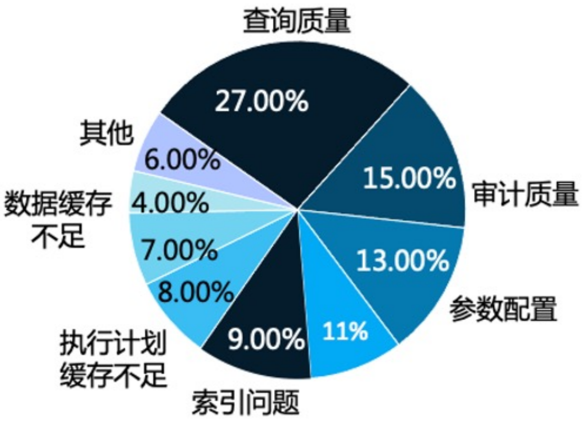
# LLM As DBA

# 背景：数据库稳定性至关重要

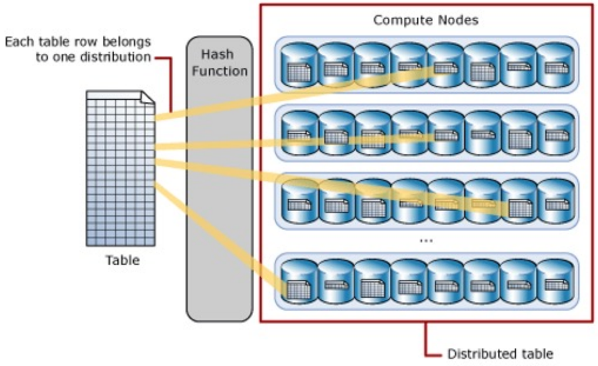
“金融机构数据库选型，最看重的三个因素：
**稳定性、安全性、性能**，分别占比80.4%、72.6%、59.6%”

# 背景：数据库运维压力与日俱增

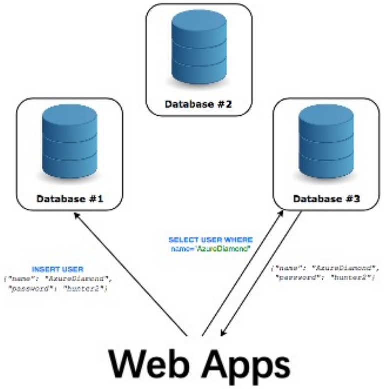
- **系统运维文档多：**单一数据库产品对外提供的 管理员指南、运维调优、故障定位手册 等达到**1w多页**，普通管理人员很难完全掌握；
- **云上运维压力大：**云数据库实例数巨大，运维工程师（DBA）短缺；
- **复杂问题诊断难：**紧急问题多层连环，运维人员难以快速响应，造成经济损失



系统根因繁杂



1M+实例难管理

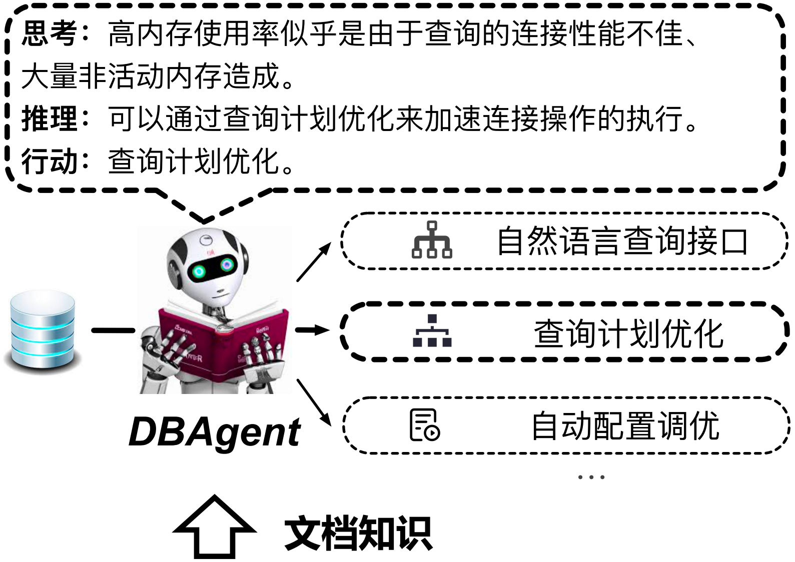

Database #2
Database #1
Database #3
SELECT USER WHERE name="AzureDiamond"
INSERT USER
{"name": "AzureDiamond", "password": "hunter2"}
{"name": "AzureDiamond", "password": "hunter2"}
Web Apps

线上问题难调优

# 愿景：智能数据库运维

- **目标：**利用大模型训练“智能运维助手”，学习人类运维经验、诊断根因
- **好处：**节省人力；提升运维工具使用能力；提升运维知识泛化能力

专家系统等难以在相似例子上复用



**思考：**高内存使用率似乎是由于查询的连接性能不佳、大量非活动内存造成。
**推理：**可以通过查询计划优化来加速连接操作的执行。
**行动：**查询计划优化。

自然语言查询接口

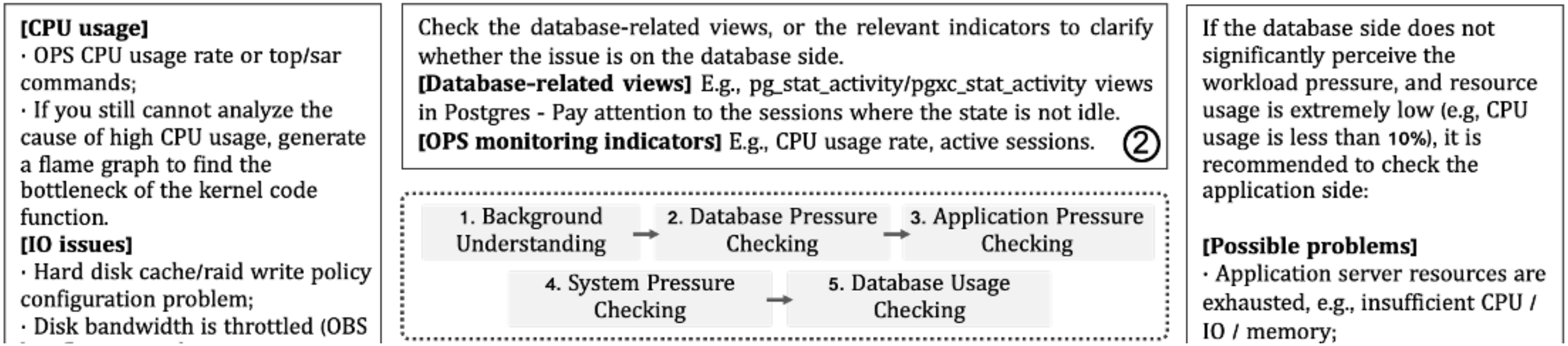
查询计划优化

自动配置调优

…

*DBAgent*

**文档知识**



**[CPU usage]**
· OPS CPU usage rate or top/sar commands;
· If you still cannot analyze the cause of high CPU usage, generate a flame graph to find the bottleneck of the kernel code function.
**[IO issues]**
· Hard disk cache/raid write policy configuration problem;
· Disk bandwidth is throttled (OBS

Check the database-related views, or the relevant indicators to clarify whether the issue is on the database side.
**[Database-related views]** E.g., pg_stat_activity/pgxc_stat_activity views in Postgres - Pay attention to the sessions where the state is not idle.
**[OPS monitoring indicators]** E.g., CPU usage rate, active sessions. ②

1. Background Understanding → 2. Database Pressure Checking → 3. Application Pressure Checking
4. System Pressure Checking → 5. Database Usage Checking

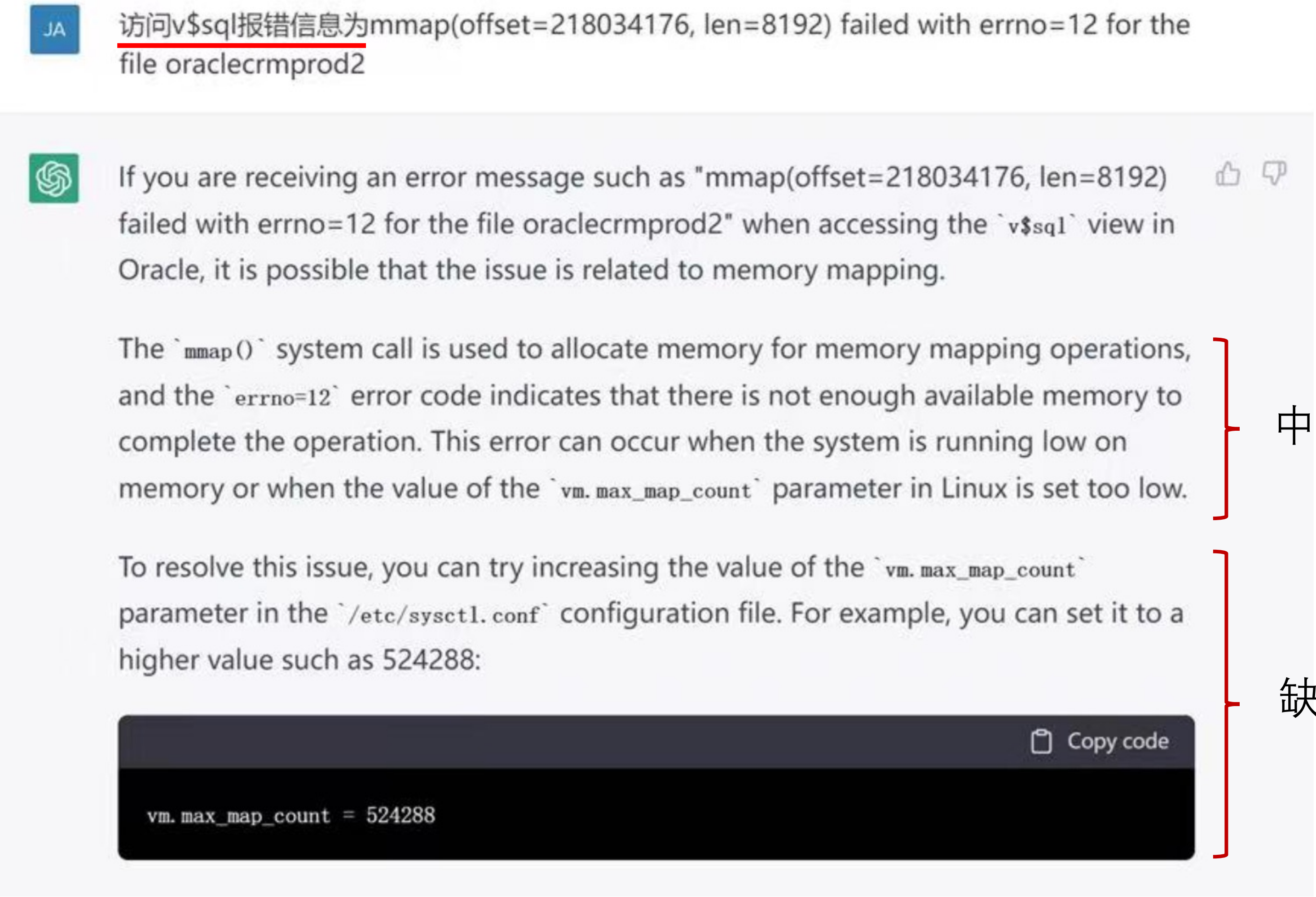
If the database side does not significantly perceive the workload pressure, and resource usage is extremely low (e.g, CPU usage is less than 10%), it is recommended to check the application side:

**[Possible problems]**
· Application server resources are exhausted, e.g., insufficient CPU / IO / memory;

# 演进一：问答助手

- 第一阶段

问题描述 ⇨ 问答助手



JA

访问v$sql报错信息为mmap(offset=218034176, len=8192) failed with errno=12 for the file oraclecrmprod2

If you are receiving an error message such as "mmap(offset=218034176, len=8192) failed with errno=12 for the file oraclecrmprod2" when accessing the `v$sql` view in Oracle, it is possible that the issue is related to memory mapping.

The `mmap()` system call is used to allocate memory for memory mapping operations, and the `errno=12` error code indicates that there is not enough available memory to complete the operation. This error can occur when the system is running low on memory or when the value of the `vm.max_map_count` parameter in Linux is set too low.

To resolve this issue, you can try increasing the value of the `vm.max_map_count` parameter in the `/etc/sysctl.conf` configuration file. For example, you can set it to a higher value such as 524288:

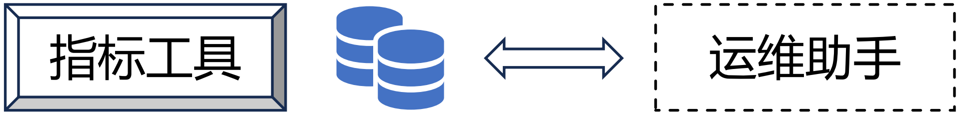
```
vm.max_map_count = 524288
```

中规中矩

缺乏依据

# 演进二：诊断助手

- 第二阶段



指标工具

运维助手

**[Chief DBA]:** The anomaly started at 1691897340 and ended at 1691897430.
**[CPU Agent]:** Based on the CPU relevant metric values, … . I suggest reviewing system logs and identifying any recent changes or updates. We should also analyze other relevant metrics such as memory usage, disk I/O, network traffic, and check for any abnormal behavior from specific processes or applications.

----------------------------

*(matched knowledge) missing_index*

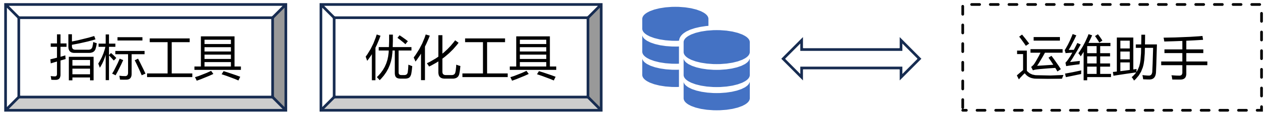
**[Memory Agent]:** The root cause of the high memory usage could be disk spill and insufficient memory allocation. I suggest analyzing the actual resource usage during query execution, performing a thorough analysis of the query plans, and evaluating the system configuration settings related to memory allocation, such as 'work_mem' and 'shared_buffers.' More information about the sort spill count or hash spill count and the threshold values is needed to confirm if disk spill is the root cause. If the free memory is consistently low and close to the total memory capacity, consider increasing the memory allocation for the system.

----------------------------

*(matched knowledge) disk_spill*

# 演进三：优化助手

- 第三阶段



指标工具 优化工具 ⟺ 运维助手

**[Chief DBA]:** The anomaly started at 1691897340 and ended at 1691897430.
**[CPU Agent]:** The anomaly is caused by high CPU due to the high number of running processes and the relatively high load averages, especially in the past 1 minute. In addition, slow queries also contribute to the high CPU usage. … 5. Add the recommended indexes: ['movie_info#movie_id', 'movie_companies#movie_id', 'keyword#keyword', 'aka_name#person_id'] to reduce cost from 46856716.92000003 to 31393898.919999983.

----------------------------

*(matched knowledge) high number of running processes*

…

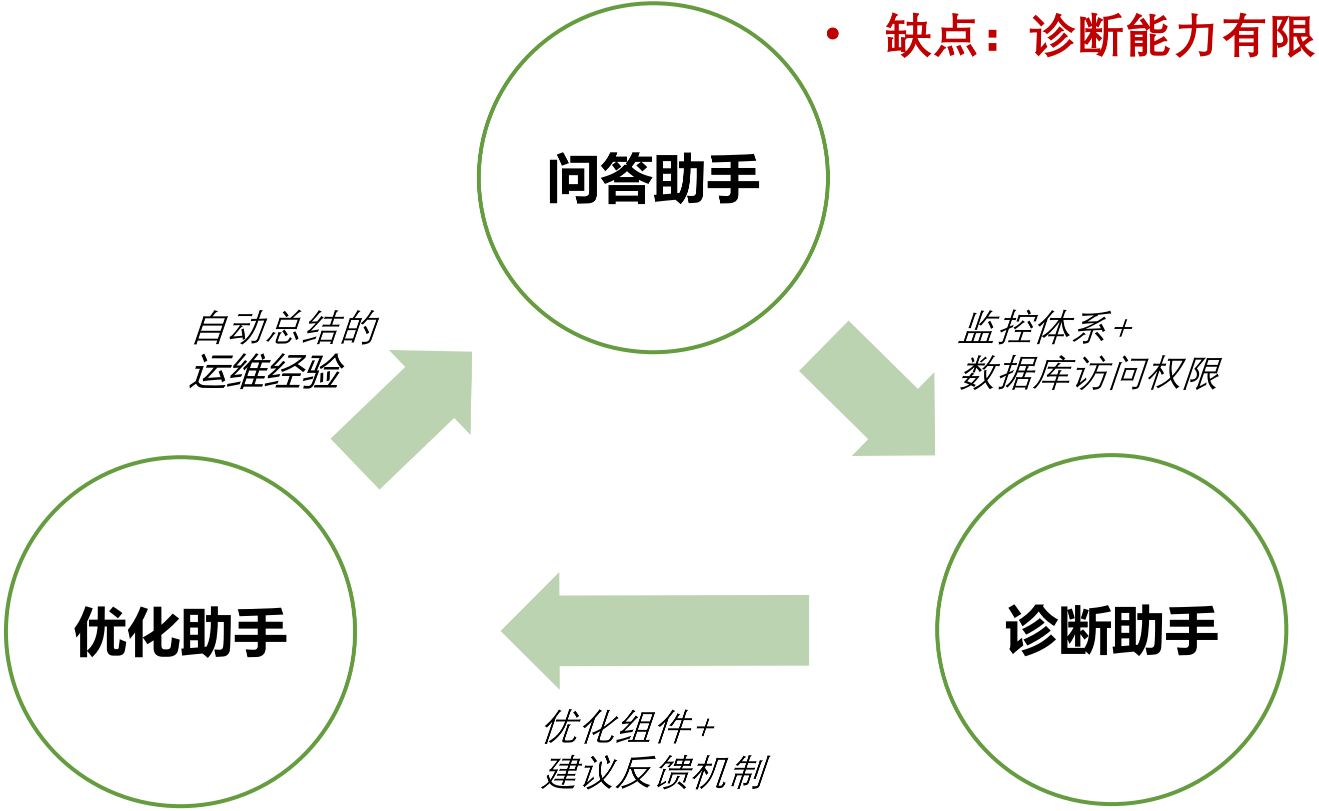
**[Chief DBA]:** The identified root causes of the anomaly are high CPU usage due to the high number of running processes and high load averages, slow queries, and high memory due to complex queries, insufficient memory allocation, and the possibility of disk spill.

----------------------------

*(solution) Add the recommended indexes: ['movie_info#movie_id', 'movie_companies#movie_id', 'keyword#keyword', 'aka_name#person_id'] to reduce cost.*

# 三个阶段的优缺点对比

- 优点：通识、易部署
- 缺点：诊断能力有限、幻觉



**问答助手**

*监控体系+*
*数据库访问权限*

**诊断助手**

*优化组件+*
*建议反馈机制*

**优化助手**

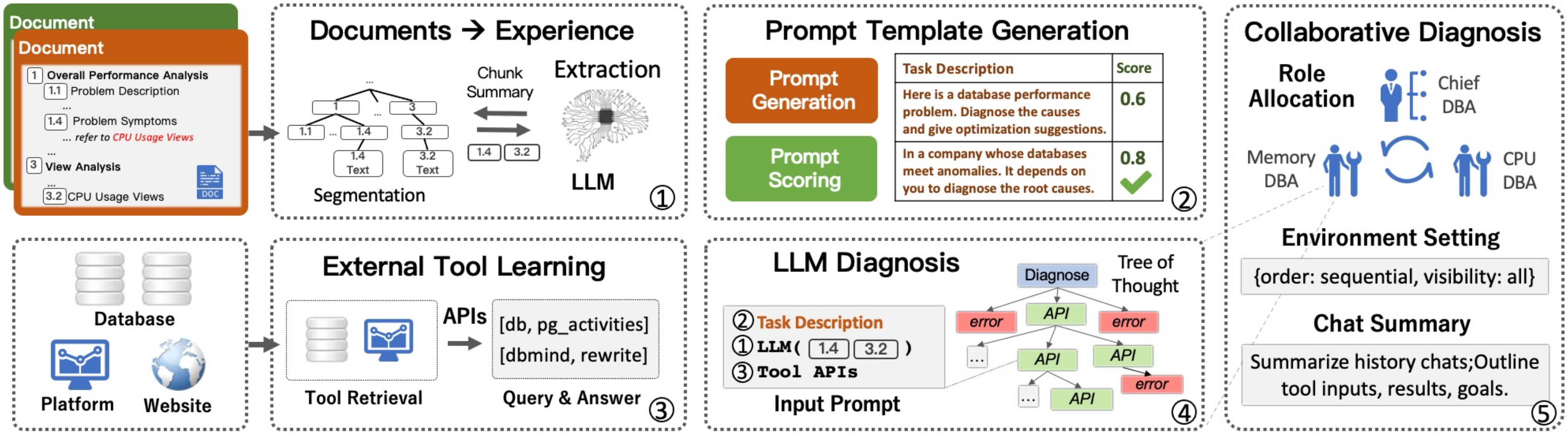
*自动总结的*
*运维经验*

- 优点：诊断精度更高
- 缺点：模型能力、部署门槛高

- 优点：诊断有根据、粒度更细
- 缺点：模型能力、部署门槛高

# 架构设计

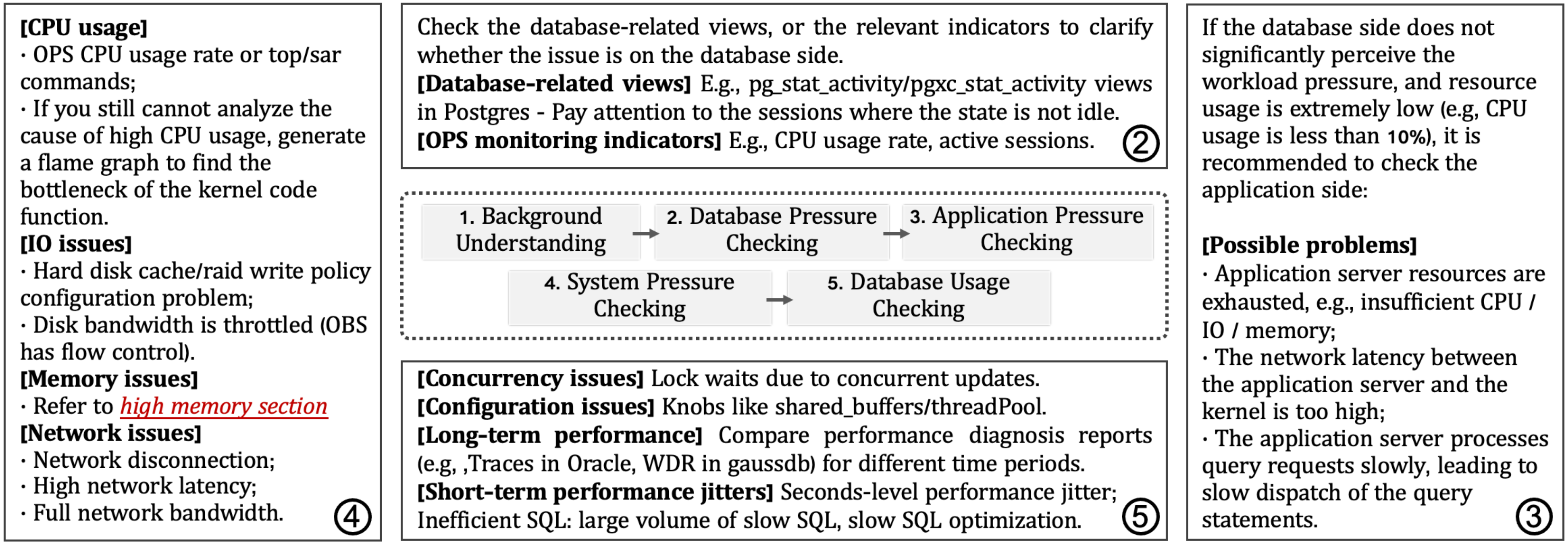
- **可观测体系构建：**日志、视图、指标等数据的原始值和加工值；
- **运维知识抽取：**分割文本块、生成摘要索引、抽取形式化知识；
- **外部工具学习：**学习API调用方式、基于树搜索提高工具的使用能力；
- **协同诊断优化：**DBA主管（诊断调度、总结发言）；子领域专家（如资源异常专家、查询优化专家等）；对话摘要员；用户（有效性反馈）



| Task Description | Score |
|---|---|
| Here is a database performance problem. Diagnose the causes and give optimization suggestions. | 0.6 |
| In a company whose databases meet anomalies. It depends on you to diagnose the root causes. | 0.8 ✓ |

# 典型异常

1. **整体性能慢背景：**客户预期、业务类型、近期业务变化、系统是否发生变化等；
2. **数据库内核压力：**数据库所在主机的CPU使用率、数据库相关视图、OPS相关指标；
3. **业务侧压力：**应用服务器资源耗尽、应用服务器和内核网络时延过高；
4. **数据库系统资源：**CPU满、IO满/IO异常、内存不足、网络异常；
5. **数据库使用不优：**并发、配置、异常等待、不优SQL。



**[CPU usage]**
- OPS CPU usage rate or top/sar commands;
- If you still cannot analyze the cause of high CPU usage, generate a flame graph to find the bottleneck of the kernel code function.

**[IO issues]**
- Hard disk cache/raid write policy configuration problem;
- Disk bandwidth is throttled (OBS has flow control).

**[Memory issues]**
- Refer to *high memory section*

**[Network issues]**
- Network disconnection;
- High network latency;
- Full network bandwidth. ④

Check the database-related views, or the relevant indicators to clarify whether the issue is on the database side.
**[Database-related views]** E.g., pg_stat_activity/pgxc_stat_activity views in Postgres - Pay attention to the sessions where the state is not idle.
**[OPS monitoring indicators]** E.g., CPU usage rate, active sessions. ②

1. Background Understanding → 2. Database Pressure Checking → 3. Application Pressure Checking
4. System Pressure Checking → 5. Database Usage Checking

**[Concurrency issues]** Lock waits due to concurrent updates.
**[Configuration issues]** Knobs like shared_buffers/threadPool.
**[Long-term performance]** Compare performance diagnosis reports (e.g, ,Traces in Oracle, WDR in gaussdb) for different time periods.
**[Short-term performance jitters]** Seconds-level performance jitter;
Inefficient SQL: large volume of slow SQL, slow SQL optimization. ⑤

If the database side does not significantly perceive the workload pressure, and resource usage is extremely low (e.g, CPU usage is less than 10%), it is recommended to check the application side:

**[Possible problems]**
- Application server resources are exhausted, e.g., insufficient CPU / IO / memory;
- The network latency between the application server and the kernel is too high;
- The application server processes query requests slowly, leading to slow dispatch of the query statements. ③

# 典型异常

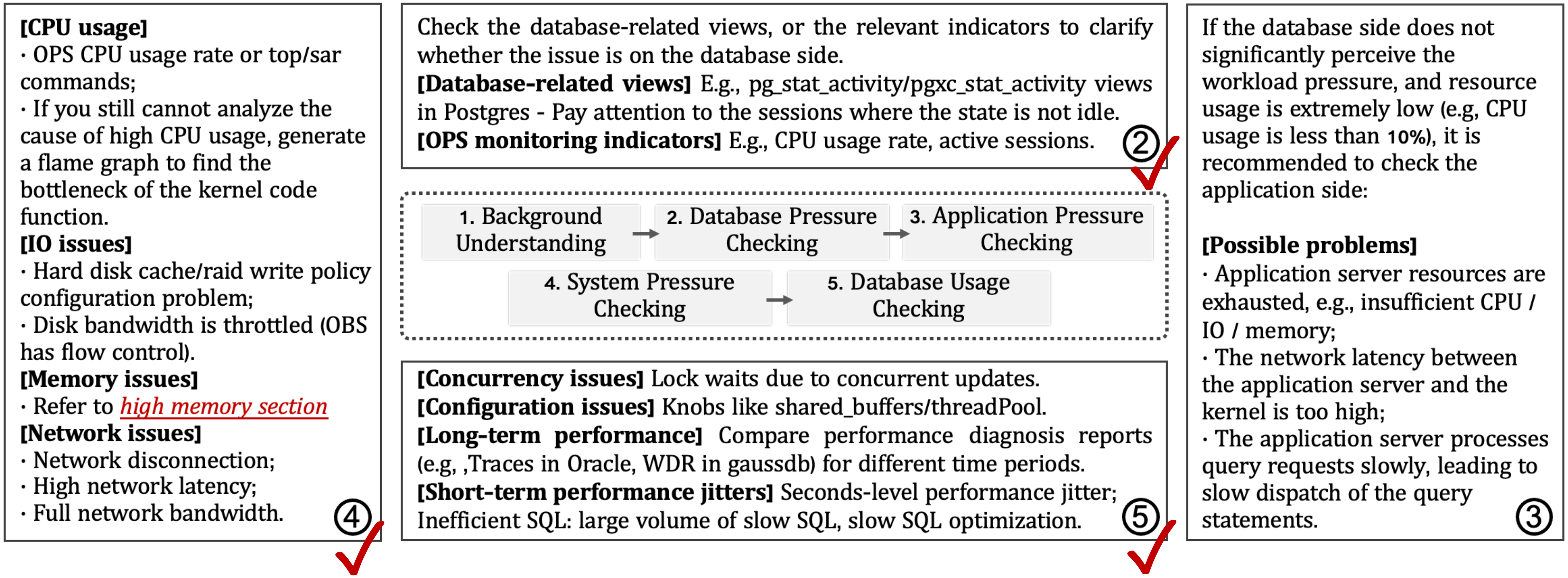
1. **整体性能慢背景：** 客户预期、业务类型、近期业务变化、系统是否发生变化等；

2. **数据库内核压力：** 数据库所在主机的CPU使用率、数据库相关视图、OPS相关指标；

3. **业务侧压力：** 应用服务器资源耗尽、应用服务器和内核网络时延过高；

4. **数据库系统资源：** CPU满、IO满/IO异常、内存不足、网络异常；

5. **数据库使用不优：** 并发、配置、异常等待、不优SQL。



**[CPU usage]**
- OPS CPU usage rate or top/sar commands;
- If you still cannot analyze the cause of high CPU usage, generate a flame graph to find the bottleneck of the kernel code function.

**[IO issues]**
- Hard disk cache/raid write policy configuration problem;
- Disk bandwidth is throttled (OBS has flow control).

**[Memory issues]**
- Refer to *high memory section*

**[Network issues]**
- Network disconnection;
- High network latency;
- Full network bandwidth.

④ ✓

Check the database-related views, or the relevant indicators to clarify whether the issue is on the database side.
**[Database-related views]** E.g., pg_stat_activity/pgxc_stat_activity views in Postgres - Pay attention to the sessions where the state is not idle.
**[OPS monitoring indicators]** E.g., CPU usage rate, active sessions. ② ✓

1. Background Understanding → 2. Database Pressure Checking → 3. Application Pressure Checking

4. System Pressure Checking → 5. Database Usage Checking

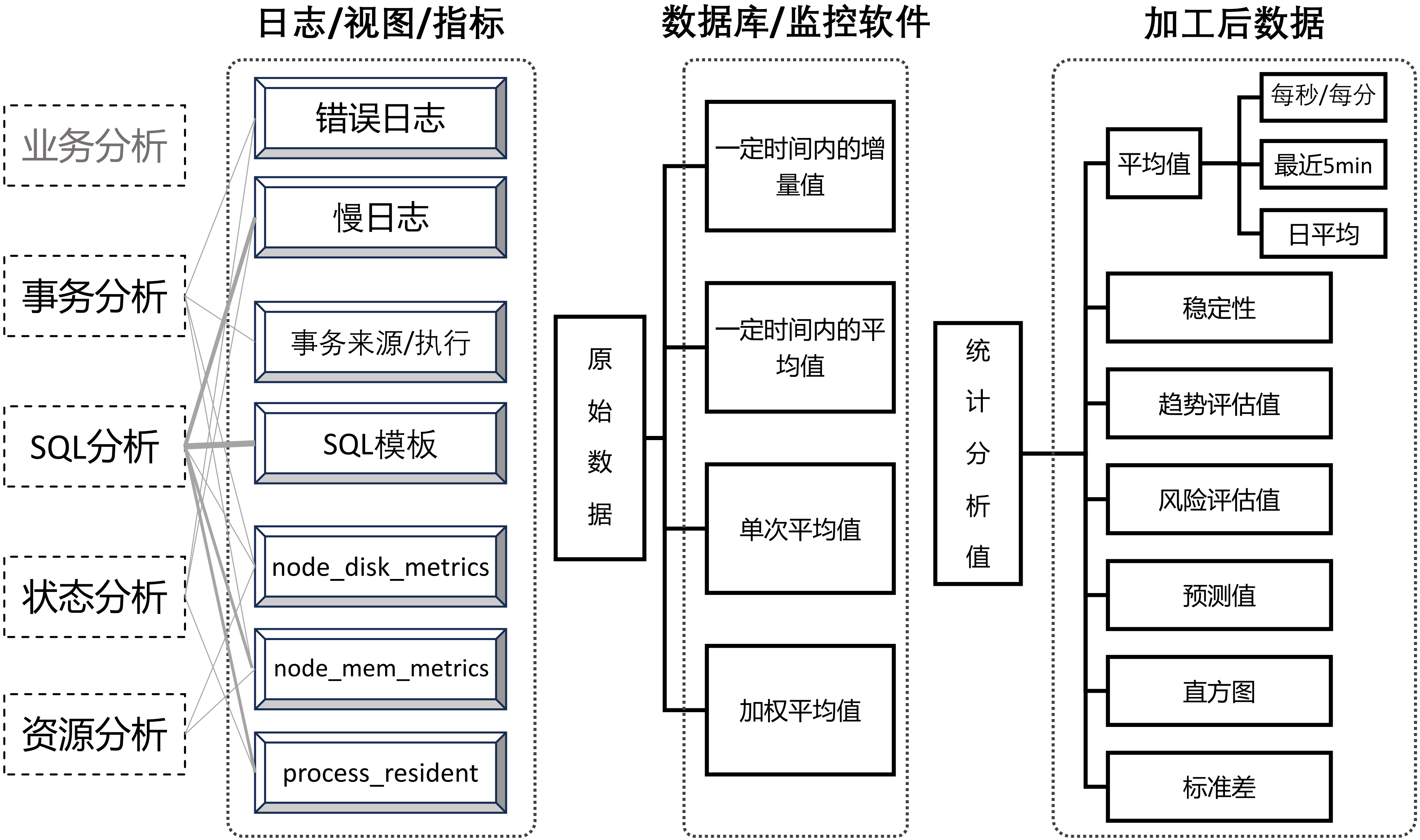
**[Concurrency issues]** Lock waits due to concurrent updates.
**[Configuration issues]** Knobs like shared_buffers/threadPool.
**[Long-term performance]** Compare performance diagnosis reports (e.g, ,Traces in Oracle, WDR in gaussdb) for different time periods.
**[Short-term performance jitters]** Seconds-level performance jitter;
Inefficient SQL: large volume of slow SQL, slow SQL optimization. ⑤ ✓

If the database side does not significantly perceive the workload pressure, and resource usage is extremely low (e.g, CPU usage is less than 10%), it is recommended to check the application side:

**[Possible problems]**
- Application server resources are exhausted, e.g., insufficient CPU / IO / memory;
- The network latency between the application server and the kernel is too high;
- The application server processes query requests slowly, leading to slow dispatch of the query statements.

③

# 技术1：可观测体系



业务分析

事务分析

SQL分析

状态分析

资源分析

**日志/视图/指标**

- 错误日志
- 慢日志
- 事务来源/执行
- SQL模板
- node_disk_metrics
- node_mem_metrics
- process_resident

**数据库/监控软件**

原始数据

- 一定时间内的增量值
- 一定时间内的平均值
- 单次平均值
- 加权平均值

统计分析值

**加工后数据**

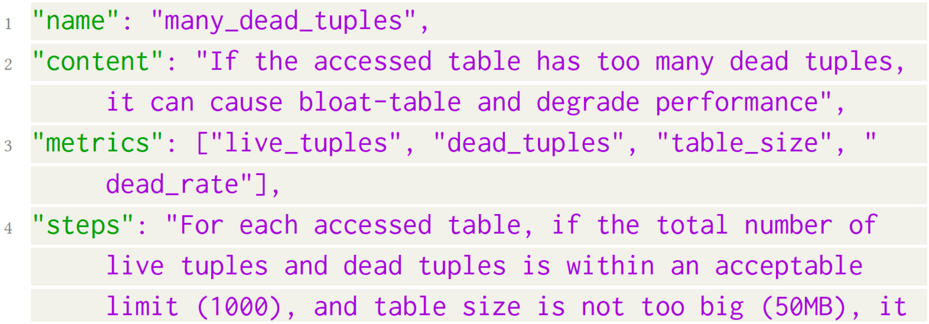
- 平均值
  - 每秒/每分
  - 最近5min
  - 日平均
- 稳定性
- 趋势评估值
- 风险评估值
- 预测值
- 直方图
- 标准差

# 技术2：文档中的知识抽取

## 知识抽取：长文本、章节之间有关联性

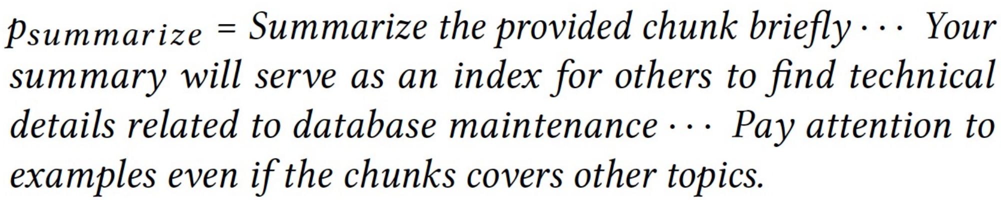
### 1. 基于语义的文本拆分

- 识别章节标识符，基于章节拆分
- 针对超过1k token的章节，树状拆分



```
"name": "many_dead_tuples",
"content": "If the accessed table has too many dead tuples,
     it can cause bloat-table and degrade performance",
"metrics": ["live_tuples", "dead_tuples", "table_size", "
     dead_rate"],
"steps": "For each accessed table, if the total number of
     live tuples and dead tuples is within an acceptable
     limit (1000), and table size is not too big (50MB), it
```

### 2. 文本的摘要索引构建

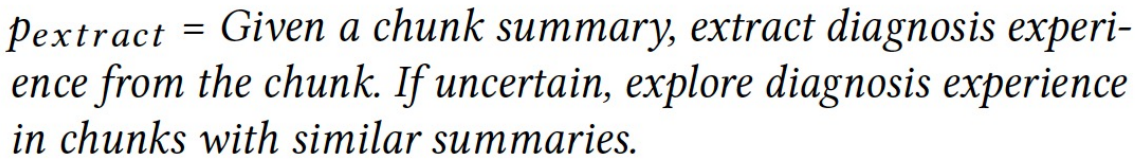


$p_{summarize}$ = *Summarize the provided chunk briefly ··· Your summary will serve as an index for others to find technical details related to database maintenance ··· Pay attention to examples even if the chunks covers other topics.*

文本块的摘要索引组成：1）该文本块内容；2）孩子节点的标题和摘要

### 3. 运维经验抽取

- LLM解析每个文本块的内容，并与相似内容的其他块的摘要进行比较



$p_{extract}$ = *Given a chunk summary, extract diagnosis experience from the chunk. If uncertain, explore diagnosis experience in chunks with similar summaries.*

# 技术3：基于树搜索的工具调用

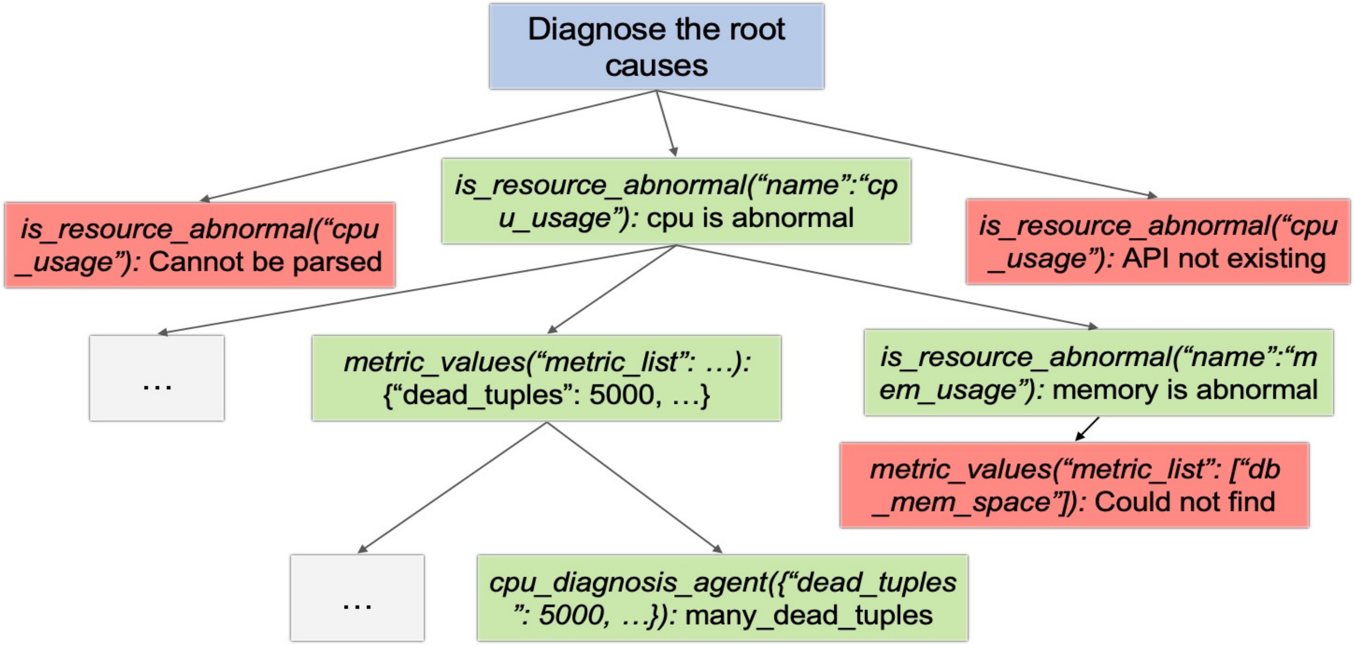
## 不使用外部工具导致幻觉 / 提前停止诊断 / 生成错误API

**1. 工具调用**

- BM25 / LLM Embedding / Dense Retrival

**2. 树搜索算法**

- **初始化：**根节点-诊断请求；**价值计算：**UCT函数（ $\mathrm{UCT}(v) = \frac{w(v)}{n(v)} + C \cdot \sqrt{\frac{\ln(N)}{n(v)}}$ ）
- **模拟运维：**如从当前节点，探索能找到新异常指标的子节点；
- **反思：**如果LLM决定当前路径无法找到有用信息，减少UCT值（如变为父节点的值）



Diagnose the root causes

*is_resource_abnormal(“cpu_usage”):* Cannot be parsed

*is_resource_abnormal(“name”:“cpu_usage”):* cpu is abnormal

*is_resource_abnormal(“cpu_usage”):* API not existing

…

*metric_values(“metric_list”: …):* {“dead_tuples”: 5000, …}

*is_resource_abnormal(“name”:“mem_usage”):* memory is abnormal

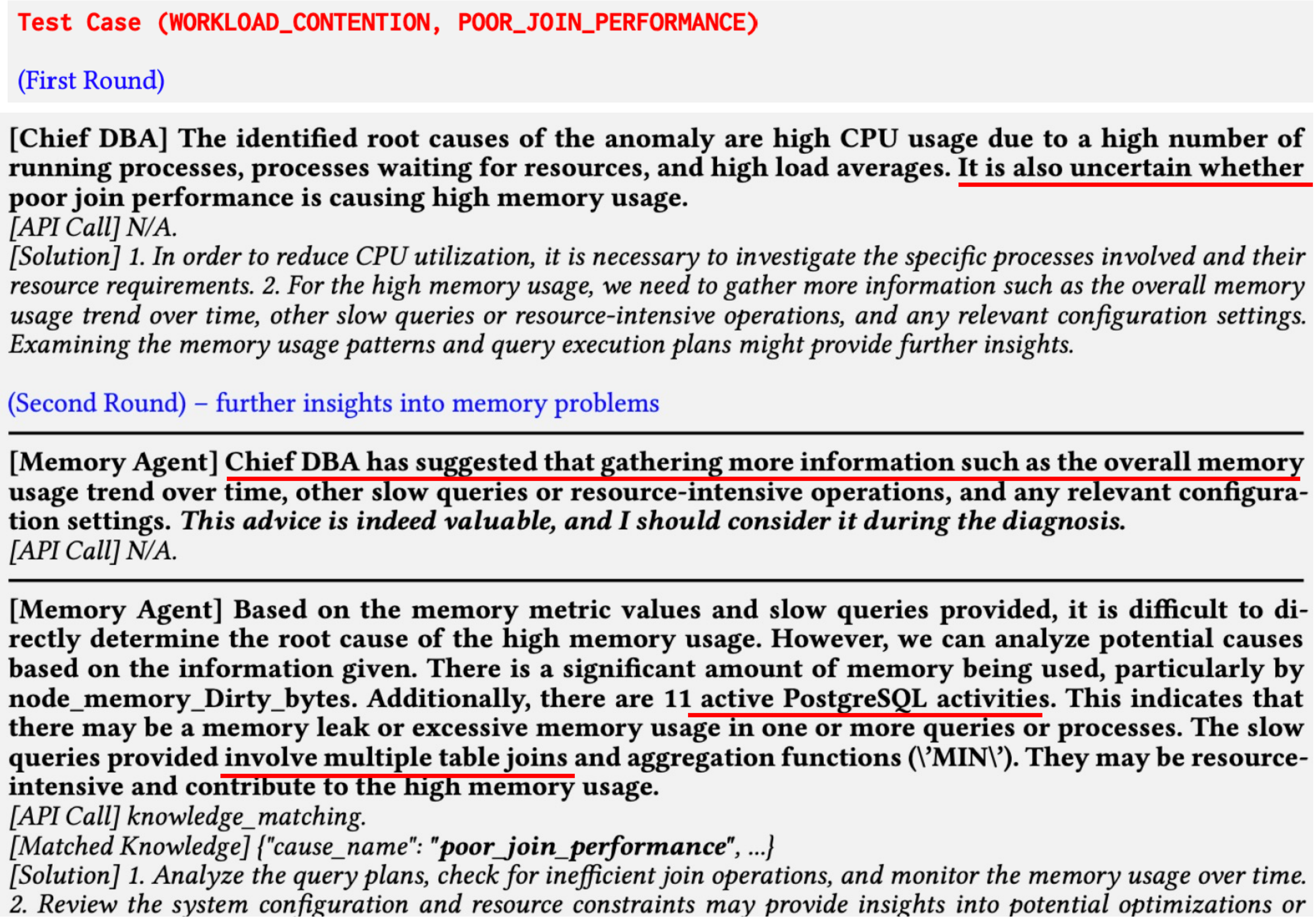
*metric_values(“metric_list”: [“db_mem_space”]):* Could not find

…

*cpu_diagnosis_agent({“dead_tuples”: 5000, …}):* many_dead_tuples

# 技术4：多模型会诊机制

## “沟通”机制对于解决复杂问题非常重要 （1 LLM vs 3 LLMs）



`Test Case (WORKLOAD_CONTENTION, POOR_JOIN_PERFORMANCE)`

(First Round)

**[Chief DBA] The identified root causes of the anomaly are high CPU usage due to a high number of running processes, processes waiting for resources, and high load averages. It is also uncertain whether poor join performance is causing high memory usage.**
*[API Call] N/A.*
*[Solution] 1. In order to reduce CPU utilization, it is necessary to investigate the specific processes involved and their resource requirements. 2. For the high memory usage, we need to gather more information such as the overall memory usage trend over time, other slow queries or resource-intensive operations, and any relevant configuration settings. Examining the memory usage patterns and query execution plans might provide further insights.*

(Second Round) – further insights into memory problems

---

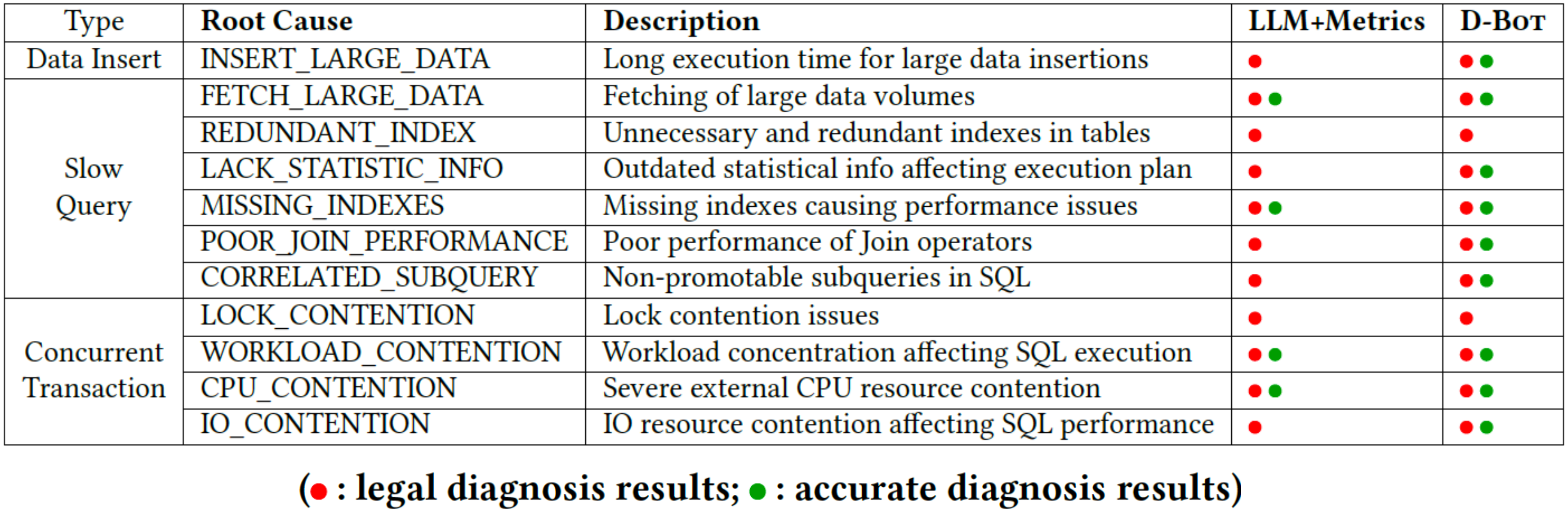
**[Memory Agent] Chief DBA has suggested that gathering more information such as the overall memory usage trend over time, other slow queries or resource-intensive operations, and any relevant configuration settings. *This advice is indeed valuable, and I should consider it during the diagnosis.***
*[API Call] N/A.*

---

**[Memory Agent] Based on the memory metric values and slow queries provided, it is difficult to directly determine the root cause of the high memory usage. However, we can analyze potential causes based on the information given. There is a significant amount of memory being used, particularly by node_memory_Dirty_bytes. Additionally, there are 11 active PostgreSQL activities. This indicates that there may be a memory leak or excessive memory usage in one or more queries or processes. The slow queries provided involve multiple table joins and aggregation functions (\'MIN\'). They may be resource-intensive and contribute to the high memory usage.**
*[API Call] knowledge_matching.*
*[Matched Knowledge] {"cause_name": "**poor_join_performance**", ...}*
*[Solution] 1. Analyze the query plans, check for inefficient join operations, and monitor the memory usage over time. 2. Review the system configuration and resource constraints may provide insights into potential optimizations or*

# 实验结果（诊断表现）

- 异常：11种单根因异常 + 部分多根因异常
- **Baseline**：vanilla GPT4 + 相关的监控指标
- 结果对比：相对baseline取得较高的准确率
  - 推理能力更强，因此能够探索更加细粒度系统运行指标/视图
  - 使用领域知识，避免“捏造根因”的情况
  - 学会使用工具，给出有实用价值的优化建议



| Type | **Root Cause** | **Description** | **LLM+Metrics** | **D-Bot** |
|---|---|---|---|---|
| Data Insert | INSERT_LARGE_DATA | Long execution time for large data insertions | 🔴 | 🔴 🟢 |
| Slow Query | FETCH_LARGE_DATA | Fetching of large data volumes | 🔴 🟢 | 🔴 🟢 |
| | REDUNDANT_INDEX | Unnecessary and redundant indexes in tables | 🔴 | 🔴 |
| | LACK_STATISTIC_INFO | Outdated statistical info affecting execution plan | 🔴 | 🔴 🟢 |
| | MISSING_INDEXES | Missing indexes causing performance issues | 🔴 🟢 | 🔴 🟢 |
| | POOR_JOIN_PERFORMANCE | Poor performance of Join operators | 🔴 | 🔴 🟢 |
| | CORRELATED_SUBQUERY | Non-promotable subqueries in SQL | 🔴 | 🔴 🟢 |
| Concurrent Transaction | LOCK_CONTENTION | Lock contention issues | 🔴 | 🔴 |
| | WORKLOAD_CONTENTION | Workload concentration affecting SQL execution | 🔴 🟢 | 🔴 🟢 |
| | CPU_CONTENTION | Severe external CPU resource contention | 🔴 🟢 | 🔴 🟢 |
| | IO_CONTENTION | IO resource contention affecting SQL performance | 🔴 | 🔴 🟢 |

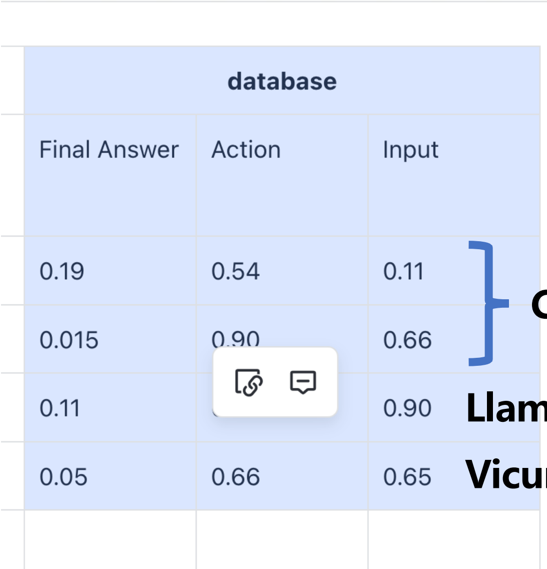
**(🔴 : legal diagnosis results; 🟢 : accurate diagnosis results)**

# 实验结果（诊断表现）

- Data Insert
  - INSERT_LARGE_DATA：批量插入数据时执行耗时较长
  - UPDATE_LARGE_TABLE
  - DELETE_LARGE_DATA
- Slow Query
  - FETCH_LARGE_DATA
  - REDUNDANT_INDEX：SQL相关表存在无用索引与冗余索引 (DML)
  - DISK_SPILL：SQL执行过程中产生SORT或HASH算子产生落盘行为
  - LACK_STATISTIC_INFO：统计信息未更新，导致基数估计不准确，影响执行计划
  - MISSING_INDEXES：由于SQL缺失索引导致性能问题
  - POOR_JOIN_PERFORMANCE: Join算子性能较差
  - STRING_MATCHING索引失效
  - COMPLEX_EXECUTION_PLAN：典型场景为SQL中join或者子查询过多
  - CORRELATED_SUBQUERY：SQL中存在子查询不能提升的场景
  - POOR_AGGREGATION_PERFORMANCE算子代价较大
- Conccurent Transactions
  - LOCK_CONTENTION
  - WORKLOAD_CONTENTION：数据库负载集中影响了SQL执行效率
  - CPU_RESOURCE_CONTENTION：数据库外进程CPU资源抢占严重
  - IO_RESOURCE_CONTENTION：IO资源紧张，导致部分SQL性能较差
  - TIMED_TASK_CONFLICT：定时任务冲突

# 实验结果（训练本地私有模型）

- 训练数据：文档手册 - tool的产品说明；诊断数据 - tool的使用案例
  - 文档手册：80+页运维宝典
  - 诊断数据：4类脚本 * 超参（读写比、并发等）* GPT-4 诊断案例
- 开源模型：如 Llama2 (语言理解能力)、Vicuna (代码理解能力)、chatcpm（中文理解能力）



| database | | | |
|---|---|---|---|
| Final Answer | Action | Input | |
| 0.19 | 0.54 | 0.11 | **Cpm-conv-0517** |
| 0.015 | 0.90 | 0.66 | |
| 0.11 | [illegible] | 0.90 | **Llama7b-finetuned** |
| 0.05 | 0.66 | 0.65 | **Vicuna7b-finetuned** |

RougeL: Evaluate the similarity between two text

**ROUGE-L = (LCS / R) * (LCS / C) * ((1 + beta^2) / (beta^2 * R + C))**

- **LCS**: Length of the **longest common subsequence** between the candidate summary and reference text.
- **R**: **Total number of words** in the reference text.
- **C**: **Total number of words** in the candidate text.
- **beta**: A parameter that controls the importance of **precision** in the F1 score calculation. Typically, beta^2 is set to 1.