So only using vector search is not enough to query accurately. Supplement information is needed to assist the query, e.g., attribute information such as entity types, to filter some inaccurate items. However, how to efficiently search in this scenario is also a challenge, which is known as attribute filtering [71]. For this hybrid search that involves both vector and non-vector data, one key consideration is the order of filtering. If there is a small number of candidates after the attribute search, then it is better to search by attributes first. Conversely, the vector search should be performed first. An adaptive mechanism should be designed to determine the order of attribute search and vector data search for different scenarios. We envision that learning-based methods can mitigate this limitation. For example, we can extract some significant features of the searched data and historical queries, and then train a classification model to predict which order to use for a new query. We can also exploit reinforcement learning-based methods to make decisions based on past query workloads.

Another challenge for attribute filtering comes from the approach of "vector search first". Specifically, we denote k as the number of items returned by the vector search. In reality, all the k items may not meet the attribute constraints, leading to null value returned. So in production environments, k is often set as a large number to improve the accuracy, which may degrade the efficiency of attribute filtering. A promising solution is to develop machine learning models to predict an appropriate k value for each query based on data distribution and query workload or to take advantage of hyperparameter optimization methods like Bayesian optimization to obtain a good k value. Recent works [72], [73], which propose to tune the knobs used in approximate nearest neighbor algorithms through learning-based methods, are a good starting point.

C. LLM Cache Optimization

In reality, different data management users may process similar tasks. For example, LLMs can aid users with limited programming proficiency in generating code for each operation in the data preparation pipeline, such as generating the codes of normalization, feature selection, etc. These operations may be reused by different users, leading to potentially redundant queries for code generation. As a result, caching the previous queries and the corresponding responses from LLMs can improve the efficiency and effectiveness of similar queries. On one hand, if a new query bears a high degree of similarity to the cached responses or the cached queries, it is possible that the responses of these similar cached queries can be reused to reduce the response time and LLM costs, i.e., there is no need to call LLMs again. On the other hand, the cached

queries can be used to augment the new query, which is related to the challenge of prompt selection described above. Overall, by caching the commonly accessed history queries or responses, better results are likely provided with reduced response time [74]. However, there are two key factors we need to consider regarding for effective and efficient caching of vector representations.

First, different from traditional cache systems [75], [76], which utilize an exact match between the new query and cached queries, for LLMs, since the stored queries are in the form of vectors[3], the matching relies on the comparison between vectors. In such a scenario, the exact match is not effective. In contrast, identifying similar query vectors instead of exactly the same query vector is a more practical solution. However, how to determine the cache hits for the LLM cache is challenging. Since similarity matching is adopted here, how to define the similarity function is thus important. In addition, we need to set the appropriate similarity threshold for determining whether the history query is matched and this similarity threshold should be different for various scenarios.

Second, how to design a memory-efficient caching algorithm is also challenging. In most cases, the memory budget is limited. We need to evict some useless data when the cache space is full. However, traditional caching algorithms, such as LRU and LFU [77], are not suitable as the scenario here is more intricate. As described above, the cached data can be (1) reused or (2) used as additional prompts for the new query, both of which refer to cache hits. However, the two cases are of different importance for LLMs. Specifically, case (1) does not need to call LLMs again, but case (2) needs to call LLMs. Consequently, they should have different weights when considering eviction. For example, we can assign the cached data of case (1) with a larger weight and evict the cached data with the smallest weight when the cache is full. Additionally, since query decomposition helps enhance the similarity between queries as described in Section III-B, caching sub-queries could improve the hit ratio. As a result, after LLMs generate the results for the queries, we need to decide whether to cache the original queries and sub-queries, or refrain from caching based on the likelihood of future access. Predictive methods, such as machine learning models, can be designed to predict the probability of future access.

Table III shows some preliminary results. We use the same dataset as in LLM Cascade in Section III-B. Note that we randomly select 10 queries and query them twice to verify the cache performance. In Table III, Cache(O) means we only cache original queries, and Cache(A) means we cache both original and sub-queries. It is observed that caching can reduce the cost and Cache(A) can further improve performance. This is because Cache(O) may cache incorrectly answered queries, which are not helpful in improving performance. However, in Cache(A), caching sub-queries, which exhibits higher accuracy, is beneficial for enhancing overall performance.

[3]Vectors can be regarded as "search key" of the cache, where both the original queries and responses are also stored.

TABLE III
PRELIMINARY RESULTS ON LLM CACHE OPTIMIZATION.

	w/o Cache	Cache(O)	Cache(A)
Accuracy	77.5%	77.5%	85%
API Cost	\$1.123	\$0.842	\$0.887

D. LLM Security and Privacy Concerns

A crucial consideration in data management is to ensure that user's data security and privacy is preserved for complying with the increasingly strict regulations, such as GDPR [17] and CCPA [18]. For example, when doctors utilize LLMs to transform patient's semi-structured data (e.g., diagnostic report in XML format) or unstructured data (e.g., discharge summary) to structured data (e.g., the CSV format), the patient's health data needs to be carefully protected in both training stage and inference stage due to its high sensitivity. There are three main challenges for privacy-preserving LLMs in data management applications.

First, LLMs are typically deployed on the cloud or hosted by third-party services, and users access the LLMs via API requests with specific input data for inference. However, as mentioned in the above example, the doctors need to send the whole table of the patient's health data to LLMs, which is often not acceptable in practice. A straightforward solution is to adopt open-source LLMs so that users can deploy the services within their own regions. However, the open-sourced LLMs may not be suitable and effective for various data management applications. Another approach is to use privacy-preserving technologies, such as trusted execution environment (TEE) (e.g., Intel SGX [78], [79]) or cryptographic encryptions [80] to ensure the cloud can provide services without knowing the input data. Nevertheless, the TEE technique is shown to be vulnerable to various side channel attacks [81], [82] while the cryptographic technique incurs huge communication and computation overhead. It calls for research to design secure TEE algorithms that can resist various attacks and lightweight cryptographic techniques for efficient LLM inference.

Second, for applications that require customized LLMs, users need to utilize their own databases to train or fine-tune the LLMs. For example, the doctors may want to train (or fine-tune) a specific LLM to do data transformation for high accuracy. Nonetheless, one user may not have sufficient data for such training or fine-tuning. A natural solution is data collaboration [83], where multiple users pool their data together for more meaningful insights. Federated learning (FL) [84], [85] has emerged as a promising paradigm for multiple users to collaboratively train or fine-tune a machine learning model [86] without disclosing the private data to each other. However, how to securely and efficiently support the LLM training under the FL paradigm still needs to be investigated. Furthermore, the users tend to be heterogeneous with regard to data distributions, qualities, quantities, and computation capabilities. This makes the design space of the FL strategies for LLMs complicated and challenging. A potential solution is to use the reinforcement learning technique to

adjust the FL training strategies for LLMs adaptively.

Third, since LLMs for the aforementioned data management applications could be trained over a set of private datasets, the LLMs must contain sensitive information about the training datasets (e.g., health data). We note that there are numerous attacks, such as model inversion attacks [87], membership inference attacks [88], and feature inference attacks [89], which allow malicious users to extract sensitive information from the original training datasets in the inference stage. To address these issues, a potential solution is to integrate differential privacy (DP) [90] into the training process of LLMs [91], such that the trained models are indistinguishable with respect to whether a specific data is in or not in the training dataset. Still, it is important to design new algorithms that inject minimal noise into the training process while maximizing the model utility of LLMs.

E. LLM Output Validation

Data management tasks typically have a high demand for the reliability of the data. For example, in the data integration step, e.g., data cleaning, an error will make the data less usable [92]. Even 10% error may make the data meaningless for real-world applications like healthcare analytics. Consequently, the LLM outputs for data management applications must be of high quality and should be verified and validated before being used.

However, given that LLMs are probabilistic in nature and the outputs are provided with uncertainty, how to ensure that the results generated by LLMs are trustworthy and reliable is challenging. In contrast, data management tasks involve less uncertainty, and the errors generated by LLMs will greatly affect the quality of adapting LLMs to data management tasks. We envision two research directions to validate the LLM outputs for data management tasks.

1) Interpretable LLMs: Interpretable techniques that provide easy-to-understand interpretations for the model are valuable tools for validating the LLM outputs. We can design specific interpretable mechanisms such as causal inference, attention learning and Bayesian modeling for the LLMs. Meanwhile, we input database-specific data, e.g., database documents (database manual, white paper, blogs, etc.), database logs, database constraints, and structural information to the model as external knowledge to guide the training and fine-tuning, so that the model can integrate the database knowledge to generate database-specific explanations for validation. For this direction, we need to consider that the formats and forms of interpretation should be different and customized for various data management applications.

2) Human-in-the-loop LLM Exploitation: Feedback [93] is an important source of information that can be incorporated to validate the LLM outputs for data management applications. We envision that the human-in-the-loop techniques developed by the database community [94] can be a viable option. We can define a score function, and then utilize crowdsourcing [95] for scoring the LLM outputs. Specifically, we can invite humans to participate in different reasoning steps of LLMs, (e.g., prompts can be provided for humans to interact with the LLM in each intermediate reasoning step), gathering feedback and eliciting suggestions for LLM output validation.

IV. Conclusions

The LLM technologies have revolutionized a wide range of real-world applications with their impressive emergent abilities. While data management is indispensable for informed decisions in the big data era, it is thus important for the data management community to explore how LLMs can contribute to better data management. In this paper, we first present a number of data management applications where LLMs can be adapted, and subsequently, we discuss the corresponding challenges and envision research directions for these challenges. We expect this paper to be one of the first steps towards exploring the relationships between LLMs and data management, and we encourage members from the data management field to join us!

Acknowledgment

We would like to thank the anonymous reviewers for their constructive comments. This work is supported by National Key Research and Development Program of China (2022YFB2405700), National Natural Science Foundation of China (62072033) and National Natural Science Foundation of China (62050099). Yuncheng Wu's work is supported by the National Research Foundation, Singapore under its Emerging Areas Research Projects (EARP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. Chengliang Chai is supported by the NSFC (62102215), CCF-Huawei Populus Grove Fund (CCF-HuaweiDB202306).

References

[1] C. Mohan, "Big data: Hype and reality," *Datenbanksysteme für Business, Technologie und Web*, 2015.

[2] B. C. Ooi, K.-L. Tan, S. Wang, W. Wang, Q. Cai, G. Chen, J. Gao, Z. Luo, A. K. Tung, Y. Wang *et al.*, "Singa: A distributed deep learning platform," in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 685–688.

[3] Z. Luo, S. Cai, J. Gao, M. Zhang, K. Y. Ngiam, G. Chen, and W.-C. Lee, "Adaptive lightweight regularization tool for complex analytics," in *International Conference on Data Engineering*, 2018, pp. 485–496.

[4] Z. Luo, S. Cai, Y. Wang, and B. C. Ooi, "Regularized pairwise relationship based analytics for structured data," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–27, 2023.

[5] M. Ghazvininejad, H. Gonen, and L. Zettlemoyer, "Dictionary-based phrase-level prompting of large language models for machine translation," *arXiv preprint arXiv:2302.07856*, 2023.

[6] Y. Cai, S. Mao, W. Wu, Z. Wang, Y. Liang, T. Ge, C. Wu, W. You, T. Song, Y. Xia *et al.*, "Low-code llm: Visual programming over llms," *arXiv preprint arXiv:2304.08103*, 2023.

[7] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in *International Conference on Machine Learning*, 2020, pp. 3929–3938.

[8] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He *et al.*, "A comprehensive survey on pretrained foundation models: A history from bert to chatgpt," *arXiv preprint arXiv:2302.09419*, 2023.

[9] Y. Lu, S. Chaudhuri, C. Jermaine, and D. Melski, "Data-driven program completion," *arXiv preprint arXiv:1705.09042*, 2017.

[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," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[11] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *arXiv preprint arXiv:2305.10601*, 2023.

[12] J. Mohoney, A. Pacaci, S. R. Chowdhury, A. Mousavi, I. F. Ilyas, U. F. Minhas, J. Pound, and T. Rekatsinas, "High-throughput vector similarity search in knowledge graphs," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 1–25, 2023.

[13] Y. Li, J. Li, Y. Suhara, J. Wang, W. Hirota, and W.-C. Tan, "Deep entity matching: Challenges and opportunities," *Journal of Data and Information Quality*, vol. 13, no. 1, pp. 1–17, 2021.

[14] Y. Li, J. Li, Y. Suhara, A. Doan, and W.-C. Tan, "Deep entity matching with pre-trained language models," *arXiv preprint arXiv:2004.00584*, 2020.

[15] Y. Suhara, J. Li, Y. Li, D. Zhang, Ç. Demiralp, C. Chen, and W.-C. Tan, "Annotating columns with pre-trained language models," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 1493–1503.

[16] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data," *arXiv preprint arXiv:2005.08314*, 2020.

[17] European Commission, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)," 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[18] "California consumer privacy act. bill no. 375 privacy: personal information: businesses. https://leginfo.legislature.ca.gov/." 2018.

[19] I. Absalyamov, M. J. Carey, and V. J. Tsotras, "Lightweight cardinality estimation in lsm-based systems," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 841–855.

[20] M. Rigger and Z. Su, "Testing database engines via pivoted query synthesis," in *USENIX Symposium on Operating Systems Design and Implementation*, 2020, pp. 667–682.

[21] P. Sioulas and A. Ailamaki, "Scalable multi-query execution using reinforcement learning," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1651–1663.

[22] A. Dutt, C. Wang, A. Nazi, S. Kandula, V. Narasayya, and S. Chaudhuri, "Selectivity estimation for range predicates using lightweight models," *Proceedings of the VLDB Endowment*, vol. 12, no. 9, pp. 1044–1057, 2019.

[23] L. Zhang, C. Chai, X. Zhou, and G. Li, "Learnedsqlgen: Constraint-aware sql generation using reinforcement learning," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 945–958.

[24] W. Wang, M. Zhang, G. Chen, H. Jagadish, B. C. Ooi, and K.-L. Tan, "Database meets deep learning: Challenges and opportunities," *ACM Sigmod Record*, vol. 45, no. 2, pp. 17–22, 2016.

[25] Z. Luo, S. Cai, C. Cui, B. C. Ooi, and Y. Yang, "Adaptive knowledge driven regularization for deep neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 8810–8818.

[26] B. Denham, E. M. Lai, R. Sinha, and M. A. Naeem, "Witan: unsupervised labelling function generation for assisted data programming," *Proceedings of the VLDB Endowment*, vol. 15, no. 11, pp. 2334–2347, 2022.

[27] H. Zhang, L. Cao, S. Madden, and E. Rundensteiner, "Lancet: labeling complex data at scale," *Proceedings of the VLDB Endowment*, vol. 14, no. 11, 2021.

[28] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, and X. Hu, "Data-centric ai: Perspectives and challenges," in *Proceedings of the 2023 SIAM International Conference on Data Mining*, 2023, pp. 945–948.

[29] Y. He, X. Chu, K. Ganjam, Y. Zheng, V. Narasayya, and S. Chaudhuri, "Transform-data-by-example (tde) an extensible search engine for data transformations," *Proceedings of the VLDB Endowment*, vol. 11, no. 10, pp. 1165–1177, 2018.

[30] P. Li, Y. He, C. Yan, Y. Wang, and S. Chauduri, "Auto-tables: Synthesizing multi-step transformations to relationalize tables without using examples," *arXiv preprint arXiv:2307.14565*, 2023.

[31] H. Wang, Z. Luo, J. W. Yip, C. Ye, and M. Zhang, "Ecggan: A framework for effective and interpretable electrocardiogram anomaly detection," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 5071–5081.

[32] K. Zheng, G. Chen, M. Herschel, K. Y. Ngiam, B. C. Ooi, and J. Gao, "Pace: learning effective task decomposition for human-in-the-loop healthcare delivery," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2156–2168.

[33] K. Zheng, S. Cai, H. R. Chua, M. Herschel, M. Zhang, and B. C. Ooi, "Dyhealth: making neural networks dynamic for effective healthcare analytics," *Proceedings of the VLDB Endowment*, vol. 15, no. 12, pp. 3445–3458, 2022.

[34] M. Wawrzoniak, I. Müller, R. Fraga Barcelos Paulus Bruno, and G. Alonso, "Boxer: Data analytics on network-enabled serverless platforms," in *Conference on Innovative Data Systems Research*, 2021.

[35] S. Cai, K. Zheng, G. Chen, H. Jagadish, B. C. Ooi, and M. Zhang, "Arm-net: Adaptive relation modeling network for structured data," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 207–220.

[36] P. Bohannon, X. Dong, S. Ganguly, H. F. Korth, C. Li, P. Narayan, and P. Shenoy, "Rolex: relational on-line exchange with xml," in *Proceedings of the 2003 International Conference on Management of Data*, 2003, pp. 673–673.

[37] C. Lei, A. Quamar, V. Efthymiou, F. Özcan, and R. Alotaibi, "Hermes: data placement and schema optimization for enterprise knowledge bases," *The VLDB Journal*, vol. 32, no. 3, pp. 549–574, 2023.

[38] Z. Jin, M. R. Anderson, M. Cafarella, and H. Jagadish, "Foofah: Transforming data by example," in *Proceedings of the 2017 International Conference on Management of Data*, 2017, pp. 683–698.

[39] M. Gertz, M. T. Özsu, G. Saake, and K.-U. Sattler, "Data quality on the web (dagstuhl seminar 03362)," 2021.

[40] S. Sadiq, T. Dasu, X. L. Dong, J. Freire, I. F. Ilyas, S. Link, M. J. Miller, F. Naumann, X. Zhou, and D. Srivastava, "Data quality: The role of empiricism," *ACM SIGMOD Record*, vol. 46, no. 4, pp. 35–43, 2018.

[41] Y. Li, X. Wang, Z. Miao, and W.-C. Tan, "Data augmentation for ml-driven data preparation and integration," *Proceedings of the VLDB Endowment*, vol. 14, no. 12, pp. 3182–3185, 2021.

[42] Z. Luo, S. H. Yeung, M. Zhang, K. Zheng, L. Zhu, G. Chen, F. Fan, Q. Lin, K. Y. Ngiam, and B. C. Ooi, "Mlcask: Efficient management of component evolution in collaborative data analytics pipelines," in *International Conference on Data Engineering*, 2021, pp. 1655–1666.

[43] A. Halevy, A. Rajaraman, and J. Ordille, "Data integration: The teenage years," in *Proceedings of the 2006 International Conference on Very Large Data Bases*, 2006, pp. 9–16.

[44] J. Zhang, Z. Luo, Q. Xu, and M. Zhang, "Pa-feat: Fast feature selection for structured data via progress-aware multi-task deep reinforcement learning," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 2023, pp. 394–407.

[45] R. Fu, Y. Wu, Q. Xu, and M. Zhang, "Feast: A communication-efficient federated feature selection framework for relational data," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–28, 2023.

[46] Z. Cai, C. Jermaine, Z. Vagena, D. Logothetis, and L. L. Perez, "The pairwise gaussian random field for high-dimensional data imputation," in *2013 IEEE 13th International Conference on Data Mining*, 2013, pp. 61–70.

[47] Z. Zhong, M. Zhang, J. Fan, and C. Dou, "Semantics driven embedding learning for effective entity alignment," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 2127–2140.

[48] N. Tang, J. Fan, F. Li, J. Tu, X. Du, G. Li, S. Madden, and M. Ouzzani, "Rpt: relational pre-trained transformer is almost all you need towards democratizing data preparation," *arXiv preprint arXiv:2012.02469*, 2020.

[49] A. Halevy and J. Dwivedi-Yu, "Learnings from data integration for augmented language models," *arXiv preprint arXiv:2304.04576*, 2023.

[50] N. Chen, L. Shou, M. Gong, J. Pei, C. You, J. Chang, D. Jiang, and J. Li, "Bridge the gap between language models and tabular understanding," *arXiv preprint arXiv:2302.09302*, 2023.

[51] H. Dong, Z. Cheng, X. He, M. Zhou, A. Zhou, F. Zhou, A. Liu, S. Han, and D. Zhang, "Table pre-training: A survey on model architectures, pre-training objectives, and downstream tasks," *arXiv preprint arXiv:2201.09745*, 2022.

[52] H. Doraiswamy, E. Tzirita Zacharatou, F. Miranda, M. Lage, A. Ailamaki, C. T. Silva, and J. Freire, "Interactive visual exploration of spatio-

temporal urban data sets using urbane," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 1693–1696.
[53] P. Sinthong and M. J. Carey, "Exploratory data analysis with database-backed dataframes: A case study on airbnb data," in *2021 IEEE International Conference on Big Data*, 2021, pp. 3119–3129.
[54] C. Lee, Z. Luo, K. Y. Ngiam, M. Zhang, K. Zheng, G. Chen, B. C. Ooi, and W. L. J. Yip, "Big healthcare data analytics: Challenges and applications," *Handbook of large-scale distributed computing in smart healthcare*, pp. 11–41, 2017.
[55] J. Dai, M. Zhang, G. Chen, J. Fan, K. Y. Ngiam, and B. C. Ooi, "Fine-grained concept linking using neural networks in healthcare," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 51–66.
[56] S. Ahmetaj, V. Efthymiou, R. Fagin, P. G. Kolaitis, C. Lei, F. Özcan, and L. Popa, "Ontology-enriched query answering on relational databases," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 17, 2021, pp. 15 247–15 254.
[57] K. Yang, Z. Luo, J. Gao, J. Zhao, B. C. Ooi, and B. Xie, "Lda-reg: Knowledge driven regularization using external corpora," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5840–5853, 2022.
[58] Z. Luo, S. Cai, G. Chen, J. Gao, W.-C. Lee, K. Y. Ngiam, and M. Zhang, "Improving data analytics with fast and adaptive regularization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 2, pp. 551–568, 2019.
[59] Z. Chen, Z. Gu, L. Cao, J. Fan, S. Madden, and N. Tang, "Symphony: Towards natural language query answering over multi-modal data lakes," in *Conference on Innovative Data Systems Research*, 2023, pp. 8–151.
[60] M. Saeed, N. De Cao, and P. Papotti, "Querying large language models with sql," *arXiv preprint arXiv:2304.00472*, 2023.
[61] A. Müller, C. Curino, and R. Ramakrishnan, "Mothernet: A foundational hypernetwork for tabular classification," *arXiv preprint arXiv:2312.08598*, 2023.
[62] S. Wang, D. Maier, and B. C. Ooi, "Lightweight indexing of observational data in log-structured storage," *Proceedings of the VLDB Endowment*, vol. 7, no. 7, pp. 529–540, 2014.
[63] Z. Liu, J. Wang, T. Dao, T. Zhou, B. Yuan, Z. Song, A. Shrivastava, C. Zhang, Y. Tian, C. Re *et al.*, "Deja vu: Contextual sparsity for efficient llms at inference time," in *International Conference on Machine Learning*, 2023, pp. 22 137–22 176.
[64] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–21.
[65] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi, "Vector approximation based indexing for non-uniform high dimensional data sets," in *Proceedings of the 2000 International Conference on Information and Knowledge Management*, 2000, pp. 202–209.
[66] H. D. Chon, D. Agrawal, and A. E. Abbadi, "Range and k nn query processing for moving objects in grid model," *Mobile Networks and Applications*, vol. 8, pp. 401–412, 2003.
[67] "Openai api pricing," 2023, https://openai.com/pricing.
[68] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," *arXiv preprint arXiv:1809.09600*, 2018.
[69] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," *arXiv preprint arXiv:1809.08887*, 2018.
[70] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, "Text-to-sql empowered by large language models: A benchmark evaluation," *arXiv preprint arXiv:2308.15363*, 2023.
[71] C. Wei, B. Wu, S. Wang, R. Lou, C. Zhan, F. Li, and Y. Cai, "Analyticdb-v: A hybrid analytical engine towards query fusion for structured and unstructured data," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3152–3165, 2020.
[72] C. Li, M. Zhang, D. G. Andersen, and Y. He, "Improving approximate nearest neighbor search through learned adaptive early termination," in *Proceedings of the 2020 International Conference on Management of Data*, 2020, pp. 2539–2554.
[73] P. Zhang, B. Yao, C. Gao, B. Wu, X. He, F. Li, Y. Lu, C. Zhan, and F. Tang, "Learning-based query optimization for multi-probe approximate nearest neighbor search," *The VLDB Journal*, vol. 32, no. 3, pp. 623–645, 2023.
[74] F. Bang, "Gptcache: An open-source semantic cache for llm applications enabling faster answers and cost savings," in *3rd Workshop for Natural Language Processing Open Source Software*, 2023.
[75] M. Altinel, Q. Luo, S. Krishnamurthy, C. Mohan, H. Pirahesh, B. G. Lindsay, H. Woo, and L. Brown, "Dbcache: Database caching for web application servers," in *Proceedings of the 2002 International Conference on Management of Data*, 2002, pp. 612–612.
[76] J. DeBrabant, A. Pavlo, S. Tu, M. Stonebraker, and S. Zdonik, "Anti-caching: A new approach to database management system architecture," *Proceedings of the VLDB Endowment*, vol. 6, no. 14, pp. 1942–1953, 2013.
[77] D. Lee, J. Choi, J.-H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "On the existence of a spectrum of policies that subsumes the least recently used (lru) and least frequently used (lfu) policies," in *Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 1999, pp. 134–143.
[78] V. Costan and S. Devadas, "Intel SGX explained," *IACR Cryptology ePrint Archive*, p. 86, 2016.
[79] T. Lee, Z. Lin, S. Pushp, C. Li, Y. Liu, Y. Lee, F. Xu, C. Xu, L. Zhang, and J. Song, "Occlumency: Privacy-preserving remote deep-learning inference using SGX," in *MobiCom*, 2019, pp. 46:1–46:17.
[80] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *USENIX*, S. Capkun and F. Roesner, Eds., 2020, pp. 2505–2522.
[81] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *IEEE S&P*, 2015, pp. 640–656.
[82] J. V. Bulck, F. Piessens, and R. Strackx, "Nemesis: Studying microarchitectural timing leaks in rudimentary CPU interrupt logic," in *CCS*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds., 2018, pp. 178–195.
[83] Y. Wang, Y. Wu, X. Chen, G. Feng, and B. C. Ooi, "Incentive-aware decentralized data collaboration," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 158:1–158:27, 2023.
[84] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 2090–2103, 2020.
[85] Y. Wu, N. Xing, G. Chen, T. T. A. Dinh, Z. Luo, B. C. Ooi, X. Xiao, and M. Zhang, "Falcon: A privacy-preserving and interpretable vertical federated learning system," *Proc. VLDB Endow.*, vol. 16, no. 10, pp. 2471–2484, 2023.
[86] Z. Zhang, Y. Yang, Y. Dai, Q. Wang, Y. Yu, L. Qu, and Z. Xu, "Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models," in *ACL*. Association for Computational Linguistics (ACL), 2023, pp. 9963–9977.
[87] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *CCS*, 2015, pp. 1322–1333.
[88] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," *IEEE S&P*, pp. 3–18, 2016.
[89] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi, "Feature inference attack on model predictions in vertical federated learning," in *2021 IEEE 37th International Conference on Data Engineering*, 2021, pp. 181–192.
[90] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds., 2016, pp. 308–318.
[91] Z. Xu, Y. Zhang, G. Andrew, C. A. Choquette-Choo, P. Kairouz, H. B. McMahan, J. Rosenstock, and Y. Zhang, "Federated learning of gboard language models with differential privacy," *arXiv preprint arXiv:2305.18465*, 2023.
[92] F. Nargesian, A. Asudeh, and H. Jagadish, "Responsible data integration: Next-generation challenges," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 2458–2464.
[93] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.
[94] G. Li, "Human-in-the-loop data integration," *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 2006–2017, 2017.
[95] E. Estellés-Arolas and F. González-Ladrón-de Guevara, "Towards an integrated crowdsourcing definition," *Journal of Information science*, vol. 38, no. 2, pp. 189–200, 2012.