

# LLM for Data Management

Guoliang Li, Xuanhe Zhou, Xinyang Zhao

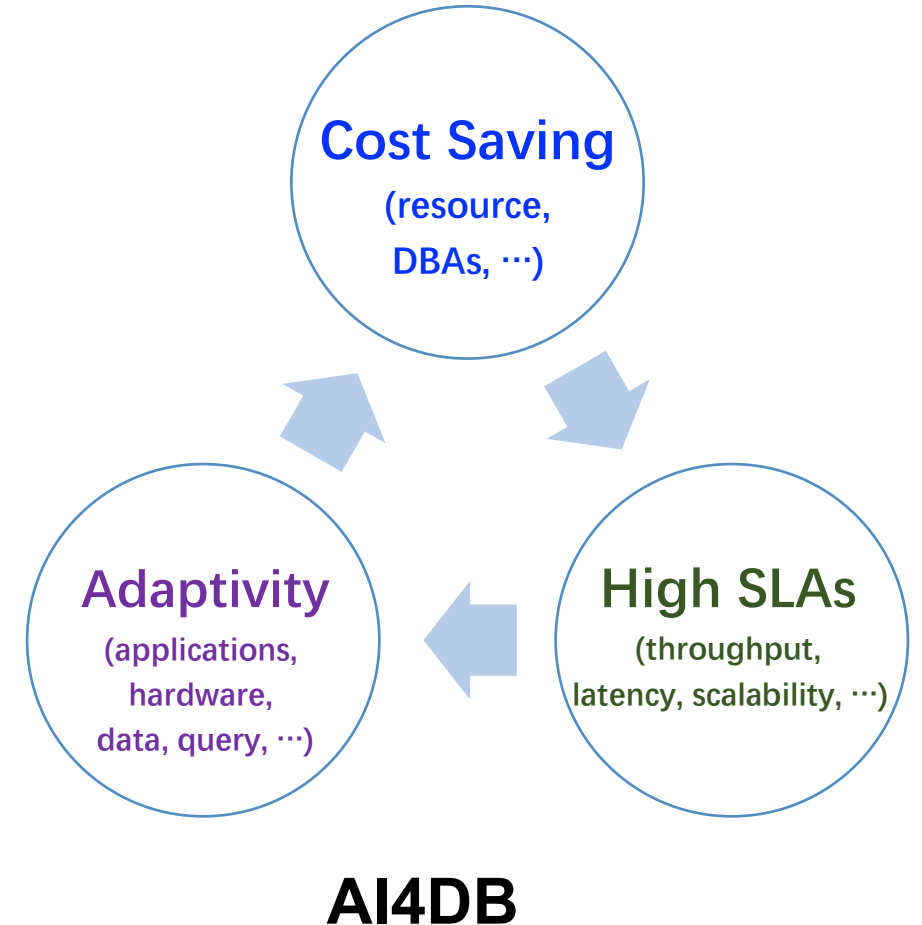
Department of Computer Science, Tsinghua University

<https://github.com/code4DB/LLM4DB>



# AI4DB/ML4DB

- **Cost Saving: Manual → Autonomous**
  - Auto Knob Tuner: ↓ Maintenance cost
  - Auto Index Advisor: ↓ Optimization latency
- **High SLAs: Heuristic → Intelligent**
  - Intelligent Optimizer: ↓ Query plan costs
  - Intelligent Scheduler: ↑ Workload performance
- **Adaptivity: Empirical → Data-Driven**
  - Learned Index: ↑ Data access efficiency
  - Learned Layout: ↑ Data manipulation efficiency



# Challenges of AI4DB

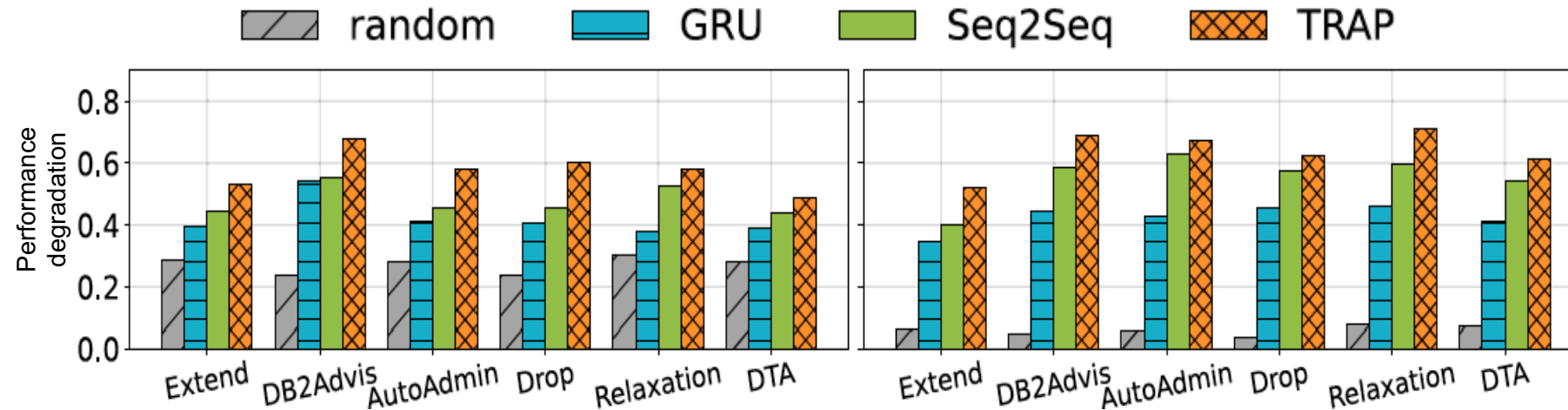
## ❑ Challenges in Traditional AI4DB

### • Adaptivity

- Dynamic changing **schema**
- Dynamic changing **data**
- Dynamic changing **workload**
- Dynamic changing **hardware**

### • Generalization

- **Cold-start**
- **High-quality** training data
- **Interpretability**



Workload changing, 38.9% performance degradation for learned index tuning

# Challenges of AI4DB

## ❑ Challenges in Traditional AI4DB

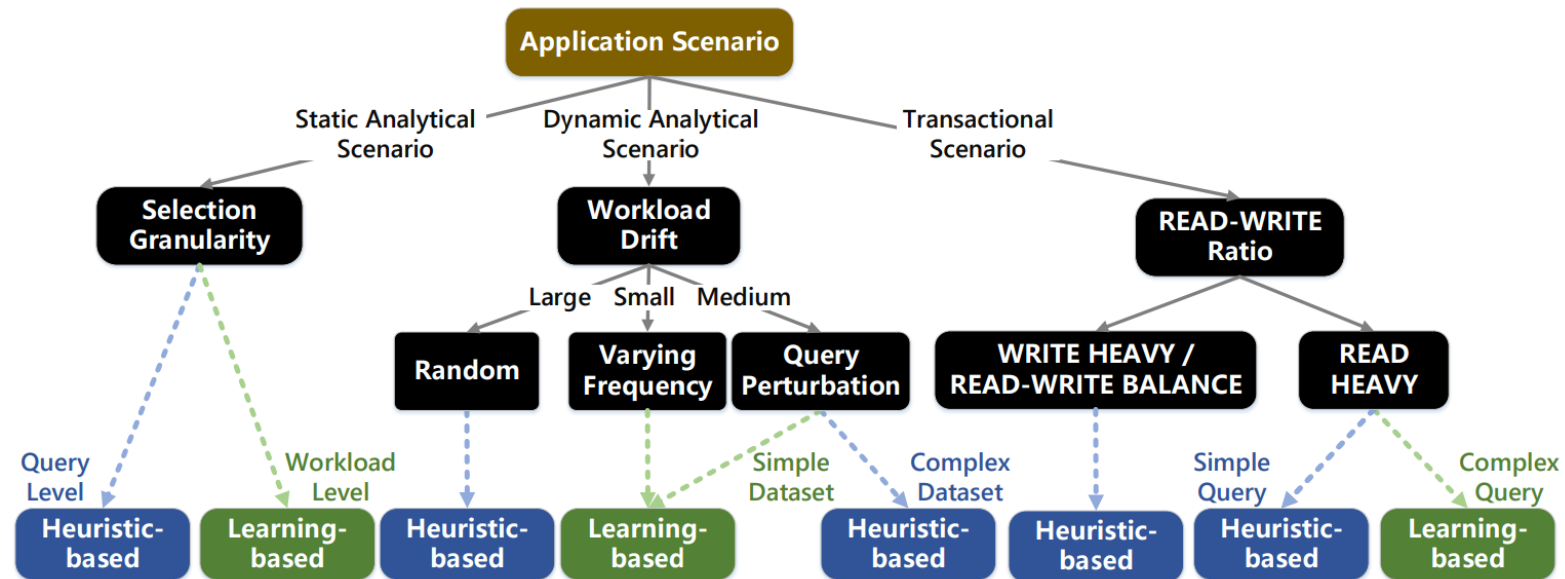
### • Adaptivity

- Dynamic changing **schema**
- Dynamic changing **data**
- Dynamic changing **workload**
- Dynamic changing **hardware**

### • Generalization

- **Cold-start**
- **High-quality** training data

### • Interpretability

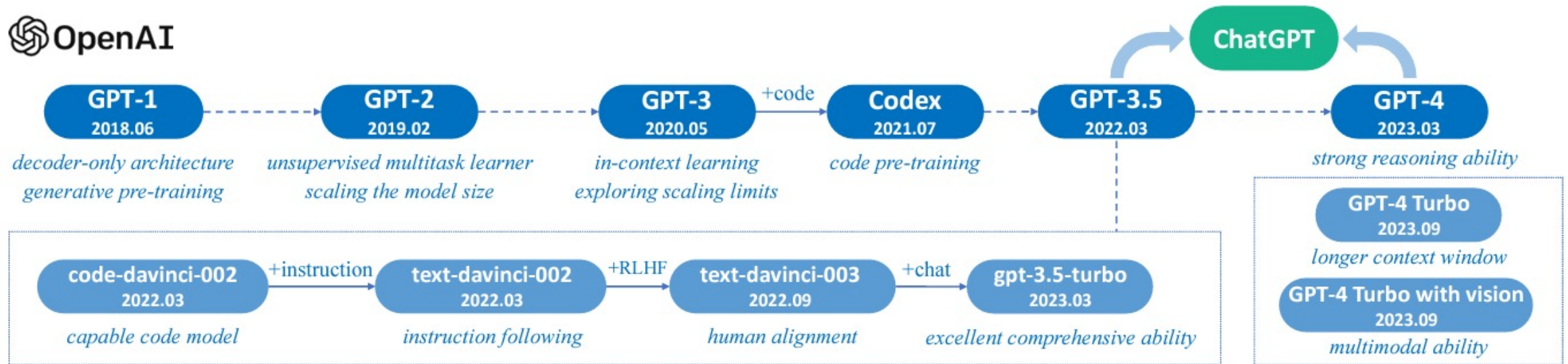


Traditional Learned Methods May not Work for some Scenarios.

# Motivations of LLM for Data Management

## □ Excellent performance and generalization capability

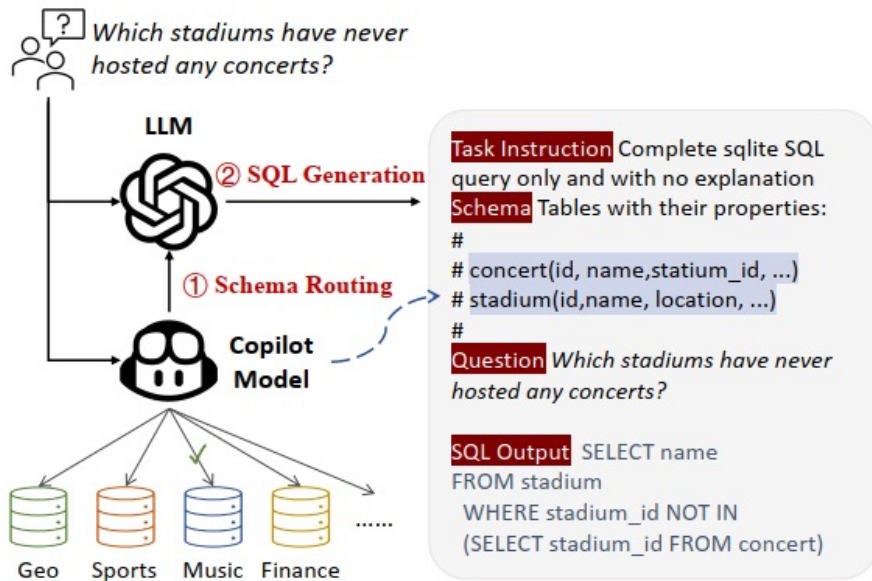
- Support various natural and programming languages
- Improved reasoning ability (v.s. traditional PLM)
- Solve various real-world tasks (e.g., coding, report writing)



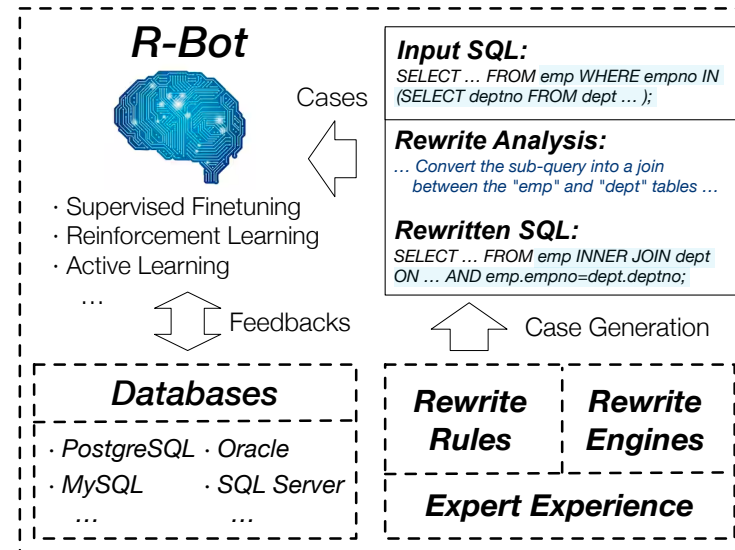
# Motivations of LLM for Data Management

## □ Opportunities of LLM for data management

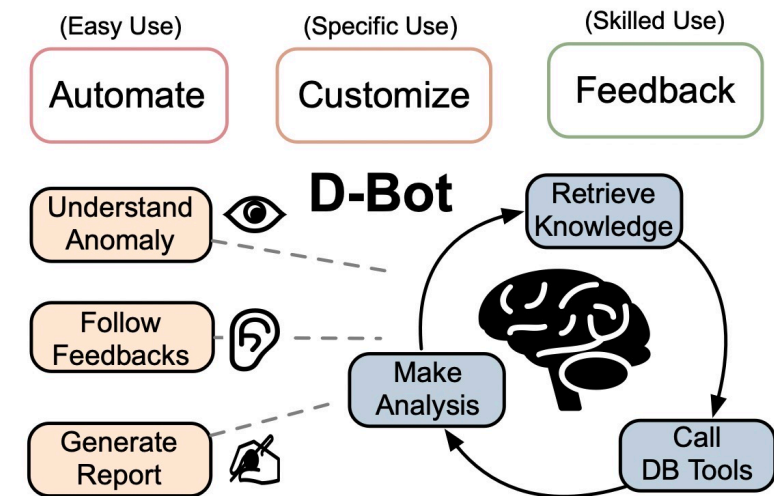
- Improved usability (e.g., Text2SQL)
- Improved performance (e.g., Query Rewrite)
- Improved maintainability (e.g., Database Diagnosis)
- Improved interpretability (e.g., Database Tuning)



Text2SQL



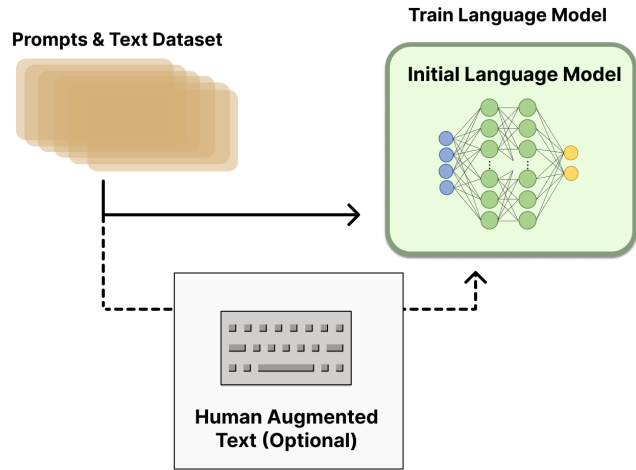
Query Rewrite



Diagnosis

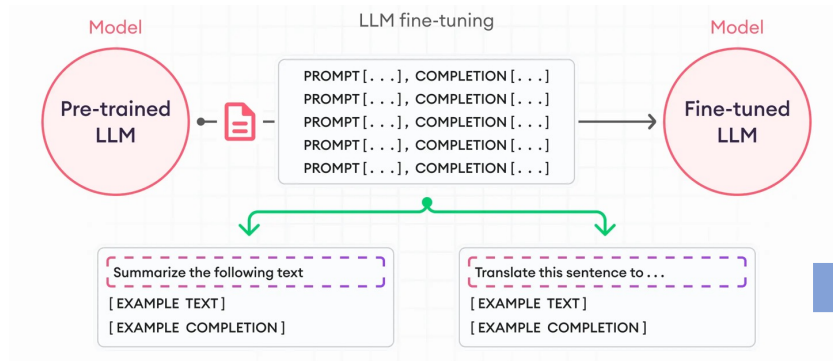
# Different Stages of LLM

## 1. (Incremental) Pretraining



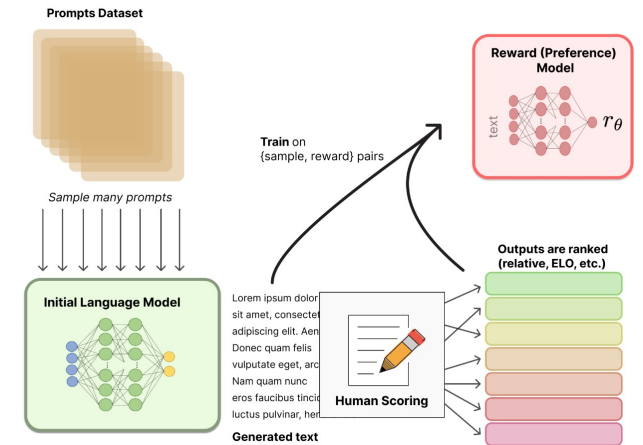
- *Common Knowledge Acquisition*
- *Understanding Diverse Texts*

## 2. (SFT) Finetuning



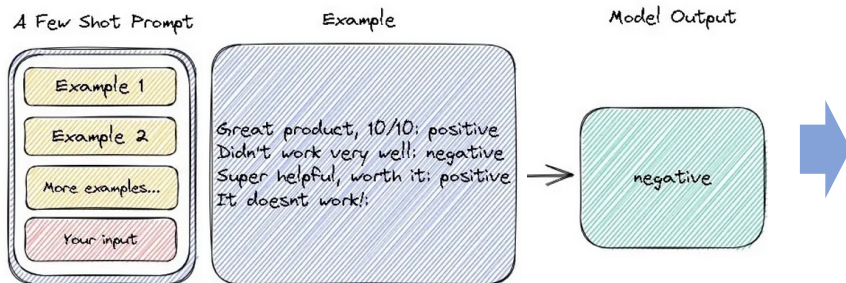
- *Instruction Following*
- *Task Adaption like Traslation/Q&A*

## 3. (RLHF) Finetuning



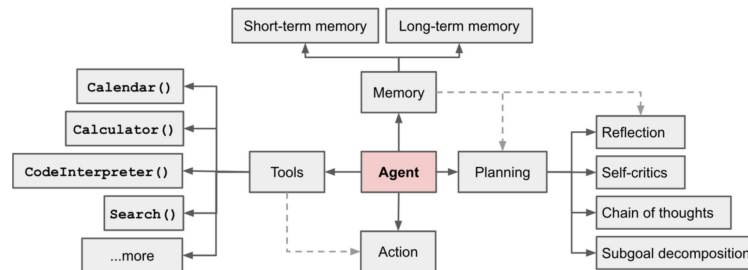
- *Align with human preferences*

## 4. Prompting



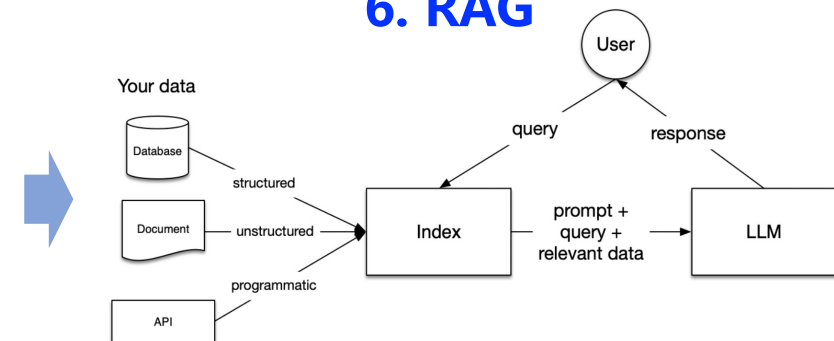
- *Context Comprehension*
- *Learn from demo examples*

## 5. Agent



- *LLM system equipped with reasoning, tools, and memory*

## 6. RAG



- *External Knowledge Integration*
- *Contextual Relevance / QA Accuracy*

# LLM Pre-Training

## □ Pretrain LLM as the foundation model for database

### Tasks

### Text corpus



(Self-supervised)  
Training

### Pretrained LM



Adaptation

Question  
Answering



Text  
Classification



Information  
Retrieval



**Transformer-based LLM:**  
*Predict the next word  
given a sequence of  
previous text*



Doc 1

Doc 2

Doc N

List of documents

Language Model

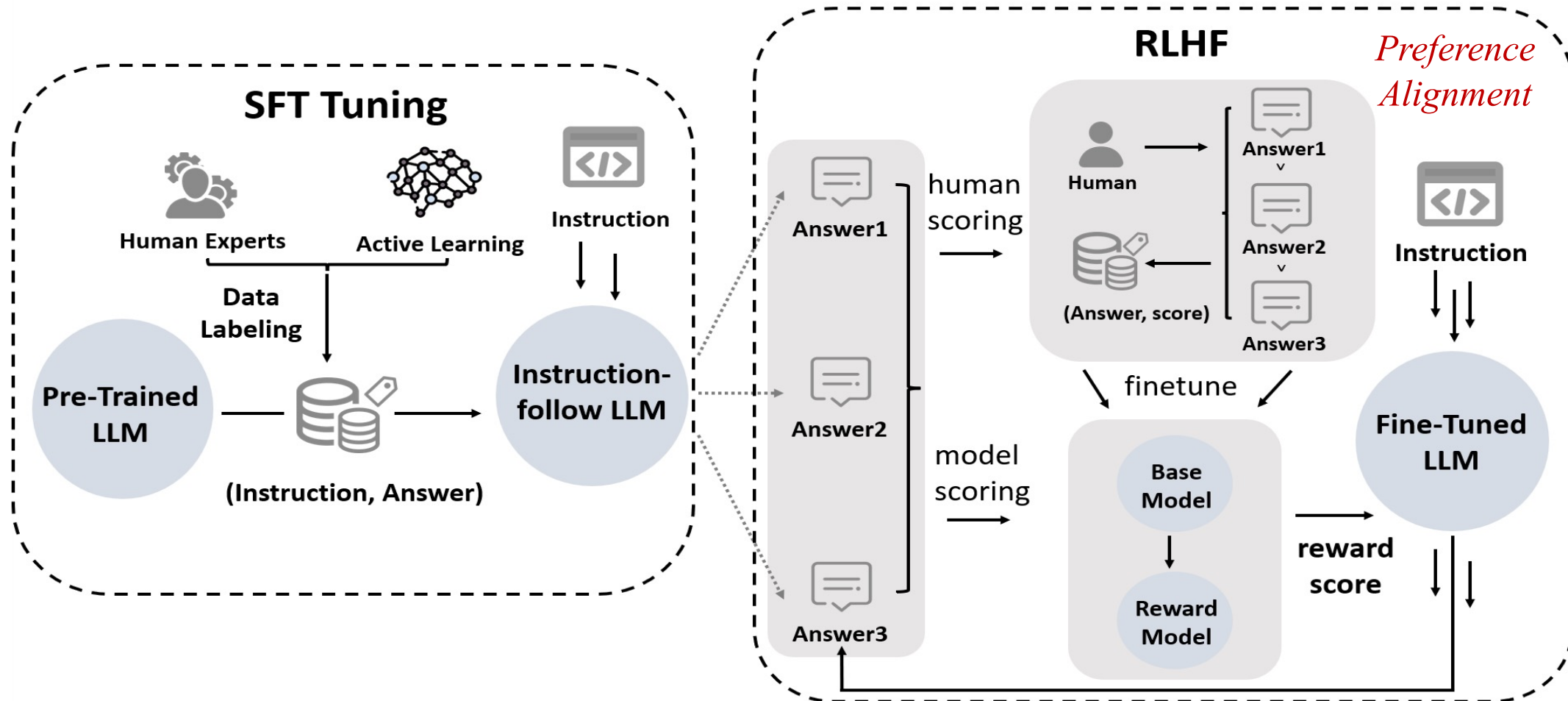


Pretrain the LM



# LLM Finetuning

- ❑ Finetuning LLM over labeled dataset is used for instruction-following and understanding task-specific knowledge
- ❑ RLHF is used for human feedback and alignment



# Prompt for LLM Inference

---

## □ Input text for LLM to generate response or execute a task

- **Simple Prompt**

- **(task)** "Explain the theory of relational tables."

- **Contextual Prompt**

- **(context)** "Undergraduate students are studying database concepts for the first time and is curious about fundamental theories."
- **(task)** "Explain the theory of relational table in a way beginners can understand."

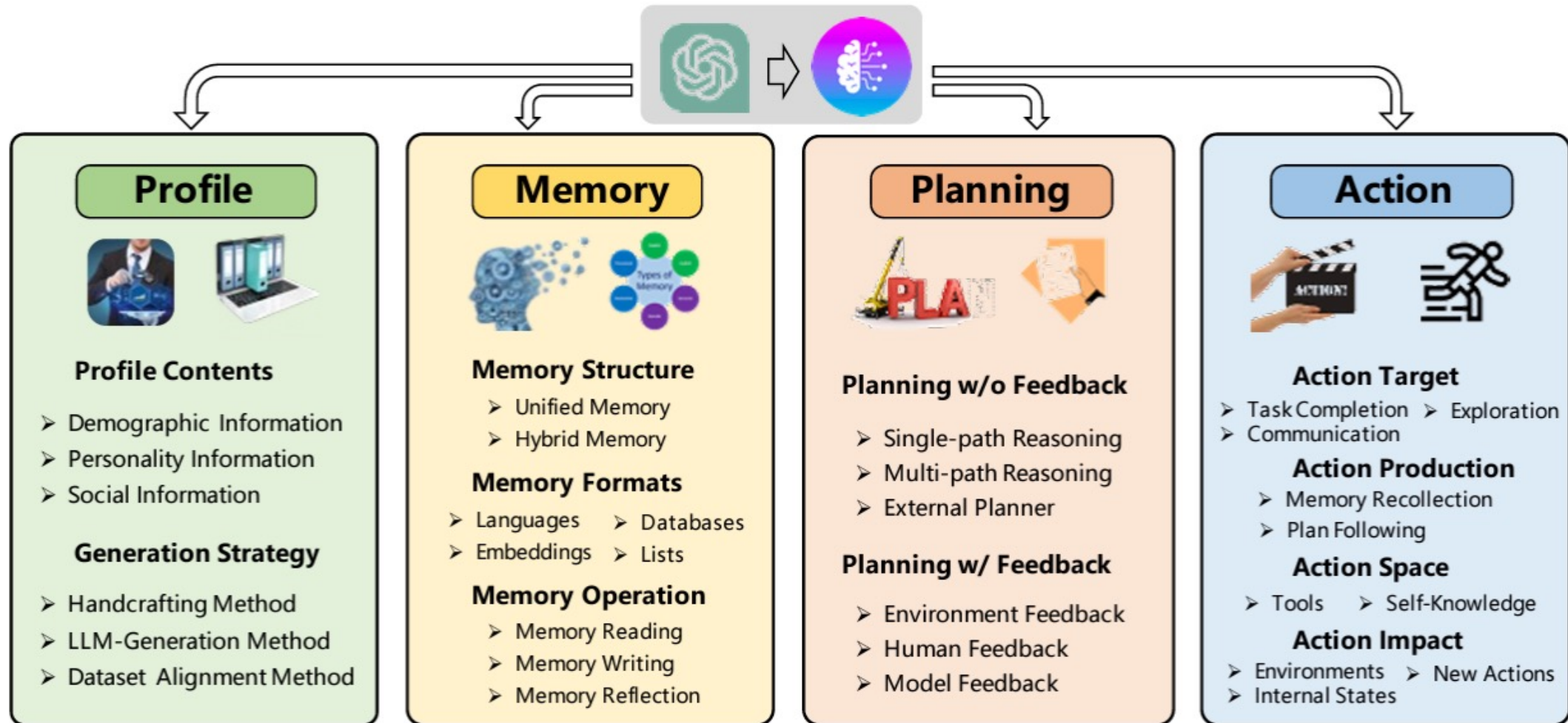
- **Contextual Prompt with Instructions**

- **(context)** " Undergraduate students are studying database ..."
- **(task)** "Explain the theory of relational tables ..."
- **(instructions)** "1. Make sure the explanation is clear and engaging for someone new to databases; 2. Limit the explanation to a few paragraphs with examples."

- **Contextual Prompt with Instructions + Demonstration Examples ...**

# LLM Based Autonomous Agent

- LLM Agent: **Perceiving** the surrounding environment, **planning**, **executing** actions to complete tasks, and **memorize** past executions



# RAG for LLM Inference

## ❑ Drawbacks of LLMs

- Hallucination
- Outdate information
- Low efficiency in LLM training
- Weak reasoning capability

## ❑ Practical Requirements

- Domain-Specific Accurate Q&A
- Frequent Data Update
- Explainability of Responses
- Controllable Cost
- Data Privacy Protection

**Question:** What color of my cat's eyes?

**Correct Answer:** Green.

### Direct QA

Question → LLM → I don't know the color of your cat's eyes.

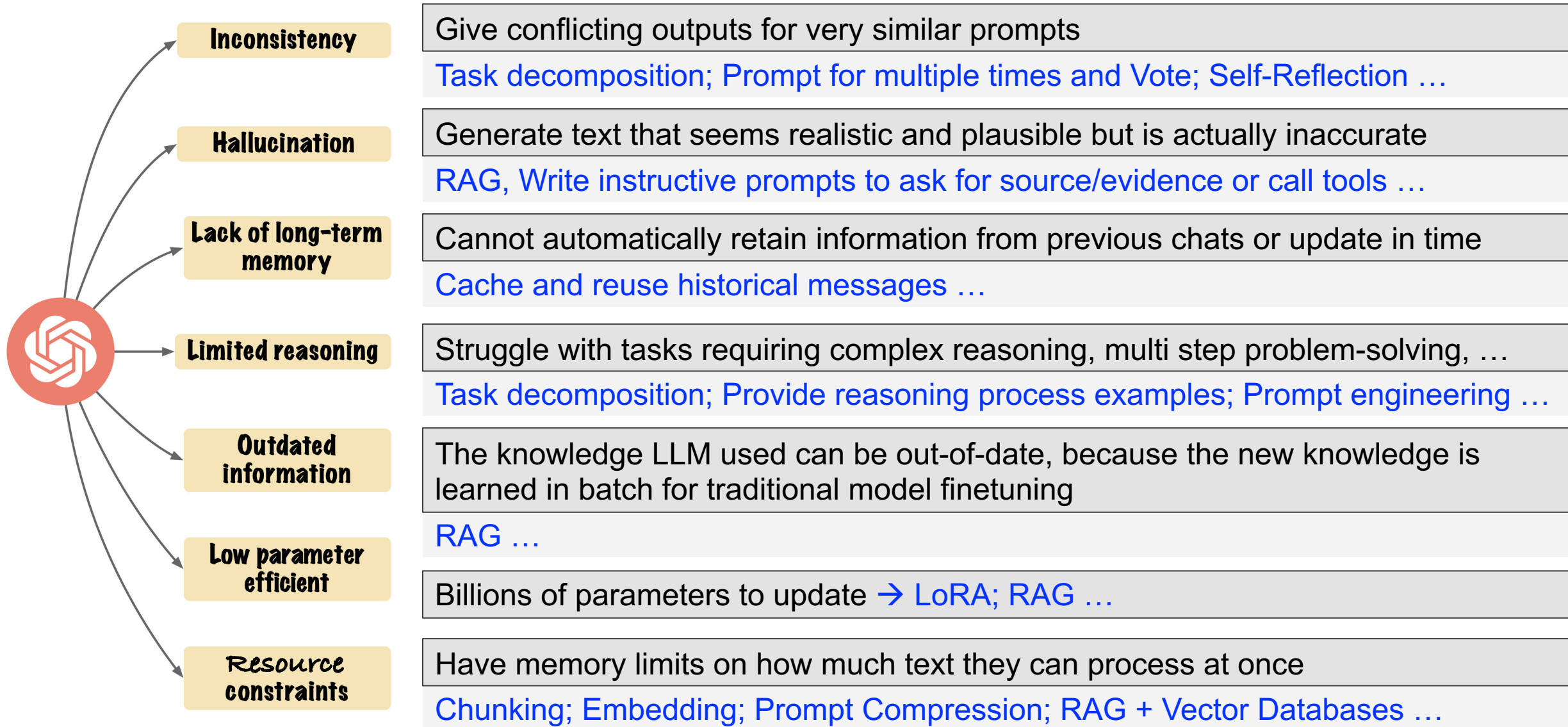
### RAG

Question → LLM → Green

↑  
"I have a cat. He has bright green eyes." (Retrieved context)

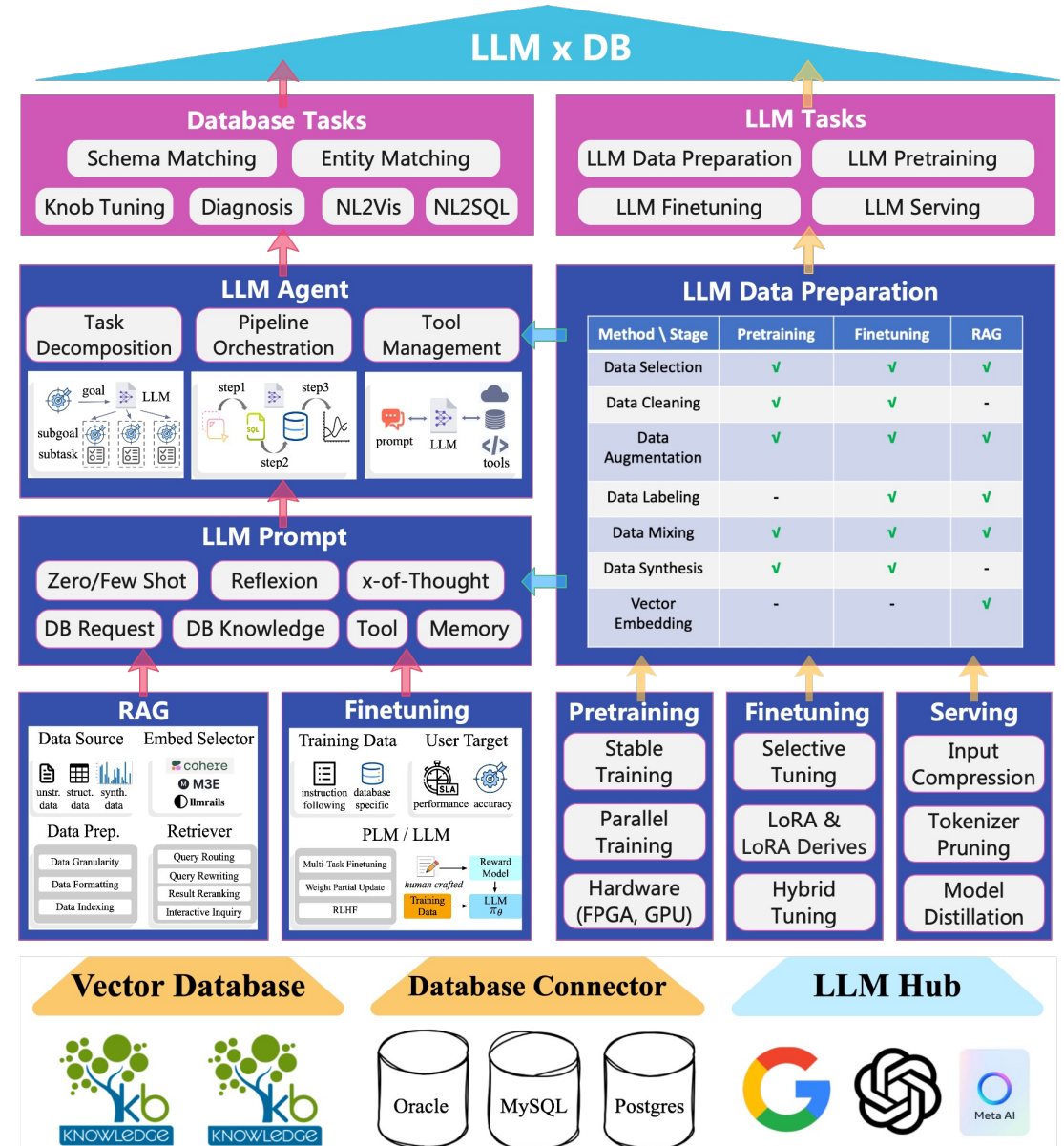
A motivative example.

# Overview of LLM Challenges and Solutions



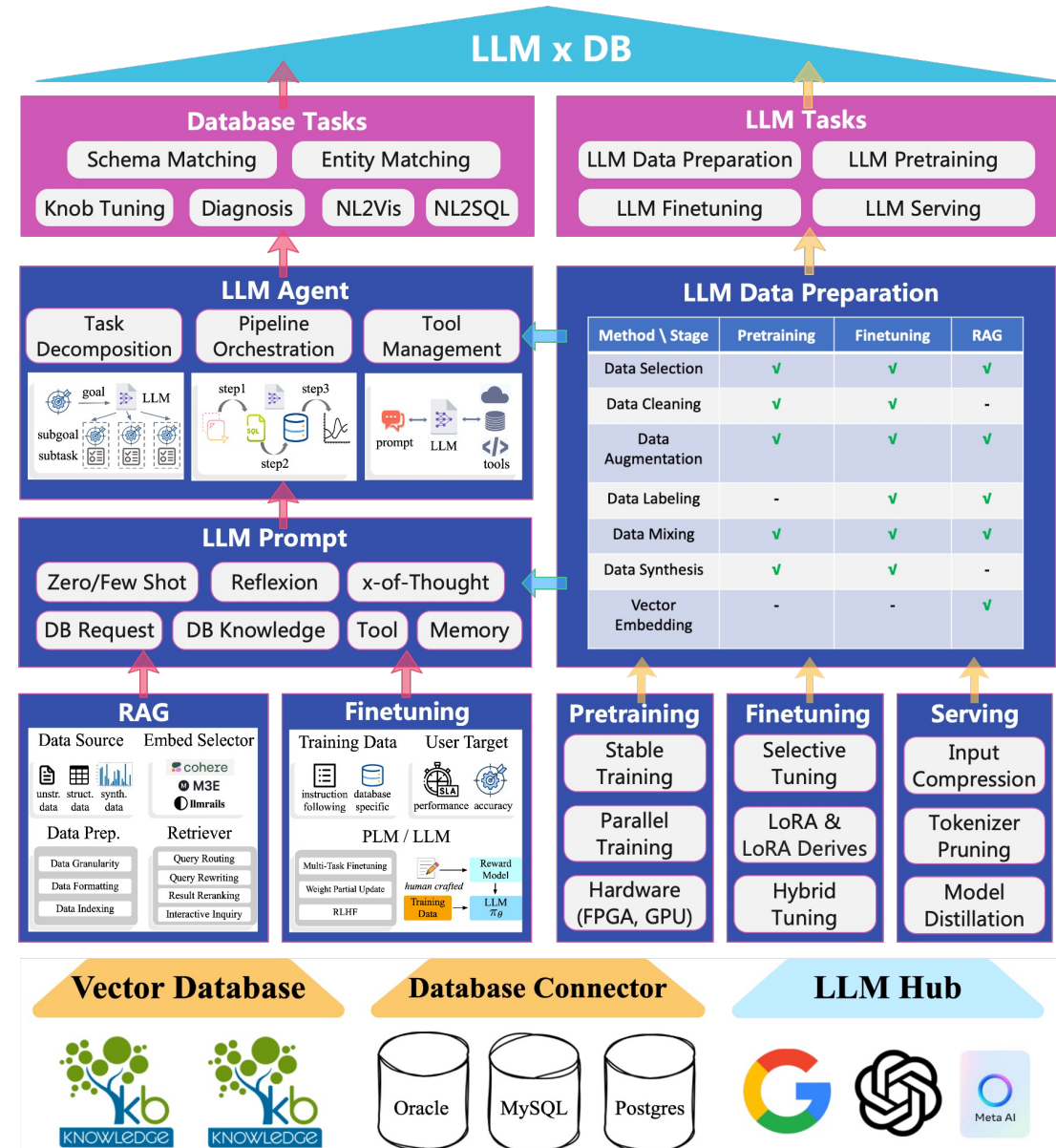
# Overview of LLM4DB Framework

- ❑ Data Management tasks
- ❑ LLM Prompt for Data Management
  - Instruction Prompting
  - Few-Shot Prompting
- ❑ LLM Agent for Data Management
  - Agent Models + Memory
  - Reasoning / Planning Strategies
  - Tool Management & Learning
- ❑ RAG for Data Management
  - Semantic Segmentation
  - Result Retrieval
  - Result Reranking
- ❑ Finetuning for Data Management
  - Reparameterization / LLM Adapter
- ❑ Data Preparation for LLM
- ❑ Open Problems



# Overview of Data Management Tasks

- ❑ Data Management tasks
- ❑ LLM Prompt for Data Management
  - Instruction Prompting
  - Few-Shot Prompting
- ❑ LLM Agent for Data Management
  - Agent Models + Memory
  - Reasoning / Planning Strategies
  - Tool Management & Learning
- ❑ RAG for Data Management
  - Semantic Segmentation
  - Result Retrieval
  - Result Reranking
- ❑ Finetuning for Data Management
  - Reparameterization / LLM Adapter
- ❑ Data Preparation for LLM
- ❑ Open Problems



# Overview of Data Management Tasks

## ❑ Data Management Tasks

### ➤ Data Preprocessing

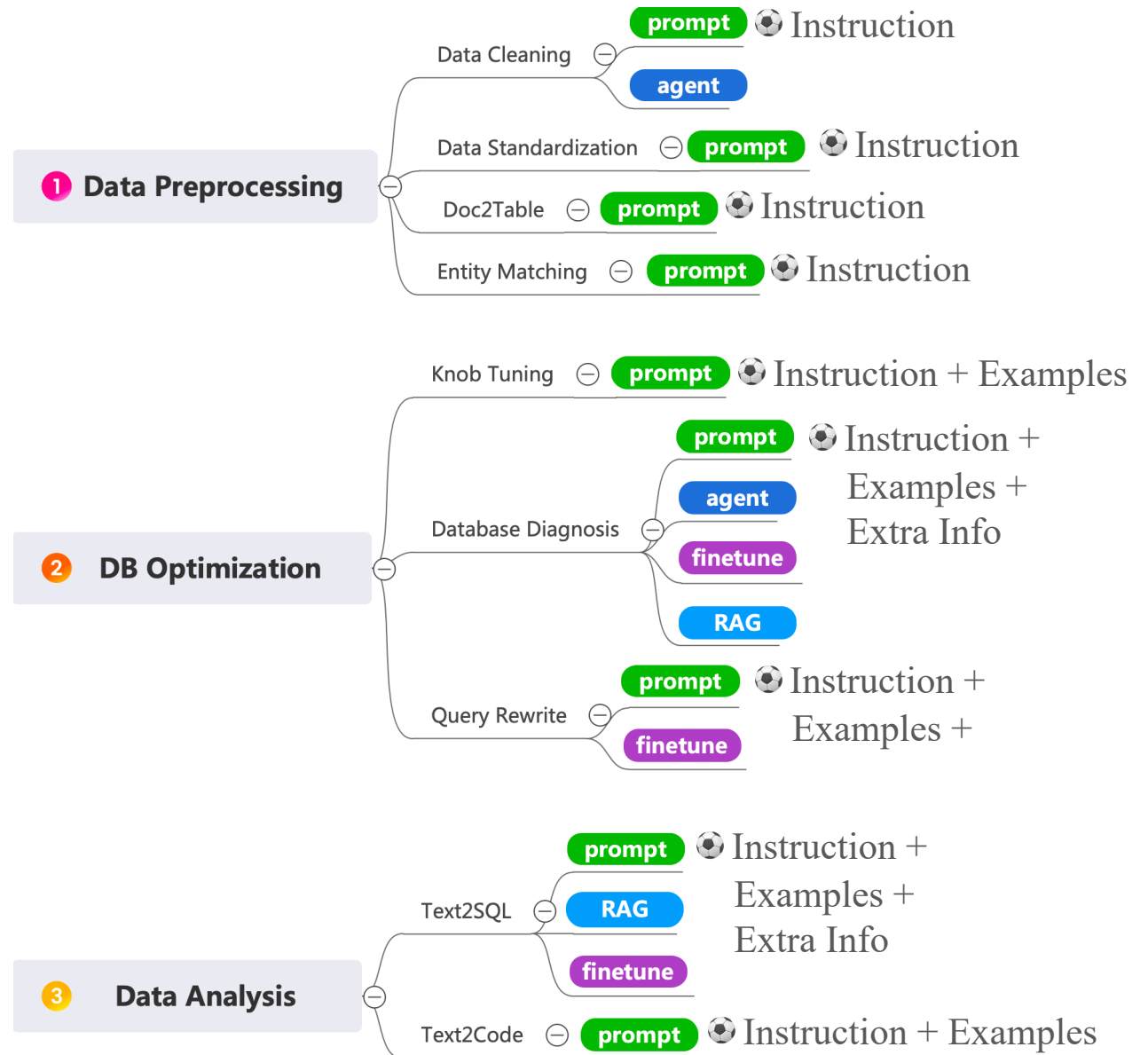
- Data cleaning
- Data Standardization
- Doc2Table
- Entity matching

### ➤ Database Optimization

- Knob tuning
- Database diagnosis
- Query rewrite

### ➤ Data Analysis

- Text2SQL
- Text2Code
- ...





# LLM Prompt for Data Management

## ❑ Data Management tasks

## ❑ LLM Prompt for Data Management

- Instruction Prompting
- Few-Shot Prompting

## ❑ LLM Agent for Data Management

- Agent Models + Memory
- Reasoning / Planning Strategies
- Tool Management & Learning

## ❑ RAG for Data Management

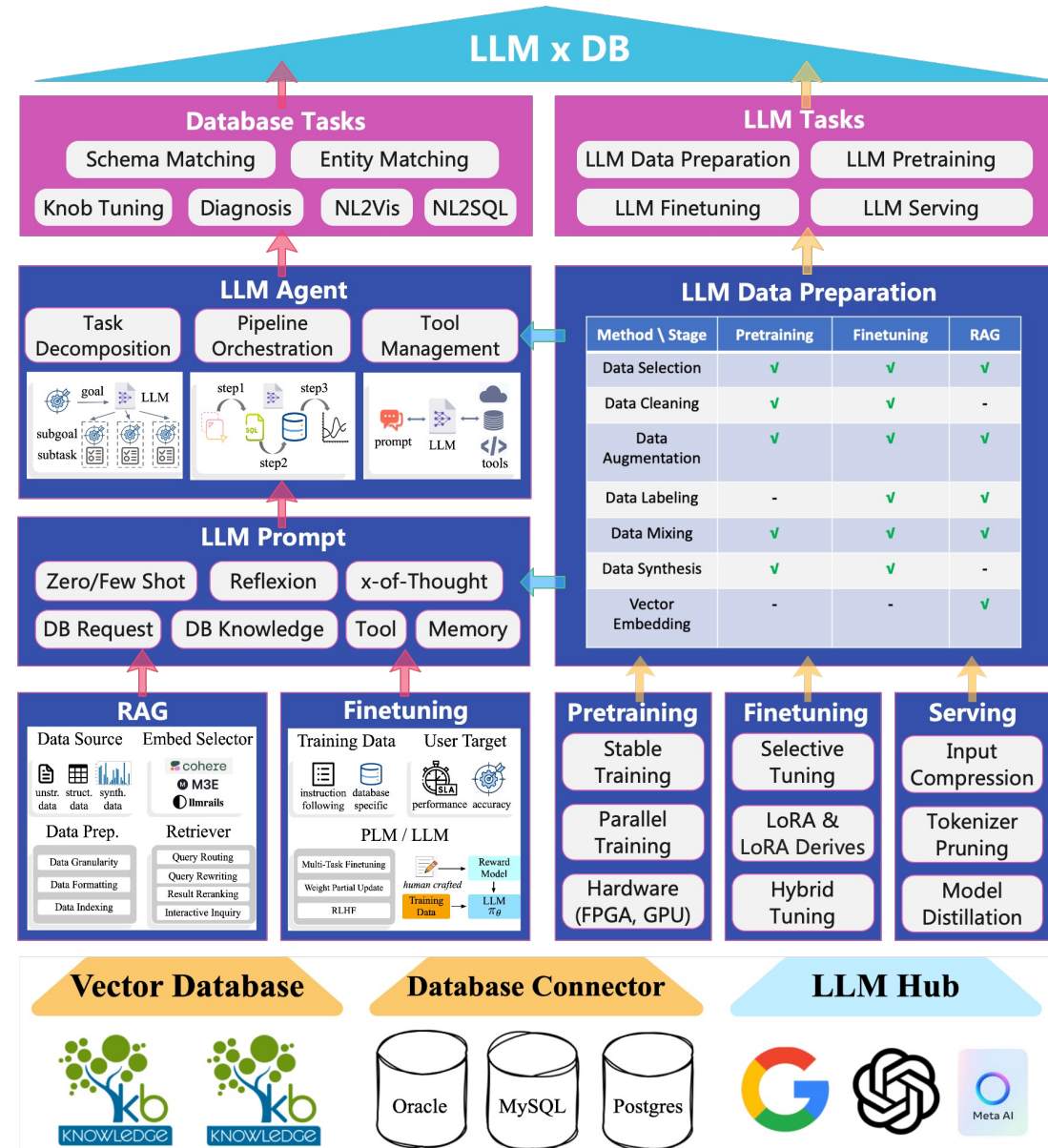
- Semantic Segmentation
- Result Retrieval
- Result Reranking

## ❑ Finetuning for Data Management

- Reparameterization / LLM Adapter

## ❑ Data Preparation for LLM

## ❑ Open Problems



# Prompt Engineering (PE)

## ❑ High-Quality Prompt can instruct LLM to optimize DB tasks without training

### ➤ Zero-shot Prompting

- Input LLM with a **task description**, without training over labeled data

### • Instruction Prompting

- Input LLM with **explicit instructions** on approaching the task, e.g., detailing the format, tone, or type of output response

### ➤ Few-shot Prompting

- Provide LLM with a few **examples** of the task within the prompt to guide the model on how to generate responses

#### Prompt of Query Rewrite

##### Task Description

Write an equivalent SQL query that can be executed on a Postgres database with decreased latency.

##### Instruction

1. Ensure output query is semantical-equivalent to the input query ...

##### Example Input

```
select ... from t1 where t1.a=(select avg(a) from t3 where t1.b=t3.b);
```

##### Example Output

```
select ... from t1 inner join (select avg(a) avg,t3.b from t3 group by t3.b) as t3 on (t1.a=avg and t1.b=t3.b);
```

##### Input

```
select t1.* from t1 where t1.col1>(
  select max(t2.col2) from t2 where t2.col1 in (
    select t1.col1 from t1 where t1.col1=t2.col1));
```

##### Output

```
select t1.* from t1 inner join (
  select max(t2.col2) max, t2.col1 from t2
  group by t2.col1) as t2 on (
  t1.col1=t2.col1)
where t1.col1>max;
```

# Challenges of PE for Data Management

## ❑ Challenge 1: How to Automatically Generate Input Prompt?

- Automatically generate proper *instructions* and select *demonstration examples* from large space within limited prompt tokens (or limited interaction rounds).

## ❑ Challenge 2: How to Efficiently Interact with LLM Using Prompts?

- *Iteratively adjust prompt* for input request (e.g., select suitable examples from candidate ones) is time consuming
- *Long prompts* often include more useful information, but require long inference time for LLM and hard to understand

### Prompt of Query Rewrite

#### Task Description

Write an equivalent SQL query that can be executed on a Postgres database with decreased latency.

#### Instruction

1. Ensure output query is semantical-equivalent to the input query ...

#### Example Input

```
select ... from t1 where t1.a=(select avg(a) from t3 where t1.b=t3.b);
```

#### Example Output

```
select ... from t1 inner join (select avg(a) avg,t3.b from t3 group by t3.b) as t3 on (t1.a=avg and t1.b=t3.b);
```

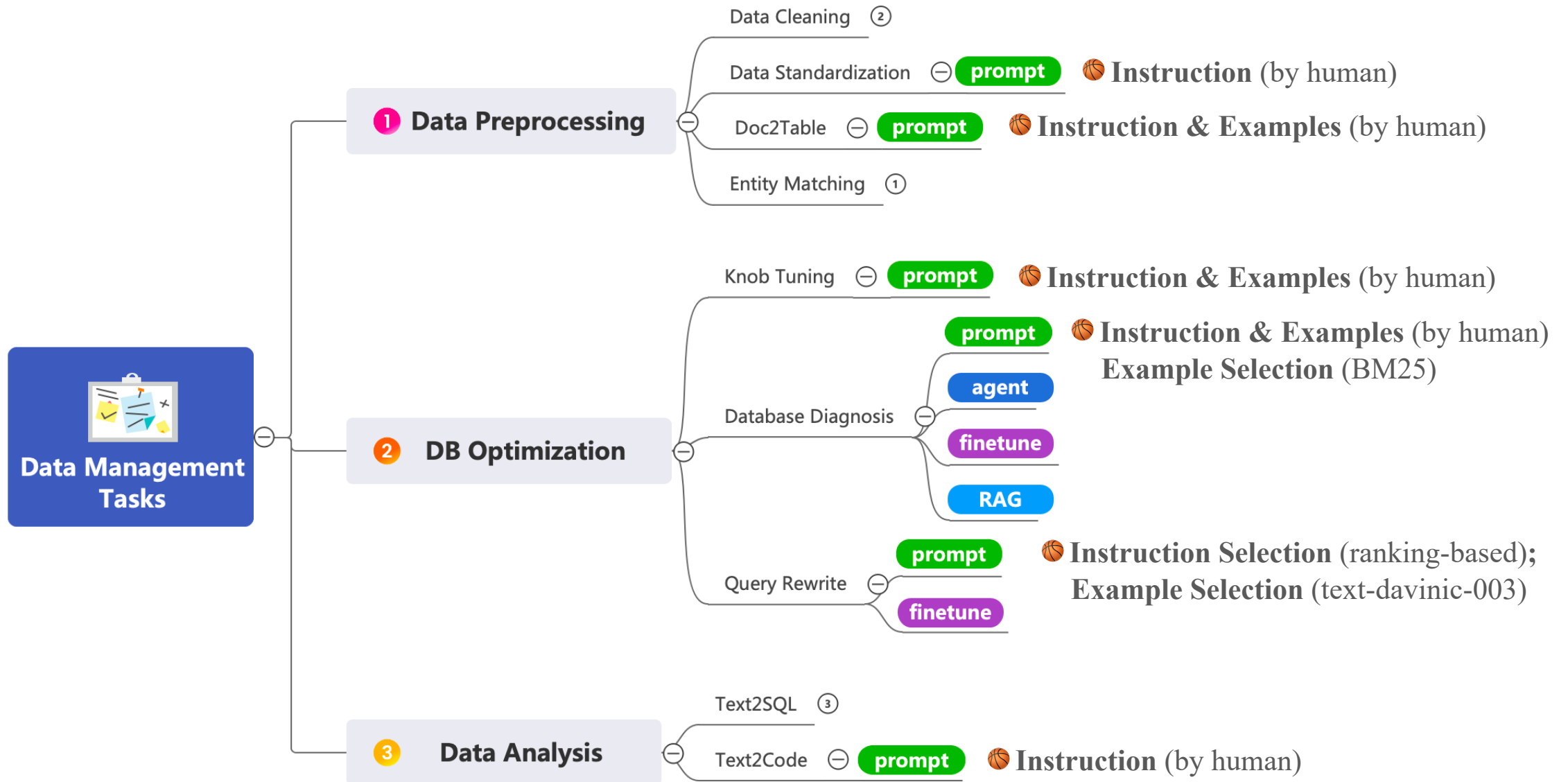
#### Input

```
select t1.* from t1 where t1.col1>(
  select max(t2.col2) from t2 where t2.col1 in (
    select t1.col1 from t1 where t1.col1=t2.col1));
```

#### Output

```
select t1.* from t1 inner join (
  select max(t2.col2) max, t2.col1 from t2
  group by t2.col1) as t2 on (
  t1.col1=t2.col1)
where t1.col1>max;
```

# Prompt Engineering Techniques for Data Management



# Prompt Engineering for Data Preprocessing

## ❑ Problems in Traditional Data Preprocessing

- Require substantial programming skills to write UDFs
- Require complex running environment to run the UDFs
- Prior experience cannot be utilized in UDFs

### Decompose into Operations

- data structural consistency
- data type conversion
- data standardization
- data anomaly detection

## ❑ Manually write NL prompts to avoid writing complex functions

```
# define an UDF
def user_define_function(inputs):
    # import the dependencies
    # implement the logic based on the use cases
    # process the inputs
    # return the processed data
    return processed_inputs

# SQL query
SELECT user_define_function(inputs)
```

*User-Defined Function*



```
# define a user-defined prompt template
prompt_template = "
<define the system prompt parameter.>
<define the prompt incorporating with the {inputs} and the
output.>"

def user_defined_function(inputs):
    # an LLM_call function communicates with LLM engine
    # and returns the processed result
    processed_inputs = llm_call(prompt_template.format(inputs))
    return processed_inputs
```

*User-Defined LLM Prompt*

# Prompt Engineering for Data Preprocessing

## ❑ Problems in Traditional Data Preprocessing

- Require substantial programming skills to write UDFs
- Require complex running environment to run the UDFs
- Prior experience cannot be utilized in UDFs

### Decompose into Operations

- data structural consistency
- data type conversion
- data standardization
- data anomaly detection

## ❑ Manually write NL prompts to avoid writing complex functions

- E.g., for date data structuralization,
- **Traditional UDFs:** Require enumeration of the date format / utilize different date processing packages
- **Manually define the output format** (YYYYMMDD) in the prompt and let LLMs handle the data processing

1		+	-	-	-	+	
2		item_id		user_id		user_rating	
3		+	-	-	-	+	
4		"101"		"201"		3	
5		+	-	-	-	+	
6		"102"		"201"		4	
7		+	-	-	-	+	
8		"101"		"202"		5	
9		+	-	-	-	+	
10		"101"		"203"		2	
11		+	-	-	-	+	

# Prompt Engineering for Query Rewrite

---

## ❑ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** Insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** Different programming languages and SQL syntax

```
SELECT ...  
WHERE ...  
AND (  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=1  
  )  
OR  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=2 )  
)
```

# Prompt Engineering for Query Rewrite

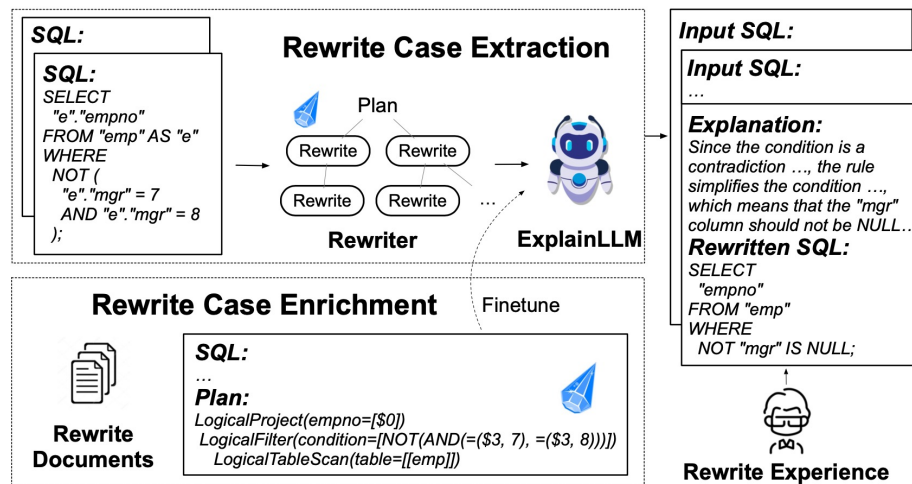
## ❑ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** Insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** different programming languages and SQL syntax

## ❑ 1. Prompt-based Rewrite Case Generation

- **Utilize LLM to generate well-explained rewrite cases**, i.e., (original query, rewritten query, rewrite rule, rewrite analysis).

```
SELECT ...  
WHERE ...  
AND (  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=1  
  )  
OR  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=2 )  
)
```



*$p_{plan} = \dots$  The rewriter first translates input SQL into equivalent input plan. Second, it uses the given rewrite rule to transform the input plan into rewritten plan. Third, it translates the rewritten plan into equivalent rewritten SQL. ... You should not mention the input plan and rewritten plan in your explanation, as if the rewriter directly transforms the input SQL into the rewritten SQL. ...*



# Prompt Engineering for Query Rewrite

## ❑ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** different programming languages and SQL syntax

## ❑ 2. Prompt-based Query Rewrite

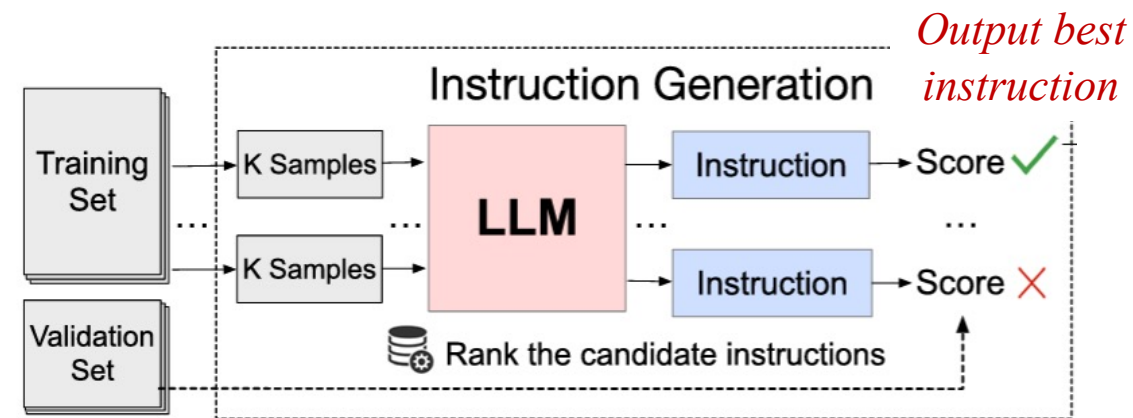
- The quality of prompt impacts the performance of LLM on different rewrites → **Automatic Prompt Generation**

```
SELECT ...  
WHERE ...  
AND (  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=1  
  )  
OR  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=2 )  
)
```

### 2.1 Instruction Generation

- Write prompt to instruct LLM to generate instruction candidates by examples (e.g., five pairs):

- I followed the instruction to rewrite the input SQL query to produce an equivalent SQL query...
- Based on the instruction, they produced the following input-output pairs: \n\n[**example pairs**] \n\nInstruction:"



# Prompt Engineering for Query Rewrite

## ❑ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** different programming languages and SQL syntax

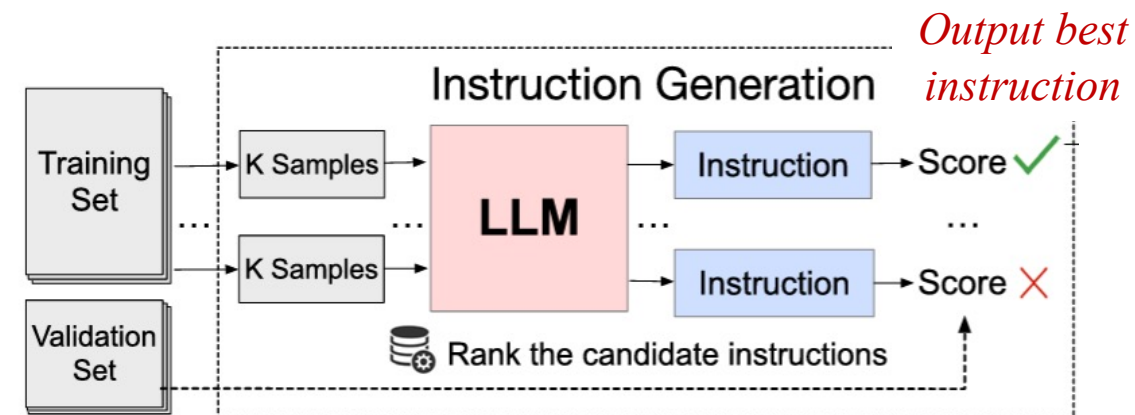
## ❑ 2. Prompt-based Query Rewrite

- The quality of prompt impacts the performance of LLM on different rewrites → **Automatic Prompt Generation**

```
SELECT ...  
WHERE ...  
AND (  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=1  
  )  
OR  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=2 )  
)
```

### 2.1 Instruction Generation

- Rank the generated instruction candidates based on their benefit for validation set (e.g., the cost reduction after rewriting with the instructed llm)



# Prompt Engineering for Query Rewrite

## ❑ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** different programming languages and SQL syntax

## ❑ 2. Prompt-based Query Rewrite

- The quality of prompt impacts the performance of LLM on different rewrites → **Automatic Prompt Generation**

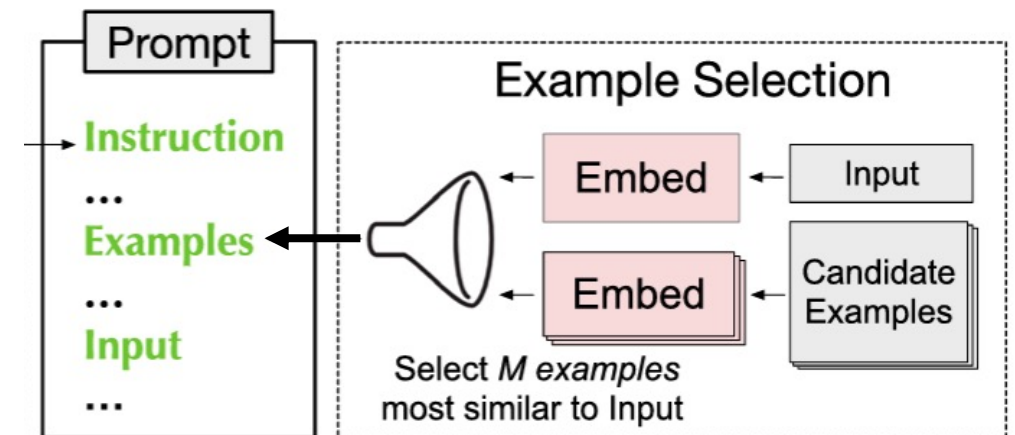
```
SELECT ...  
WHERE ...  
AND (  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=1  
  )  
OR  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=2 )  
)
```

### 2.1 Instruction Generation

- Rank the generated instruction candidates based on their benefit for validation set (e.g., the cost reduction after rewriting with the instructed llm)

### 2.2 Demonstration Example Generation

- **Match** the current query  $q$  with a few candidate rewrites whose input queries are similar to  $q$



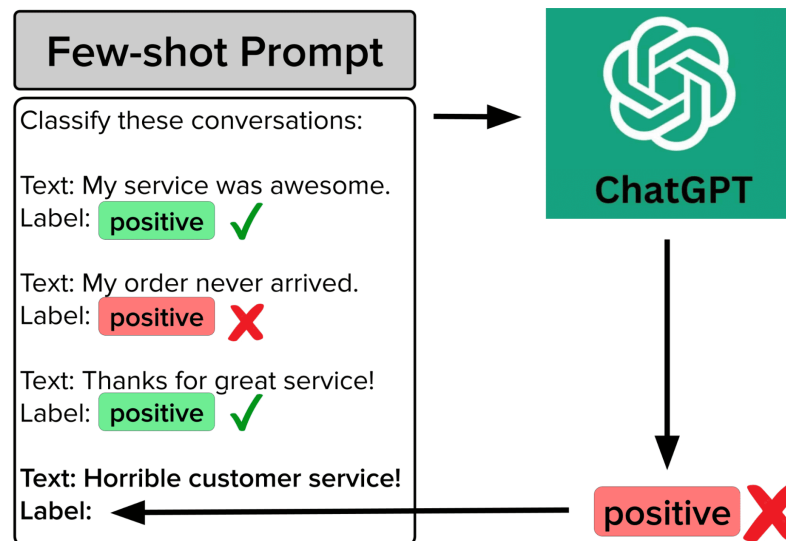
# Prompt Engineering for Query Rewrite

## ❑ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** different programming languages and SQL syntax

## ❑ 3. Efficiency Issues

- Search-based example matching is time-consuming  
→ Prompt / Finetune a model to **identify the most suitable demo examples**



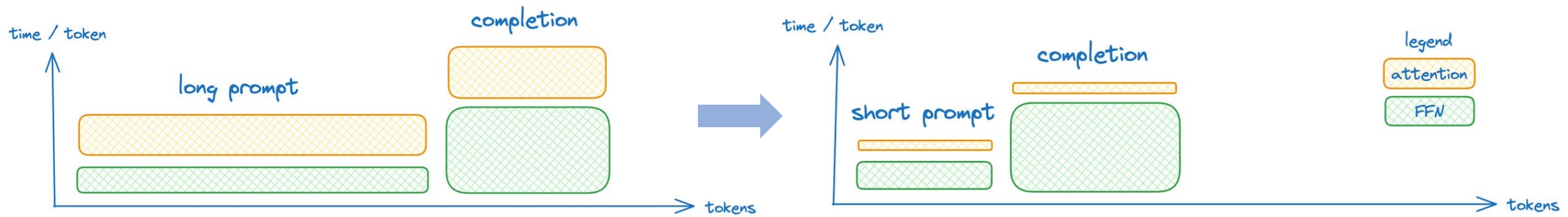
# Prompt Engineering for Query Rewrite

## ❑ Problems of Rule-driven Rewriters

- **Inadequacy of rules:** insufficient for handling complex query transformations (e.g., merging sub-queries)
- **Cross-system migration:** different programming languages and SQL syntax

## ❑ 3. Efficiency Issues

- Excessively long prompt can slow down the LLM inference  
→ Strike a balance between **prompt length and LLM performance** (e.g., generate summary for the detailed description of rewrite rules)



# Prompt Engineering for Table Data Extraction

❑ LLM cannot directly extract structured table from long-context and semi-structured documents

## ❑ Core Idea

❑ For long document, feed sampled documents to the LLM, and **prompt it to generate useful information (schema and cell values)** that can form a structured table (e.g., writing code to extract the values of important attributes)

### Input

**Data lake:** A collection of semi-structured documents (e.g. HTML, TXT, XML)



EVAPORATE-CODE+  
(Doc2Table)



### Output

**Tables:** A structured view of the data in the input documents.



name	draft year	position
Jayson Tatum	2017	Power Forward
Anthony Davis	2012	Center
Kevin Durant	2007	Small Forward
Steph Curry	2009	Point Guard

# Prompt Engineering for Table Data Extraction

- ❑ LLM cannot directly extract structured table from long-context and semi-structured documents

- ❑ Prompt-based Table Data Extraction

  - ❑ Schema Synthesis

    - ❑ With a sampling subset of documents, it **prompts LLM to extract attributes** based on their occurrence frequencies

    - ❑ Rerank the extracted attributes by adjusting their frequency weights with LLM

  - ❑ Function Synthesis

    - ❑ A heavy job to extract attribute values from every document → **Prompt LLM to write code** to extract the attribute values

*Function  
Prompt*

Here is a file sample:  
<title>U.S. GDP Rose 2.9% in the Fourth Quarter </title>  
<meta itemProp="datePublished"  
content="2023-01-26T10:30:00Z"/>  
...

**Question:** Write a python function called "get\_date\_published\_field" to extract the "datePublished" field from the text. Include any imports.



```
from bs4 import BeautifulSoup
def get_date_published_field(text: str):
    soup = BeautifulSoup(
        text, parser="html.parser"
    )
    date_published_field = soup.find(
        'meta', itemprop="datePublished"
    )
    return date_published_field['content']
```

# Prompt Engineering for NL2Code

## □ Motivation: Textual instructions → Execute by generating code

### • Core Idea

- Design prompt to drive LLM to (1) **decompose the input query** into a series of simple processing steps and (2) **translate** the instructed steps into **executable code**

*Database description*



*Processing steps*



*Natural language instructions*



### Example Prompt

```
"""
Table Data with columns 'Player','No_','Nationality','Position',
'Years_in_Toronto','School_Club_Team', stored in 'Data.csv'.
Processing steps:
1. Load data for table Data.
2. Print progress updates.
3. Check if 'Player' equals 'dell curry'.
4. Print progress updates.
5. Filter results of Step 1 using results of Step 3.
6. Print progress updates.
7. Create table with columns 'Years_in_Toronto'
   (aka. result ) from results of Step 5.
8. Print progress updates.
9. Write results of Step 7 to file 'result.csv' (with header).
10. Print progress updates.
"""
```



# Prompt Engineering for Knob Tuning (Workload Generation)

## □ Motivation

- ML-based methods require numerous workloads as training data

## • Main Steps

- Workload generation: Use manual-crafted GPT-4 to generate diversified workloads for specific database schema and workload types

### Workload Generation Prompt

*You are a helpful DBA, tasked with generating challenging OLAP / OLTP workloads and fulfill my goals.*

**OLAP:** Recall the complex queries in the TPC-H, TPC-DS, and JOB databases... Ensure that the queries do not involve write operations like 'insert', 'update', or 'delete'.

Note 1: The key attributes of OLAP queries are as follows: .....

**OLTP:** Recall the simple queries in the Sysbench, TPC-C and OLTPBench databases...

Generate several 'select', 'insert', 'update' or 'delete' queries...

Note 1: The key attributes of OLTP queries are as follows: .....

Goal: Executable Workloads

Here are the database schema followed by some column values that might assist you in generating predicates.....

```
CREATE TABLE public.events_relevant (  
  event_id integer NOT NULL: "336", "444"...  
  device_id text: "-918417362937292"...
```

Goal: Diverse Workloads

You should craft <number> highly intricate <type> queries, incorporating elements such as multi-table JOINS — with a minimum of <x> and up to <y> tables.

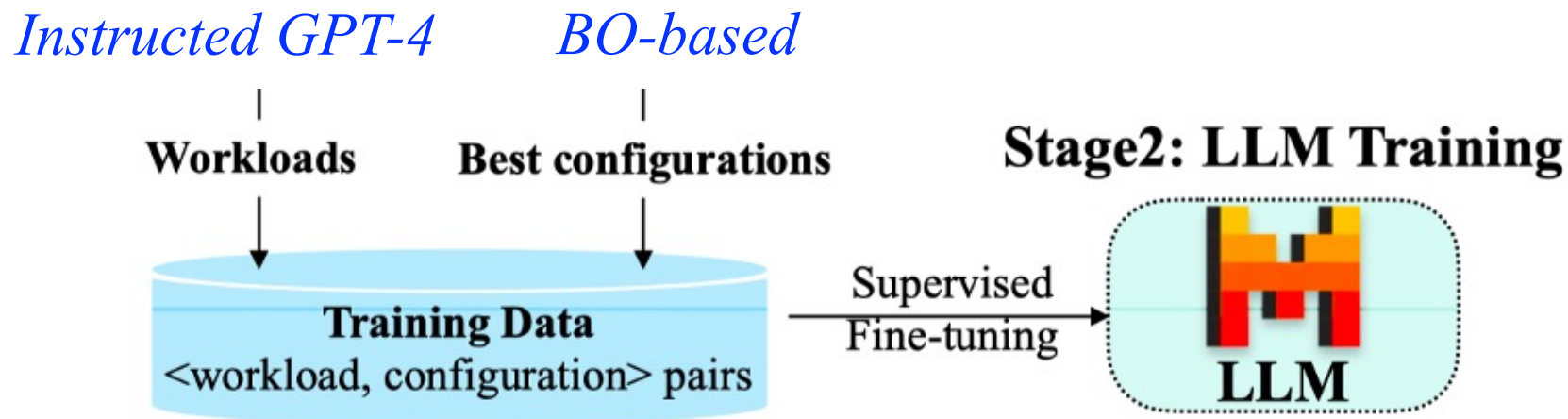
# Prompt Engineering for Knob Tuning (Workload Generation)

## □ Motivation

- ML-based methods require numerous workloads as training data

## • Main Steps

- LLM Finetuning: Train Mistral-7B with workloads labeled with configurations recommended by Bayesian Optimization (BO)-based algorithm
  - **LLM Input:** Workload features, internal metrics, query plan;
  - **LLM Output:** Generate the configuration change based on previous configuration

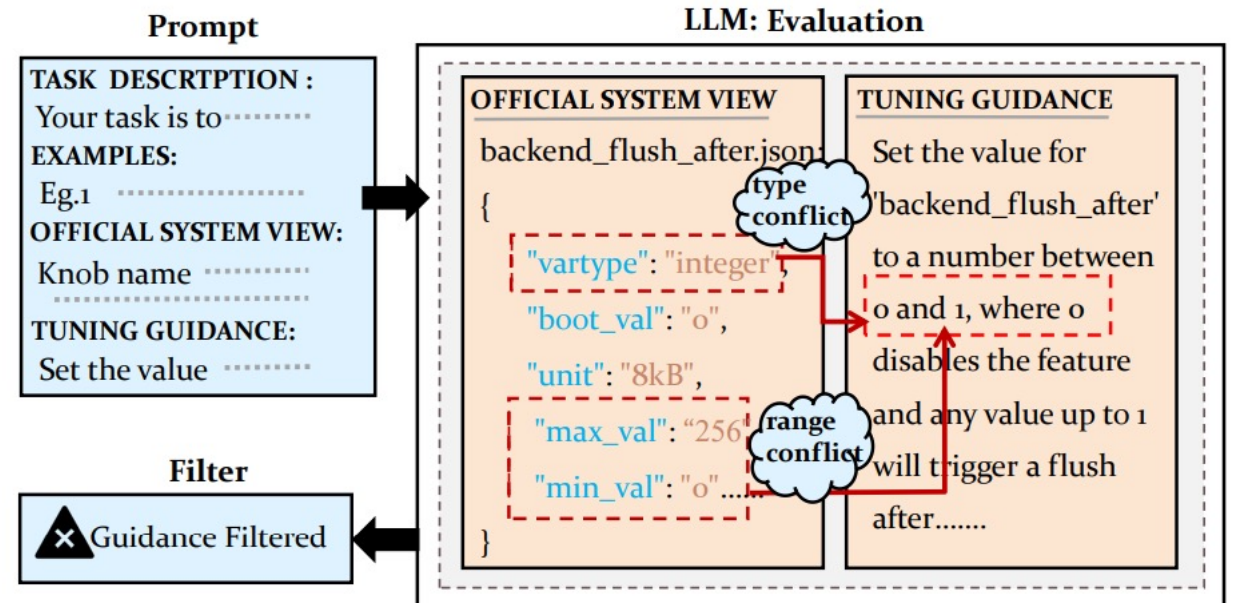
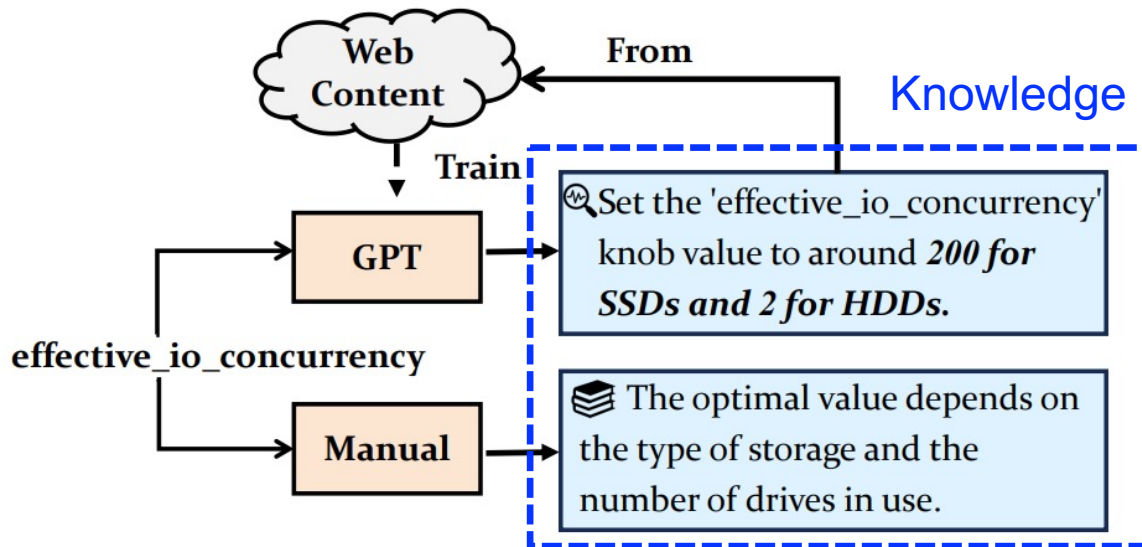


# Prompt Engineering for Knob Tuning (Knowledge Extraction)

□ Textual experience are not well utilized in knob tuning

→ Split knowledge extraction task into four main steps, and manually design prompts to guide LLM in each step

- 1. *Extract knowledge from LLM*: retrieval knowledge by (1) directly asking GPT or (2) prompting LLM to summarize from documents
- 2. *Filter noisy knowledge*: Prompt LLM to **evaluate** whether the tuning knowledge conflicts with the system view



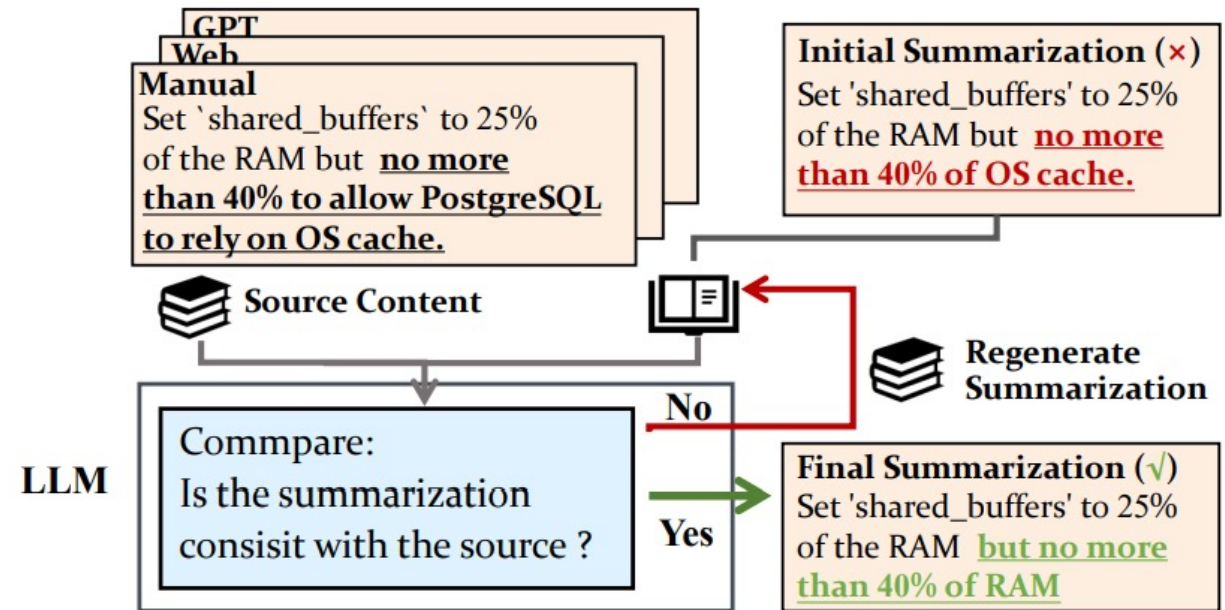
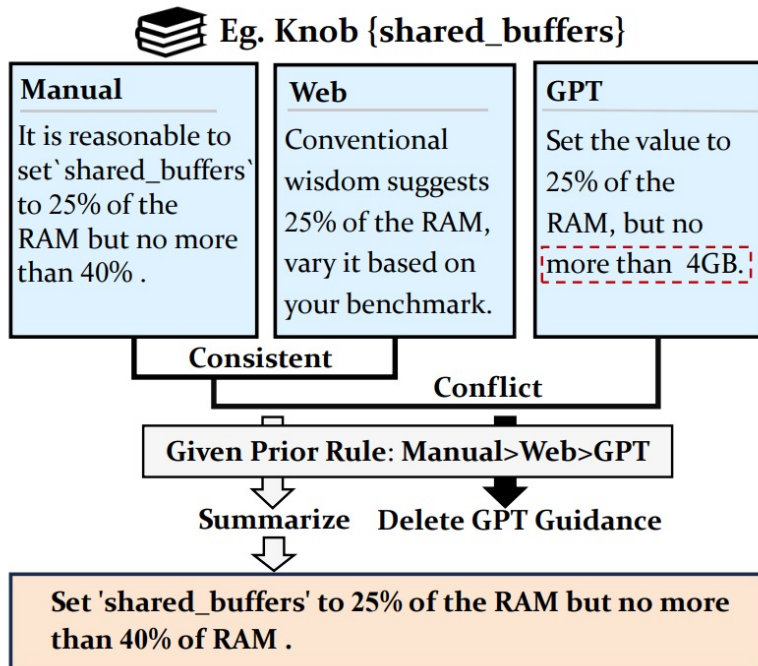
# Prompt Engineering for Knob Tuning (Knowledge Extraction)

□ Textual experience are not well utilized in knob tuning

→ Split knowledge extraction task into four main steps, and manually design prompts to guide LLM in each step

• 3. Summarize knowledge from various resources: **Manual > Web > GPT**

• 4. Check **factual inconsistency** with the document sources



# LLM Prompt for Database Diagnosis

## □ Improve diagnosis prompt with matched *Knowledge* and *Tools*

### • 1. Anomaly Description for triggered alerts

During the inspection, it was identified that from 13:20:49 to 13:37:49 on October 15, 2023, the load on the node 'ins:stdload1' was exceedingly high, reaching 160% of the standard capacity. This exceeded the threshold of 100%, thus triggering a warning alert. Although

### • 2. Tool Matching with finetuned embed model

$$\text{Score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgDL}})}$$

$\text{emb}(\cdot)$ : the embedding function of the **fine-tuned Sentence-BERT model**;  $s$ : context;  $t_j$ : tool API

### • 3. Knowledge Matching by metric attributes

$$\text{sim}(s, t_j) = \frac{\text{emb}(s) \cdot \text{emb}(t_j)}{\|\text{emb}(s)\|_2 \|\text{emb}(t_j)\|_2},$$

- $D$ : A candidate knowledge chunk;  $q_i$ : abnormal metric

#### Prompt Template

- Expert Description: <Role> ; <Task> ; <Steps>
- The anomaly alerts are {Anomaly Info}
- In this step, you can use the following tools: {Tool APIs}
- The matched knowledge is: {Knowledge}

=====  
(Demonstration Examples of available tool usage)

#### Example Knowledge Chunk from Past Diagnosis Report

```
"name": "large_data_insert",  
"content": "Identify excessive inserted tuples in a table or  
query operations.",  
"metrics": ["inserts", "query", "index_schema"],  
"steps": "For each inserted table, if the count of inserted  
tuple is equal to or exceeds the {threshold}, it's  
flagged as a potential root cause."
```

# Summarization of Prompt-based Data Management

	LLM Job	Operations	Prompt (Content)	Prompt (Examples)	Efficiency Issue
<i>Data Standardization</i>	<b>GPT:</b> Modify Data format	Four Preprocessing Operations	Instruction	Manual	N/A
<i>Query Rewrite</i>	<b>GPT:</b> Query Rewrite	Typical Logical Transformations	Instruction + Example	Ranking based Selection	1. A small-sized model for Example Selection; 2. Prompt Compress; 3. Past rewrite Reuse
<i>Table Data Extraction</i>	<b>GPT:</b> Doc2table	Schema Extraction; Value Extraction	Instruction + Example	Manual	N/A
<i>NL2Code</i>	<b>GPT:</b> Text2code	Processing steps and comments	Instruction	Manual	N/A
<i>Knob Tuning</i>	<b>GPT:</b> Workload Generation; <b>Mistral:</b> Knob Tuning	Workload Generation + Knob Tuning	Instruction	Manual	N/A
<i>Knob Tuning</i>	<b>GPT:</b> Knowledge Extraction	Four steps for knowledge extraction	Instruction + Example	Manual	N/A
<i>Database Diagnosis</i>	<b>GPT / Llama:</b> Root Cause Analysis	Three analysis and tool calling steps	Instruction + Example + Knowledge / Tool	Matched by external info	N/A

# Take-aways

---

## □ Prompt generation

- Existing prompts mostly depend on human experts to craft high-quality instructions and examples, which have the **scalability** problem
- Prompt examples can be generated from humans, real cases, LLMs (e.g., add explanations), and traditional algorithms (e.g., search-based for configs)

## □ Instruction selection

- Instruction **format** can affect LLM performance (especially weak LLMs)
- The order of instructions / different examples can affect LLM performance

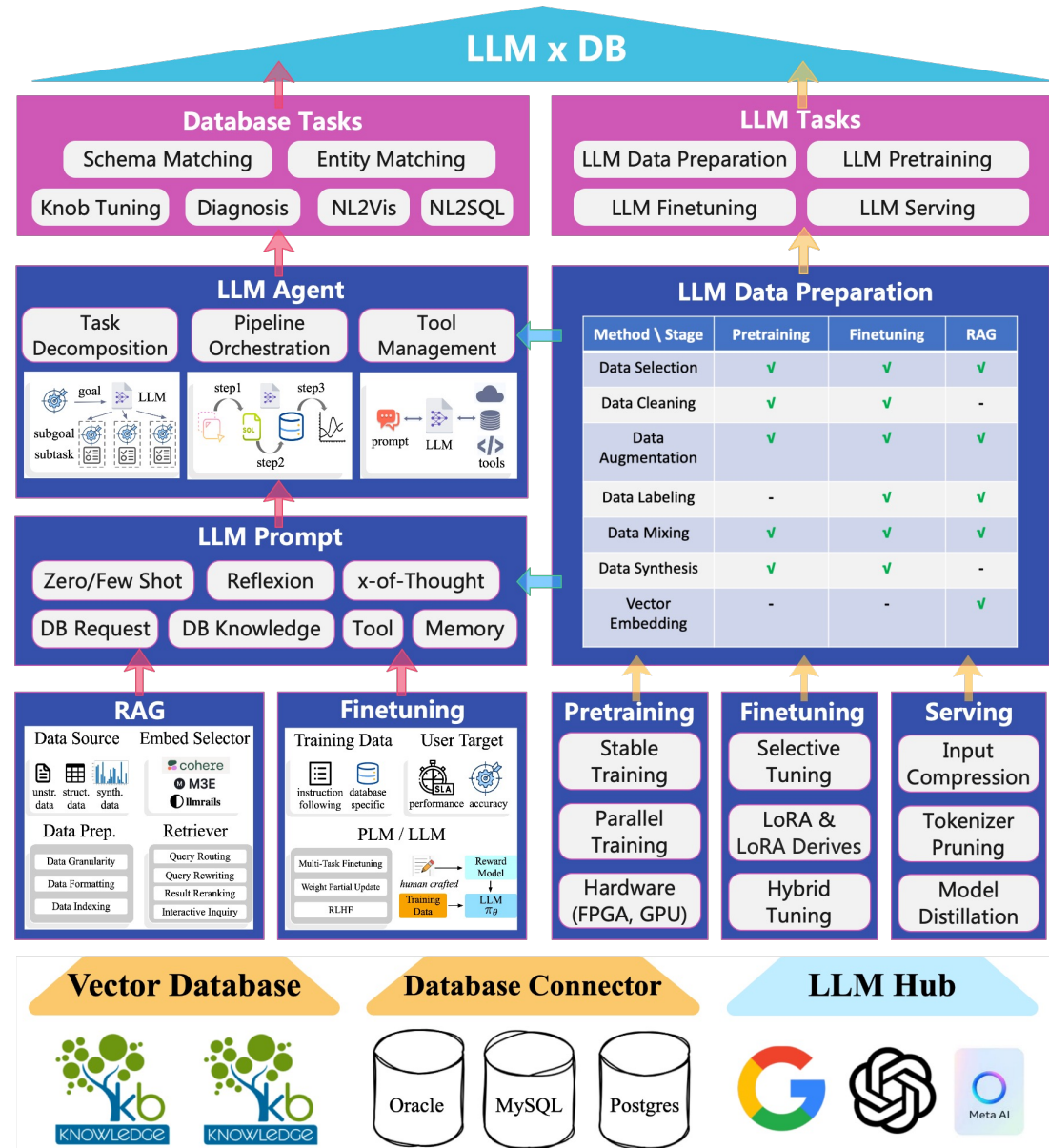
## □ Prompt Selection

- Prompt **selection** are critical to the success of the current task
- No one-size-fit-all-model prompt, which needs modifications for different LLMs

## □ Efficiency improvement: prompt compression, model-based example selection, and batch inference

# LLM Agent for Data Management

- ❑ Data Management tasks
- ❑ LLM Prompt for Data Management
  - Instruction Prompting
  - Few-Shot Prompting
- ❑ LLM Agent for Data Management
  - Agent Models + Memory
  - Reasoning / Planning Strategies
  - Tool Management & Learning
- ❑ RAG for Data Management
  - Semantic Segmentation
  - Result Retrieval
  - Result Reranking
- ❑ Finetuning for Data Management
  - Reparameterization / LLM Adapter
- ❑ Data Preparation for LLM
- ❑ Open Problems





# LLM Agent

❑ **Human-Crafted Prompt** for each task → Design **Automatic Agents** to automatically execute complex tasks with predefined prompt templates

- **LLM agents:** An LLM system that can automatically resolve a series of domain-specific tasks with minor human involvement



# Advantages of LLM Agent

## □ Autonomy

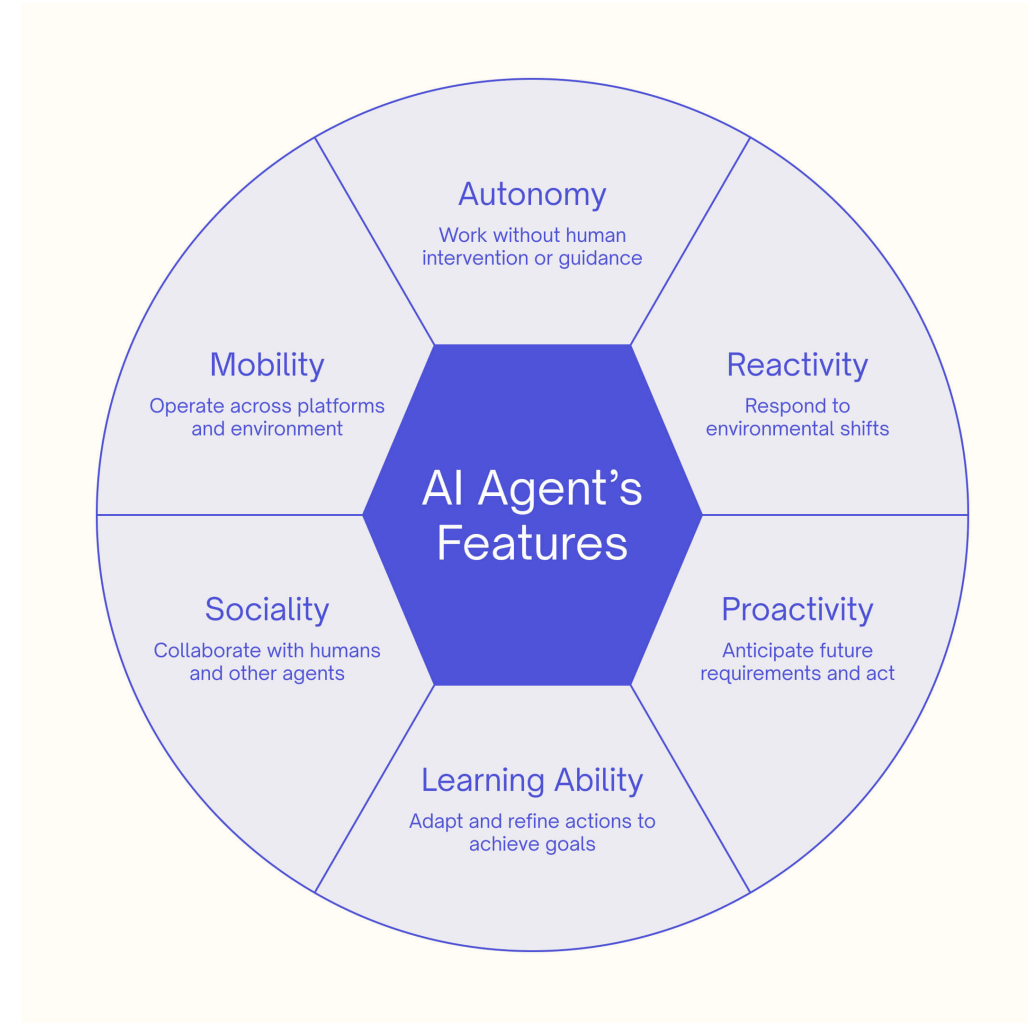
- Execute human instructions and complete tasks;
- Start and execute operations without human requirements

## □ Reactivity

- Respond rapidly to immediate changes and interact with its environment

## □ Pro-activeness

- Anticipate future. Make plans, and take proactive measures in their actions to achieve specific goals



# Advantages of LLM Agent

## ❑ Social (Multi-Agent) ability

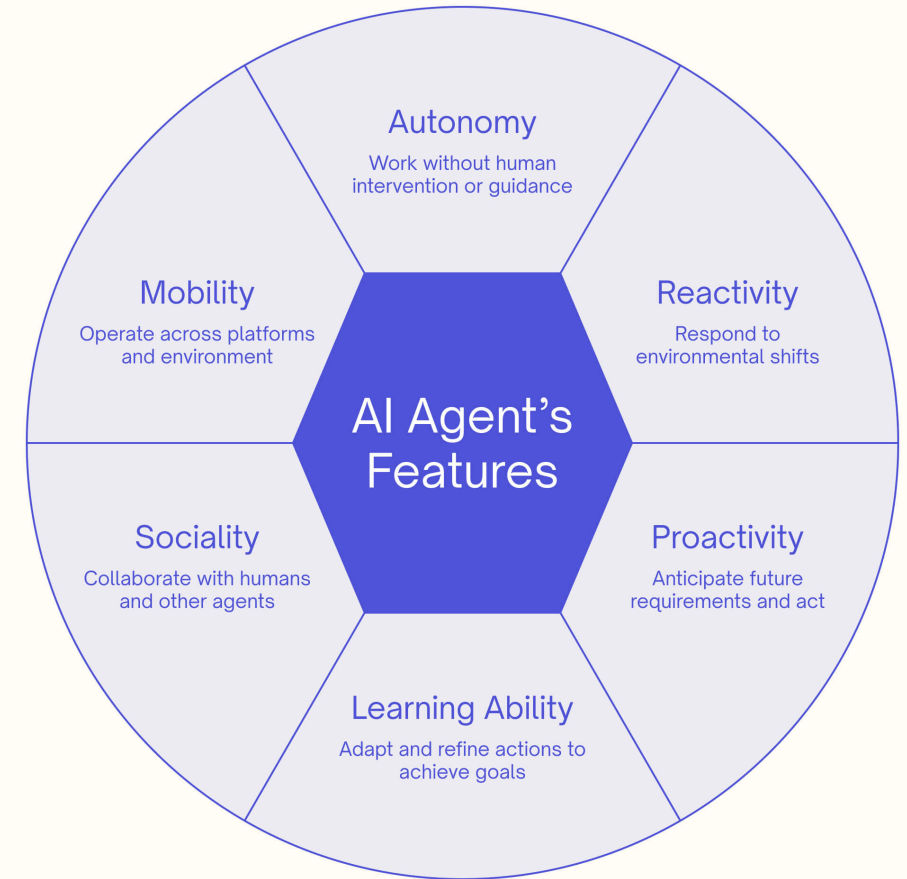
- Interact with other agents (including humans), generate and understand natural language

## ❑ Learning Ability

- Adaptively integrate new tools and refine the execution pipelines based on environment feedback
- Memorize experience in both external knowledge base and model parameters

## ❑ Mobility

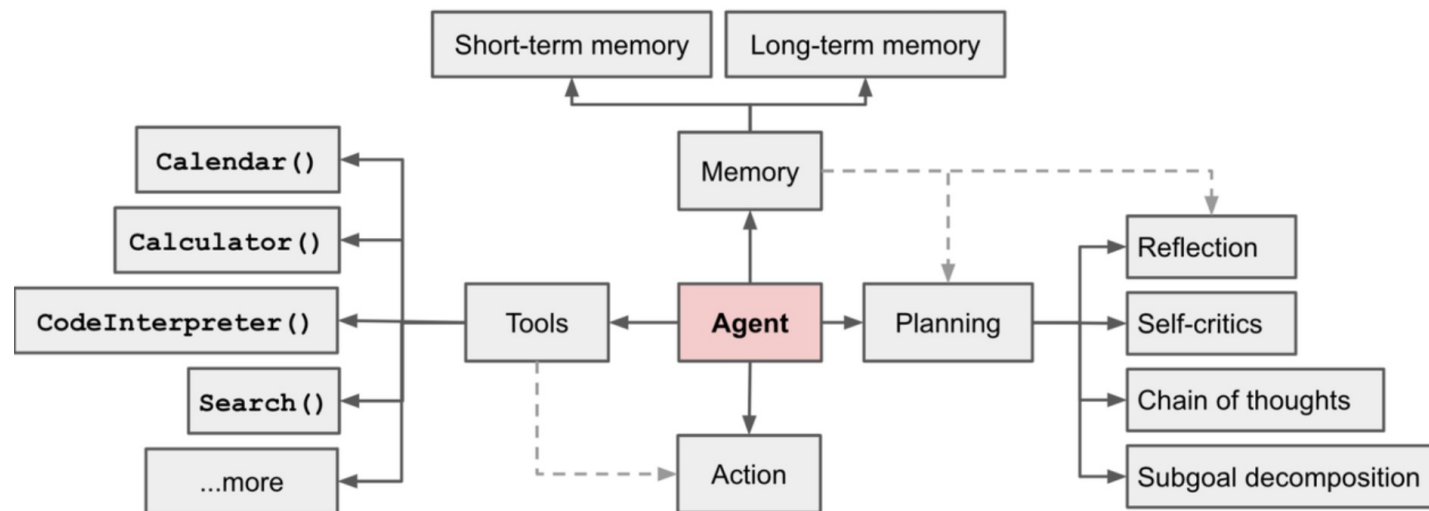
- Easy to generalize to new platforms (e.g., Postgres → MySQL) and scenarios (e.g., specifying new workloads) via prompt-level modification



# Main Components of LLM Agent

## □ Main components

- **Agent Model:** A powerful language model that can (1) conduct basic reasoning by following the instruction; (2) generate tool calling commands and understand tool outputs ...
- **Planning:** Decompose complex task and conduct step-by-step reasoning
- **Memory:** Store records of interactions from the agent's (long/short term) previous tasks
- **Tools:** Manage agent's calling of external API tools

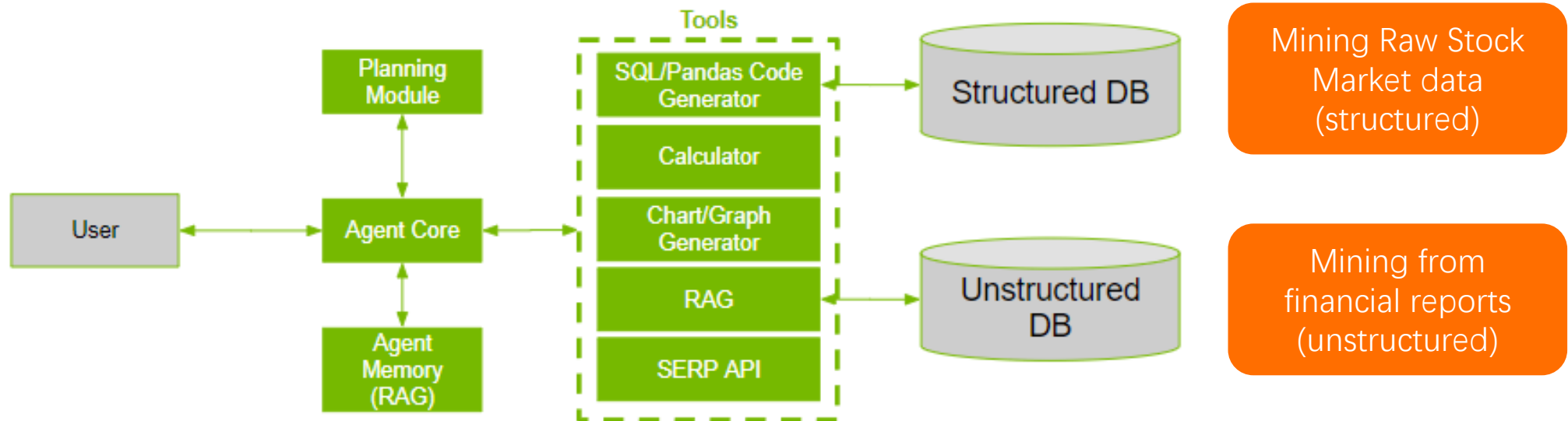


# Example LLM Agent

## □ Main components

- **Agent Model:** A powerful language model that can (1) conduct basic reasoning by following the instruction; (2) generate tool calling commands and understand tool outputs ...
- **Planning:** Decompose complex task and conduct step-by-step reasoning
- **Memory:** Store records of interactions from the agent's (long/short term) previous tasks
- **Tools:** Manage agent's calling of external API tools

*Data-Analytics  
Agent  
By Nvidia*



# Challenges & Techniques of Agent Components

## □ Main components

- *Agent Model: How to (1) conduct basic reasoning by following the instruction; (2) generate tool calling commands and understand tool outputs ...*
  - The **selection / finetuning of suitable agent model** is vital to the performance
    - GPT-3.5 Turbo is effective as code interpreter, particularly in its ability to understand and translate complex task descriptions into functional code;
    - GPT-4 demonstrates weaker consistency compared with GPT-3.5

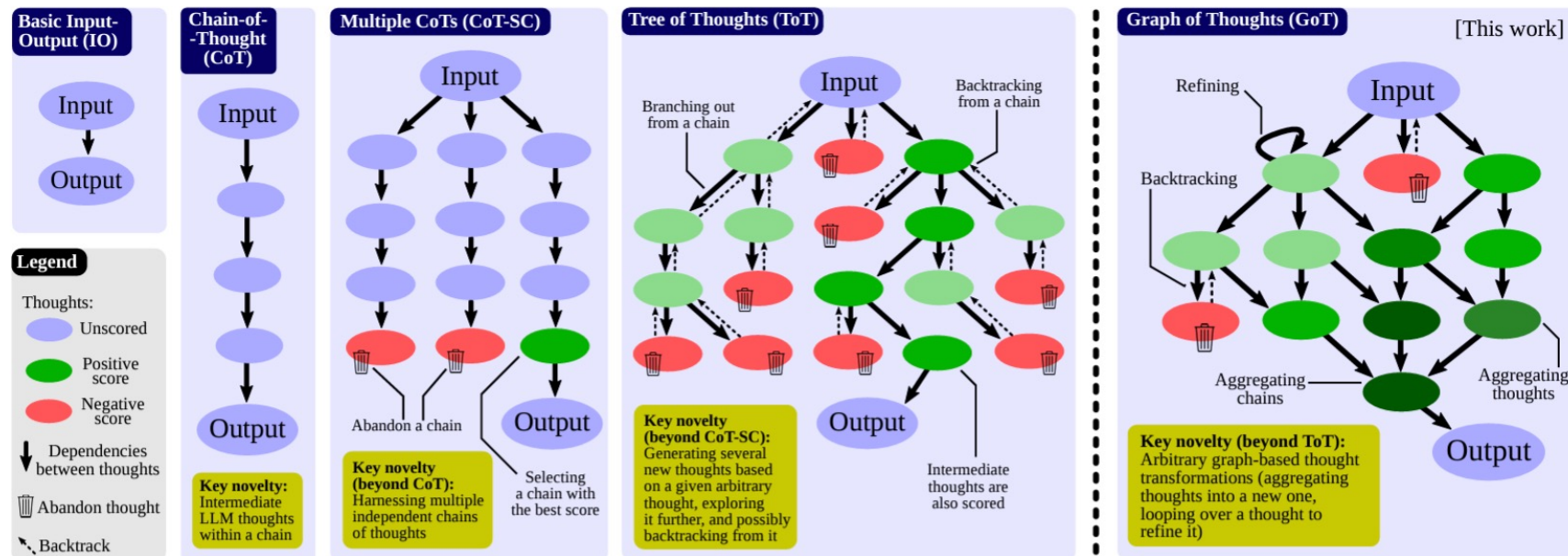
Table 1: Workflow of the proposed model: Application across diverse datasets and output formats

LLM Model	Product	Parameter	I/P Description	Accurate Result	Quality
GPT-4 Turbo	OpenAI	1.96 trillion	10	6	★★★★☆
GPT-4	OpenAI	1.76 trillion	10	5	★★★★☆
GPT-3.5 Turbo	OpenAI	154 billion	10	7	★★★★☆
GPT-3.5	OpenAI	125 billion	10	5	★★★★☆
Google Bard	Google	1.56 trillion	10	4	★★☆☆☆
LLama	Meta	70 billion	10	2	★☆☆☆☆
Hugging Face	Hugging Face	355M	10	2	★☆☆☆☆

# Challenges & Techniques of Agent Components

## □ Main components

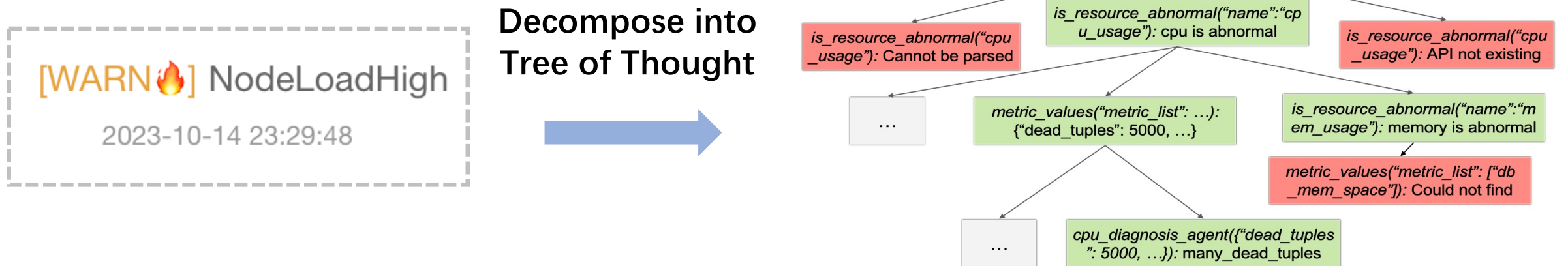
- *Planning: How to decompose complex task and conduct step-by-step reasoning*
  - **Chain-of-Thought (CoT):** Explicitly break down the task into **a sequence of intermediate steps**
  - **Tree-of-Thought (ToT):** Decompose the thought process into **multiple branches and sub-nodes**
  - **Graph-of-Thought (GoT):** LLM thoughts as vertices, edges as thought dependencies → **Arbitrary thoughts can be aggregated** by constructing vertices with multiple incoming edges



# Challenges & Techniques of Agent Components

## □ Main components

- *Planning: How to decompose complex task and conduct step-by-step reasoning*
  - **Chain-of-Thought (CoT):** Explicitly break down the task into **a sequence of intermediate steps**
  - **Tree-of-Thought (ToT):** Decompose the thought process into **multiple branches and sub-nodes**
  - **Graph-of-Thought (GoT):** LLM thoughts as vertices, edges as thought dependencies → **Arbitrary thoughts can be aggregated** by constructing vertices with multiple incoming edges





# Challenges & Techniques of Agent Components

## □ Main components

➤ *Memory: How to store historical messages for caching or effective task-solving*

➤ **LLM may forget past actions when resolving the same task**

- → **Short-Term Memory:** Memory information is directly written into prompt (e.g., for maintaining the internal state during executing a task)

➤ **The effective plans for historical task may be useful for current task**

- → **Long-Term + Short-Term Memory:** Long-term memory for stable knowledge (e.g., reuse behaviors and thoughts in past plans for current situation); Short-term memory for flexible planning (e.g., adjust the plan with recent feedback)
- **Memory formats:** Different memory formats possess distinct strengths

Memory Format	Advantage	Case
Natural language	Flexible; Rich semantics;	Reflexion: Stores <b>experiential feedback</b> in natural language within a sliding window.
Embeddings	Benefit retrieval	ChatDev: Encode <b>dialogue history</b> into vectors for reuse
Databases	Support Complex Query	DB-GPT: Agents are fine-tuned to <b>understand and execute SQL queries</b> , enabling them to interact with databases

# Challenges & Techniques of Agent Components

## ❑ Main components

- *Tools: How to prepare tool APIs for better tool calling and result understanding*

- Vast number of Tools (length and latency issues)

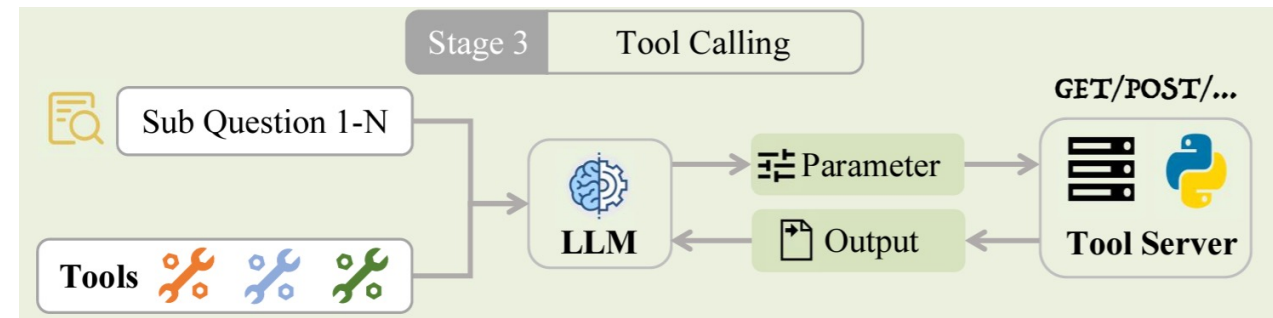
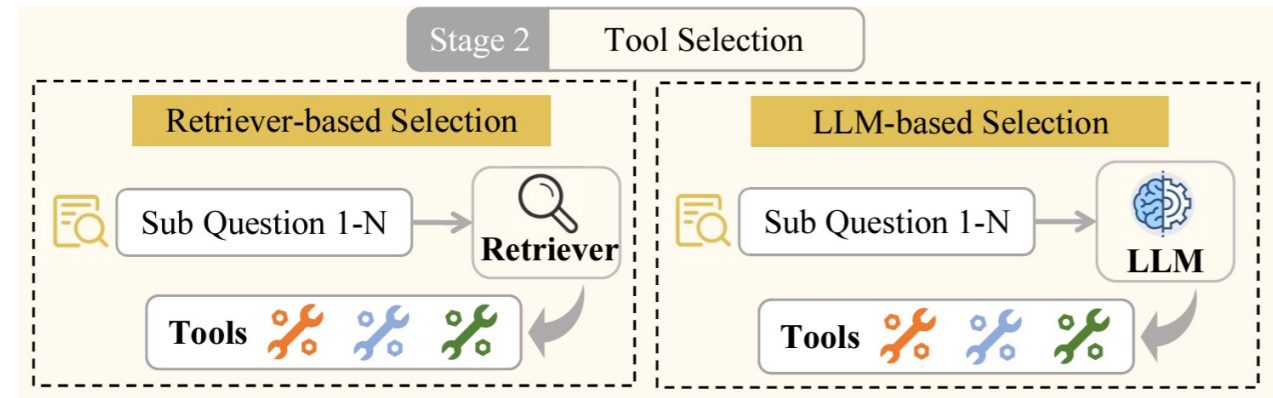
- → **Tool Selection**

- Retriever-based tool selection
- LLM-based tool selection

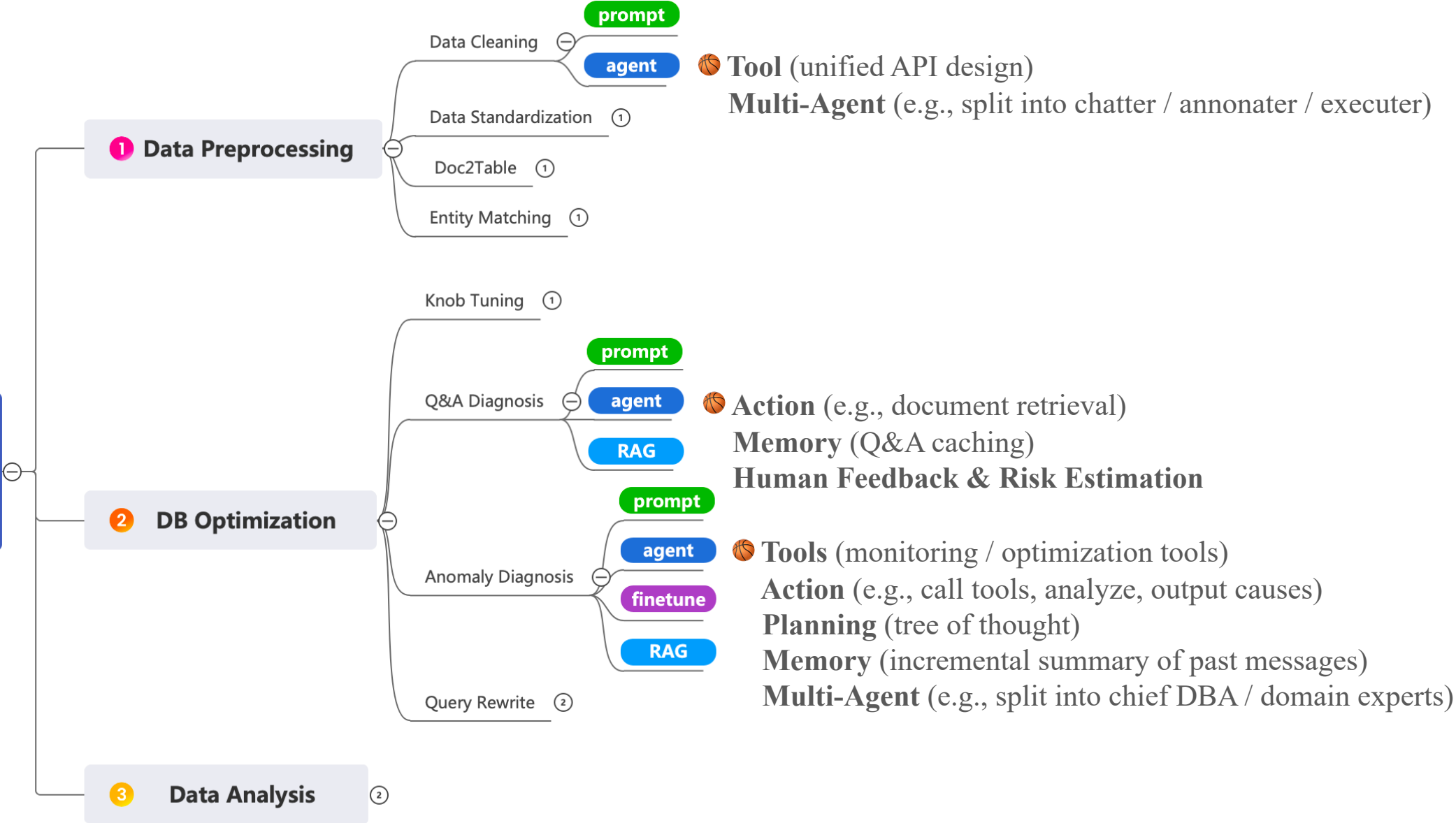
- Adherence to API parameter and formats

- → **Tool Calling**

- LLM Finetuning
- Output Constraints (e.g., in json format)
- ...



# LLM Agent Techniques for Data Management



# LLM Agent for Data Standardization

- ❑ Problem: The complexity of tools like Pandas require great human efforts to write code for various column types

Input Table  $T$

Name	Admission Date	Address
Abby	Fri Jan 1st 10:36:28 2021	1234 west main heights LA 57033
Scott	1996.07.10 AD at 15:08:56	1111 S Figueroa St, LA, 90015

Standardized Table  $T'$

Name	Admission Date	Address
Abby	01/01/2021 10:36:28	1234 W. Main Hts., LA, 57033
Scott	01/15/2020 15:08:56	1111 S. Figueroa St., LA, 90015

*Data Standardization: Unify the format of values within the same column (e.g., the “Admission Date” column)*

```
1 def standardize_address(addr):
2     # Extract street number and street name
3     street = pd.Series(addr).str.extract(r'(\d+ [^,]+)').squeeze()
4     # Extract state name
5     state = "LA"
6     # Extract zipcode
7     zipcode = pd.Series(addr).str.extract(r'(\d{5})').squeeze()
8     # Output standardized address
9     return f"{street}, {state}, {zipcode}"
```

*Case-by-Case Analysis and Coding* ✘

# LLM Agent for Data Standardization

❑ **Core Idea: Convert textual instructions into declarative API calls and automate data standardization with LLM agent**

❑ **Challenges**

- (1) How to design declarative and unified tool APIs for data standardization?
- (2) How to optimize the interaction between data scientists and LLM agent?

❑ **CleanAgent**

➤ **Tools**

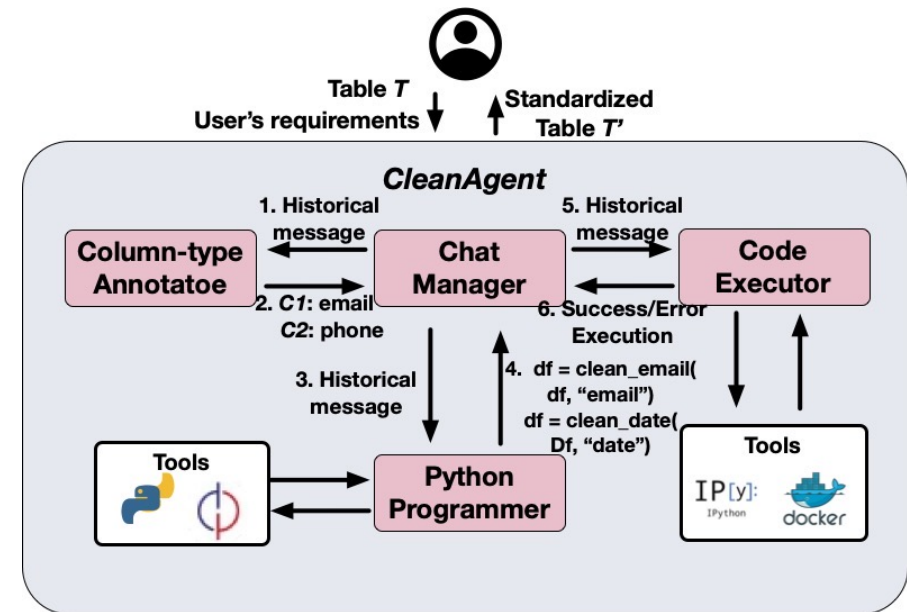
- Unified tool API:

`clean_type(df, column_name, target_format)`

*df*: input dataframe (table)

*column\_name*: the column needs to standardize

*target\_format*: the target standardization format users specified



# LLM Agent for Data Standardization

❑ **Core Idea: Convert textual instructions into declarative API calls and automate data standardization with LLM agent**

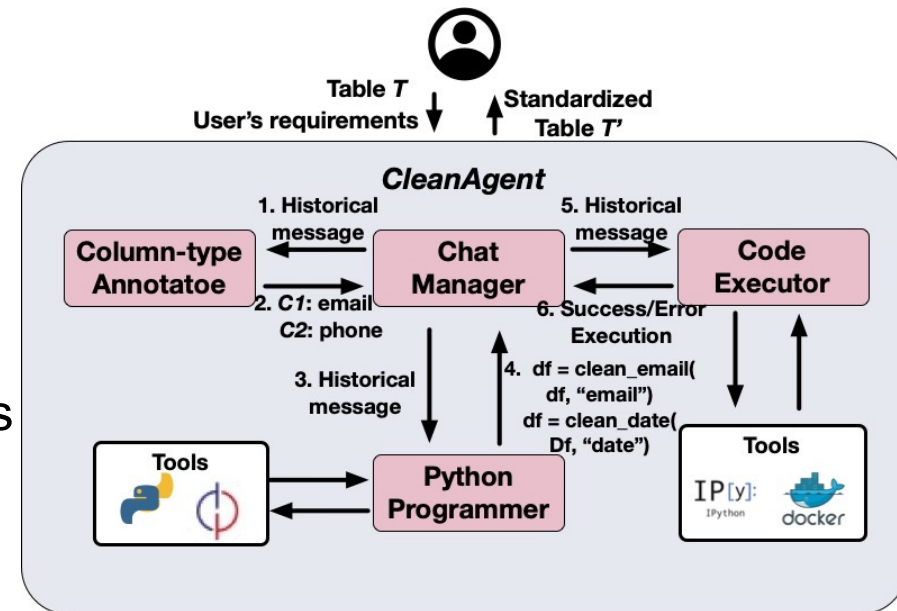
❑ **Challenges**

- (1) How to design declarative and unified tool APIs for data standardization?
- (2) How to optimize the interaction between data scientists and LLM agent?

❑ **CleanAgent**

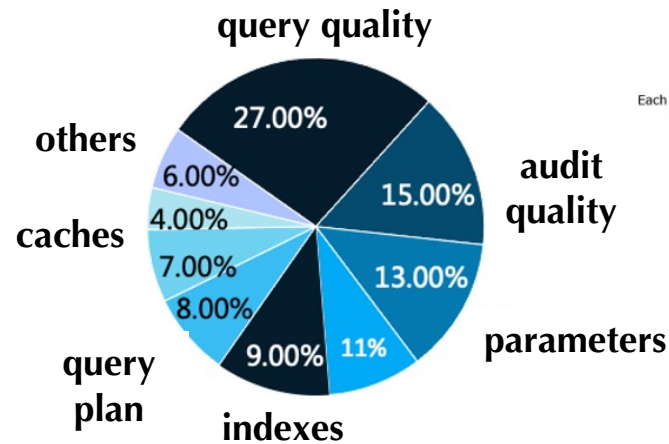
➤ **Multi-Agent for Planning (ChatGPT as the model)**

- **Chat Manager (Mem):** Store historical message
- **Column Annotator (Model):** Annotate type for each table column
- **Python Programmer (Model):** Generate code with candidate APIs
- **Code Executor (Action):** Execute code and feed result to Chat Manager

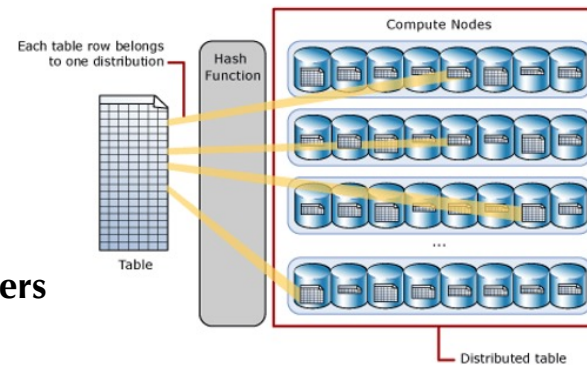


# LLM Agent for Database Diagnosis

- **Numerous Documents:** A single database product provides over **10,000+ pages** of materials (e.g., operational guides and instructions). It's tricky for junior DBAs to fully grasp all of this.
- **Significant Operational Pressure:** The number of cloud database instances is massive, but there's a shortage of DBAs.
- **Complex Issues:** Many urgent problems are interconnected, making it hard to respond quickly, resulting in economic losses.



Various Root Causes



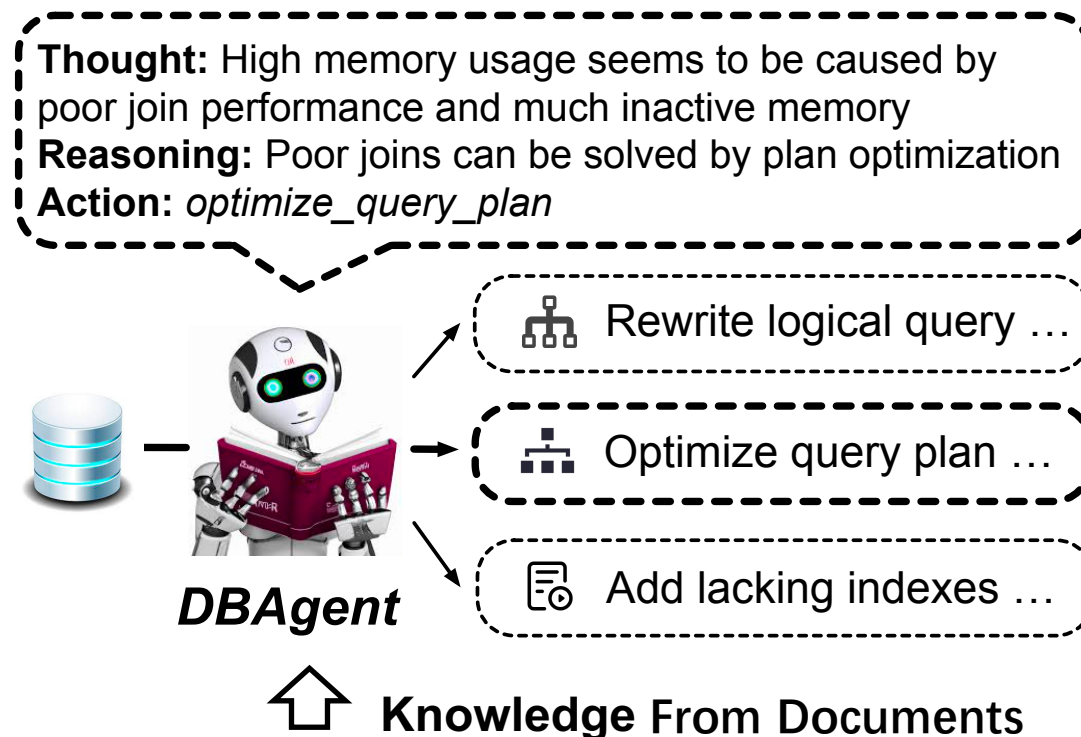
1M+ Cloud Instances



Correlated Online Issues

# LLM Agent for Database Diagnosis

- **Goal:** Utilize LLM as an “intelligent DBA assistant”, learning from human operational experience and diagnose root causes.
- **Benefits:** Save manpower; Enhance proficiency in using maintenance tools; Improve the ability to generalize operational knowledge.

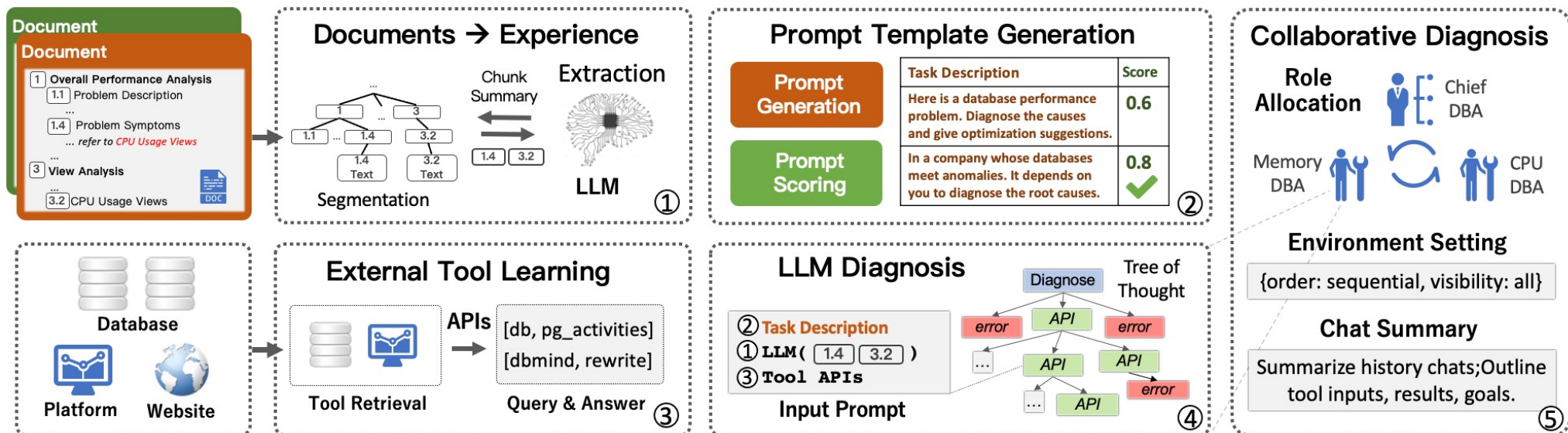


Other methods like *expert systems* are hard to reuse similar cases



# LLM Agent for Database Diagnosis

- **Tool API:** Monitoring Tools (e.g., logs, views, metrics); Optimization Tools (e.g., index)
- **External knowledge Extraction:** Segmentation of text blocks; generation of summary indexes; extraction of formalized knowledge.
- **Planning:** Improving tool usage accuracy through tree-search-based method.
- **Multi-Agent optimization:** **Chief DBA** (diagnostic scheduling, finding summarization); **Domain Experts** (e.g., resource anomaly, query optimization), **Chat Summarizers**; Users (providing effective feedback).



# Agent-Based Data Management

Methods	Model	Memory	Tool	Planning	Extra Knowledge	Multi-Agent
Data Standardization	GPT	Historical messages	Declarative APIs	Iterative	N/A	N/A
Database Diagnosis	GPT-4 Llama	Results of historical actions	Monitoring Tools; Optimization Tools	Tree of Thought	Documents	√

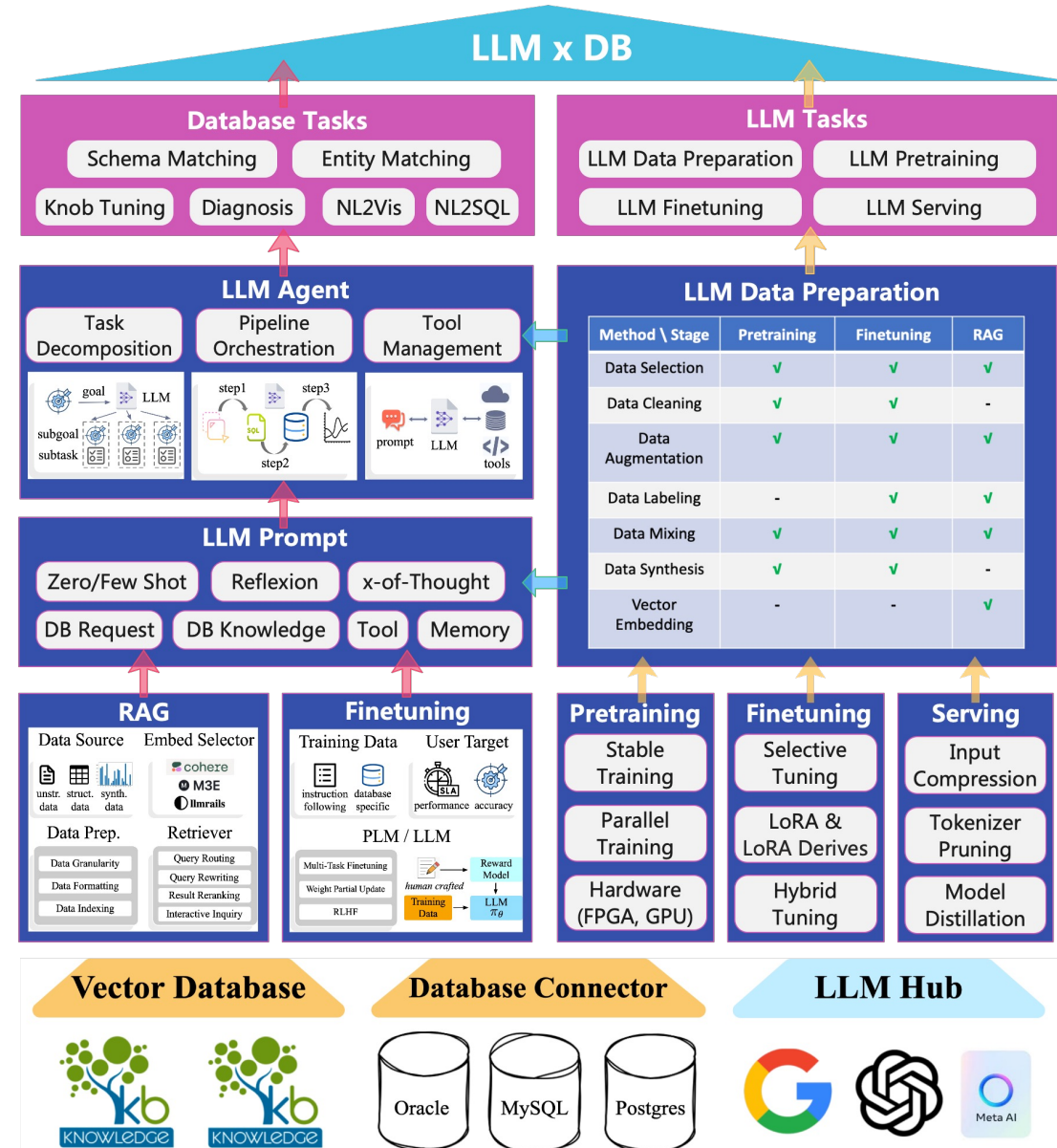
# Take-aways

---

- ❑ **Agent integrates capabilities like planning, obtaining external knowledge and reacting (i.e., tool learning), and so works better than prompt-only ones**
- ❑ **Existing Tool APIs and prompt templates for Agent are manually written, which also have the scalability problem**
- ❑ **Planning is vital for Agent to resolve problems like early-stop, hallucination (e.g., inaccurate tool calling), and self-inconsistent**
- ❑ **Memory reminds LLM of the historical messages (avoiding repeated actions or serving as cache), but also causes great prompt redundancy**
- ❑ **Multi-Agent offers new mechanisms like collaboration (v.s., single-agent), but the superiority needs to further explore, especially in real-world cases (where single-agent already causes great overhead)**

# RAG for Data Management

- ❑ Data Management tasks
- ❑ LLM Prompt for Data Management
  - Instruction Prompting
  - Few-Shot Prompting
- ❑ LLM Agent for Data Management
  - Agent Models + Memory
  - Reasoning / Planning Strategies
  - Tool Management & Learning
- ❑ RAG for Data Management
  - Semantic Segmentation
  - Result Retrieval
  - Result Reranking
- ❑ Finetuning for Data Management
  - Reparameterization / LLM Adapter
- ❑ Data Preparation for LLM
- ❑ Open Problems



# Motivation of Retrieval Augmented Generation

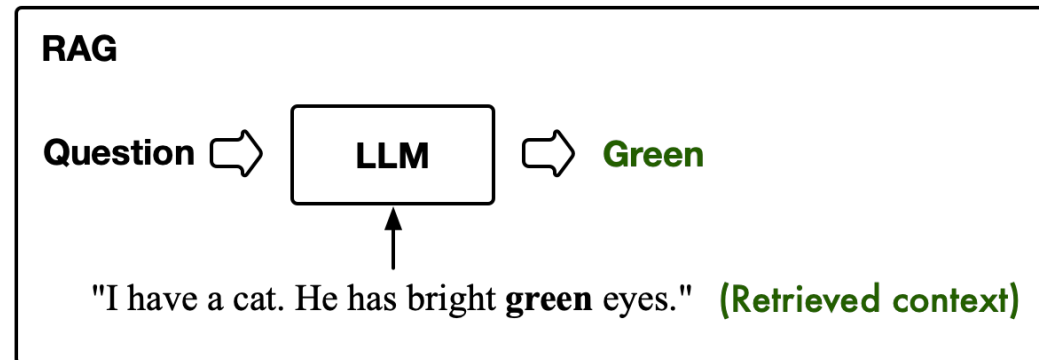
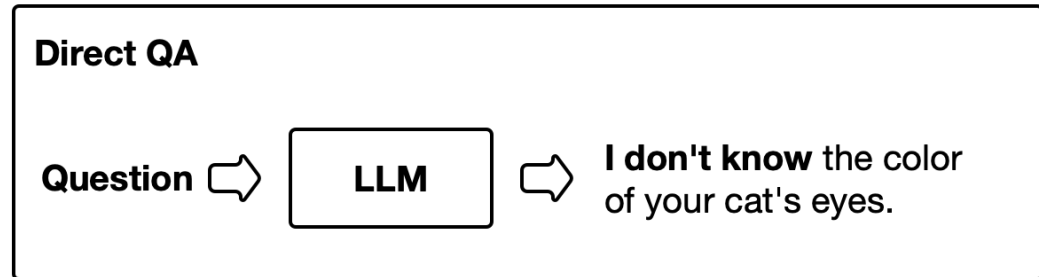
## Drawbacks of LLMs

- Hallucination
- Outdate information
- Low efficiency in parameterized knowledge
- Weak reasoning capability

## Practical Requirements

- Domain-Specific Accurate Q&A
- Frequent Data Update
- Explainability of Generated Answer
- Controllable Cost
- Data Privacy Protection

**Question:** What color of my cat's eyes?  
**Correct Answer:** Green.



A motivative example.

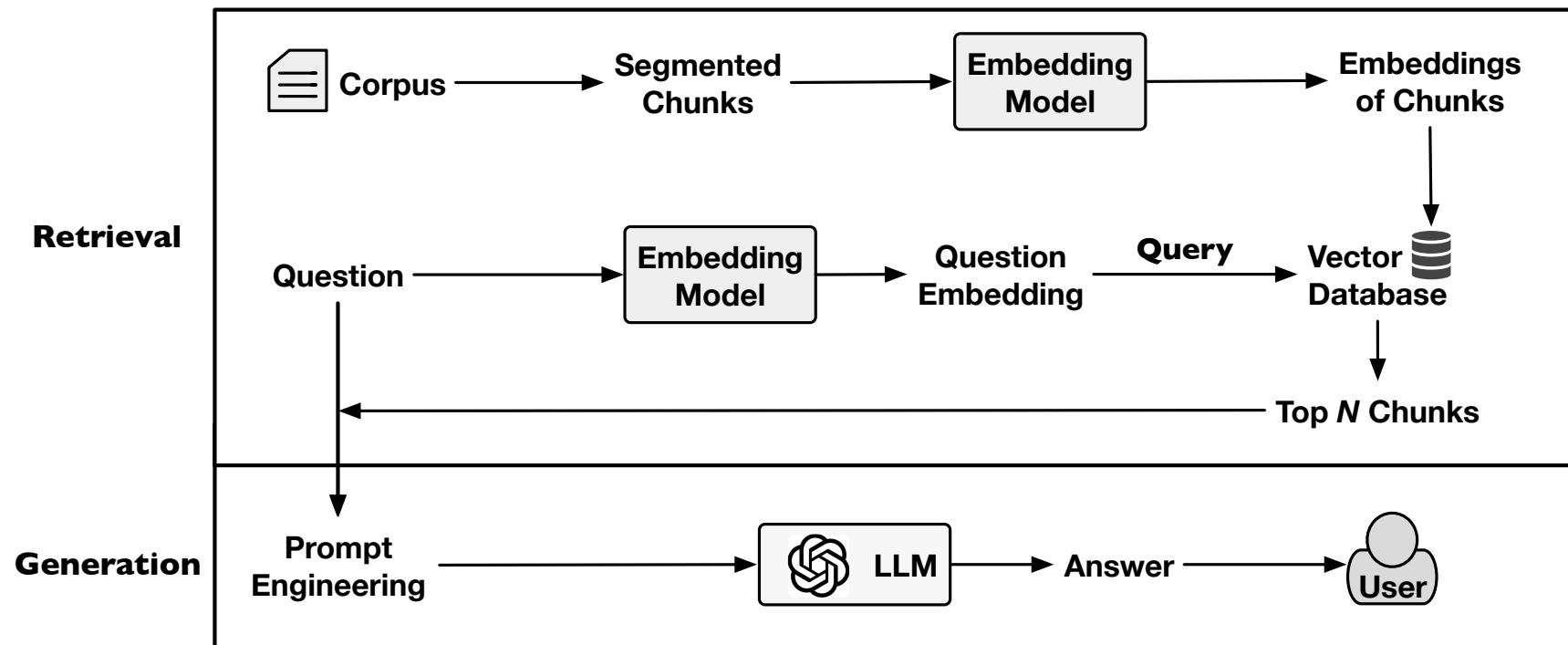
# Retrieval and Generation Pipeline

## Retrieval phase:

- The given corpus will be segmented into many chunks.
- For a question, a retriever identifies and selects the top K most related chunks as context.

## Generation phase:

- The question, alongside the context, will be inputted into a LLM to generate an answer.



# Limitations of RAG

---

## Retrieval limitation:

- **Chunk selection based on embedding similarity may lead to misleading or irrelevant chunks and missing crucial information.**
- **Methods:**
  - *Semantic segmentation*
  - *Reranking*
  - *Retrieval techniques*

## Generation limitation:

- **LLMs may produce content that is not supported by the retrieved context.**
- **Methods:**
  - *Prompt engineering*
  - *Using LLMs with high proficiency levels*

# Semantic segmentation

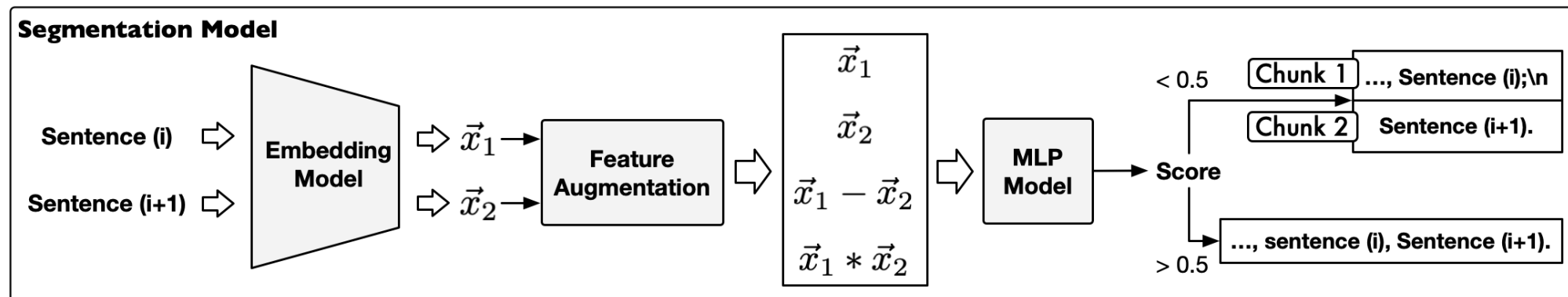
**Motivation:** Segment the corpus into semantically coherent chunks, ensuring that the retrieved information is semantically complete and relevant.

## Training:

- Collecting *sentence pairs with labels*. (*label=1 means they are semantically relevant.*)
- Fitting the mapping between sentence pairs and labels using a model (Embedding model + MLP).

## Inference:

- Each two adjacent sentences in the corpus is judged by the segmentation model.





# Reranking Technique

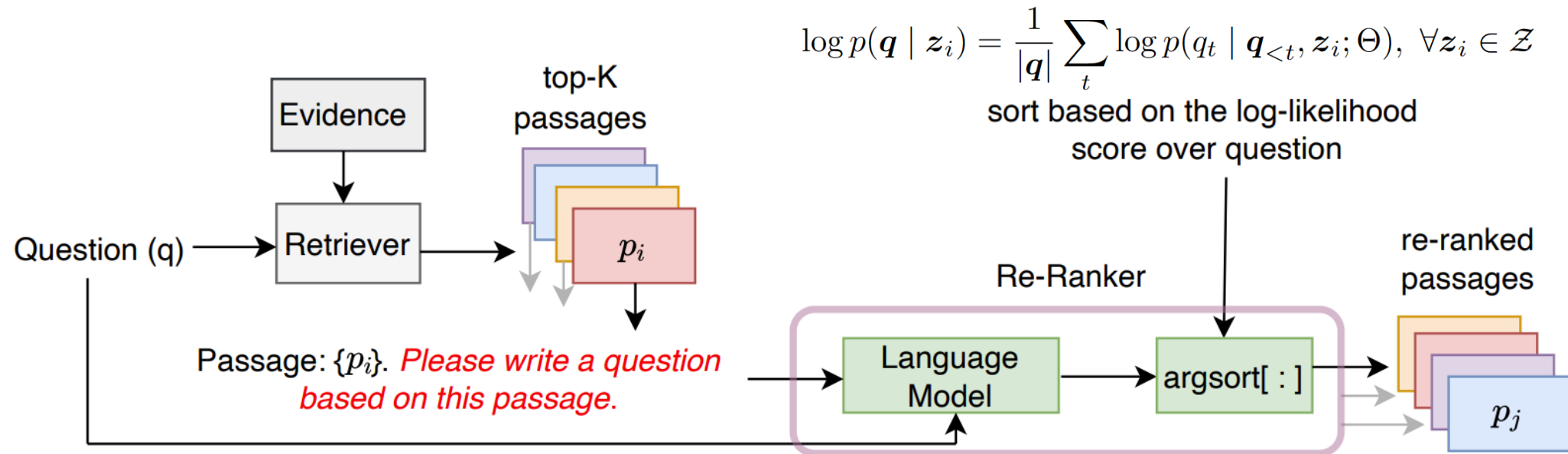
**Reranking:** **Reorder chunks** according to their relevance to the question, and then **select the top K** based on specific metric (e.g. relevance).

- Enhancer: Boost the relevance and quality of context.
- Filter: Filter out misleading or irrelevant chunks.

**Example:**

**UPR | EMNLP 2022**

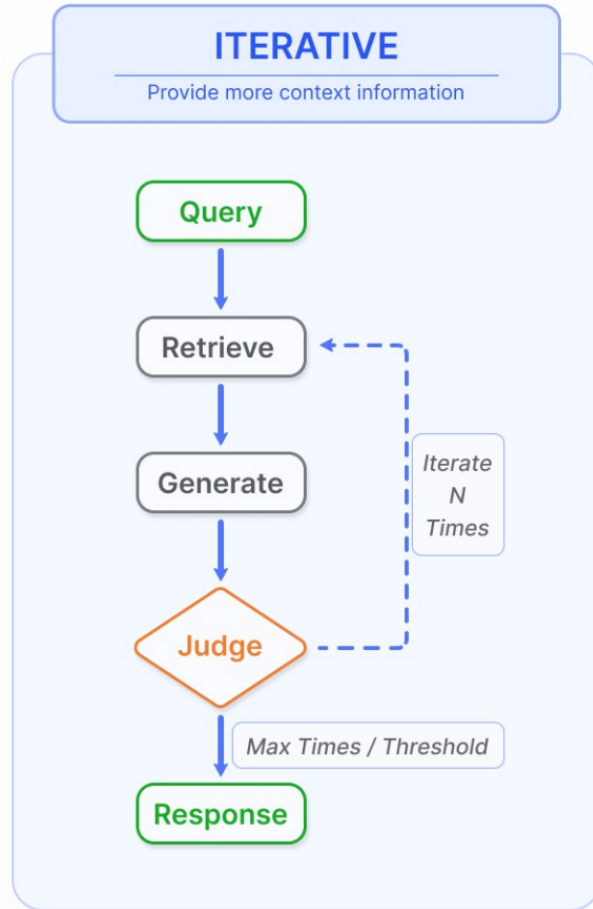
Using a **PLM** to compute the probability of the input question conditioned on a retrieved passage.



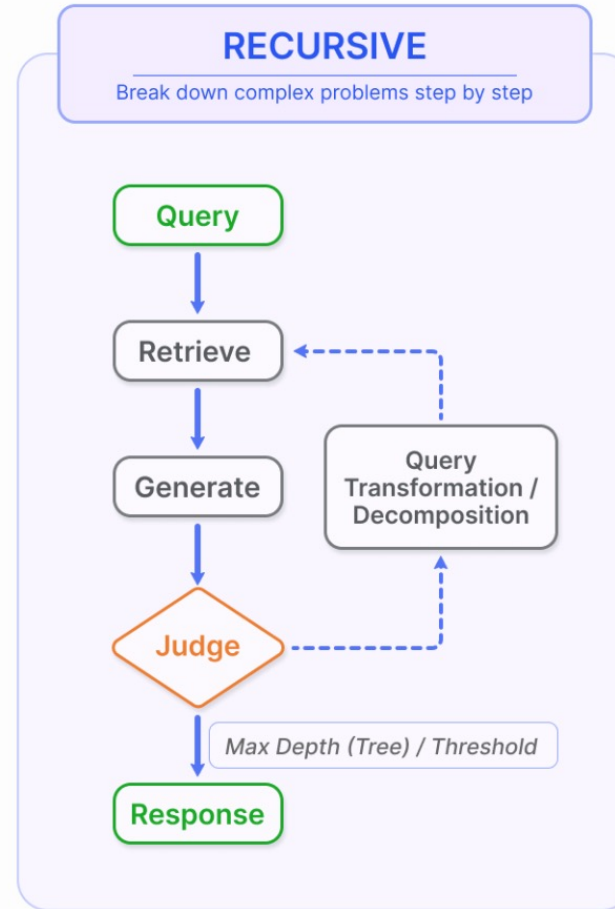
# Retrieval Technique

## Retrieval augmentation

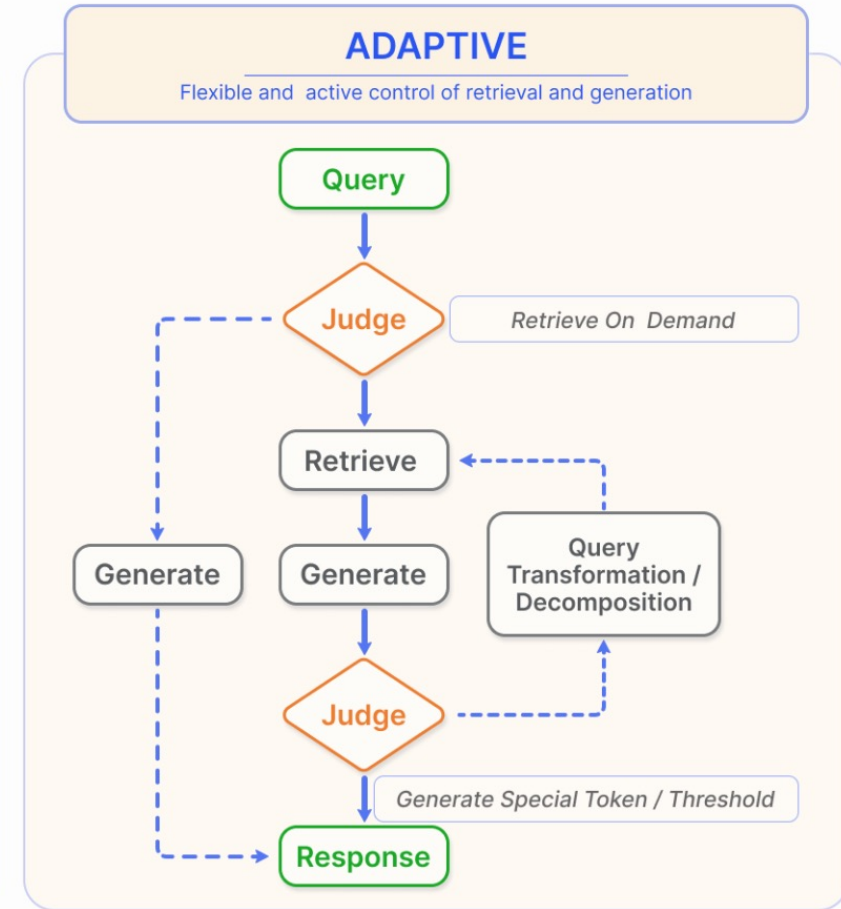
- Iterative Retrieval



- Recursive Retrieval

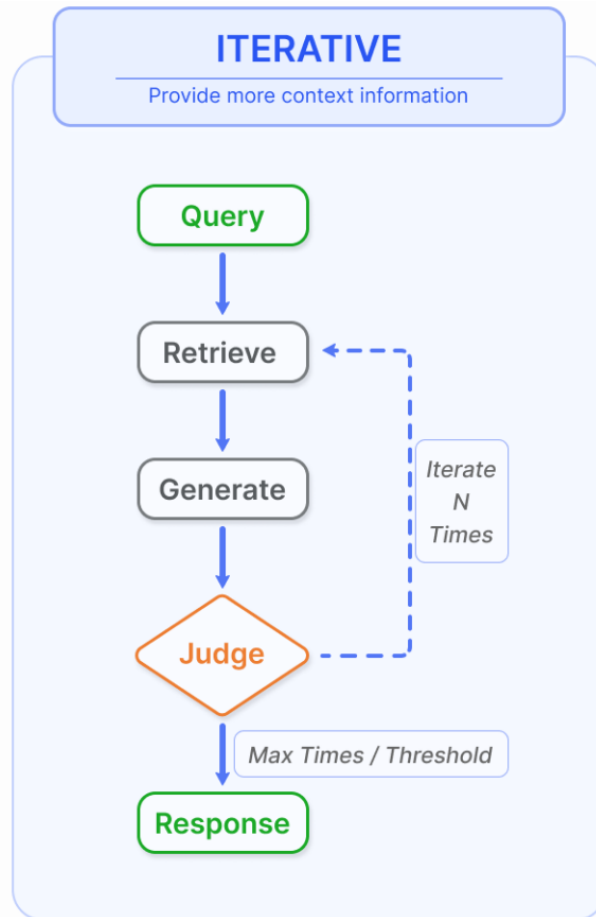


- Adaptive Retrieval



# Retrieval Technique

- **Iterative Retrieval:** Repeatedly search based on the initial query and the generation result.  
**Advantage:** Enhance robustness of subsequent answer generation.



## ITER-RETGEN | EMNLP 2023

ITER-RETGEN will iterate for specific times, in each iteration, it uses **generated content and the initial query** to retrieve.

Question:  $q$

What is the height of the player who won the 2015 AFL Rising Star award?

Retrieval:  $q \rightarrow \mathcal{D}_q$

**Title:** 2015 AFL Rising Star **Context:** The NAB AFL Rising Star award is given annually to a stand out young player in the Australian Football League. **The award was won by Jesse Hogan of Melbourne**

Retrieval-Augmented Generation:  $\mathcal{D}_q \parallel q \rightarrow y_1$

**The 2015 AFL Rising Star award was won by Jesse Hogan of Melbourne. Jesse Hogan is a professional Australian rules footballer. He is 198 cm tall. So the answer is 198 cm**

Iteration 1

Question:  $q$

What is the height of the player who won the 2015 AFL Rising Star award?

Generation-Augmented Retrieval:  $y_1 \parallel q \rightarrow \mathcal{D}_{y_1 \parallel q}$

**Title:** Jesse Hogan **Context:** Jesse Hogan ... playing for the Melbourne Football Club. A key forward, **Hogan is 1.95 m tall ...** made his AFL debut in the 2015 season and won the Ron Evans Medal as the AFL Rising Star

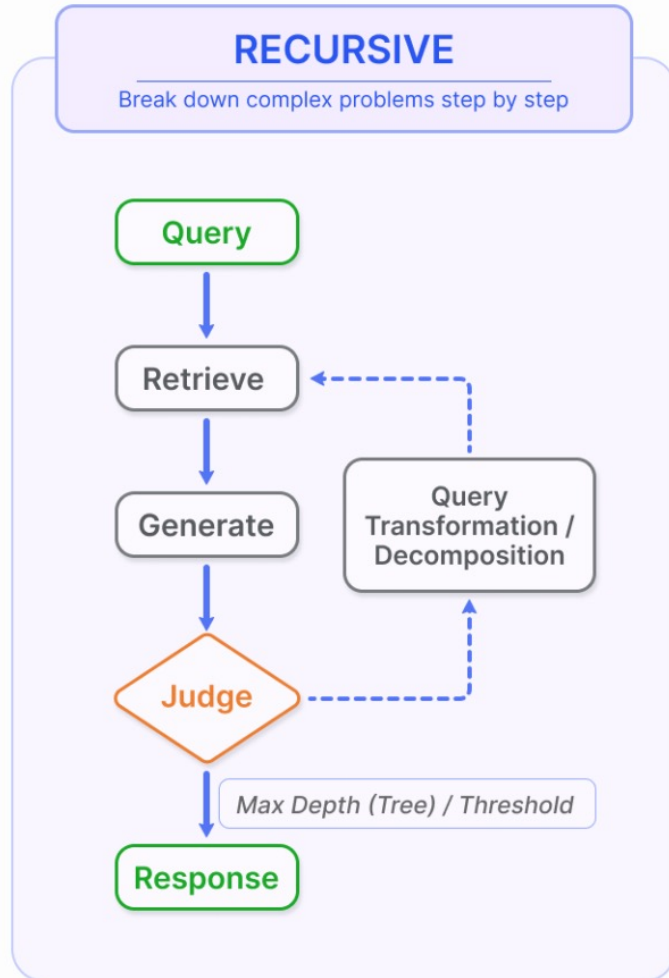
Retrieval-Augmented Generation:  $\mathcal{D}_{y_1 \parallel q} \parallel q \rightarrow y_2$

**The 2015 AFL Rising Star award was won by Jesse Hogan of Melbourne. Jesse Hogan is 1.95 m tall. So the answer is 1.95 m**

Iteration 2

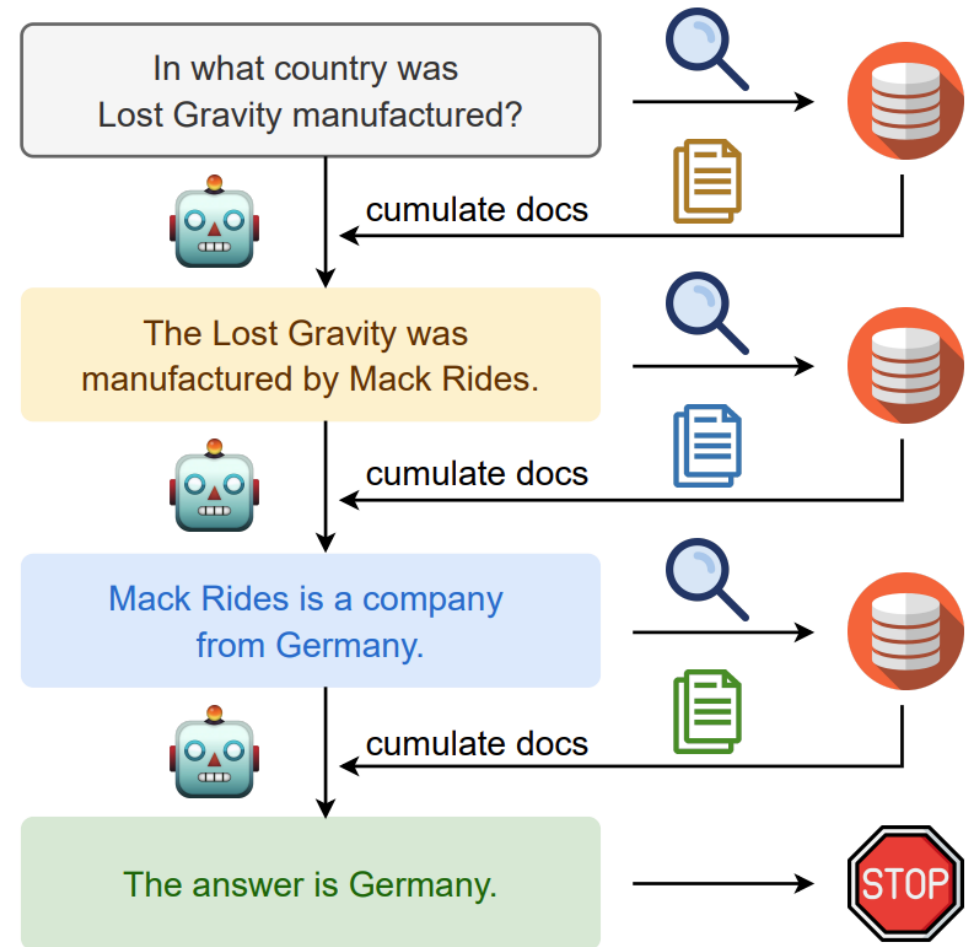
# Retrieval Technique

- **Recursive Retrieval:** Iteratively refining search queries based on previous retrieval result.  
**Advantage:** Improve the depth and relevance of search results.



**Example:** IRCOT will iteratively repeat two steps:

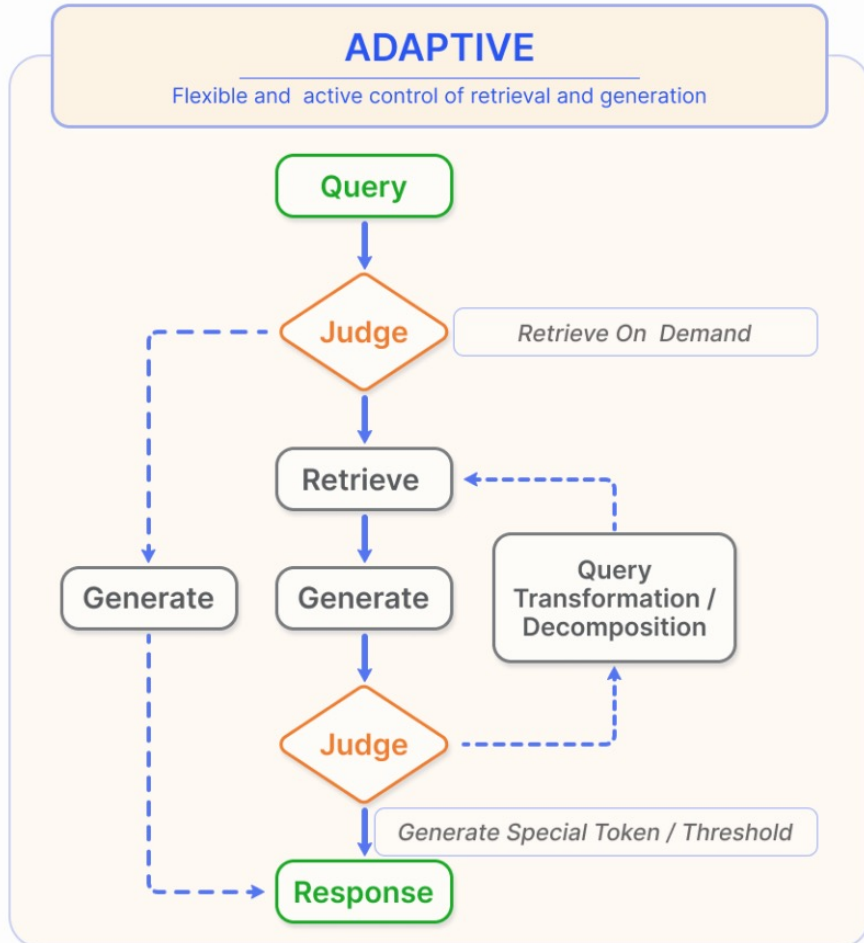
- **Generate** CoT (chain-of-thought) question based on retrieved corpus and question.
- **Retrieve** over the previous CoT question.



# Retrieval Technique

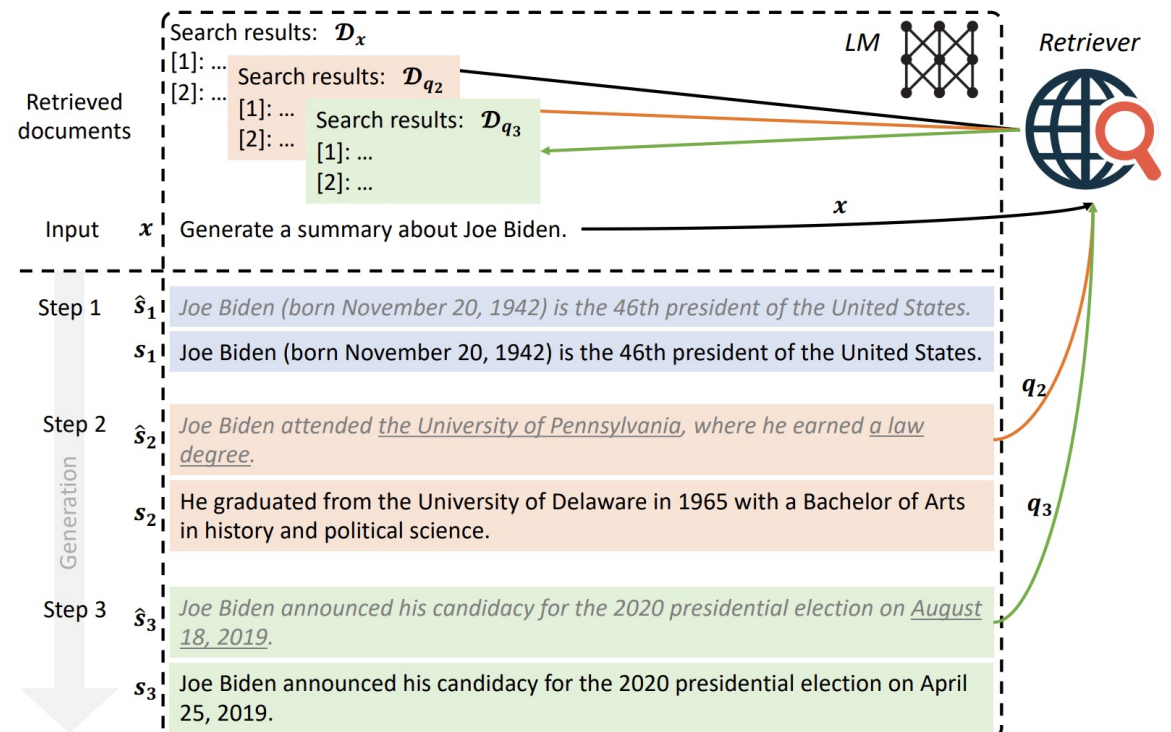
- **Adaptive Retrieval:** Enable LLMs to actively determine the moments for retrieval.

**Advantage:** Enhance the efficiency and relevance of the information sourced.



## Example: FLARE

Retrieve when LLM generates tokens with probabilities below a set threshold.



# Insights with experimental examples

- Noisy chunks retrieved considerably undermine the effectiveness of RAG systems.

<p><b>Question:</b> Who are genetically considered “kin”?</p> <p><b>Option_1:</b> [Full siblings]</p> <p><b>Option_2:</b> [All humans]</p> <p><b>Option_3:</b> [Adoptive children and full siblings]</p> <p><b>Option_4:</b> [Friends]</p>	<p><b>Scores of Chunks</b></p> <table border="1"><tr><td>Chunk 1</td><td>13.79</td></tr><tr><td>Target Chunk</td><td>13.58</td></tr><tr><td>Chunk 3</td><td>11.91</td></tr><tr><td>Chunk 4</td><td>11.55</td></tr><tr><td>Chunk 5</td><td>10.94</td></tr><tr><td>Chunk 6</td><td>7.815</td></tr><tr><td>Chunk 7</td><td>7.665</td></tr><tr><td>Chunk 8</td><td>5.490</td></tr><tr><td>Chunk 9</td><td>4.416</td></tr><tr><td>Chunk 10</td><td>1.304</td></tr><tr><td>Chunk 11</td><td>0.800</td></tr><tr><td>Chunk 12</td><td>0.255</td></tr><tr><td>Chunk 13</td><td>0.198</td></tr><tr><td>Chunk 14</td><td>0.093</td></tr><tr><td>Chunk 15</td><td>0.089</td></tr></table>	Chunk 1	13.79	Target Chunk	13.58	Chunk 3	11.91	Chunk 4	11.55	Chunk 5	10.94	Chunk 6	7.815	Chunk 7	7.665	Chunk 8	5.490	Chunk 9	4.416	Chunk 10	1.304	Chunk 11	0.800	Chunk 12	0.255	Chunk 13	0.198	Chunk 14	0.093	Chunk 15	0.089
Chunk 1	13.79																														
Target Chunk	13.58																														
Chunk 3	11.91																														
Chunk 4	11.55																														
Chunk 5	10.94																														
Chunk 6	7.815																														
Chunk 7	7.665																														
Chunk 8	5.490																														
Chunk 9	4.416																														
Chunk 10	1.304																														
Chunk 11	0.800																														
Chunk 12	0.255																														
Chunk 13	0.198																														
Chunk 14	0.093																														
Chunk 15	0.089																														
<p><b>Noisy Chunks</b></p> <p>Because nowadays, copies of these genes do reside in non-kin in your next-door neighbor and, for that matter, your worst enemy.</p> <p>..., But in truth, you share virtually all your genes with any randomly selected homo sapien on any continent.</p> <p>Genes that natural selection fully endorsed long ago-- the basic genes for hunger, for lust, for familial love-- are in everyone.</p>																															
<p>Get the correct answer [Option_1] when <math>2 \leq K \leq 10</math>. Might get wrong answers when <math>11 \leq K \leq 13</math>. Get the wrong answer [Option_2] When <math>K = 14</math>.</p>																															

# Insights with experimental examples

- Precise retrieval is a predominant part in RAG.
- The proficiency level of LLMs plays a crucial role in RAG.
- Embedding models, though useful, are not as important as LLMs.

COMPARISON ON THE QUALITY DATASET (USING GPT-4).

Model \ Metric	Accuracy in Test Set	Accuracy in Hard Set
GPT-4	77.2%	70.3%
RAPTOR+GPT-4	82.6%	76.2%
SAGE +GPT-4	<b>90.10%</b>	<b>76.3%</b>

ACCURACY ON QUALITY DATASET WITH DIFFERENT LLMs.

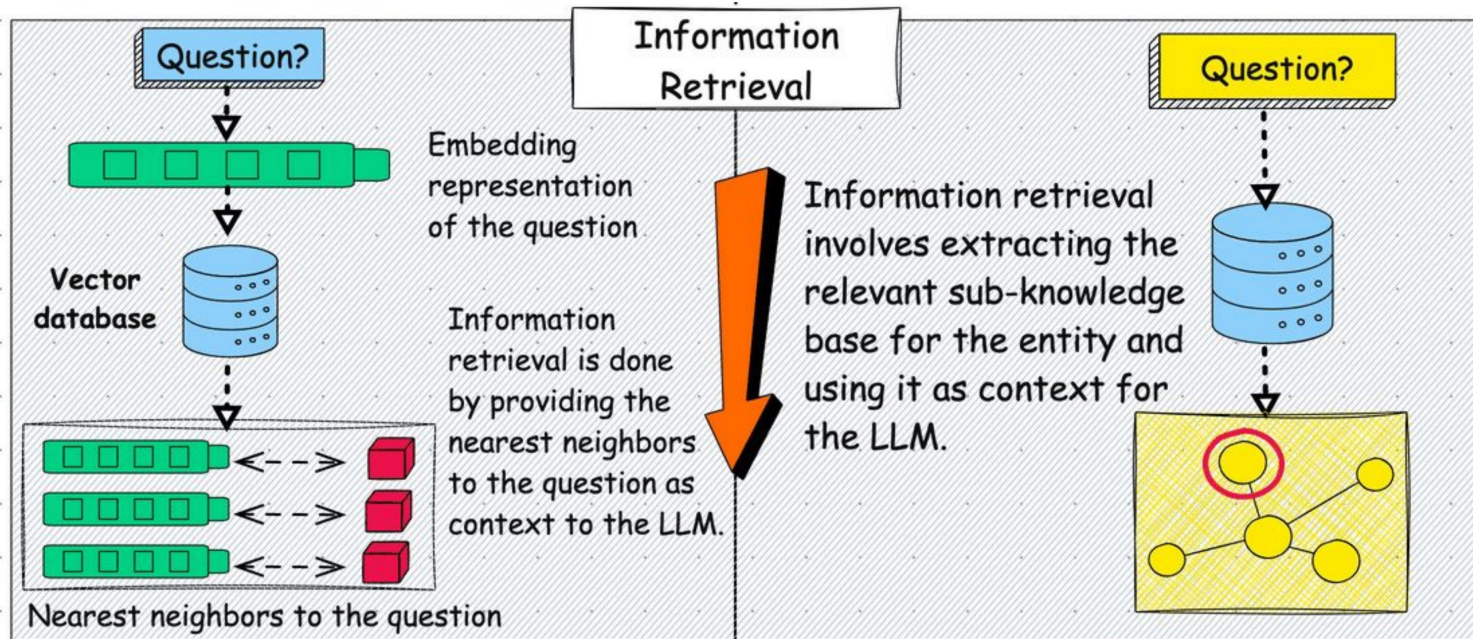
Model \ Metric	GPT-3.5 Accuracy	GPT-4o-mini Accuracy
BM25	62.70%	73.50%
DPR	60.4%	73.0%
SAGE	<b>64.5%</b>	<b>77.1%</b>

EFFECTIVENESS EVALUATION ON QUALITY AND QASPER DATASET (USING GPT-4o-MINI).

Model \ Metric	Accuracy (QuALITY)	F1-Match (QASPER)
SBERT	72.48%	37.57%
BM25	72.18%	37.30%
DPR	72.38%	37.41%
OpenAI Embedding	75.32%	38.94%

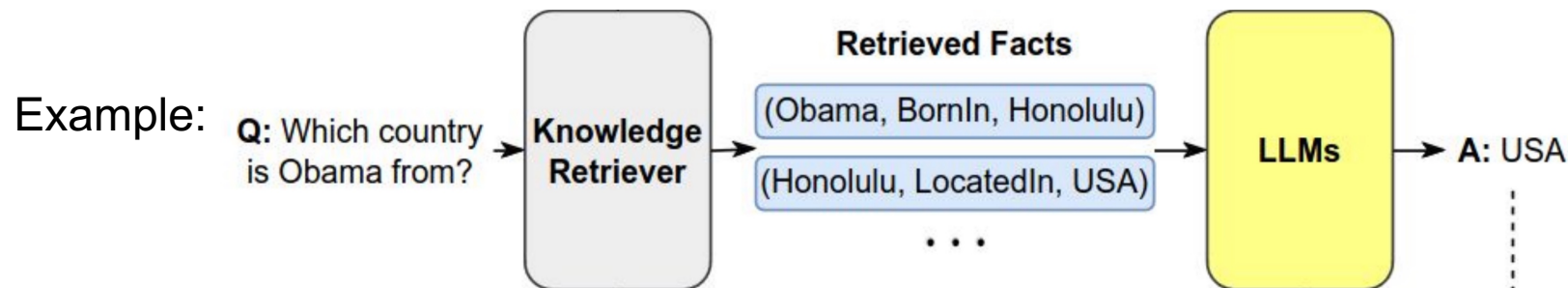
# Graph RAG

- **Graph RAG** includes a **graph database** as a source of the retrieval context sent to the LLM.  
**Retrieval Method:** Entity Linking & NL-to-Graph-query.



## Advantages:

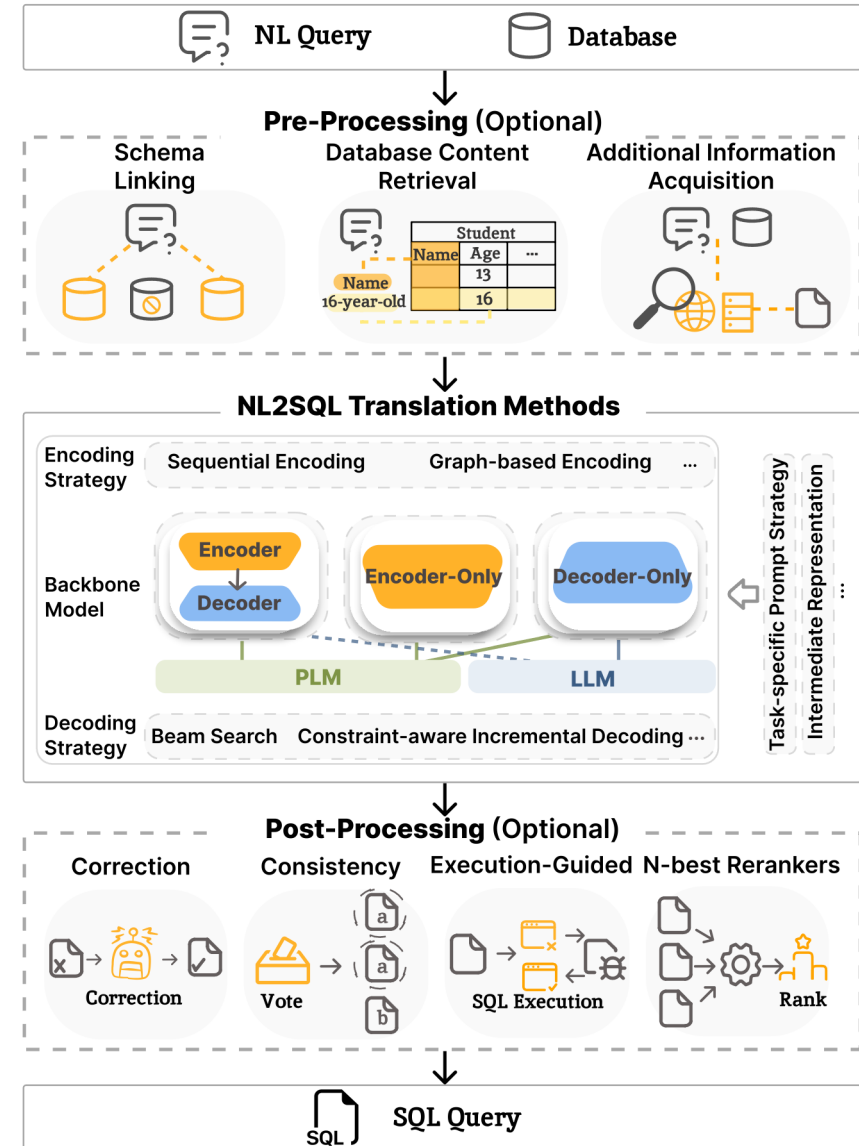
- Providing extra information like metadata for generation.
- Handling complex and nuanced queries.
- Supporting data update.





# RAG for Text2SQL

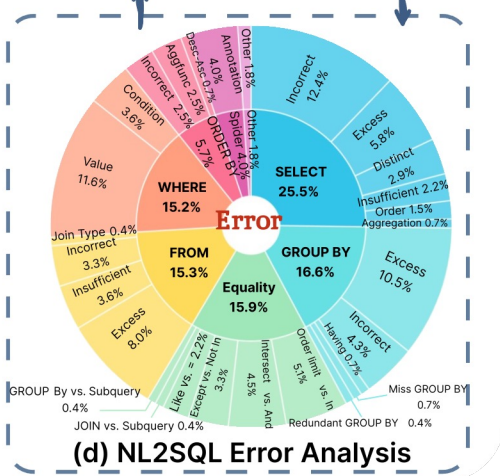
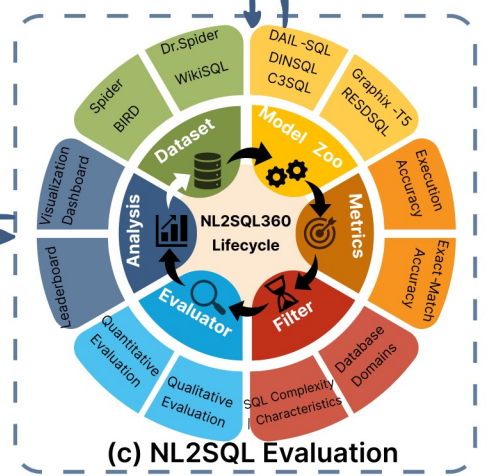
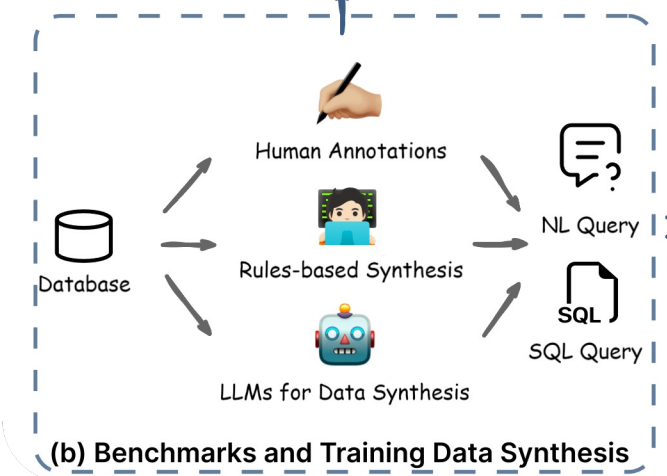
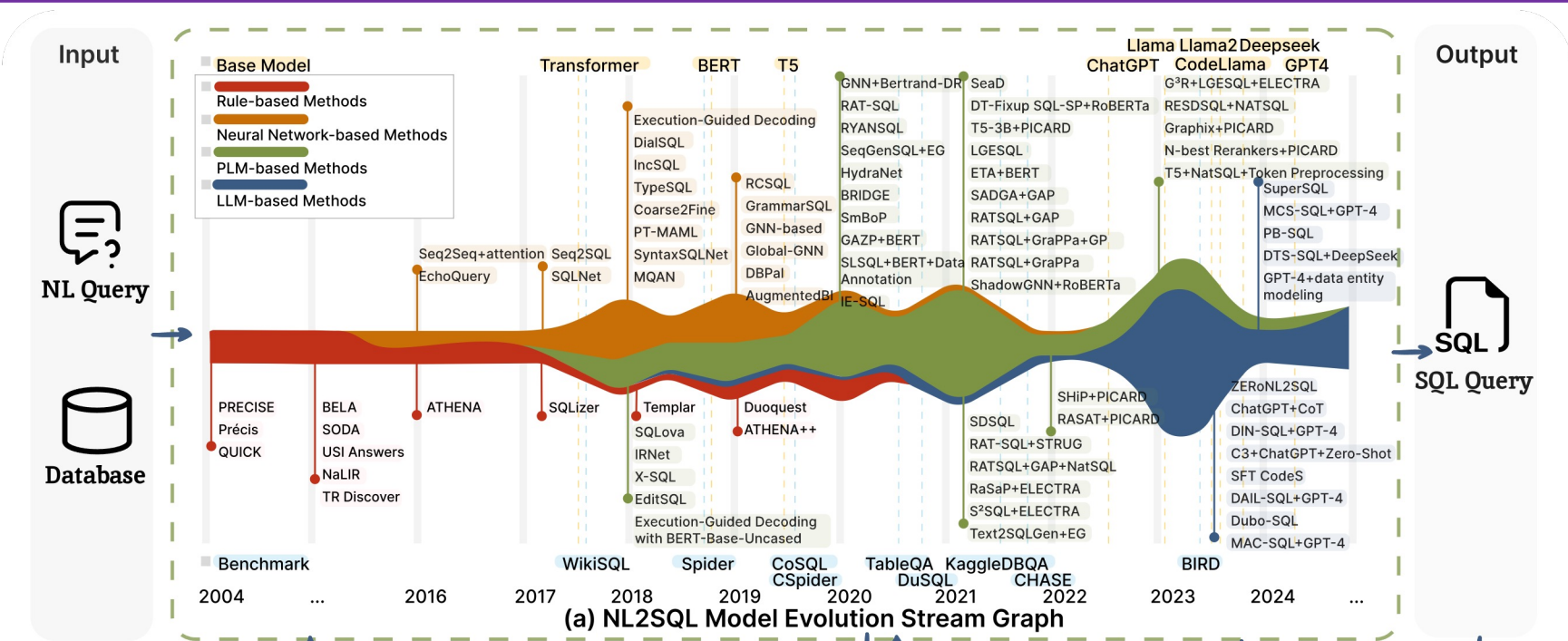
- **Pre-Processing**
  - Schema Linking
  - Database Content Retrieval
  - Additional Information Acquisition
- **NL2SQL Translation Methods**
  - Encoding Strategy
  - Decoding Strategy
  - Task-specific Prompt Strategies
  - Intermediate Representation
- **Post-Processing**
  - Correction
  - Consistency
  - Execution-Guided
  - N-best Rerankers



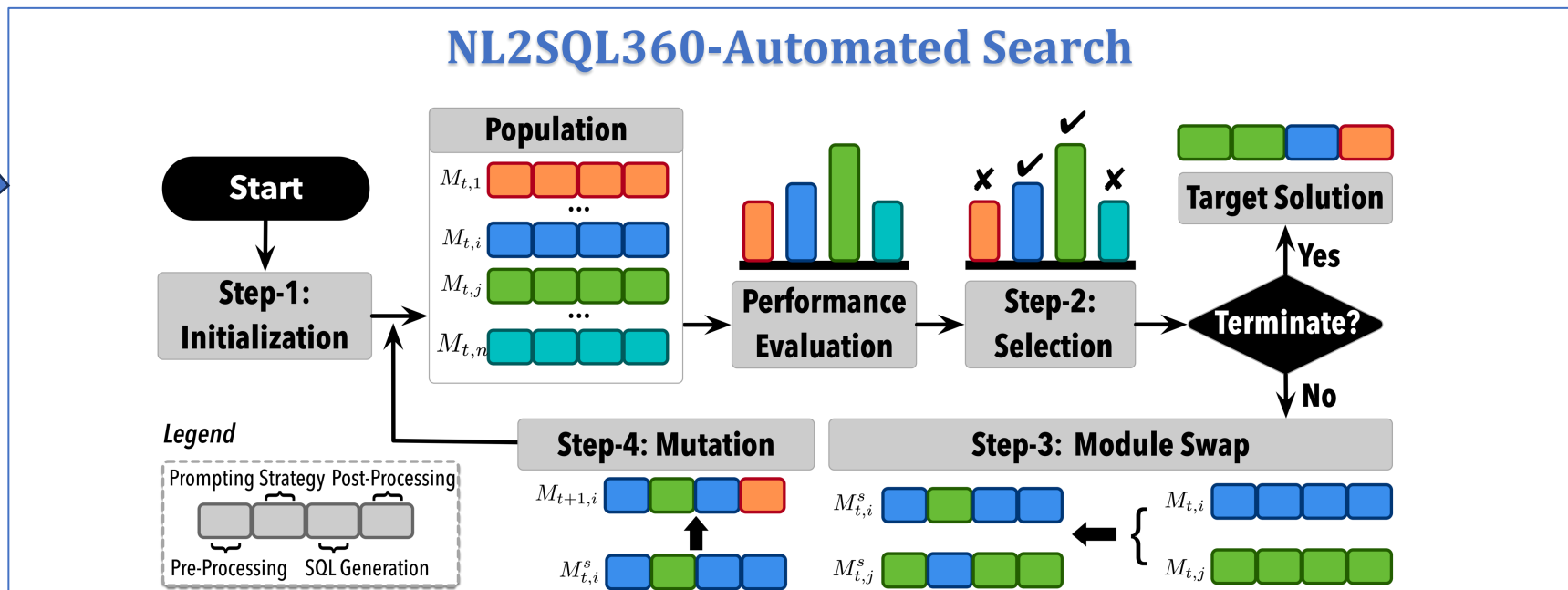
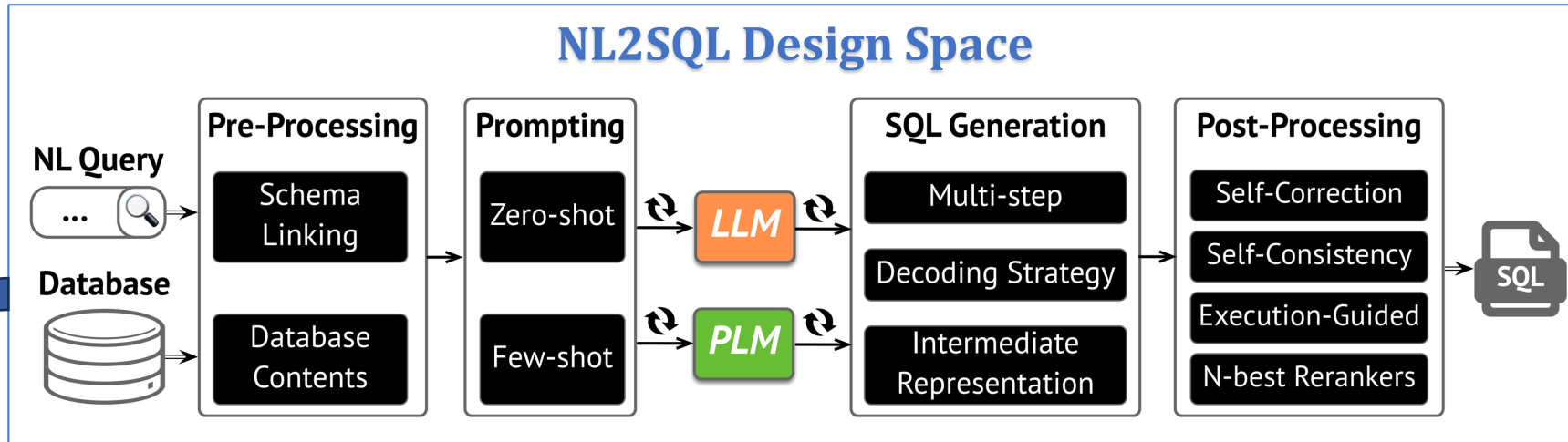
Paper: <https://arxiv.org/pdf/2408.05109>

NL2SQL Handbook: [https://github.com/HKUSTDial/NL2SQL\\_Handbook](https://github.com/HKUSTDial/NL2SQL_Handbook)

# RAG for Text2SQL



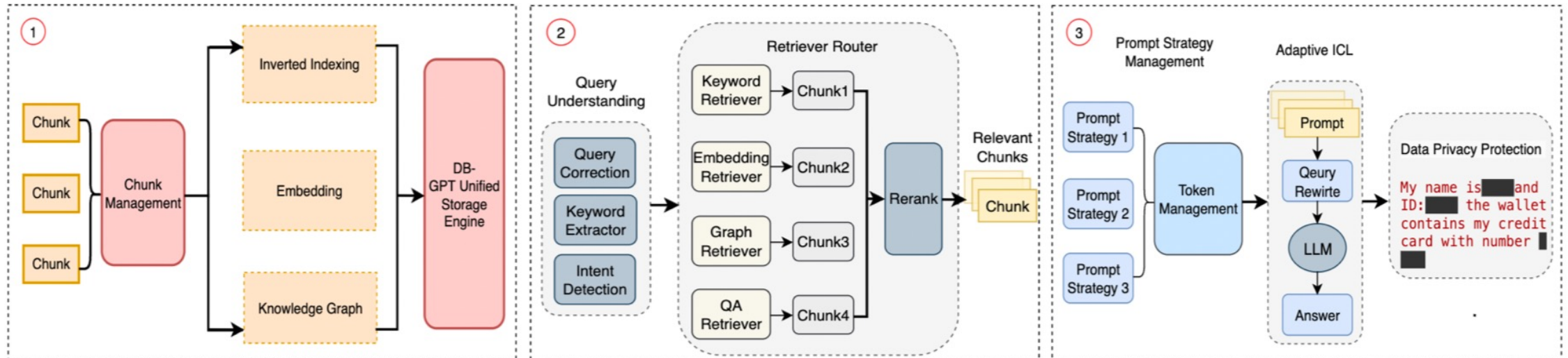
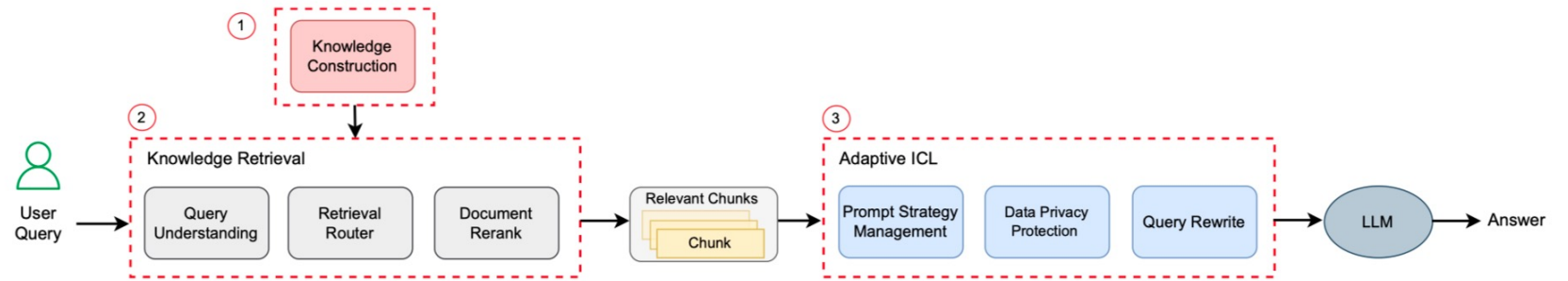
# RAG for Text2SQL



# RAG for Text2SQL

## Multi-source RAG for QA

- Knowledge construction
- Knowledge retrieval
- Learning to Embed and Search
- Adaptive In-Contextual Learning (ICL) and Generation by LLM
  - ICL is a technique used to improve LLMs' performance in handling contextual information by incorporating additional context during the training or inference phase



# RAG for Q&A Diagnosis

## Hard to accurately answer debugging questions with citations

- **1. RAG based Document Retrieval**
- **2. Feature Retrieval:** Identify Top-K feature names (e.g., queries, database, schema) from the retrieved documents
- **3. Context Aggregator:** Merge retrieved docs (taking Top-3 chunks if exceeding prompt limit) and features in json format → Give the root causes by LLM
- **4. Source Citation:** Append retrieved docs in passages (generated by LLM)

**Query:** For my Aurora MySQL database, I see a lot of sessions waiting for cpu and io/redo\_log\_flush. What's the issue, and how can I fix it?

**Panda:** Based on the task information and reference documents, the likely cause of the performance issue is an increase in wait events related to cpu, redo log flush. This is typically caused by excessive commits and write I/O operations<sup>[s1]</sup>.

To fix this issue, the recommended action is to [group your write operations together to reduce the number of commits](#)<sup>[s2]</sup>. Instead of inserting records row by row or in small batches, you should consider [larger batch sizes](#)<sup>[s3]</sup> or using bulk insert operations. By reducing the number of commits you can minimize the wait events related to write I/O operations and, mutex lock contention thereby improving the performance of your Aurora MySQL database.

**Sources:**

[s1]: Amazon Aurora User Guide for Aurora Pg.901, L.26.

[s2]: Amazon Aurora User Guide for Aurora Pg.902, L.22.

[s3]: Amazon Aurora User Guide for Aurora Pg.903, L.44.

**GPT-4:** If you are experiencing a high number of sessions waiting for 'redo log flush', and 'CPU' in your Aurora MySQL database, it could indicate potential performance issues:

Optimize your database design and queries to reduce the number and duration of transactions. Ensure your transactions are committing or rolling back promptly. Consider breaking down large transactions into smaller ones to reduce contention. Monitor and tune the innodb\_buffer\_pool\_size and innodb\_buffer\_pool\_instances configuration parameters to optimize memory usage.

Monitor your disk I/O performance to identify any bottlenecks. Consider optimizing your queries and transaction management to minimize the amount of data written to the redo log. Ensure that your database server has sufficient disk throughput and IOPS for your workload. Adjust the innodb\_log\_file\_size configuration parameter to an appropriate value based on your workload.

**DBE Feedback:** GPT-4 is incorrect. Aurora doesn't use innodb\_log\_file\_size. Also, "breaking down large transactions into smaller ones to reduce contention" is the opposite of what should be done here. Panda clearly outperforms GPT-4 in this scenario.

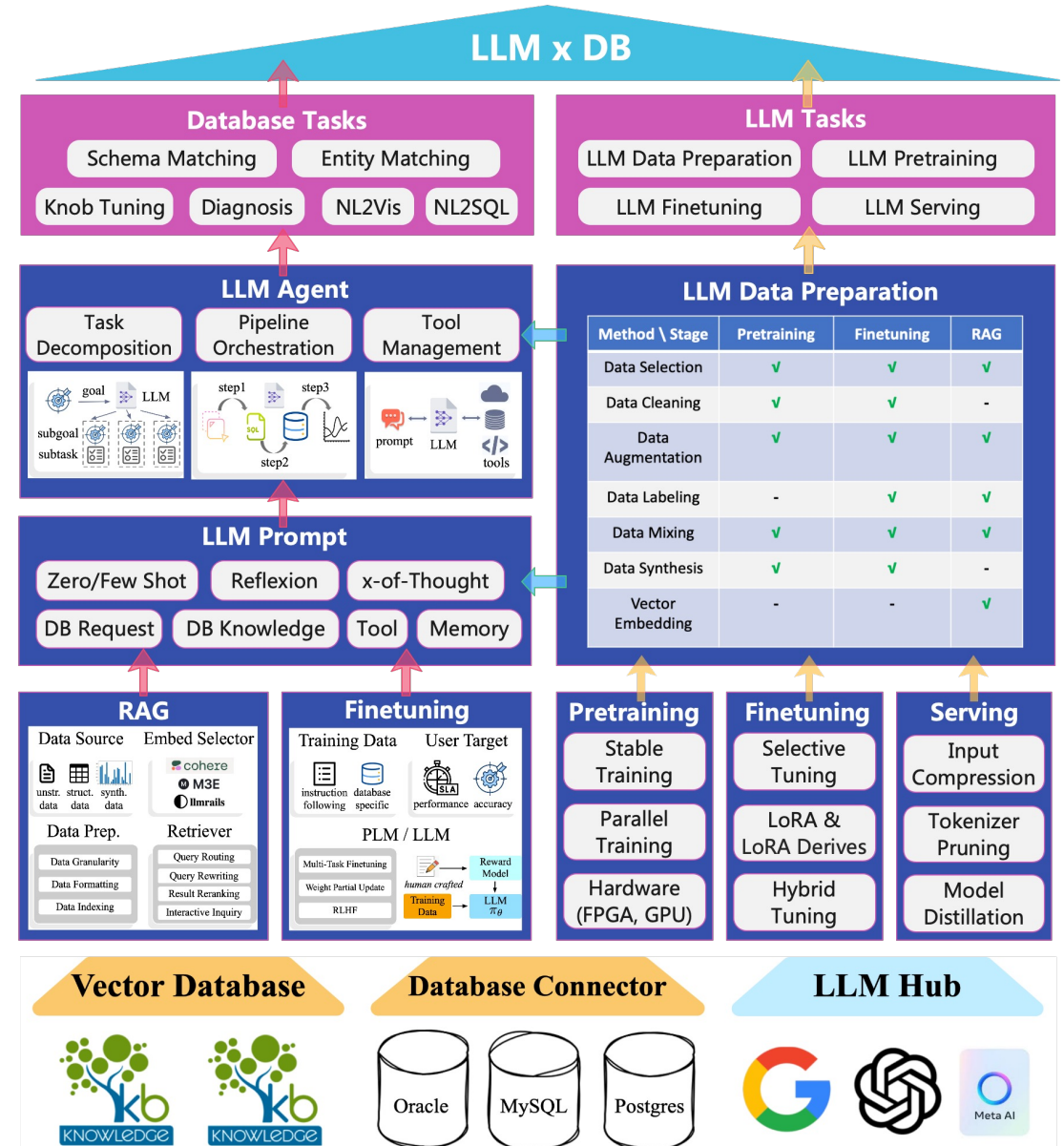
# Take-aways

---

- ❑ **RAG is very important to adapt to vertical domains and support data updates**
- ❑ **Not only improve the efficiency (vector database, vector index) but also improve the recall (multi retriever, segmentation, embedding models)**
- ❑ **It is important combine Prompt, RAG, and Agent to answer complex tasks**
- ❑ **It requires to build a RAG system**

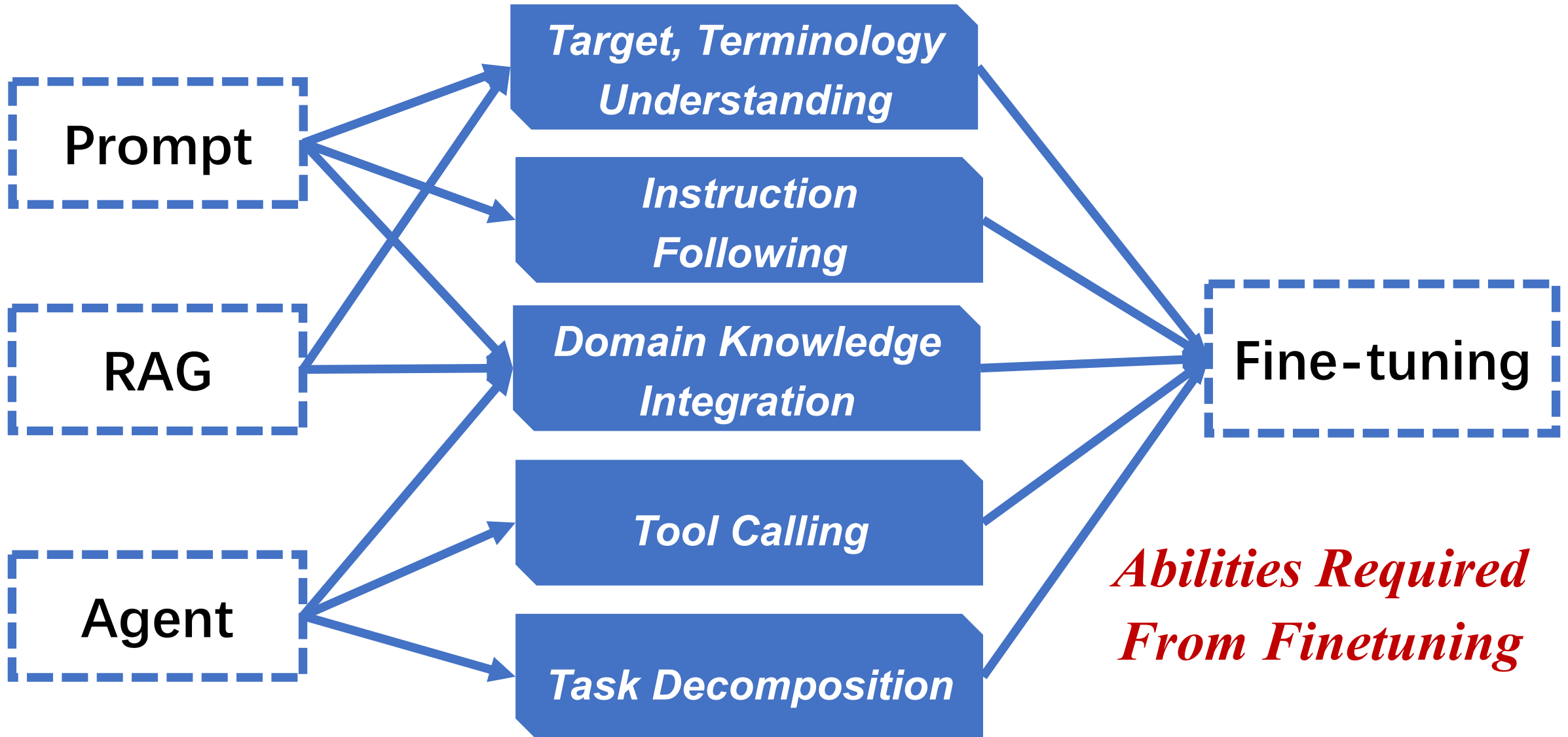
# Finetuning for Data Management

- ❑ Data Management tasks
- ❑ LLM Prompt for Data Management
  - Instruction Prompting
  - Few-Shot Prompting
- ❑ LLM Agent for Data Management
  - Agent Models + Memory
  - Reasoning / Planning Strategies
  - Tool Management & Learning
- ❑ RAG for Data Management
  - Semantic Segmentation
  - Result Retrieval
  - Result Reranking
- ❑ Finetuning for Data Management
  - Reparameterization / LLM Adapter
- ❑ Data Preparation for LLM
- ❑ Open Problems



# Abilities Gained by Fine-Tuning

---





# Example: Prompt Only → Prompt + Finetuning

## Query Rewrite Prompt

### Task Description

**Translate** the input query into an equivalent one that can be executed on Postgres with decreased latency.

### Input

```
select t1.* from t1 where t1.col1 > (  
  select max(t2.col2) from t2 where t2.col1 in (  
    select t1.col1 from t1 where t1.col1 = t2.col1));
```

## *Prompt Only*

### Output (LLMx-7B)

(Translate the input query in Chinese)

## ***Benefits of Finetuning***

- Domain knowledge memorization
- Redundant instructions (e.g., task, example, style) → A simple prompt
- Output Quality

## *Prompt + Finetuning*

### Output (LLMx-7B finetuned over rewrite dataset) (Analysis of how to rewrite the input query)

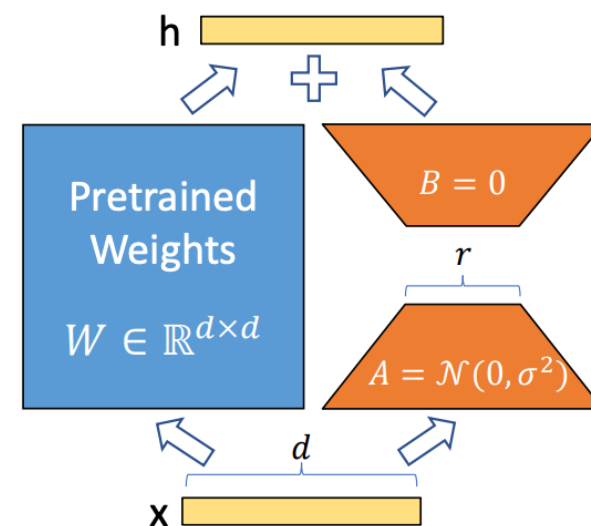
```
select t1.* from t1 inner join (  
  select max(t2.col2) max, t2.col1 from t2  
  group by t2.col1) as t2 on (  
  t1.col1 = t2.col1) where t1.col1 > max;
```

# Typical Techniques for LLM Finetuning

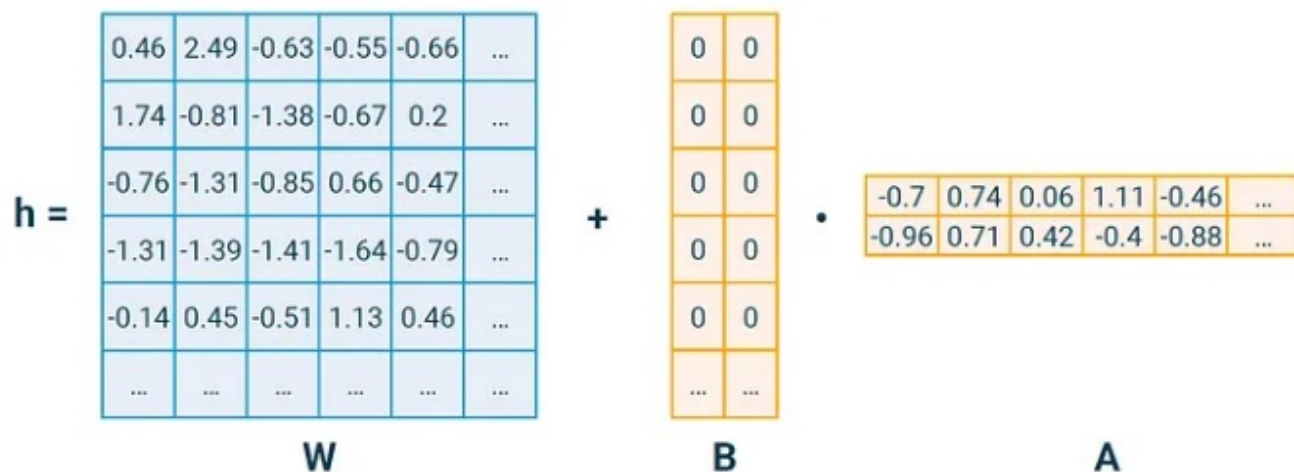
## □ Reparameterization (LoRA): Update layer parameters in a low-rank subspace

$$H_o = H_i W_0 + H_i \Delta W = H_i W_0 + H_i B A$$

- $H_i$  and  $H_o$  are the input and output of attention layer
- $W_0$  is the frozen model weights (MLP and Attention layer)
- $\Delta W$ : with much fewer parameters than  $W_0$
- $B, A$ : Low-dimensional matrices for approximating  $\Delta W$



- Given basic dimension is 1024, and LoRA rank  $r$  as 4:
  - #-parameters of weight  $W$  is  $1024 \times 1024 \approx 1M$
  - #-parameters of  $A, B$  are both  $r \times 1024 \approx 4K$
- In this way, we only train 0.8% of the parameters to update LLM



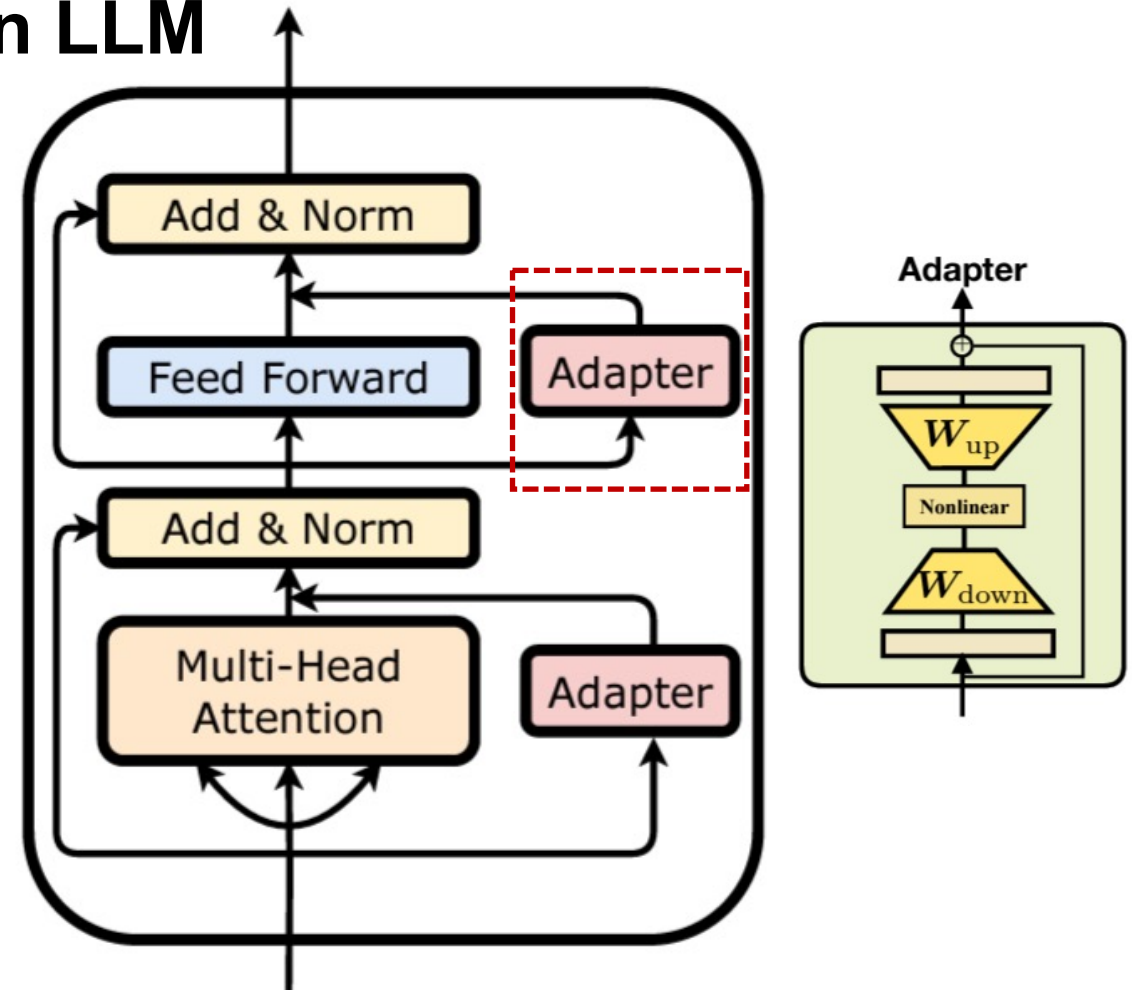
# Typical Techniques for LLM Finetuning

## □ LLM Adapter: Design and train additional learned modules for specific layers in an LLM

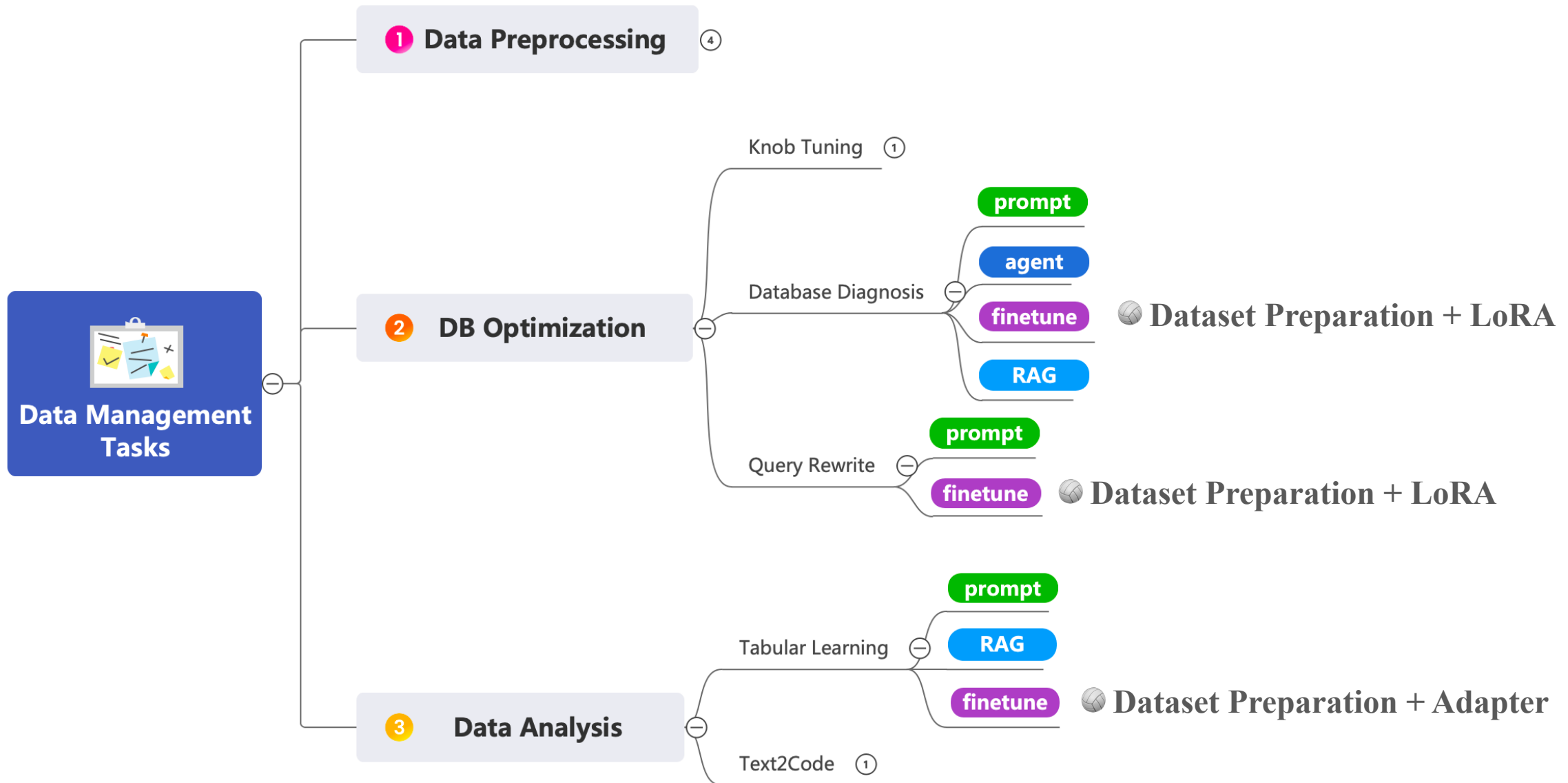
$$H_o \leftarrow H_o + f(H_i W_{down}) W_{up}$$

- $H_i$  and  $H_o$  are the input and output of attention layer
- Down-projected by  $W_{down}$  and then up-projected by  $W_{up}$

*Different from LoRA, you can **introduce new capabilities** in the adapter (e.g., image understanding) after finetuning*



# LLM Finetuning Techniques for Data Management



# LLM Finetuning for Query Rewrite

## ❑ Problems of Prompt-Only Rewriters

- Prompt engineering has been criticized for **limited knowledge capacity and unstable performance**

## ❑ Finetuning for Query Rewrite

### ❑ Training Data ~ (*origin query, rewritten query, steps*)

- **Origin Query:** (1) From various schemas; (2) Simple queries with atomic patterns; (3) Complex queries merged from simple ones using LLM like GPT-4
- **Rewritten Query:** (1) Heuristic Policy; (2) Volcano Policy; (3) Option Monte-Carlo Tree Search

```
SELECT ...  
WHERE ...  
AND (  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=1  
  )  
OR  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=2 )  
)
```

- **Targets, Terminologies**
- **Domain Knowledge**

# LLM Finetuning for Query Rewrite

## ❑ Problems of Prompt-Only Rewriters

- Prompt engineering has been criticized for **limited knowledge capacity and unstable performance**

## ❑ Finetuning for Query Rewrite

### ❑ Training Data ~ (origin query, rewritten query, steps)

- **Steps:** Prompt an LLM to explain the rewrite procedure in details:

*$p_{plan} = \dots$  The rewriter first translates input SQL into equivalent input plan. Second, it uses the given rewrite rule to transform the input plan into rewritten plan. Third, it translates the rewritten plan into equivalent rewritten SQL. ... You should not mention the input plan and rewritten plan in your explanation, as if the rewriter directly transforms the input SQL into the rewritten SQL. ...*

```
SELECT ...  
WHERE ...  
AND (  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=1  
  )  
OR  
  EXISTS (  
    SELECT a2 FROM t2  
    WHERE t2.b2=2 )  
)
```

- **Targets, Terminologies**
- **Logical Plan Structures**

# LLM Finetuning for Tool Learning in Diagnosis Agent



(Step 1)

**Thought:** Determine if the CPU usage was indeed abnormal  
**Action:** *whether\_is\_abnormal\_metric*  
**Arguments:** {metric: "cpu\_usage", time: "xxx - xxx"}

Result 1

Result 2



**Observation1:** The metric *cpu\_usage* is abnormal

**Observation2:** The service is unavailable

Matching



**Matched Knowledge:** *cpu\_relevant\_metrics*

**Diagnosis Failure**  
(Early Stop)

LLM Inference



(Step 2)

**Output 1 (Early Stop)**  
**Thought:** The anomaly is caused by high CPU usage ...  
**Action:** *Finish*

**Output 2**  
**Thought:** Investigate metrics like *node\_procs\_running* ...  
**Action:** *obtain\_metric\_values*  
**Arguments:** {metrics: ["node\_procs\_running", ...], time: ..}

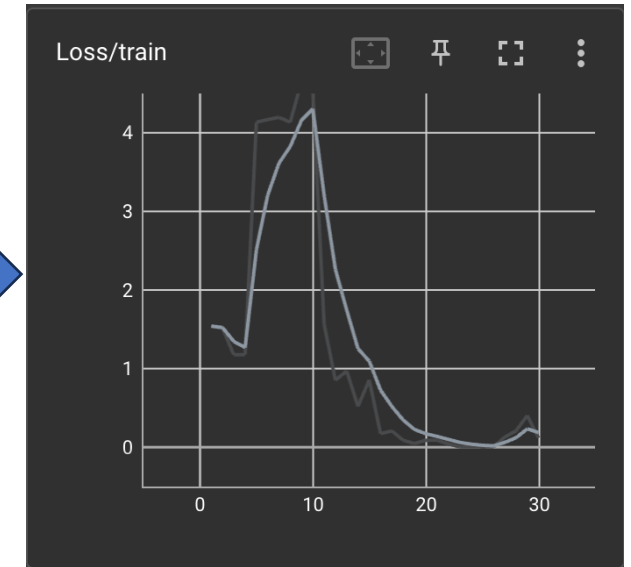
API calling by GPT-4

## □ Train LLM to select and call diagnosis APIs

- Prepare 1500 samples ~ 8:1:1
- Converge after training in 8 epoches
- Accuracy rate over the test set: 149/150

**"input":** "analyse the status of sockstat\_UDPLITE\_inuse in xxx for the next 2 hours",  
**"output":** "Action: *risk-analysis* Action Input: {*'metric': 'node\_sockstat\_UDPLITE\_in use', 'instance': '10.79.26.157:15766'}*}"

Data Format

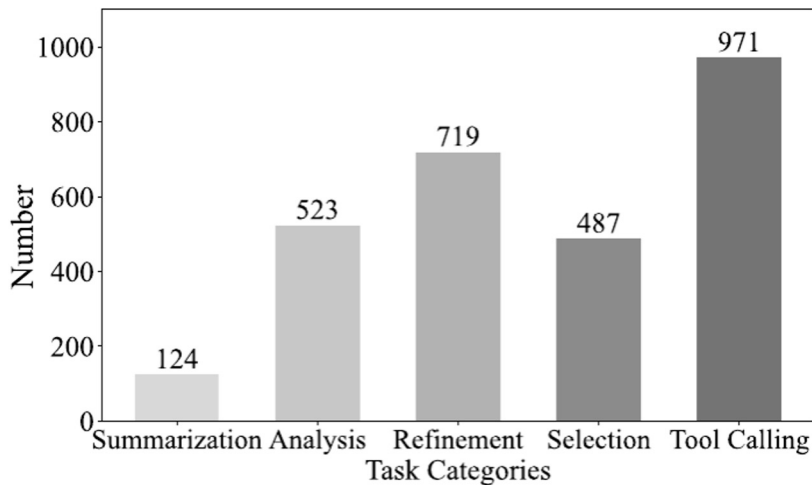


Llama-13B Finetuning

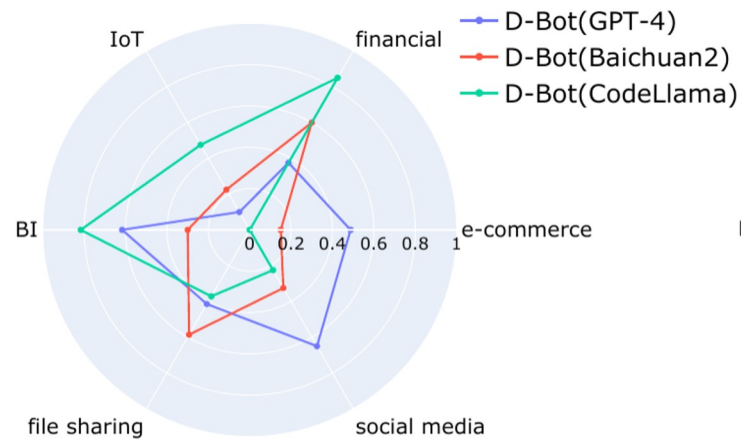
# LLM Finetuning for Diagnosis Agent

## □ Finetune a diagnosis model (LoRA) over five subtasks

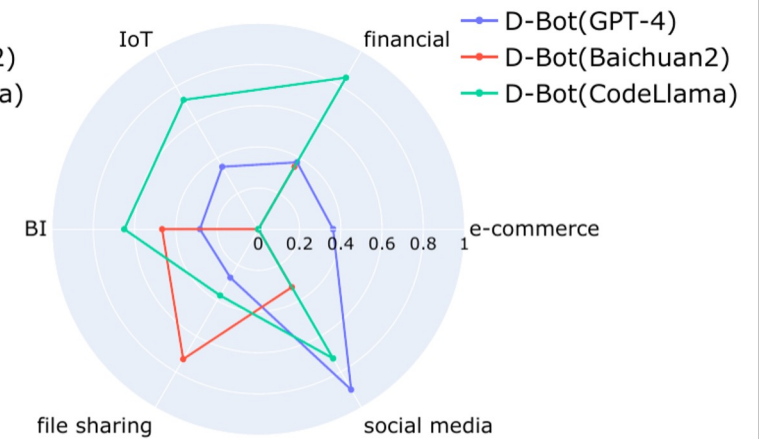
- Five diagnosis subtasks with 2819 finetuning samples
  - Analysis, Summarization(**target & terminologies**); Agent Selection (**task decompsition**); Tool Calling
- Good testing performance: Good at subtasks (e.g., tool calling, abnormal metric analysis) ; but highly rely on the quality of finetuning data



(a) Subtasks in Finetuning Samples



(c) Testing Results (Acc)



(d) Testing Results (HEval)



# LLM Finetuning for Table Learning











## □ Design and finetune a Table-Text Model for Tabular Data Tasks

- Model Structure for **New Modality Learning**

- **Visual Encoder** (pretrained ViT model) + **Vision-Language Connector** (two-layer MLP) + **Foundation LLM** (Vicuna)

- **Model Finetuning**

- Vision-Language Connector and LLM are jointly finetuned with instruction-following data of multi-modal tabular tasks

Outcome	No.	Date	Tournament	Surface	Opponent in the final
Runner-up	1.	2 September 2001	<a href="#">Mostar</a>	Clay	 <a href="#">Adriana Basaric</a>
Winner	1.	27 January 2002	<a href="#">Courmayeur</a>	Hard	 <a href="#">Stefanie Weis</a>
Winner	2.	17 February 2002	<a href="#">Bergamo</a>	Hard	 <a href="#">Rita Degli-Esposti</a>
Runner-up	2.	31 March 2002	Rome – <a href="#">Parioli</a>	Clay	 <a href="#">Dinara Safina</a>
Runner-up	3.	23 June 2002	<a href="#">Gorizia</a>	Clay	 <a href="#">Ainhoa Goni-Blanco</a>
Runner-up	4.	11 August 2002	<a href="#">Rimini</a>	Clay	 <a href="#">Laurence Andreotto</a>
Winner	3.	26 January 2003	<a href="#">Grenoble</a>	Hard	 <a href="#">Sophie Lefèvre</a>
Winner	4.	16 February 2003	<a href="#">Southampton</a>	Hard	 <a href="#">Magdalena Zđenovcová</a>
Winner	5.	23 February 2003	<a href="#">Redbridge</a>	Hard	 <a href="#">Olga Barabanschikova</a>
Winner	6.	23 March 2003	<a href="#">Castellon</a>	Clay	 <a href="#">Ludmila Cervanová</a>
Winner	7.	2 November 2003	<a href="#">Poitiers</a>	Hard	 <a href="#">Roberta Vinci</a>

Web page table

For the cell positioned in the 10th row and the 1st column of this table, provide its content. Output the target cell content as JSON in the form at {"row\_id":"m", "column\_id":"n", "cell\_value":"<Corresponding Cell Value>"}.  
↓

Textual Q&A

The target cell value in the 10th row and the 1st column is {"row\_id":"10", "column\_id":"1", "cell\_value":"Winner"}.

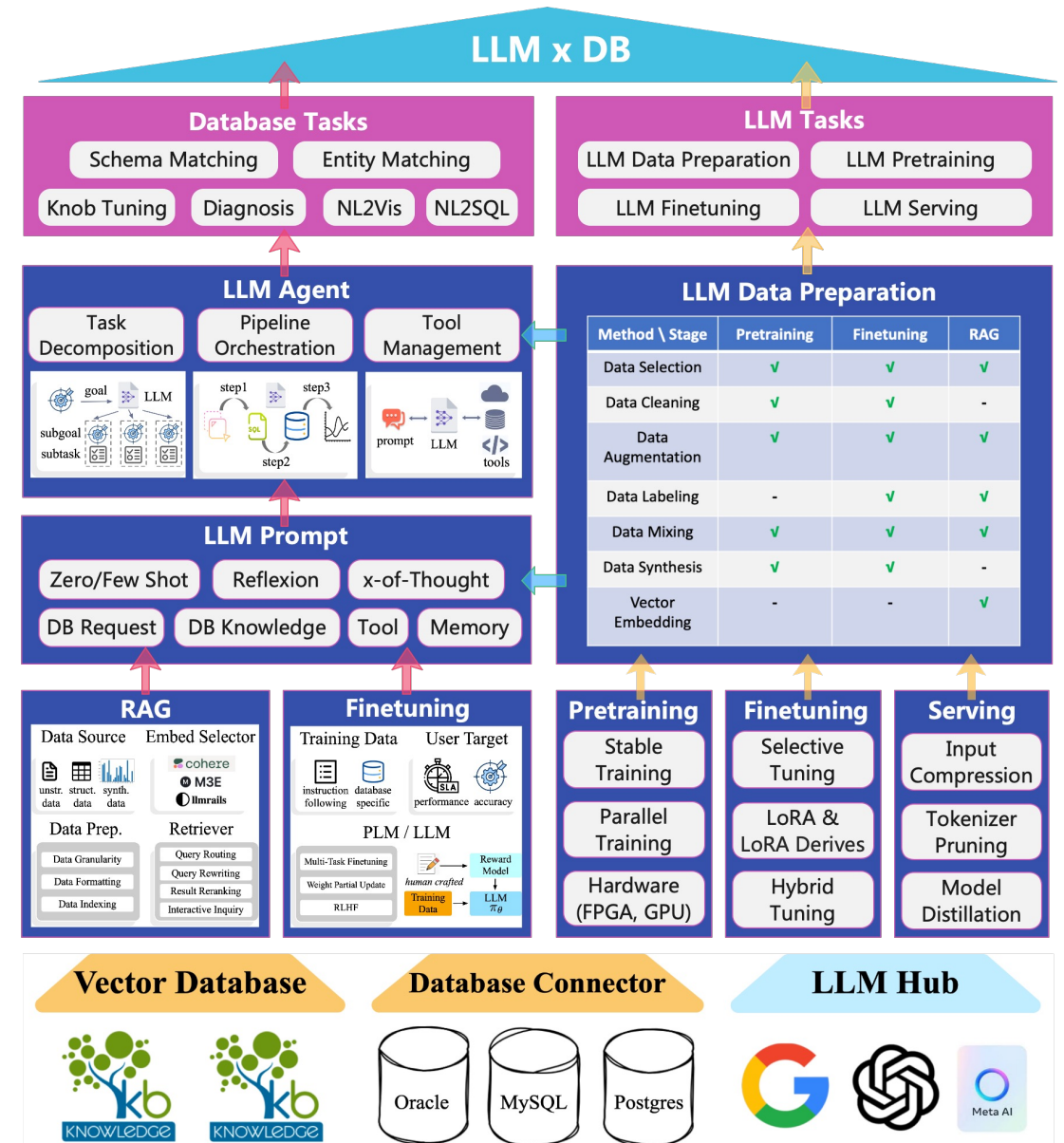
# Take-aways

---

- ❑ Finetuning is critical to domain tasks, such as (1) terminology and syntax learning, (2) instruction following, (3) tool calling, and (4) even supporting new modalities.
- ❑ The above three abilities can be achieved via **reparameterization**, but new modality learning requires carefully-designed **adapters**
- ❑ The updated parameters and finetuning techniques depend on factors, such as the task complexity and the pretraining data acquired by the LLM
- ❑ Training data is vital to the finetuning performance, which relies on data discovery, data processing, and expert insights
- ❑ LLM can be enhanced by combining both finetuning and RAG

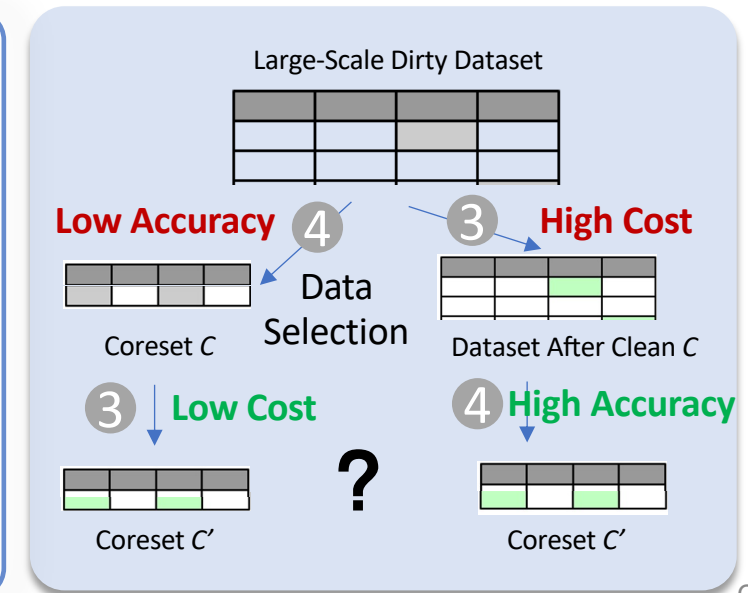
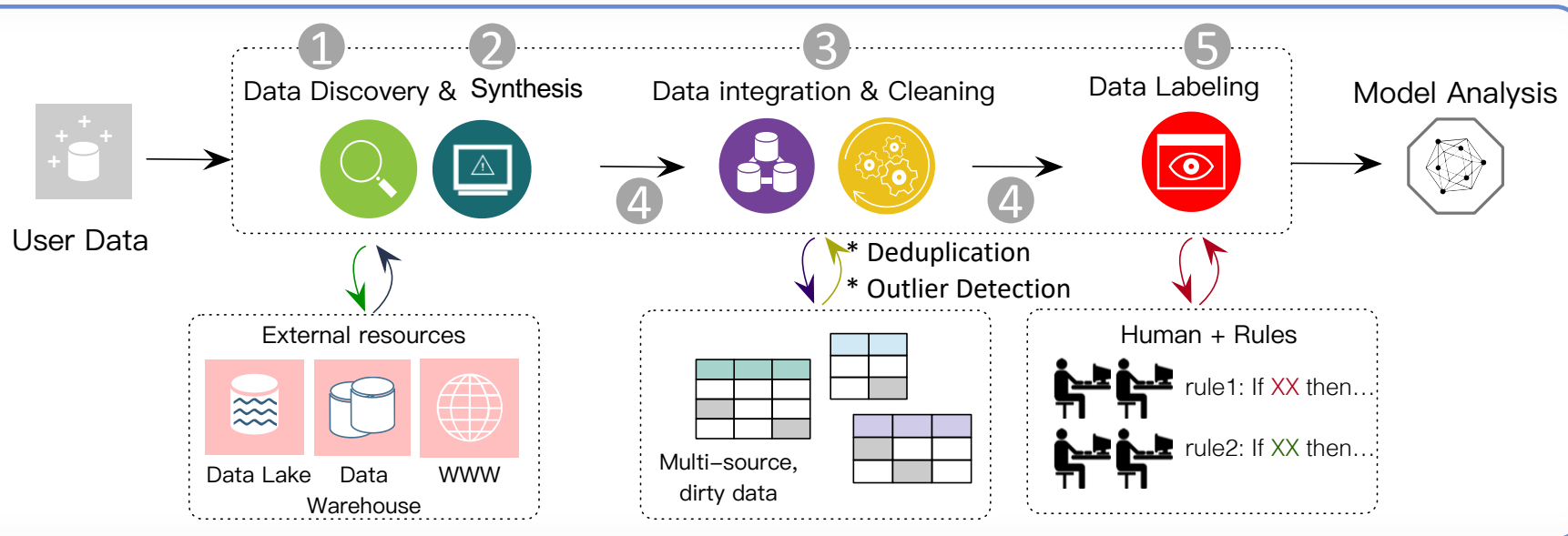
# Data Preparation for LLM

- ❑ Data Management tasks
- ❑ LLM Prompt for Data Management
  - Instruction Prompting
  - Few-Shot Prompting
- ❑ LLM Agent for Data Management
  - Agent Models + Memory
  - Reasoning / Planning Strategies
  - Tool Management & Learning
- ❑ RAG for Data Management
  - Semantic Segmentation
  - Result Retrieval
  - Result Reranking
- ❑ Finetuning for Data Management
  - Reparameterization / LLM Adapter
- ❑ Data Preparation for LLM
- ❑ Open Problems



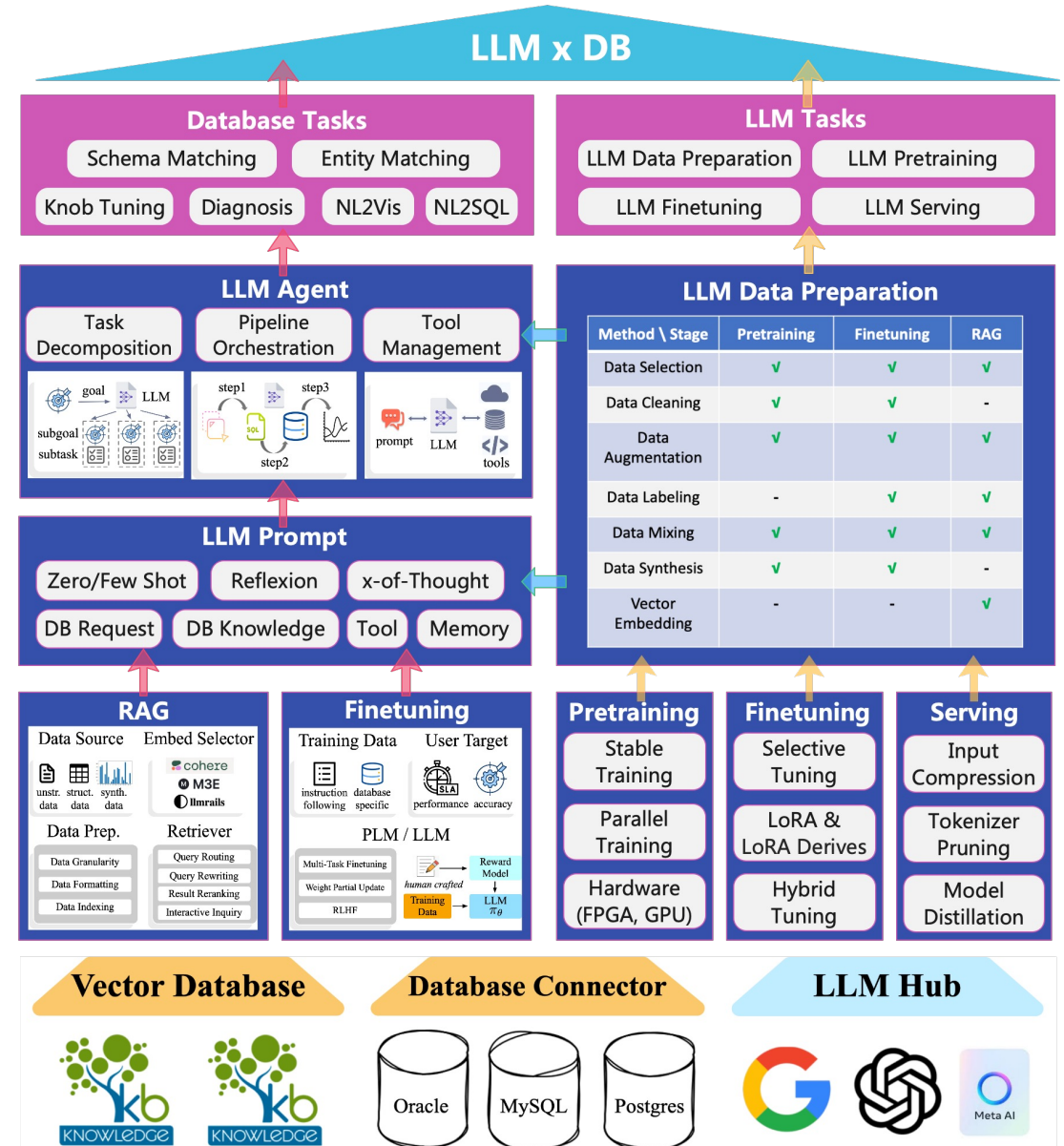
# Data Preparation for LLM

Stages \ LLM Training	Pretraining	Incr.- Pretraining	Finetuning	RLHF	RAG	Prompting
Data Selection	✓	✓	✓	✓	✓	✓
Data Cleaning	✓	✓	✓	-	-	-
Data Augmentation	✓	✓	✓	✓	✓	✓
Data Labeling	-	-	✓	✓	✓	✓
Data Mixing	✓	✓	✓	✓	✓	✓
Data Synthesis	✓	✓	✓	✓	-	-
Vector Embedding	-	-	-	-	✓	-



# Open Problems

- ❑ Data Management tasks
- ❑ LLM Prompt for Data Management
  - Instruction Prompting
  - Few-Shot Prompting
- ❑ LLM Agent for Data Management
  - Agent Models + Memory
  - Reasoning / Planning Strategies
  - Tool Management & Learning
- ❑ RAG for Data Management
  - Semantic Segmentation
  - Result Retrieval
  - Result Reranking
- ❑ Finetuning for Data Management
  - SFT Dataset Generation
- ❑ Data Preparation for LLM
- ❑ Open Problems

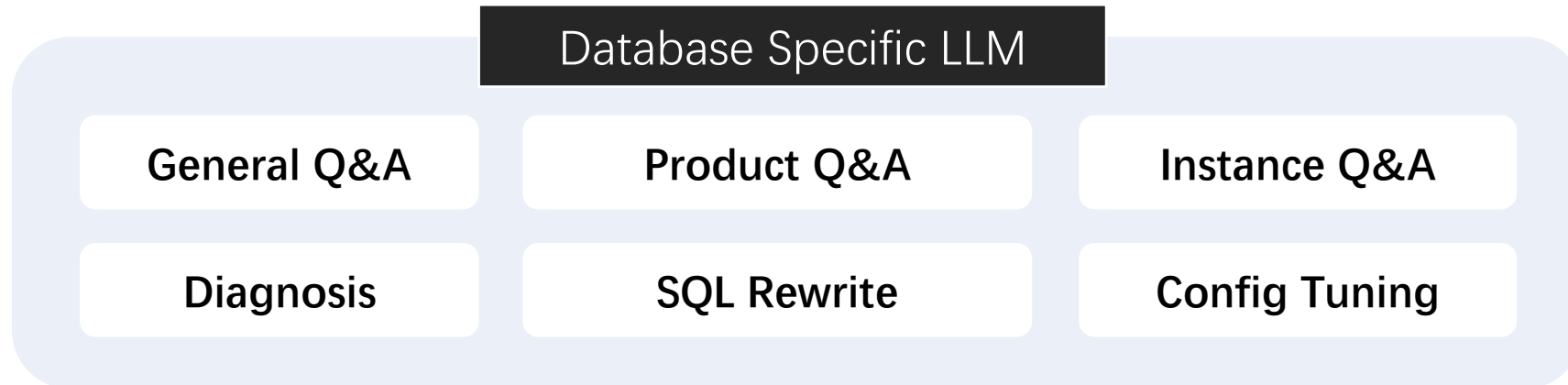


# Open Problem 1: Database Specific LLM

---

## □ Case-by-Case LLM Finetuning → Database-Specific LLM Construction

- **Pretrain:** Collect sufficient database-domain tokens (e.g., in millions) as pre-training corpora from sources like database textbook and query analysis
- **Finetune:** Instruction Understanding in SQL / Text → Basic Q&A (DB / Product / Instance)  
→ Task-Solving in DB Domains → Alignment to Database Experts
- **Evaluation:** Evaluate the accuracy and robustness of the database model with carefully-crafted validation dataset, measuring metrics, and end-to-end testbed.



# Open Problem 2: Tabular Data Learning

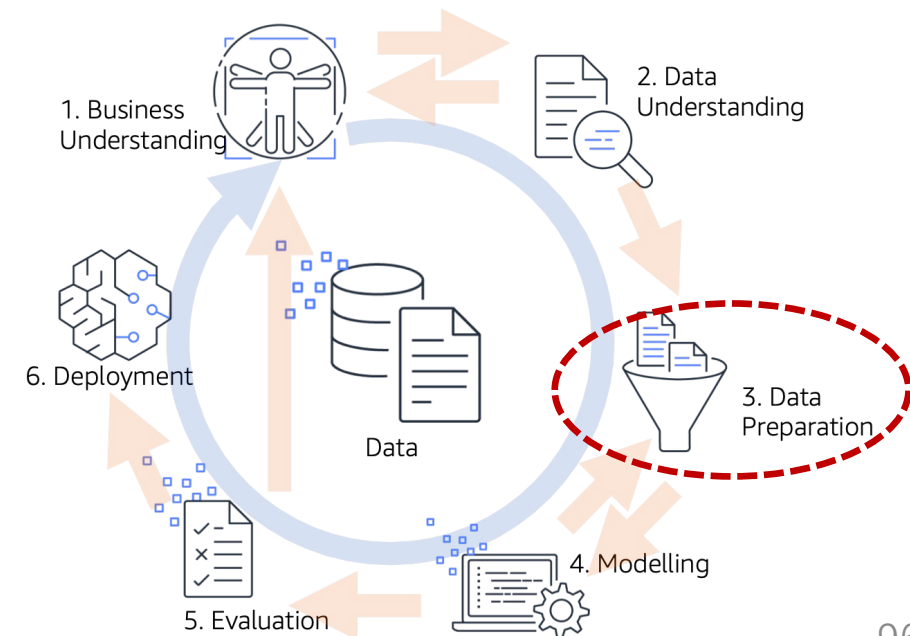
- Existing Adapters still cannot handle the following things:
  - Complex Table Structure Understanding
  - Alignment of table structure, table content, and in-context text
  - Excessively large tables processing

Childhood behavioural inhibition	Parental illness type(PIT)			All
	PD	Pure MD	Controls	
BI-	17.68 ± 2.04(n=29)	17.71 ± 1.28(n=11)	17.24 ± 1.86(n=21)	17.54 ± 1.85(n=61)
BI+	17.46 ± 1.76(n=16)	18.08 ± 2.39(n=5)	18.36(n=1)	17.64 ± 1.84(n=22)

# Open Problem 3: Data Preparation System for LLM

## □ An Effective System for Preparing LLM Data

- Data & Model Co-Design
- Big Data Curation for different LLM stages
- Data Synthesis: High LLM Perf. & Privacy-Preserving
- Data Flywheel: Self-Data-Reinforcing Loop
- Data Quality Evaluation

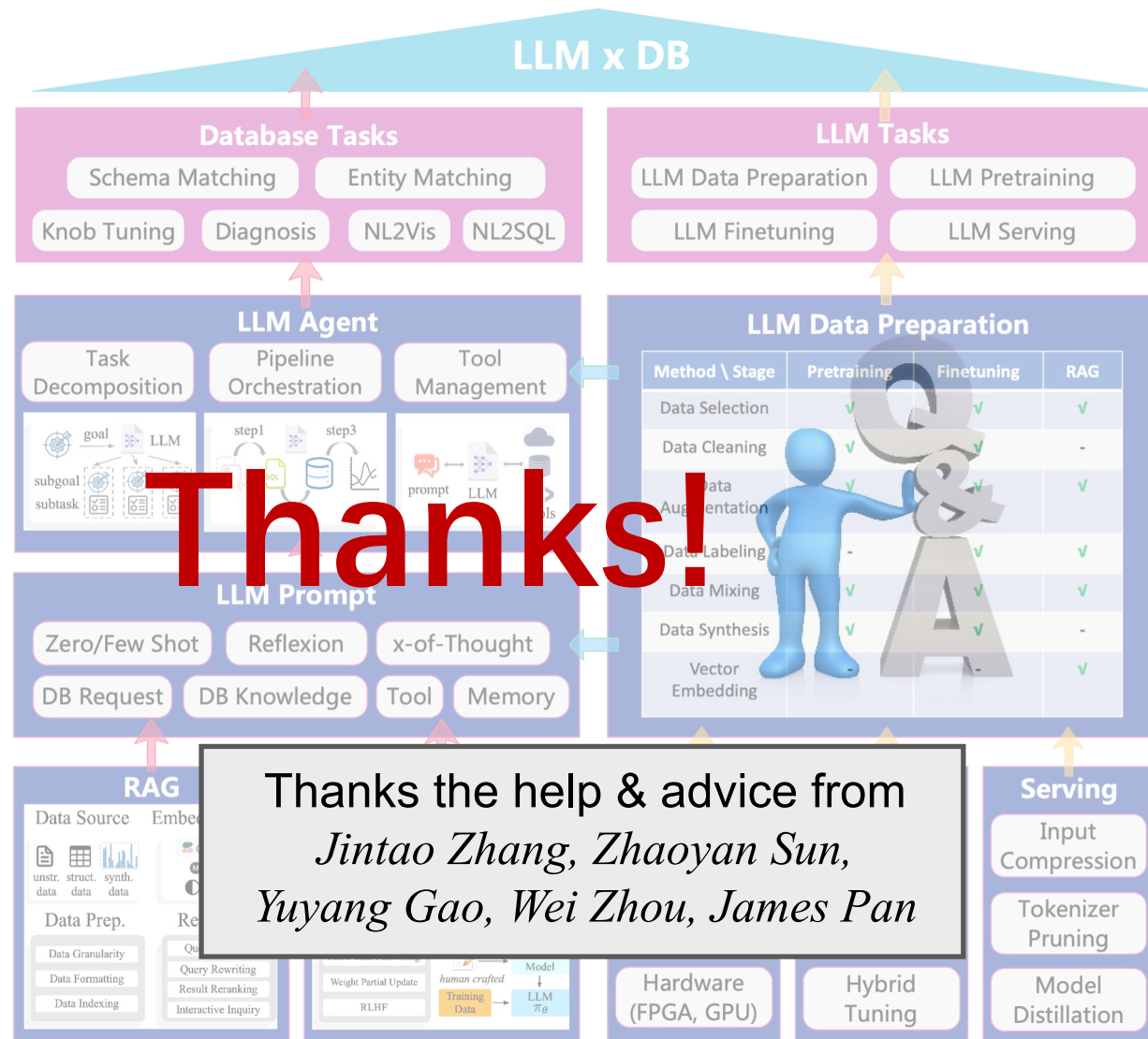




# Other Problem 4: Easy-to-Use LLM (Agent)

---

- ❑ **How to reduce the use costs of LLM / LLM-based Agent**
  - **LLM Distillation:** Large Powerful LLM → Cost-Efficient LLM
  - **Training or RAG?** (e.g., according characters like update frequency)
  - **Prompt Management:** Prompt Template Library + Automatic Template Gen
  - **Tool API:** Manual Generation → Automatic API Intergration (e.g., from programs)
  - **Agent-As-A-Service:** Careful Agent Design → One-click Agent Generation
  - ...



Thanks the help & advice from  
*Jintao Zhang, Zhaoyan Sun,*  
*Yuyang Gao, Wei Zhou, James Pan*

**Slides:** <https://dbgroup.cs.tsinghua.edu.cn/ligl/activities.html>

**Complete Paper List:** <https://github.com/code4DB/LLM4DB>