Latest updates: https://dl.acm.org/doi/10.1145/3722212.3725091

SHORT-PAPER

D-Bot: An LLM-Powered DBA Copilot

ZHAOYAN SUN, Tsinghua University, Beijing, China

XUANHE ZHOU, Shanghai Jiao Tong University, Shanghai, China

JIANMING WU, Tsinghua University, Beijing, China

WEI ZHOU, Huawei Technologies Co., Ltd., Shenzhen, Guangdong, China

GUOLIANG LI, Tsinghua University, Beijing, China

Open Access Support provided by:

Shanghai Jiao Tong University

Huawei Technologies Co., Ltd.

Tsinghua University

PDF Download
3722212.3725091.pdf
06 January 2026
Total Citations: 0
Total Downloads: 786

Published: 22 June 2025

Citation in BibTeX format

SIGMOD/PODS '25: International Conference on Management of Data
June 22 - 27, 2025
Berlin, Germany

Conference Sponsors:
SIGMOD

D-Bot: An LLM-Powered DBA Copilot

Zhaoyan Sun
Tsinghua University
Beijing, China
szy22@mails.tsinghua.edu.cn

Xuanhe Zhou
Shanghai Jiao Tong University
Beijing, China
zhouxh@cs.sjtu.edu.cn

Jianming Wu
Tsinghua University
Beijing, China
wjm@yeah.net

Wei Zhou
Shanghai Jiao Tong University
Shanghai, China
weizhoudb@gmail.com

Guoliang Li
Tsinghua University
Beijing, China
liguoliang@tsinghua.edu.cn

Abstract

Database administrators (DBAs) play an important role in maintaining the high performance and high availability of database systems. However, database diagnosis by DBAs often takes tedious efforts and time, which is insufficient for the large amount of database instances (e.g., millions of instances on the cloud). Besides, existing diagnosis tools built with rules and small-scale learned models lack the capability for flexible reasoning and report generation, and thus can only serve as "one small piece of a bigger puzzle". Recently, Large language models (LLMs) have exhibited superiority in natural language understanding, reasoning and generation, positioning them as a potential comprehensive copilot to DBAs to address these limitations. Thus, we introduce *D-Bot*, an LLM-powered DBA copilot that can automatically acquire pertinent knowledge from diagnostic documents, interact with users for self-refinement, and generate reasonable and well-founded diagnosis reports (i.e., specifying root causes, solutions, and references). We will show that *D-Bot* can effectively automate database diagnosis in real scenarios, even for complex anomalies.

CCS Concepts

• **Information systems → Autonomous database administration**.

Keywords

Database System, Database Diagnosis, Large Language Model

ACM Reference Format:
Zhaoyan Sun, Xuanhe Zhou, Jianming Wu, Wei Zhou, and Guoliang Li. 2025. D-Bot: An LLM-Powered DBA Copilot. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3722212.3725091

1 Introduction

Database diagnosis aims to ensure the high performance and high availability of database systems by detecting and resolving anomalies. In most organizations, database administrators (DBAs) are responsible for manually diagnosing daily anomalies, a process that

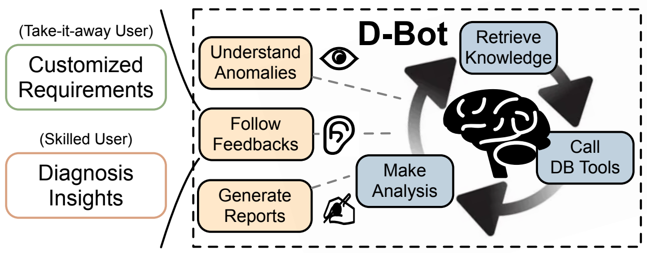


Figure 1: Example Database Diagnosis using *D-Bot*.

is often time-consuming and resource-intensive. Besides, existing diagnosis tools built with rules and small-scale learned models lack the capability for flexible reasoning and report generation, and thus can only serve as "one small piece of a bigger puzzle" [9].

Recently, large language models (LLMs) have exhibited superiority in natural language understanding, reasoning and generation, positioning them as a potential comprehensive copilot to DBAs [4, 8, 10]. As shown in Figure 1, LLM can automate the entire database diagnosis process through the following steps: (*i*) detect and understand anomalies described in natural language, (*ii*) make analysis with the help of relevant knowledge and database tools, (*iii*) follow user feedbacks (e.g., customized requirements, diagnosis insights) to refine its diagnosis, and (*iv*) generate reasonable and well-founded diagnosis reports. However, building an expert-level LLM-based database diagnosis system can still be challenging in the following aspects.

Challenges. First, since the diagnosis experiences accumulated by DBAs (e.g., documents, diagnosis cases) are highly complementary to LLMs, the first challenge is *how to enhance LLM diagnosis with private knowledge* **(C1)**. Second, given the elaborate analysis necessary for database diagnosis, it requires a deep consideration about *how to improve LLM diagnosis with an effective reasoning mechanism to tackle complex anomalies* **(C2)**. Third, since user feedback is often ambiguous, directly using it as LLM instruction may not accurately capture the underlying user intentions. Thus, the third challenge is *how to refine LLM diagnosis to align with user feedbacks* **(C3)**.

Our Methodology. To address the above challenges, we propose *D-Bot*, a DBA copilot powered by LLMs. First, to leverage the diagnostic knowledge of DBAs, we use LLM to extract knowledge chunks from diagnostic documents, where structurally or semantically pertinent chapters are processed together to form comprehensive chunks. Then, we generate embedding vectors for these

This work is licensed under a Creative Commons Attribution 4.0 International License.
SIGMOD-Companion '25, Berlin, Germany
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1564-8/2025/06
https://doi.org/10.1145/3722212.3725091

Table 1: Example refinements with user feedbacks. We develop test suites to align LLM response with user preferences, and store refinements as "if-then" statements for future diagnosis.

User Feedback	LLM Response Test Suite	LLM Response Refinement ("if-then")
Offer distinct, detailed reviews instead of repeating others.	return ask_llm("Is the response informative and detailed?")	If query performance is poor, then a specific investigation into the effectiveness of creating new indexes should be conducted.
Offer more in-depth analysis.	return ask_llm("Does it provide in-depth analysis of causes?")	If analyzing identified anomalies, then also focus on how their impacts can affect the system on a larger scale.
Focus more on root causes like CPU contention.	return not ask_llm("Does it mention any solutions?")	If it includes multiple root causes then provide all root causes (e.g., both CPU contention and high IO operations).

chunks, store them in vector databases, and retrieve relevant chunks during diagnosis to guide LLM analysis (addressing **C1**). Second, to cope with complex reasoning involved in diagnosis, we design a collaborative diagnosis framework incorporating multiple LLM experts, where they analyze anomalies with multi-step tree search reasoning, and cooperate together through cross-review to obtain a thorough diagnosis result (addressing **C2**). Third, to better align the diagnosis with user preferences, we use LLM to refine the result based on user feedback (see Table 1). Specifically, we develop test suites for each feedback intention, and guide LLM to iteratively refine the diagnosis by submitting it to these test suites. Then, we automatically analyze both the original and refined LLM responses to identify refinement patterns (e.g., "if-then" statements), which are then stored for reuse when similar diagnoses are encountered (addressing **C3**). In our demonstration, database users can view the diagnosis report generated by *D-Bot*, and back-trace the detailed diagnosis process (e.g., LLM reasoning steps) to ensure its reliability.

This demo is an extension of our research paper [9], further with a self-evolving mechanism learning from user interactions and feedbacks. *D-Bot* distinguishes itself from other database diagnosis tools mainly in three aspects: (*i*) *D-Bot* is capable of retrieving relevant knowledge from raw diagnostic documents, similar to the expertise of human DBAs, when analyzing anomalies; (*ii*) *D-Bot* can systematically identify the root cause and solutions step-by-step, and automatically generate a reasonable and well-founded diagnosis report; (*iii*) *D-Bot* enables users to participate in diagnosis decisions and aligns with user preferences by incorporating insights derived from user feedbacks. We will demonstrate that *D-Bot* can generate diagnosis reports comparable to human DBAs for real-world anomalies with an easy-to-use interaction interface. A demonstration video is available at https://vimeo.com/1070633767/440814fe24?ts=0&share=copy.

2 Overview

The architecture of *D-Bot* is presented in Figure 2. Firstly, users can prepare *D-Bot* by initializing knowledge stores and connecting to their database. Secondly, *D-Bot* automatically detects and diagnoses database anomalies using LLMs. Thirdly, users can provide feedbacks to refine the intermediate diagnoses of *D-Bot*. Fourthly, *D-Bot* generates a reasonable diagnosis report with verifiable references.

Knowledge Retrieval. To mitigate the hallucination problem commonly encountered by LLMs, we can retrieve pertinent knowledge chunks as LLM context and require LLM to self-calibrate using them [5]. Specifically, we first filter out the knowledge chunks by their tags (e.g., anomaly metrics). Second, to evaluate semantic relevance of knowledge chunks to the anomaly, we embed them

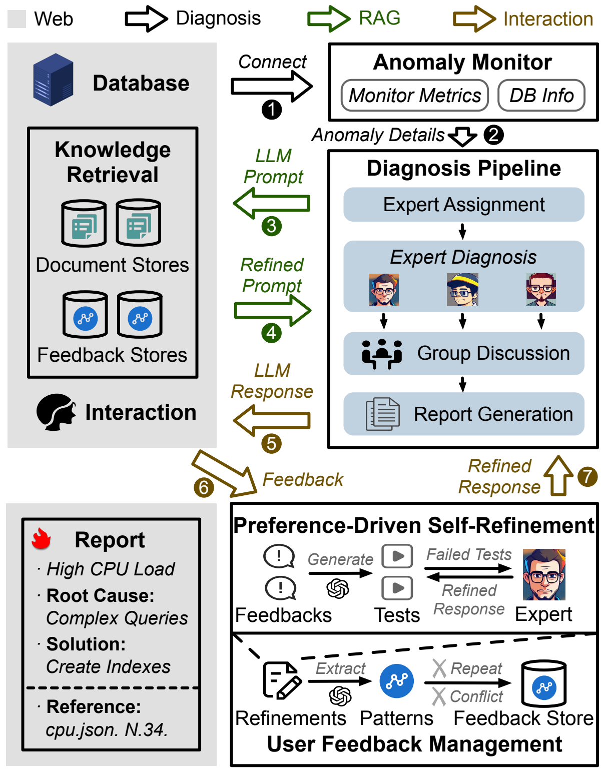
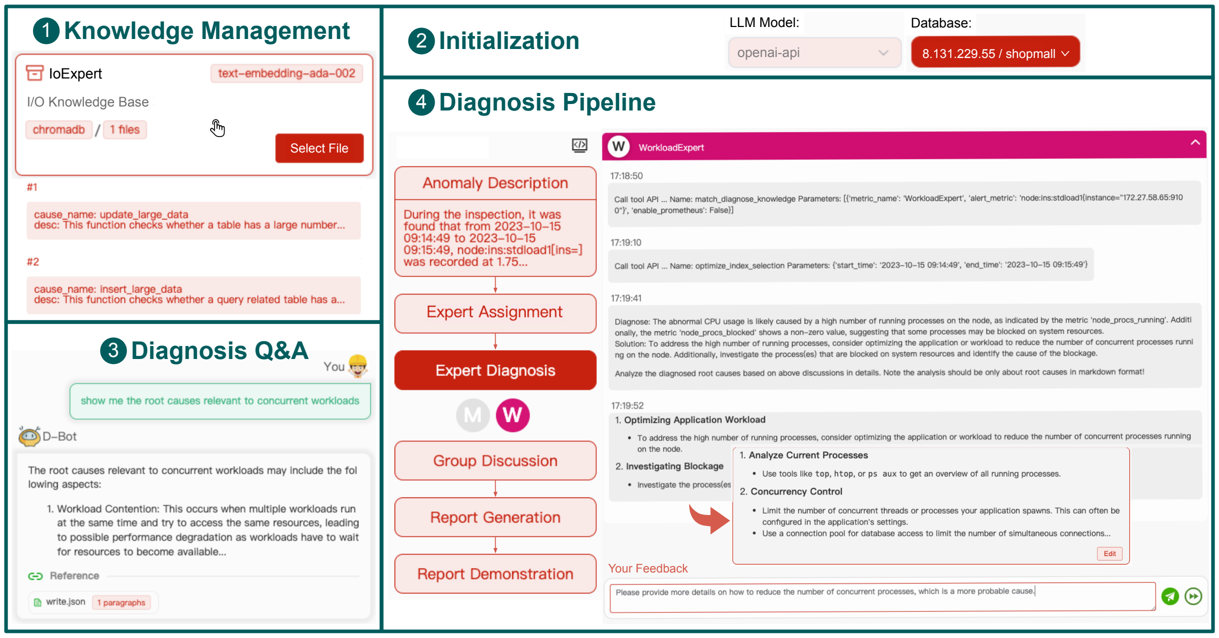


Figure 2: The Architecture of *D-Bot*.

into vectors with pre-trained text embedding models (e.g., text-embedding-3-large [2]), and retrieve the chunks with top vector similarity (e.g., using L^2-distance of vectors).

To accelerate knowledge retrieval, we can offline prepare the embedding vectors of knowledge chunks, and store them in vector databases (e.g., Chroma [1]). We have two types of knowledge stores. (*i*) *Document Store.* It extracts knowledge chunks (e.g., pairs of anomalies and solutions) from diagnostic documents. Specifically, we first divide the document based on chapter structure, use LLM to extract knowledge chunks from chapters which are structurally or semantically related, and then organize the chunks according to anomaly topics (e.g., CPU, workload). Users can also expand our prepared store with their private documents. (*ii*) *Feedback Store.* It accumulates experiences learned from user interactions and feedbacks (e.g., refinement patterns of LLM response).

Anomaly Monitor. *D-Bot* can automatically detect anomalies by monitoring critical metrics (e.g., available memory less than 10%). If any anomaly is detected, *D-Bot* collects relevant details from database (e.g., logs of slow SQL queries), and supplies them to LLMs for further analysis and diagnosis.



#1

cause_name: update_large_data
desc: This function checks whether a table has a large number...

#2

cause_name: insert_large_data
desc: This function checks whether a query related table has a...

show me the root causes relevant to concurrent workloads

The root causes relevant to concurrent workloads may include the following aspects:

1. Workload Contention: This occurs when multiple workloads run at the same time and try to access the same resources, leading to possible performance degradation as workloads have to wait for resources to become available...

During the inspection, it was found that from 2023–10–15 09:14:49 to 2023–10–15 09:15:49, node:ins:stdload1[ins=] was recorded at 1.75...

17:18:50

Call tool API ... Name: match_diagnose_knowledge Parameters: [{'metric_name': 'WorkloadExpert', 'alert_metric': 'node:ins:stdload1{instance="172.27.58.65:9100"}', 'enable_prometheus': False}]

17:19:10

Call tool API ... Name: optimize_index_selection Parameters: {'start_time': '2023–10–15 09:14:49', 'end_time': '2023–10–15 09:15:49'}

17:19:41

Diagnose: The abnormal CPU usage is likely caused by a high number of running processes on the node, as indicated by the metric 'node_procs_running'. Additionally, the metric 'node_procs_blocked' shows a non–zero value, suggesting that some processes may be blocked on system resources.
Solution: To address the high number of running processes, consider optimizing the application or workload to reduce the number of concurrent processes running on the node. Additionally, investigate the process(es) that are blocked on system resources and identify the cause of the blockage.

Analyze the diagnosed root causes based on above discussions in details. Note the analysis should be only about root causes in markdown format!

17:19:52

1. Optimizing Application Workload
   - To address the high number of running processes, consider optimizing the application or workload to reduce the number of concurrent processes running on the node.
2. Investigating Blockage
   - Investigate the process(es

1. Analyze Current Processes
   - Use tools like `top`, `htop`, or `ps aux` to get an overview of all running processes.
2. Concurrency Control
   - Limit the number of concurrent threads or processes your application spawns. This can often be configured in the application's settings.
   - Use a connection pool for database access to limit the number of simultaneous connections...

Please provide more details on how to reduce the number of concurrent processes, which is a more probable cause.

Figure 3: A Screenshot of *D-Bot* (https://github.com/TsinghuaDatabaseGroup/DB-GPT).

Diagnosis Pipeline. To enhance LLM performance in database diagnosis, we propose a collaborative diagnosis framework incorporating multiple LLM experts, where each expert is equipped with distinct knowledge and tools. It consists of four steps. (*i*) *Expert Assignment.* Given an anomaly, we first use LLM to analyze the anomaly description, and assign relevant LLM experts (e.g., CPU expert) for diagnosis. (*ii*) *Expert Diagnosis.* Then, the assigned experts simultaneously fulfill diagnosis incorporating multi-step LLM reasoning (e.g., calling tools, retrieving relevant knowledge). They also adopt a tree search algorithm, which can explore many reasoning paths, and select the optimal result through LLM reflection. (*iii*) *Group Discussion.* After individual expert diagnosis, we also instruct LLM experts to exchange intermediate diagnosis results, and refine their results based on cross-review. (*iv*) *Report Generation.* Finally, we utilize LLM to generate a reasonable report summarizing diagnosis results of the experts. Note that we also enhance its reliability by providing references like involved knowledge chunks and corresponding user feedbacks.

Preference-Driven Self-Refinement. To better align the diagnosis with user preferences, *D-Bot* refines its diagnosis results based on user feedbacks. Specifically, we first use LLM to extract a concise list of statements reflecting the intentions behind feedback. We then use LLM to synthesize executable test suites for each statement [6]. As shown in Table 1, the test suite decomposes user intentions into more manageable questions (e.g., *"Is the LLM response informative and detailed?"*), which can be reliably answered by LLMs.

Next, we instruct LLM to refine its previous response by appending user feedback, and submit the refined response to the test suites. If it fails any tests, we instruct LLM for further refinement with the test error logs. We iteratively repeat the process until it passes all the tests or reaches a threshold of times (e.g., 3).

User Feedback Management. After refining the LLM response according to user feedback, *D-Bot* extracts the refinement patterns and stores them for automatic reuse, improving future LLM diagnoses when similar cases arise. For instance, as shown in Table 1, the diagnosis refinements are summarized as "if-then" statements, which can effectively instruct LLM to align with user preferences.

Specifically, we can compare the refined response with the original one, and use LLM to extract the refinement patterns [7]. These patterns, along with the context of refined LLM response, are then stored in the feedback store. During future diagnosis, when similar contexts are input to LLM (e.g., evaluated by embedding vector similarity), relevant refinement patterns can be incorporated as part of LLM input, ensuring that LLM response aligns with user preferences. Note that before adding new patterns into the feedback store, we also check whether any existing patterns are redundant or contradictory and remove them if detected.

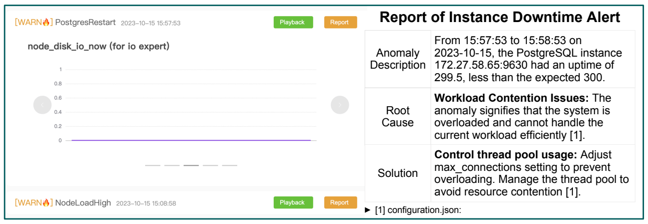
Web Service. We provide a website interface to help users easily deploy *D-Bot* and observe the diagnostic procedure. There are mainly four functions: (*i*) manage knowledge stores (e.g., uploading private documents); (*ii*) initialize *D-Bot* by setting up database and LLM models; (*iii*) answer questions on database diagnosis using relevant stored knowledge; (*iv*) demonstrate the diagnosis report automatically generated by *D-Bot* and detailed diagnosis process, where users can also actively interact with *D-Bot* to refine the intermediate diagnoses.

3 Demonstration

In this section, we describe the website interface of *D-Bot*, where users can easily deploy *D-Bot* and observe the anomaly diagnosis process. The online website will be continuously updated.

3.1 End-to-End Experience

Figure 3 showcases the front-end of *D-Bot*. Users can perform anomaly diagnosis and observe the diagnosis report generated by *D-Bot* in the following four steps.

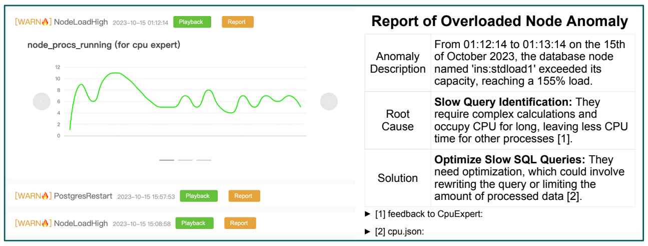

[WARN🔥] PostgresRestart 2023–10–15 15:57:53
Playback
Report
node_disk_io_now (for io expert)
1
0.8
0.6
0.4
0.2
0
[WARN🔥] NodeLoadHigh 2023–10–15 15:08:58
Playback
Report

Report of Instance Downtime Alert

Anomaly Description	From 15:57:53 to 15:58:53 on 2023-10-15, the PostgreSQL instance 172.27.58.65:9630 had an uptime of 299.5, less than the expected 300.
Root Cause	**Workload Contention Issues:** The anomaly signifies that the system is overloaded and cannot handle the current workload efficiently [1].
Solution	**Control thread pool usage:** Adjust max_connections setting to prevent overloading. Manage the thread pool to avoid resource contention [1].

▶ [1] configuration.json:

(a) Scenario 1 - A Real Single-Cause Anomaly.


[WARN🔥] NodeLoadHigh 2023–10–15 01:12:14
Playback
Report
node_procs_running (for cpu expert)
12
10
8
6
4
2
0
[WARN🔥] PostgresRestart 2023–10–15 15:57:53
Playback
Report
[WARN🔥] NodeLoadHigh 2023–10–15 15:08:58
Playback
Report

Report of Overloaded Node Anomaly

Anomaly Description	From 01:12:14 to 01:13:14 on the 15th of October 2023, the database node named 'ins:stdload1' exceeded its capacity, reaching a 155% load.
Root Cause	**Slow Query Identification:** They require complex calculations and occupy CPU for long, leaving less CPU time for other processes [1].
Solution	**Optimize Slow SQL Queries:** They need optimization, which could involve rewriting the query or limiting the amount of processed data [2].

▶ [1] feedback to CpuExpert:

▶ [2] cpu.json:

(b) Scenario 2 - A Real Multi-Cause Anomaly.

Figure 4: Example Diagnosis Reports Generated by *D-Bot* (more detailed reports in http://dbgpt.dbmind.cn/).

1) Knowledge Management. Before diagnosis, users need to prepare the knowledge stores for pertinent knowledge retrieval. We provide two functions (Fig. 3-①). First, users can click the "Select File" button to upload their documents, where knowledge chunks are automatically extracted and added to the knowledge store. Second, we also enable users to check the stored knowledge chunks by demonstrating brief descriptions (e.g., root causes of anomalies).

2) Initialization. To initialize *D-Bot*, users can connect to their database, and set up the LLM models (GPT-4 [3] by default) employed by *D-Bot* (Fig. 3-②).

3) Diagnosis Q&A. If users encounter problems with database diagnosis, they can directly ask their questions and *D-Bot* will provide detailed answers with references (e.g., source documents) using LLMs (Fig. 3-③).

4) Diagnosis Pipeline. During *D-Bot* diagnosis, database users can check out the real-time diagnosis procedure and interactively refine it. *D-Bot* demonstrates the procedure in two aspects (Fig. 3-④). First, on the left side, we display the diagnosis workflow (e.g., brief anomaly description and diagnosis stages), and highlight the ongoing stage (e.g., *Expert Diagnosis*). Second, on the right side, we demonstrate details of LLM reasoning steps (e.g., calling index optimizers) and intermediate diagnosis results (e.g., root causes identified by WorkloadExpert). Next, *D-Bot* also provides three interactive functions. (i) If users are satisfied with the diagnosis result, they can click the double-arrow button to bypass the interaction mechanism. (ii) If users require to refine the diagnosis result, they can enter their feedback into the box below, prompting *D-Bot* to automatically refine the result. (iii) If users are still unsatisfied with the refined result, they can click the "Edit" button to manually adjust it.

3.2 Scenario 1 - Real Single-Cause Anomalies.

We evaluate diagnosis performance of *D-Bot* on 51 real single-cause anomalies. Powered by superior LLM like GPT-4 [3], *D-Bot* can achieve a relatively high F2-score of 67.6% with a fully automatic pipeline. Besides, we find it possible to further correct diagnosis flaws through user interactions, even for take-it-away users. For instance, as shown in Figure 4 (a), LLM can repeat reviews from other experts during *Group Discussion*, where users can pose feedback like *"offer distinct, detailed reviews"* to ensure the effectiveness of discussion. *D-Bot* also stores the refinements of LLM response during this interaction (see the first row in Table 1), enabling ConfigurationExpert to offer more insightful advice dealing with anomalies like poor query performance.

For skilled users, a more advanced optimization is to reduce resource consumption by manually assigning fewer LLM experts. That is because LLM tends to incorporate all possible experts during diagnosis (e.g., WorkloadExpert, QueryExpert, CpuExpert), even if WorkloadExpert alone is enough to identify the root cause for simple cases (e.g., highly deletes).

3.3 Scenario 2 - Real Multi-Cause Anomalies.

We also evaluate the diagnosis performance on 9 real multi-cause anomalies. *D-Bot* powered by GPT-4 achieves an F2-score of 59.5% without user intervention, which is even higher than human DBAs. We then demonstrate that incorporating user feedback can further improve this performance. First, since LLM sometimes provides ambiguous responses without fully leveraging the retrieved knowledge and tools, take-it-away users can correct this by providing feedback like *"offer more in-depth analysis"* to trigger a self-refinement of LLM (see the second row in Table 1).

Besides, skilled users can further refine the diagnosis process by presenting insights based on their experiences. For instance, as shown in Figure 4 (b), if they identify "CPU intention" as a more probable root cause, they can instruct LLM to brainstorm more targeted analysis and solutions (see the third row in Table 1).

Acknowledgments

This paper was supported by NSF of China (62525202,62232009), National Key R&D Program of China (2023YFB4503600), Shenzhen Project (CJGJZD20230724093403007), Zhongguancun Lab, Huawei, and Beijing National Research Center for Information Science and Technology (BNRist). The corresponding author is Guoliang Li.

References

[1] 2025. *Chroma*. https://www.trychroma.com.
[2] 2025. *Embeddings - OpenAI API*. Retrieved January 26, 2025 from https://platform.openai.com/docs/models#embeddings
[3] 2025. *GPT-4*. https://openai.com/gpt-4.
[4] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv:2303.08774* (2023).
[5] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL]
[6] Shreya Shankar, Haotian Li, Parth Asawa, Madelon Hulsebos, Yiming Lin, JD Zamfirescu-Pereira, Harrison Chase, Will Fu-Hinthorn, Aditya G Parameswaran, and Eugene Wu. 2024. SPADE: Synthesizing Assertions for Large Language Model Pipelines. *arXiv preprint arXiv:2401.03038* (2024).
[7] Zeyuan Yang, Peng Li, and Yang Liu. 2023. Failures Pave the Way: Enhancing Large Language Models through Tuning-free Rule Accumulation. In *EMNLP*.
[8] Xuanhe Zhou, Chengliang Chai, Guoliang Li, and Ji Sun. 2020. Database meets artificial intelligence: A survey. *IEEE TKDE* 34, 3 (2020), 1096–1116.
[9] Xuanhe Zhou, Guoliang Li, Zhaoyan Sun, Zhiyuan Liu, Weize Chen, Jianming Wu, Jiesi Liu, Ruohang Feng, and Guoyang Zeng. 2024. D-Bot: Database Diagnosis System using Large Language Models. *VLDB* 17, 10 (Aug. 2024), 2514–2527.
[10] Xuanhe Zhou, Zhaoyan Sun, and Guoliang Li. 2024. Db-gpt: Large language model meets database. *Data Science and Engineering* 9, 1 (2024), 102–111.