

AI 大模型开发工程师 百亿级智能数据分析平台项目实战

讲师：李希沅

目录

- 1 百亿级数据分析平台项目背景
- 2 百亿级数据分析平台架构设计
- 3 百亿级数据分析平台资源评估
- 4 百亿级数据分析平台项目准备
- 5 百亿级数据分析平台代码落地
- 6 百亿级数据分析平台项目总结

1 百亿级数据分析平台项目背景



- ◆李希沅
- ◆连续创业5年
- ◆**AIGC 应用探索者**
- ◆前转转集团Hadoop平台负责人
- ◆前东方国信大数据架构师
- ◆有丰富的超大数据平台管理经验
- ◆擅长Hadoop生态, Spark生态, Flink生态

01、百亿级大数据平台演进

数据量级演进

GB级 小数据量	TB级 中等数据量	PB级 大数据量
-------------	--------------	-------------

公司发展阶段演进

0-1 初创期	1-10 高速发展期	10-100 稳定发展期
------------	---------------	-----------------

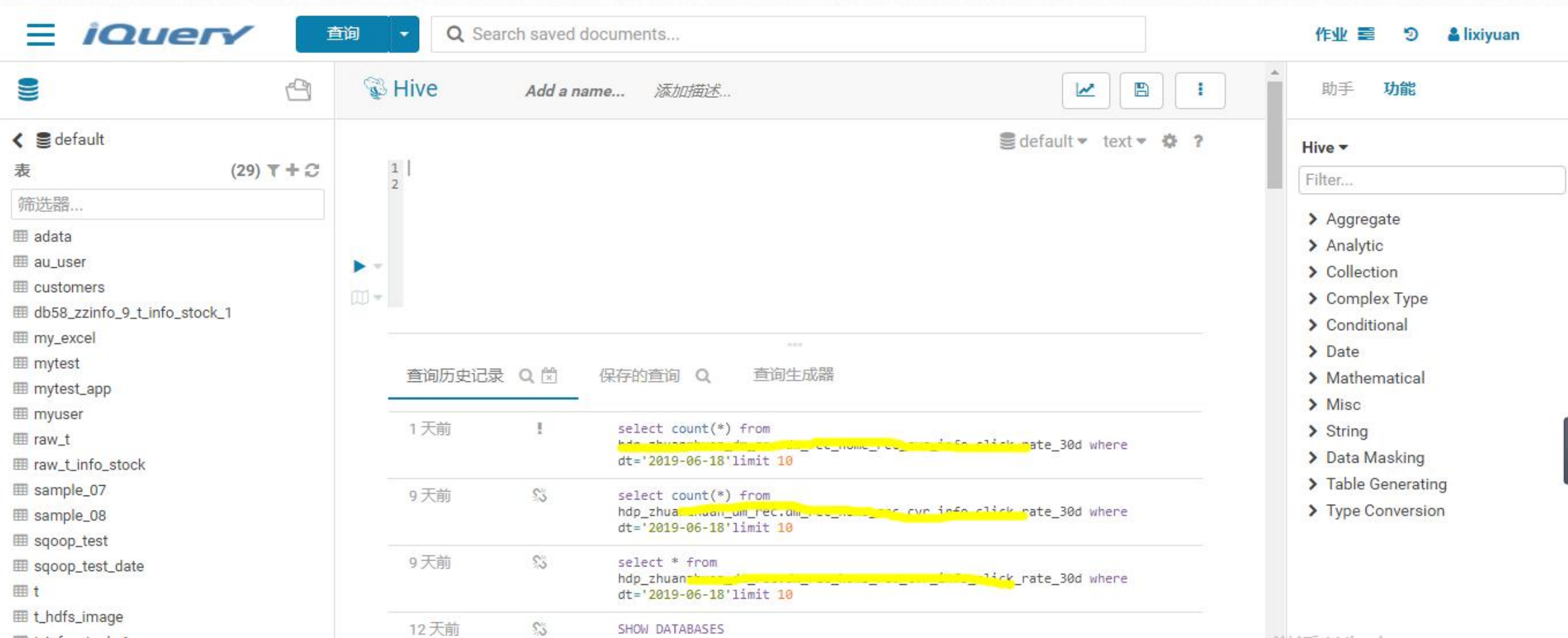
大数据架构（中台架构）演进

总体架构模式 Shell单体	总体架构模式 共建Hadoop生态分层	总体架构模式 自建Hadoop生态分层
PaaS架构设计 日志原地计算、Shell脚本	PaaS架构设计 复用云平台PaaS或者集团PaaS	PaaS架构设计 自建PaaS
DaaS架构设计 无	DaaS架构设计 自建数据仓库、数据集市	DaaS架构设计 自建全域数据仓库、数据集市
DA架构设计 无可视化、邮件发送报表	DA架构设计 可视化BI平台	DA架构设计 多端可视化业务共建BI平台

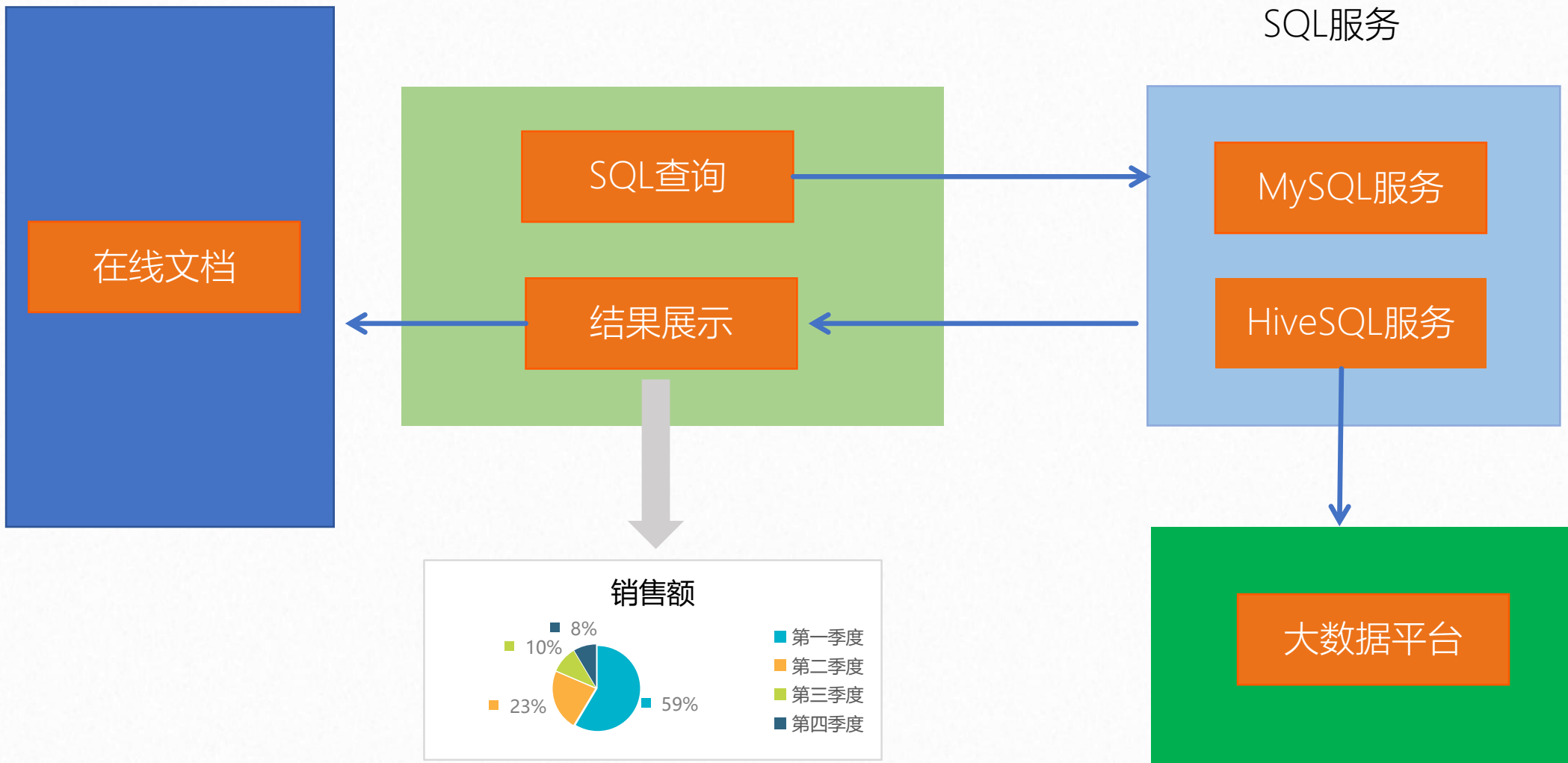
02、百亿级大数据平台

数据可视化/ 反馈	数据可视化/ 反馈	DA（数据应用层）										服务业务化	
		BI报表				数据产品				业务系统		应用治理	
		渠道分析	商品分析	交易分析		智能挖掘	自助报表	精细化推送		商品系统	财务系统	指标字典	
	数据统计/ 分析/挖掘	用户分析	订单分析	行为分析		画像档案	事件漏斗	数据地图		运营系统	客服系统	血缘关系	
		搜索推荐	竞品分析	商业分析		数据地图	监控告警	电视看板		搜索推荐	质检系统	滚动清理	
		DaaS（Data-as-a-Service）											
服务业务化		数据集市层											
资产服务化		留存模型主题表		事件模型主题表		画像提取平台		实时自助框架		生命周期管理		数据质量管理	
数据资产化	数据建模 /存储	数据仓库层											数据服务化
业务数据化		用户宽表		商品宽表		交易宽表		收入宽表		广告宽表		行为宽表	
		源数据层											
		前端埋点		后端日志		业务数据库		三方广告		竞品抓取		线下表单	
		PaaS（Platform-as-a-Service）											
Skynet调度平台	数据传输 (实时/批量)	数据计算层											
苍鹰数据治理平台		MapReduce		Spark		Storm		Flink		Kylin		Druid	
		数据存储层											
		HDFS		Hive		HBase		MySQL		TiDB		ZZRedis	
Lego日志采集平台	数据采集	数据传输层											业务数据化
		Flume		Sqoop		Kafka		Lego		Nginx		log4j	

03、百亿级iQuery自助数据分析平台



04、百亿级iQuery自助数据分析平台



05、百亿级iQuery自助数据分析平台痛点



稳定的

支持自然语言

交互式

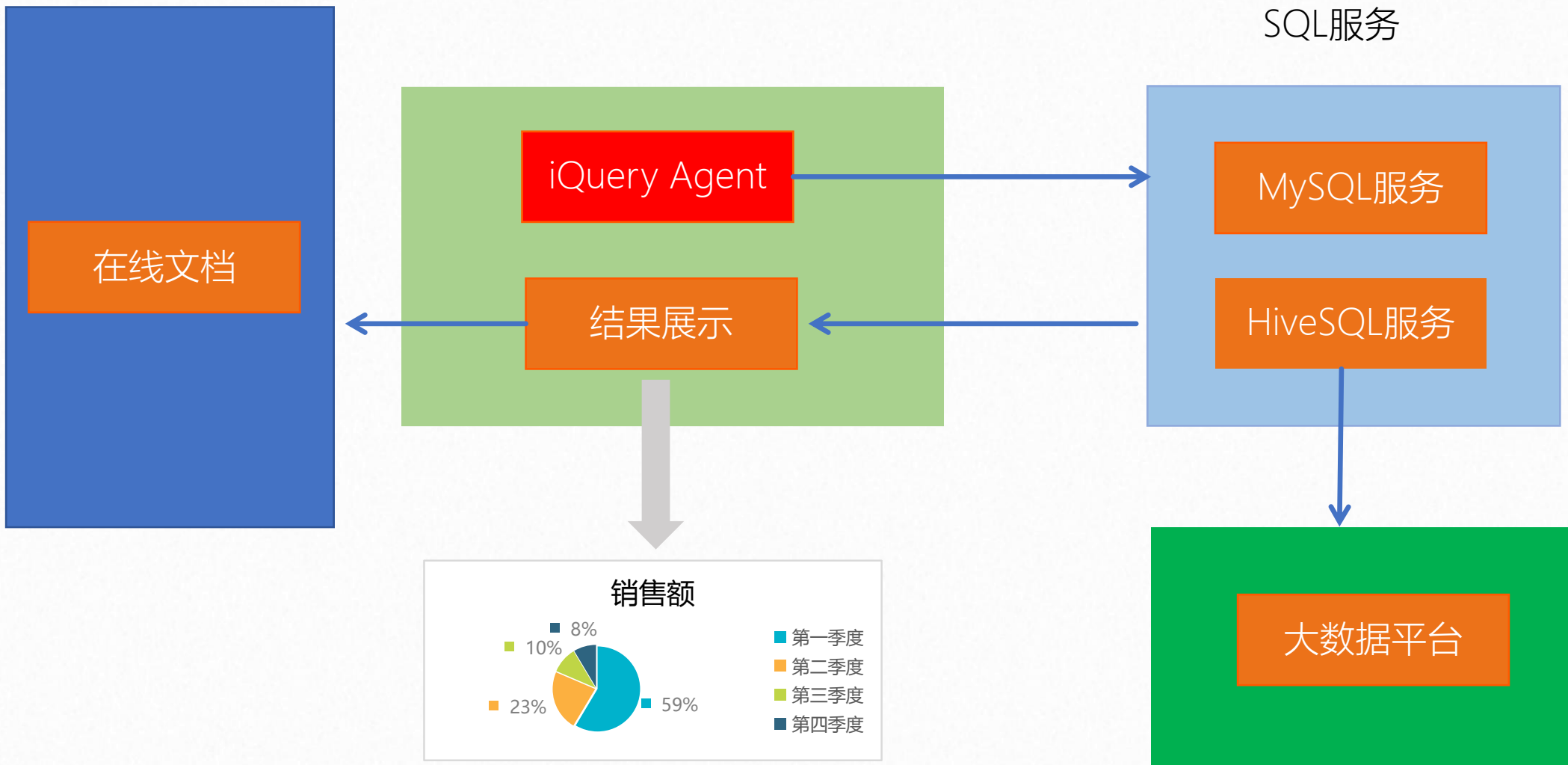
具备长/短期记忆

智能分析

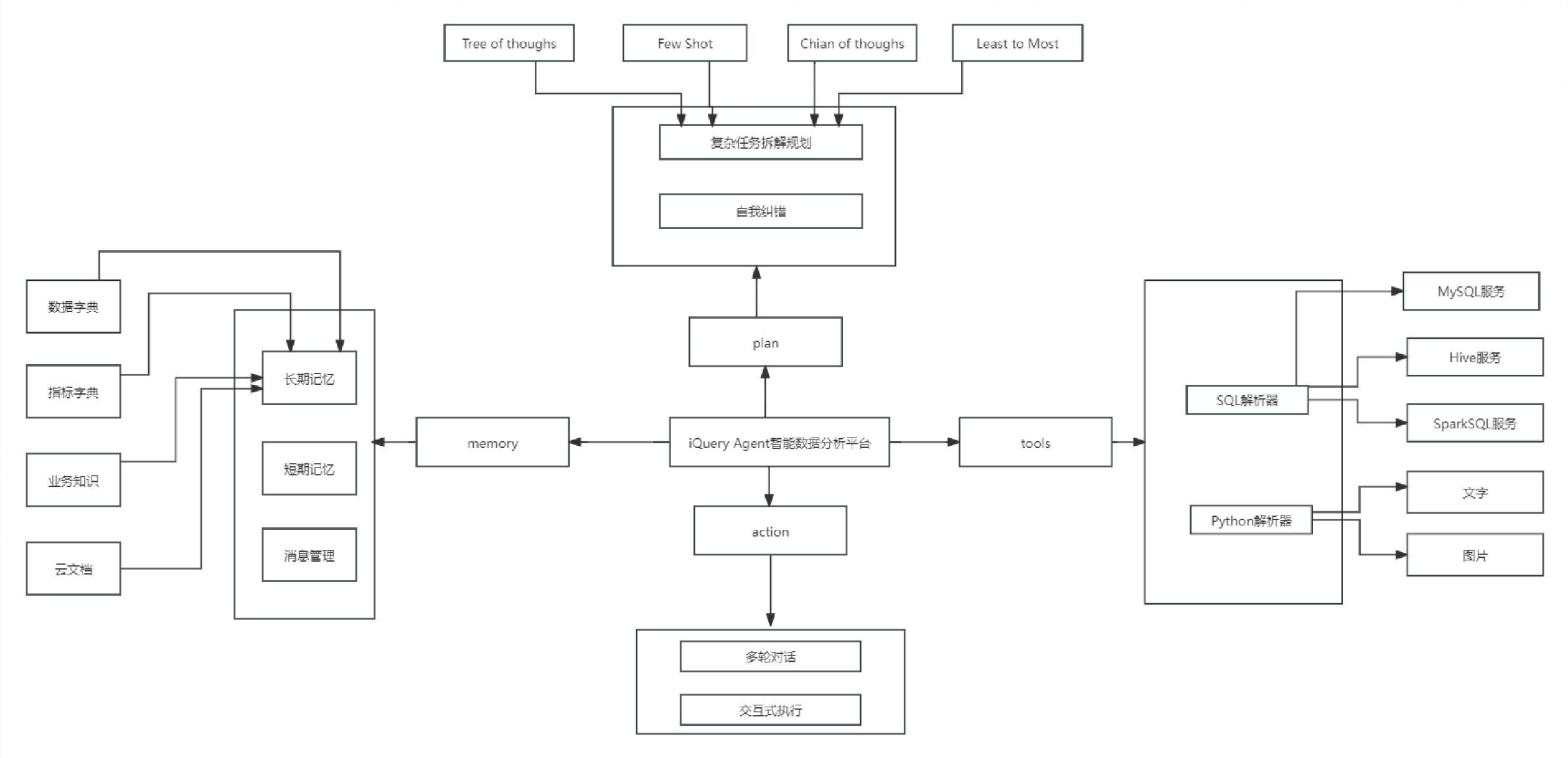
支持图表展示

② 百亿级数据分析平台架构设计

01、百亿级智能数据分析平台设计



02、百亿级智能数据分析平台设计



03、数据字典

数据字典是一个记录数据库或数据集中各种数据元素的文档或文件，它提供了对数据元素的详细描述和定义，以及数据的结构、关系和属性等信息。数据字典对于数据管理和数据分析非常重要，它有助于组织、理解和使用数据，以及确保数据的准确性、一致性和可靠性。下面将详细介绍数据字典的作用和功能，包括数据元素的定义、关系表示、数据字典的组成和应用。

数据字典通常由以下几个组成部分构成：

1. **数据元素定义**：包括数据元素名、标识符、数据类型、长度、含义、取值范围、格式等。
2. **数据结构和关系**：表明数据元素之间的逻辑和物理关系，如表的关系图、主键和外键关联等。
3. **数据属性和约束**：描述数据元素的属性和限制，如大小、精度、唯一性约束等。
4. **数据质量规则和指标**：定义数据的质量标准和要求，如完整性、准确性、一致性等。
5. **数据安全和权限**：记录数据的敏感性和访问权限，如数据分类、保密级别、用户权限等。
6. **数据源和血缘追溯**：追踪数据的来源、处理过程和传递路径，保证数据的可追溯性和可信度。
7. **数据文档和注释**：提供数据的文档和注释，包括数据定义、解释、用途和示例等信息。

04、指标字典

1. 指标字典是什么？

答:指标字典是业务数据标准化的基础，目的是对指标进行统一管理，方便共享达成对业务指标的共识，并统一修改和维护。

2、设计指标字典的目的？

1. 规范维度和量度命名，命名规则要明确，通用，易懂。
- 2.对维度或量度统一计算口径，避免歧义。
- 3.涵盖尽可能多的关注的核心维度和量度，以此为基础推动数据建设，确保指标字典里覆盖的维度都可区分、指标都可统计。

3 百亿级数据分析平台资源评估

01、百亿级智能数据分析平台技术选型

技术模块	选型
开发语言	Python
大模型选型	GPT/GLM4或者私有模型
是否需要微调	否
额外准备	1. 安装MySQL数据库 2. 安装HiveSQL服务（可选）

ChatCompletions ▾

```
1 from openai import OpenAI
2 client = OpenAI()
3
4 completion = client.chat.completions.create(
5     model="gpt-3.5-turbo",
6     messages=[
7         {"role": "system", "content": "You are a poetic assistant, skilled in explaining"},
8         {"role": "user", "content": "Compose a poem that explains the concept of recursion"}
9     ]
10 )
11
12 print(completion.choices[0].message)
```

```
1 from zhipuai import ZhipuAI
2 client = ZhipuAI(api_key="") # 填写您自己的APIKey
3 response = client.chat.completions.create(
4     model="glm-4", # 填写需要调用的模型名称
5     messages=[
6         {"role": "user", "content": "作为一名营销专家，请为我的产品创建一个吸引人的slogan"},
7         {"role": "assistant", "content": "当然，为了创建一个吸引人的slogan，请告诉我一些关于您产品的信息"},
8         {"role": "user", "content": "智谱AI开放平台"},
9         {"role": "assistant", "content": "智启未来，谱绘无限—智谱AI，让创新触手可及！"},
10        {"role": "user", "content": "创建一个更精准、吸引人的slogan"}
11    ],
12)
13 print(response.choices[0].message)
```


02、百亿级智能数据分析平台资源评估

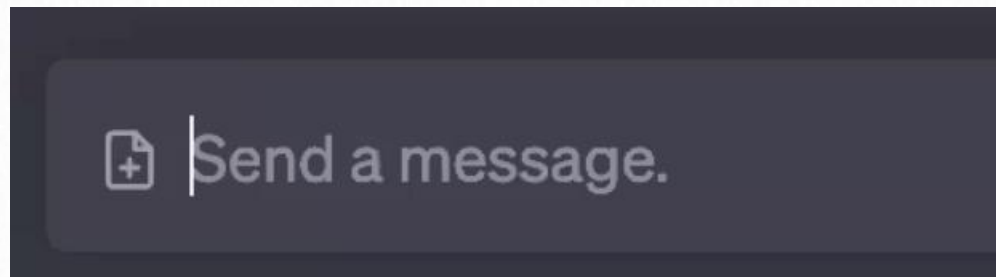
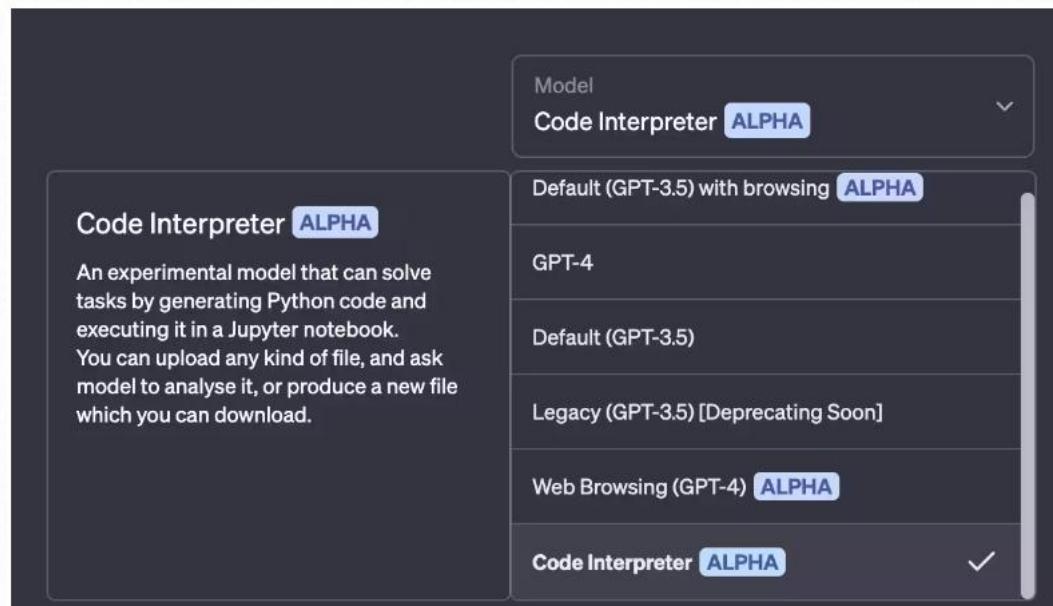
在线模型资源评估	离线模型资源评估
可忽略不计	1. 推理场景资源评估 2. 高效微调场景资源评估 3. 全量微调/预训练场景资源评估

模型参数（精度度）	大小（字节）	模型大小计算（chatglm3-6b为例）
fp32	4	24G
fp16	2	12G
int8	1	6G
int4	0.5	3G

4 百亿级数据分析平台项目准备

项目调研

01、项目调研：ChatGPT Code Interpreter



- 1、不能联网
- 2、仅限于官方内置提供的一些库，不能自己安装其他库，或调用其他库的功能。
- 3、老生常谈的数据安全话题，如果你想体验自动化的数据分析或者代码编写，就必须把本地文件进行上传。（并且有大小限制）

02、项目调研：知名开源项目

DB-GPT

<https://github.com/eosphoros-ai/DB-GPT>



DB-GPT是一个开源的AI原生**数据应用开发框架**。目的是构建大模型领域的基础设施，通过开发多模型管理(SMMF)、Text2SQL效果优化、RAG框架以及优化、Multi-Agents框架协作、AWEL(智能体 workflow 编排)等多种技术能力，让围绕数据库构建大模型应用更简单，更方便。

open-interpreter

<https://github.com/KillianLucas/open-interpreter>



Open Interpreter (**开放解释器**) 可以让大语言模型 (LLMs) 在本地运行代码 (比如 Python、JavaScript、Shell 等)。安装后，在终端上运行 `$ interpreter` 即可通过类似 ChatGPT 的界面与 Open Interpreter 聊天。

03、DB-GPT快速体验

文档: <https://github.com/eosphoros-ai/DB-GPT/blob/main/README.zh.md>

Image

 [AutoDL镜像](#)

 [小程序云部署](#)

 eosphoros-ai/DB-GPT

镜像版本: v0.5.1

框架: PyTorch: 2.0.1

CUDA: 11.8

镜像大小: 12.35GB

AutoDL创建实例

安装MySQL数据库

步骤1: 安装MySQL数据库

```
sudo apt-get update
```

```
sudo apt-get install mysql-server
```

```
sudo service mysql start
```

步骤2: 创建数据库用户

```
CREATE USER 'gpt'@'localhost' IDENTIFIED BY 'gpt';
```

步骤3: 给数据库用户赋权限

```
GRANT ALL PRIVILEGES ON *.* TO 'gpt'@'localhost';
```

```
FLUSH PRIVILEGES;
```

04、DB-GPT快速体验总结

功能强大

支持模型较多

界面体验较好

交互体验差

复杂任务拆解能力弱

03、Open Interpreter快速体验

与 ChatGPT 的代码解释器比较

OpenAI 发布的 [Code Interpreter](#) 和 GPT-4 提供了一个与 ChatGPT 完成实际任务的绝佳机会。

但是，OpenAI 的服务是托管的，闭源的，并且受到严格限制：

- 无法访问互联网。
- [预装软件包数量有限](#)。
- 允许的最大上传为 100 MB，且最大运行时间限制为 120.0 秒
- 当运行环境中途结束时，之前的状态会被清除（包括任何生成的文件或链接）。

安装工具

```
pip install open-interpreter
```

设置key

```
from interpreter import interpreter

interpreter.llm.api_key=openai.api_key
interpreter.llm.api_base=openai.api_base

interpreter.chat("你好") # 执行单一命令
```

设置模型

```
interpreter.llm.model = "gpt-3.5-turbo"
```

交互式使用

```
interpreter.chat("打印一下乘法口诀表，并且执行一下")
```

Open Interpreter will require approval before running code.

Use `interpreter -y` to bypass this.

Press `CTRL-C` to exit.

```
for i in range(1, 10):
    for j in range(1, i+1):
        print(f'{j}x{i}={i*j}', end='\t')
    print()
```

Would you like to run this code? (y/n)

y

```
for i in range(1, 10):
    for j in range(1, i+1):
        print(f'{j}x{i}={i*j}', end='\t')
    print()
```

```
1x1=1
1x2=2 2x2=4
1x3=3 2x3=6 3x3=9
1x4=4 2x4=8 3x4=12 4x4=16
1x5=5 2x5=10 3x5=15 4x5=20 5x5=25
1x6=6 2x6=12 3x6=18 4x6=24 5x6=30 6x6=36
1x7=7 2x7=14 3x7=21 4x7=28 5x7=35 6x7=42 7x7=49
1x8=8 2x8=16 3x8=24 4x8=32 5x8=40 6x8=48 7x8=56 8x8=64
1x9=9 2x9=18 3x9=27 4x9=36 5x9=45 6x9=54 7x9=63 8x9=72 9x9=81
```


04、Open Interpreter快速体验

> 帮我读取一下当前环境下的 WA_Fn-UseC_-Telco-Customer-Churn.csv 文件

```
import pandas as pd

# Load the data
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
# Show the first 5 rows of the dataframe
df.head()
```



Would you like to run this code? (y/n)

```
import pandas as pd

file_path = 'WA_Fn-UseC_-Telco-Customer-Churn.csv'
df = pd.read_csv(file_path)
df.head()
```



gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	Streaming
Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	
Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	
Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	
Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	
Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	

这个数据框包含了关于电信公司客户流失的数据。每一行都代表了一个客户，每一列代表了一个属性或特性。以下是每个列的详细解释：

- 1 customerID - 客户的ID
- 2 gender - 客户的性别 (Male, Female)
- 3 SeniorCitizen - 客户是否是老年人 (1 = 是, 0 = 不是)
- 4 Partner - 客户是否有伙伴 (Yes, No)
- 5 Dependents - 客户是否有依赖 (Yes, No)
- 6 tenure - 客户在公司上网的月数
- 7 PhoneService - 客户是否有电话服务 (Yes, No)
- 8 MultipleLines - 客户是否有多线服务 (Yes, No, No phone service)
- 9 InternetService - 客户的互联网服务提供商 (DSL, Fiber optic, No)
- 10 OnlineSecurity - 客户是否有在线安全服务 (Yes, No, No internet service)
- 11 OnlineBackup - 客户是否有在线备份服务 (Yes, No, No internet service)



04、Open Interpreter快速体验总结

轻量级，方便

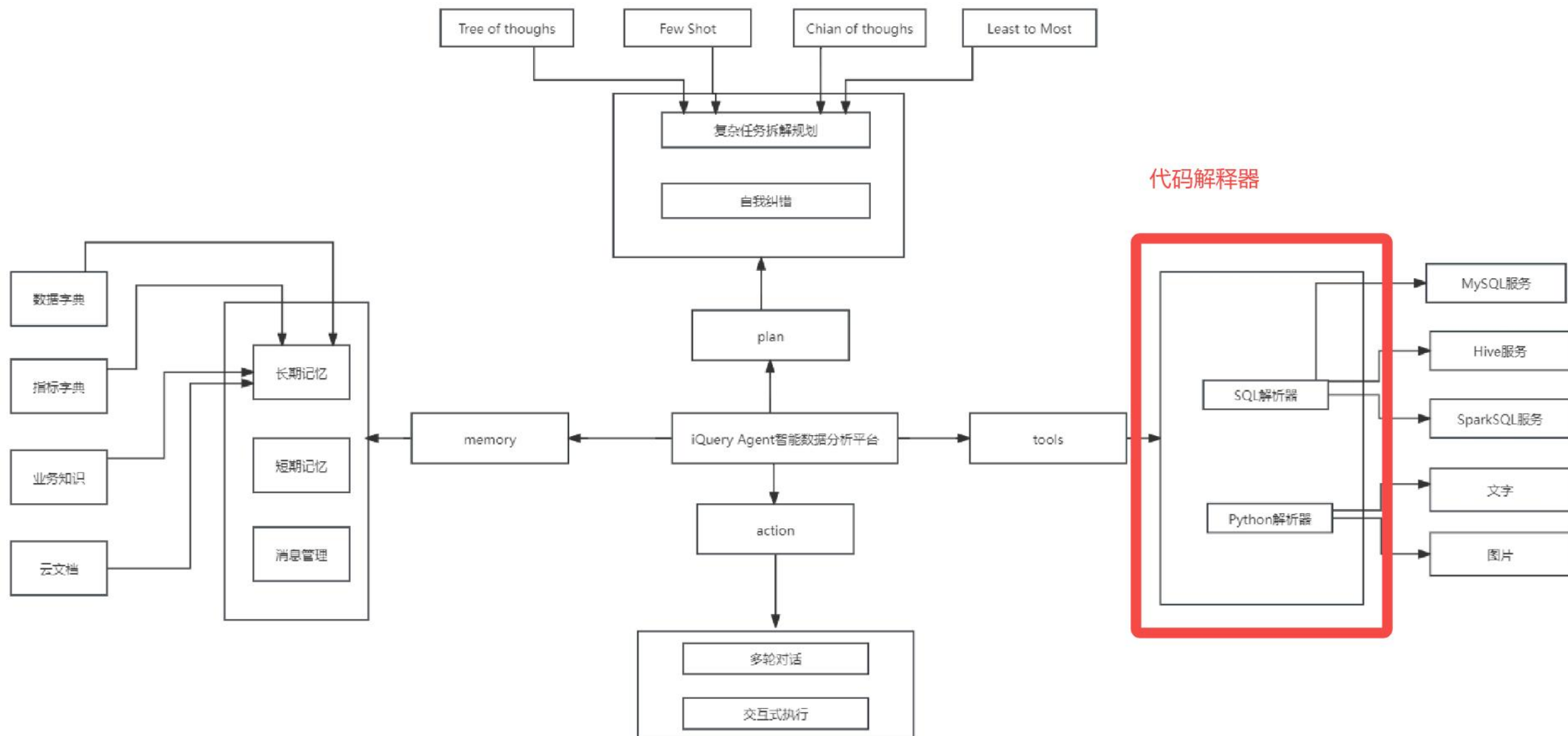
不稳定

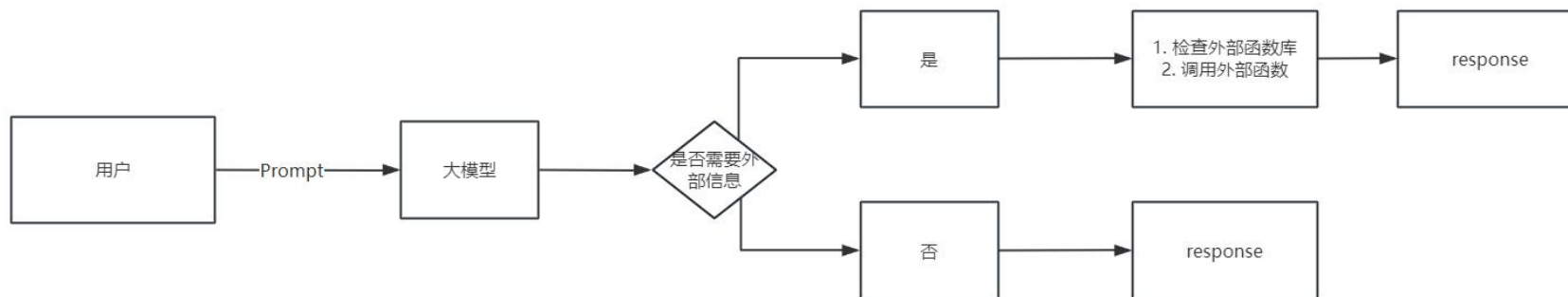
交互体验较好

复杂任务拆解能力弱

技术调研

01、技术调研之Function Calling





```
def run_conversation():
    # Step 1: send the conversation and available functions to the model
    messages = [{"role": "user", "content": "What's the weather like in San Francisco"}]
    tools = [
        {
            "type": "function",
            "function": {
                "name": "get_current_weather",
                "description": "Get the current weather in a given location",
                "parameters": {
                    "type": "object",
                    "properties": {
                        "location": {
                            "type": "string",
                            "description": "The city and state, e.g. San Francisco, CA",
                        },
                        "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]}
                    },
                    "required": ["location"],
                }
            }
        }
    ]
    response = client.chat.completions.create(
        model="gpt-3.5-turbo-0125",
        messages=messages,
        tools=tools,
        tool_choice="auto", # auto is default, but we'll be explicit
    )
```

```
from openai import OpenAI
import json

client = OpenAI()

# Example dummy function hard coded to return the same weather
# In production, this could be your backend API or an external API
def get_current_weather(location, unit="fahrenheit"):
    """Get the current weather in a given location"""
    if "tokyo" in location.lower():
        return json.dumps({"location": "Tokyo", "temperature": "10", "unit": unit})
    elif "san francisco" in location.lower():
        return json.dumps({"location": "San Francisco", "temperature": "72", "unit": unit})
    elif "paris" in location.lower():
        return json.dumps({"location": "Paris", "temperature": "22", "unit": unit})
    else:
        return json.dumps({"location": location, "temperature": "unknown"})
```

虽然大语言模型的知识储量巨大，且具备非常强大的涌现能力，但很多时候我们实际使用大语言模型时仍然会明显的感受到模型能力上的局限，例如模型无法获取最新的信息、模型只能给出文字的建议但无法直接帮我们解决某些问题（如查询数据库，查询商品价格等）

03、Function Calling实战演练--特殊语法

在Python中，双星号（**）操作符用来将字典中的键-值对解包为函数参数。

例如，假设有一个函数定义如下：

```
python 复制代码

def greet(first_name, last_name):
    print(f'Hello {first_name} {last_name}')
```

你可以通过以下方式来调用这个函数：

```
python 复制代码

greet('John', 'Doe')
```

但是，如果你的数据在字典中，你可以使用**来解包它并传递给函数：

```
python 复制代码

name_dict = {'first_name': 'John', 'last_name': 'Doe'}
greet(**name_dict) # 输出: Hello John Doe
```

这里，`**name_dict` 操作将字典解包为 `first_name='John', last_name='Doe'`，并传递给了 `greet` 函数。

如果字典中的键和函数的形参名不匹配，或者缺少必需的键，那么使用**解包会引发TypeError。

04、Function Calling实战演练--自动编写function定义

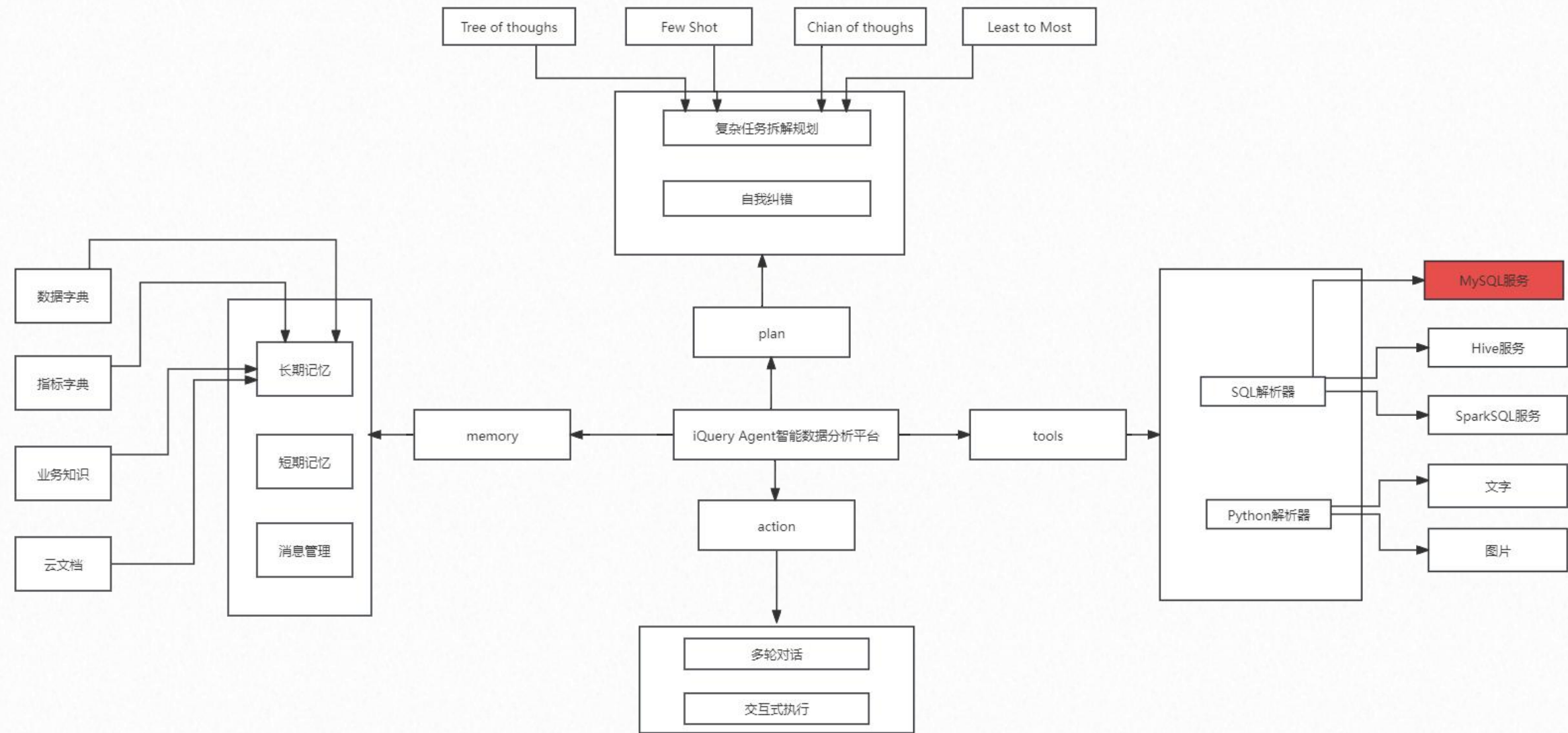
```
def get_current_weather(location, unit="fahrenheit"):
    """Get the current weather in a given location"""
    if "tokyo" in location.lower():
        return json.dumps({"location": "Tokyo", "temperature": "10", "unit": unit})
    elif "san francisco" in location.lower():
        return json.dumps({"location": "San Francisco", "temperature": "72", "unit": unit})
    elif "paris" in location.lower():
        return json.dumps({"location": "Paris", "temperature": "22", "unit": unit})
    else:
        return json.dumps({"location": location, "temperature": "unknown"})

def run_conversation():
    # Step 1: send the conversation and available functions to the model
    messages = [{"role": "user", "content": "What's the weather like in San Francisco"}]
    tools = [
        {
            "type": "function",
            "function": {
                "name": "get_current_weather",
                "description": "Get the current weather in a given location",
                "parameters": {
                    "type": "object",
                    "properties": {
                        "location": {
                            "type": "string",
                            "description": "The city and state, e.g. San Francisco, CA",
                        },
                        "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]}
                    },
                    "required": ["location"],
                },
            },
        },
    ]

    response = client.chat.completions.create(
        model="gpt-3.5-turbo-0125",
        messages=messages,
        tools=tools,
        tool_choice="auto", # auto is default, but we'll be explicit
    )
```


5 百亿级数据分析平台代码落地

01、数据准备之MySQL服务



02、数据准备之安装数据库

安装MySQL数据库

步骤1: 安装MySQL数据库

```
sudo apt-get update
```

```
sudo apt-get install mysql-server
```

```
sudo service mysql start
```

步骤2: 创建数据库用户

```
CREATE USER 'iquery_agent'@'localhost' IDENTIFIED BY 'iquery_agent';
```

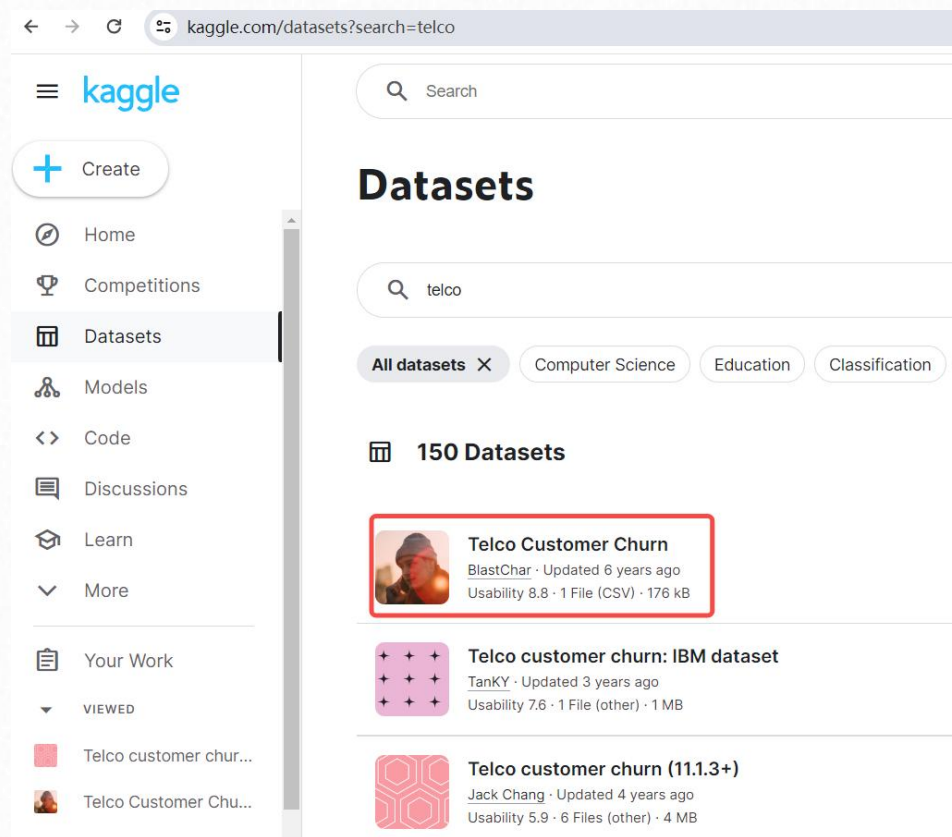
步骤3: 给数据库用户赋权限

```
GRANT ALL PRIVILEGES ON *.* TO 'iquery_agent'@'localhost';
```

```
FLUSH PRIVILEGES;
```

数据集获取

地址: <https://www.kaggle.com/datasets>



03、实现可控的多轮对话效果

1.user_demographics数据表

- 基本解释
user_demographics数据表记录了电信用户的个人基本情况，主要涵盖客户基本生物属性，包括性别、年龄状况、是否结婚以及是否经济独立等。
- 数据来源
user_demographics数据集由一线业务人员人工采集记录，并且通过回访确认相关信息，数据集的准确性和可信度都非常高。
- 各字段说明

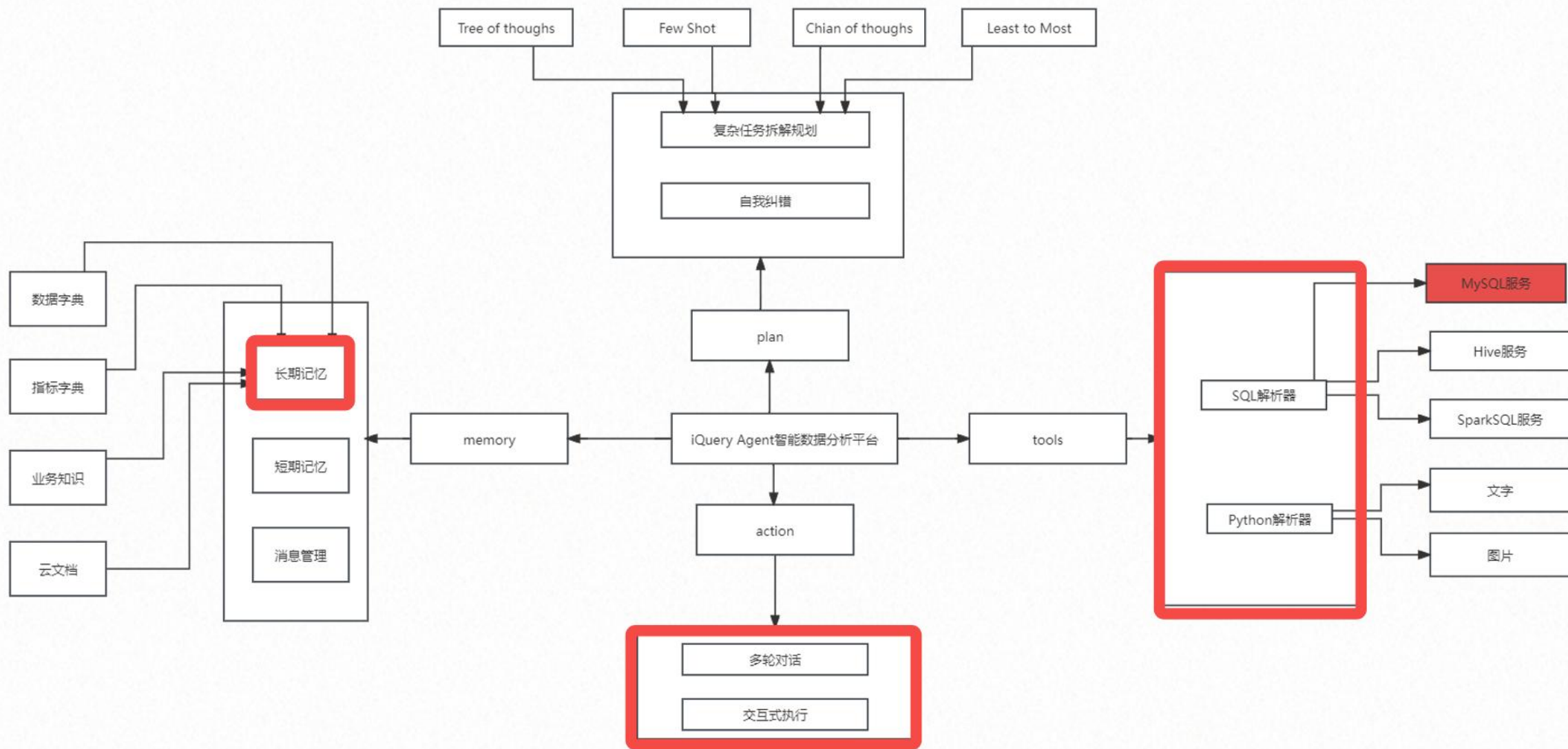
Column Name	Description	Value Range	Value Explanation	Type
customerID	客户ID， user_demographics数据表主键		由数字和字母组成的	VARCHAR(255)
gender	用户的性别	Female, Male	Female (女性), Male (男性)	VARCHAR(255)
SeniorCitizen	是否为老人	0, 1	0 (不是), 1 (是)	INT
Partner	用户是否有伴侣	Yes, No	Yes (有), No (没有)	VARCHAR(255)
Dependents	用户经济是否独立， 往往用于判断用户是否已经成年	No, Yes	Yes (有), No (没有)	VARCHAR(255)

添加数据字典

实现代码执行确认

实现多轮对话

04、实现数据分析报告功能



05、实现数据分析报告功能

数据分析报告之本地云盘创建

数据分析报告之报告展示

数据分析报告之本地Python代码解释器实现

数据分析报告之可视化代码逻辑封装

数据分析报告之整体功能初体验

06、数据分析业务知识

2024/3/14 20:18:18



完整的数据分析流程包含哪些模块？



2024/3/14 20:18:18

一个完整的数据分析流程通常包括以下模块：

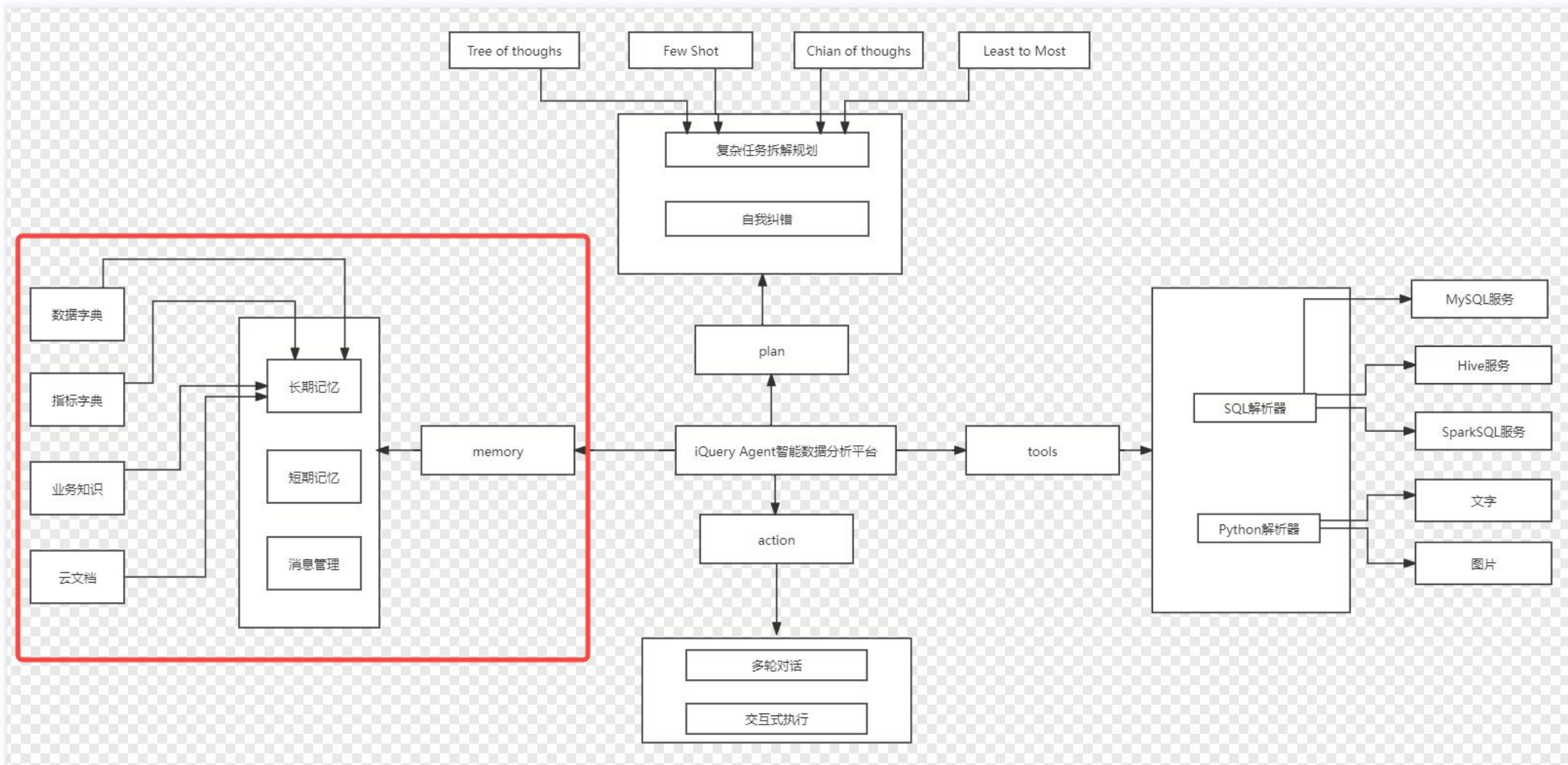
1. **问题定义**：明确要回答的问题或达到的目标。
2. **数据获取**：找到并收集需要的数据。数据可以来自多种来源，比如数据库，网络，第三方API等。
3. **数据清洗和预处理**：去除无用、重复、错误的数据，处理缺失值，可能还需要进行数据类型转换、数据归一化等操作。
4. **数据探索和理解**：通过统计分析，可视化等方式对数据进行深入理解，找到数据的规律和特点。
5. **数据建模**：选择合适的算法和模型，如回归分析，决策树，神经网络等，对数据进行建模。
6. **模型训练和评估**：使用训练数据集对模型进行训练，并用测试数据集进行模型评估，如准确率，召回率，F1值等。
7. **结果解释和报告**：将分析和模型结果进行整理和解释，形成报告，为决策提供支持。
8. **模型部署和监控**：将模型部署到实际业务中，并进行持续的监控和优化。

以上每个部分都是相互关联的，可能需要进行多次迭代才能达到满意的结果。

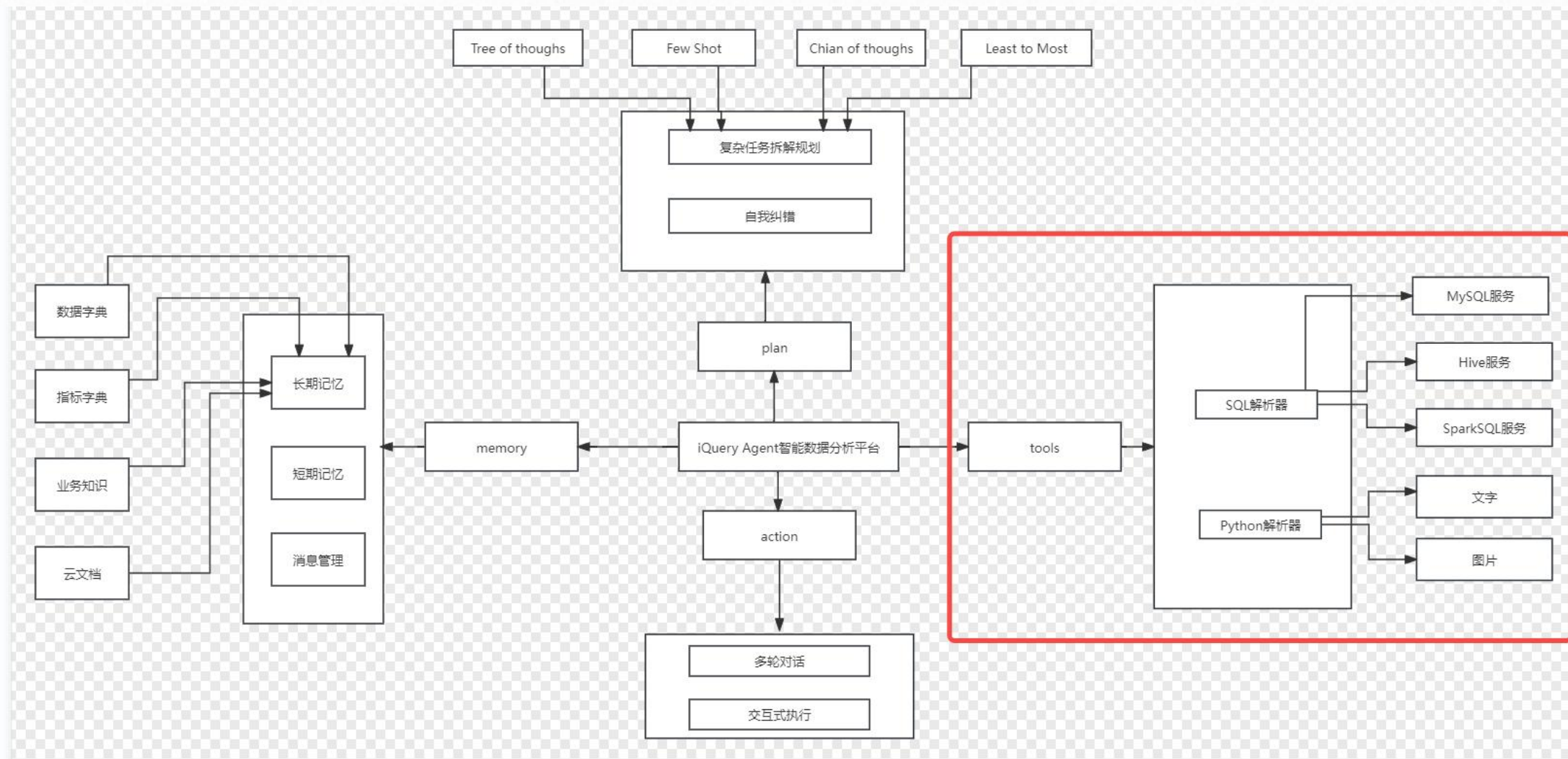


可通过微调的方式，让大模型学习这些知识

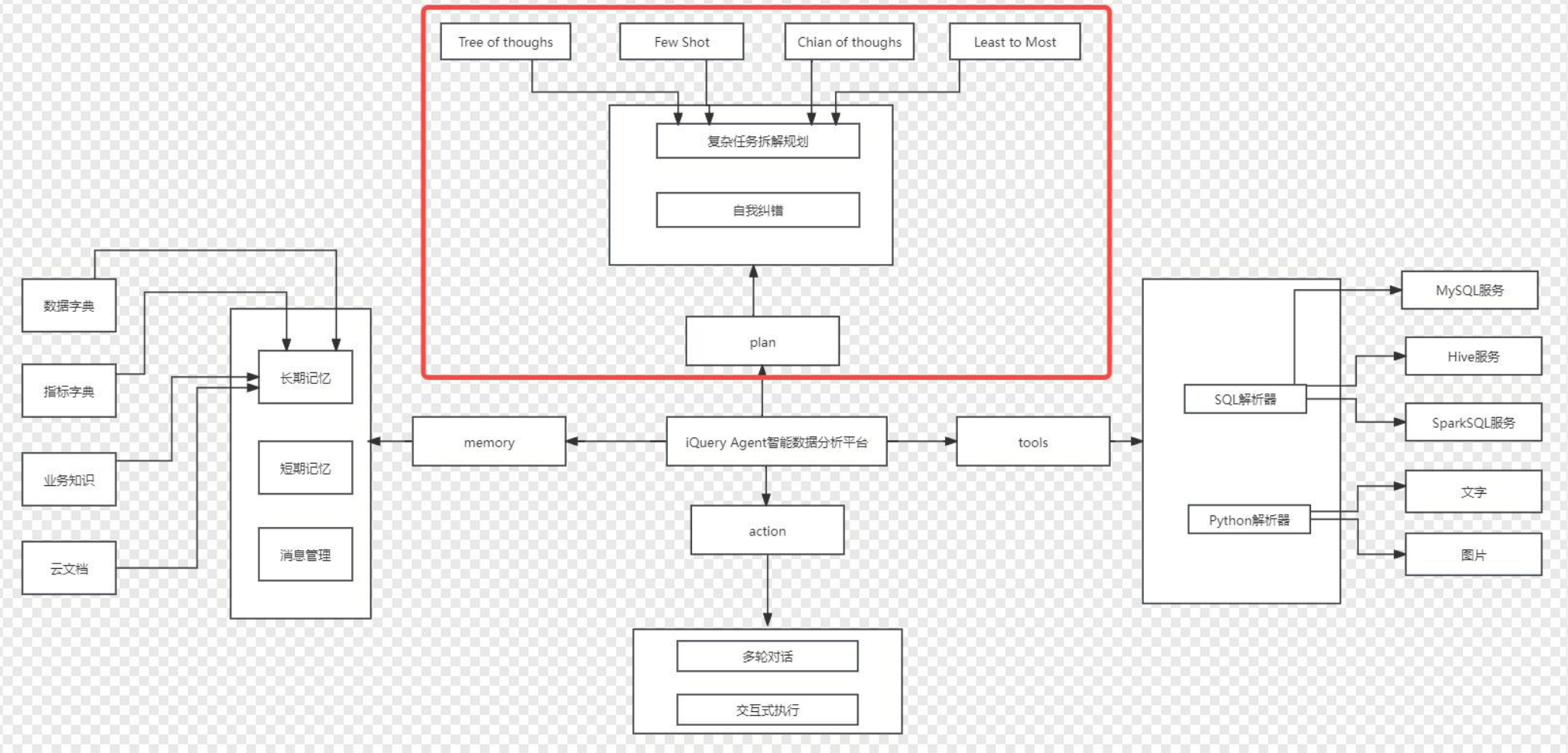
07、iQuery Agent落地



08、iQuery Agent落地



09、iQuery Agent落地



10、Planning之人类意图挖掘

```
-----
AuthenticationError                                Traceback (most recent call last)
Cell In[289], line 1
----> 1 msg_response3 = get_chat_response(model='gpt-3.5-turbo',
      2                                     messages=msg3,
      3                                     available_functions=af)

Cell In[271], line 32, in get_chat_response(model, messages, available_functions, is_developer_mode, is_enhanced_mode, delete_some_messages, is_task_decomposition)
    27 # 当且仅当围绕复杂任务拆解结果进行修改时, 才会出现is_task_decomposition=True的情况
    28 # 当is_task_decomposition=True时, 不再重新创建response_message
    29 if not is_task_decomposition:
    30     # 先获取单次大模型调用结果
    31     # 此时response_message是大模型调用返回的message
----> 32     response_message = get_gpt_response(model=model,
      33                                     messages=messages,
      34                                     available_functions=available_functions,
      35                                     is_developer_mode=is_developer_mode,
      36                                     is_enhanced_mode=is_enhanced_mode)
    38 # 复杂条件判断, 若is_task_decomposition = True,
    39 # 或者是增强模式且是执行function response任务时
    40 # (需要注意的是, 当is_task_decomposition = True时, 并不存在response_message对象)
    41 if is_task_decomposition or (is_enhanced_mode and response_message.tool_calls):
    42     # 将is_task_decomposition修改为True, 表示当前执行任务为复杂任务拆解

Cell In[242], line 40, in get_gpt_response(model, messages, available_functions, is_developer_mode, is_enhanced_mode)
    34     response = client.chat.completions.create(
    35         model=model,
    36         messages=messages.messages)
    38     # 若存在外部函数, 此时functions和function_call参数信息都从AvailableFunctions对象中获取
    39 else:
```

问的问题比较泛

key不稳定

剩余Token较少

低版本的大模型

AuthenticationError 是 OpenAI 的一个错误类, 在你试图使用无效的 API 密钥访问 API 等情况下可能会遇到这个错误。在 openai-python 库中的路径是 `openai.error.AuthenticationError`。

11、Planning之Few-shot修改提示词和增加COT

普通模式

开发者模式



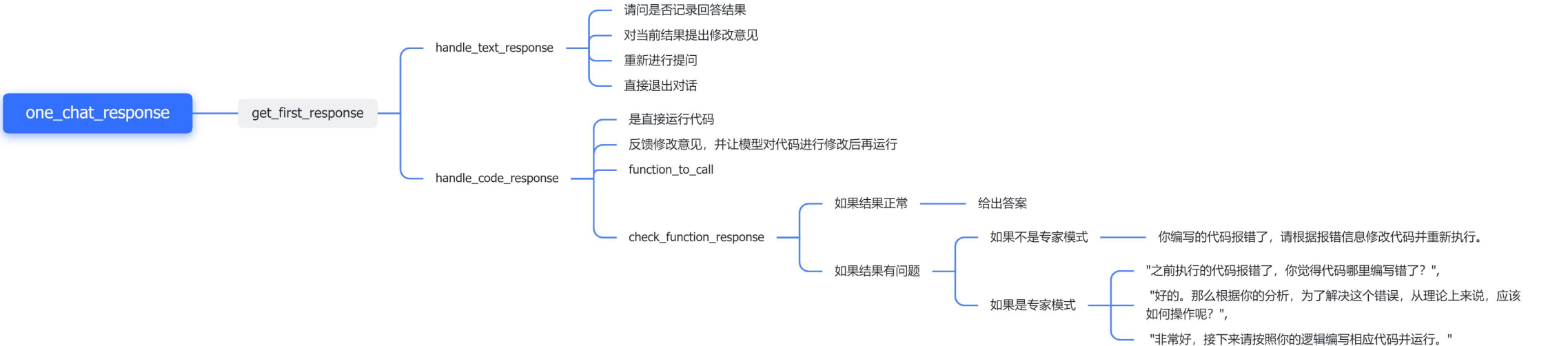
```
# 思考链提示词模板  
cot_prompt = "请一步步思考并得出结论。"
```

专家模式

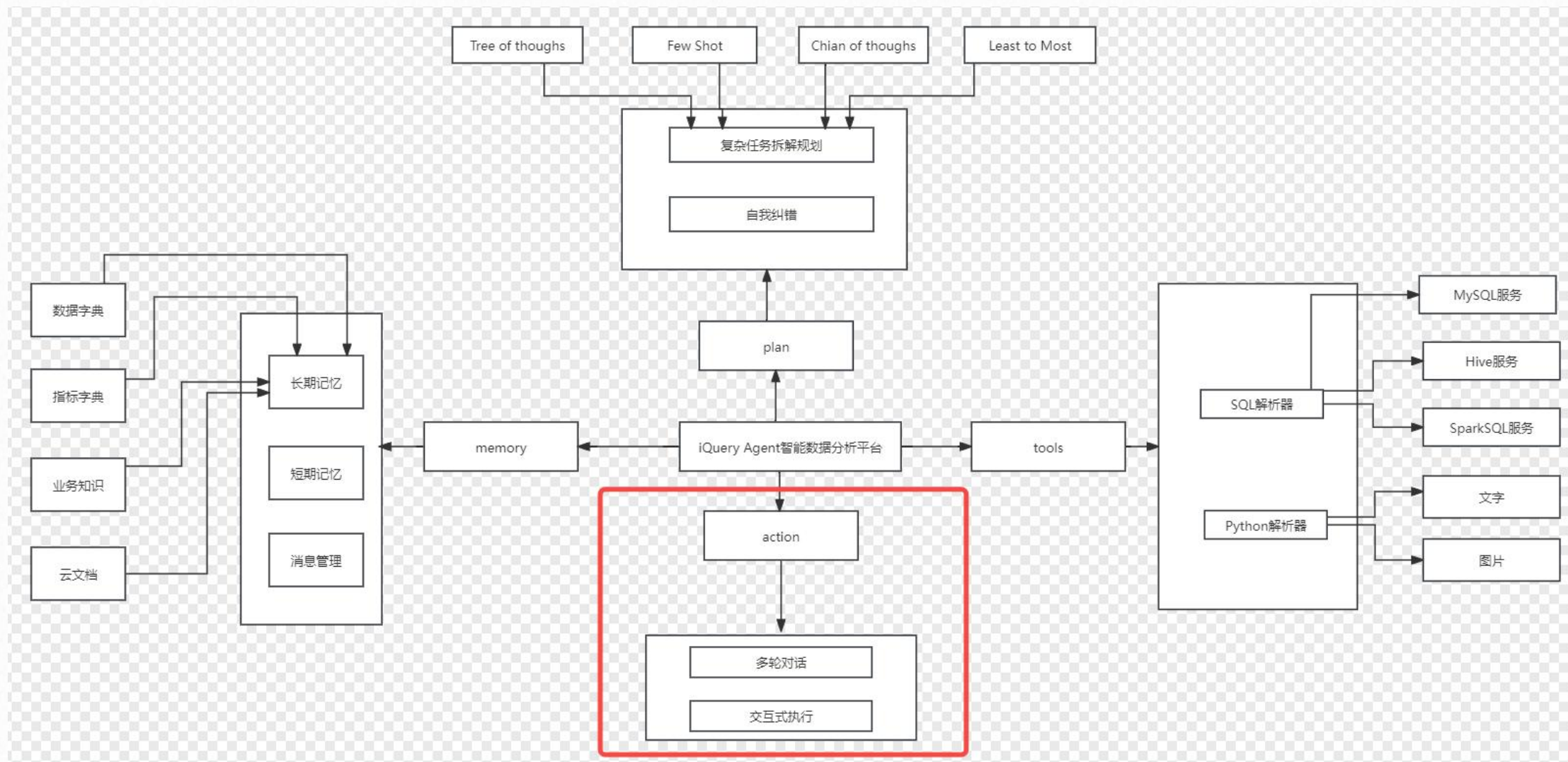


```
user_question3 = '围绕数据库中的user_payments表，我想要检查该表是否存在缺失值'  
user_message3_content = "现有用户问题如下：\"%s\"。为了回答这个问题，总共需要分几步来执行呢？\n若无需拆分执行步骤，请直接回答原始问题。" % user_question3  
assistant_message3_content = '为了检查user_payments数据集是否存在缺失值，我们将执行如下步骤：\n\n步骤1：使用`extract_data`函数将user_payments数据表读取到当前的Python环境中。\  
\n步骤2：使用`python_inter`函数执行Python代码检查数据集的缺失值。'
```

12、Planning之一次完整对话逻辑封装

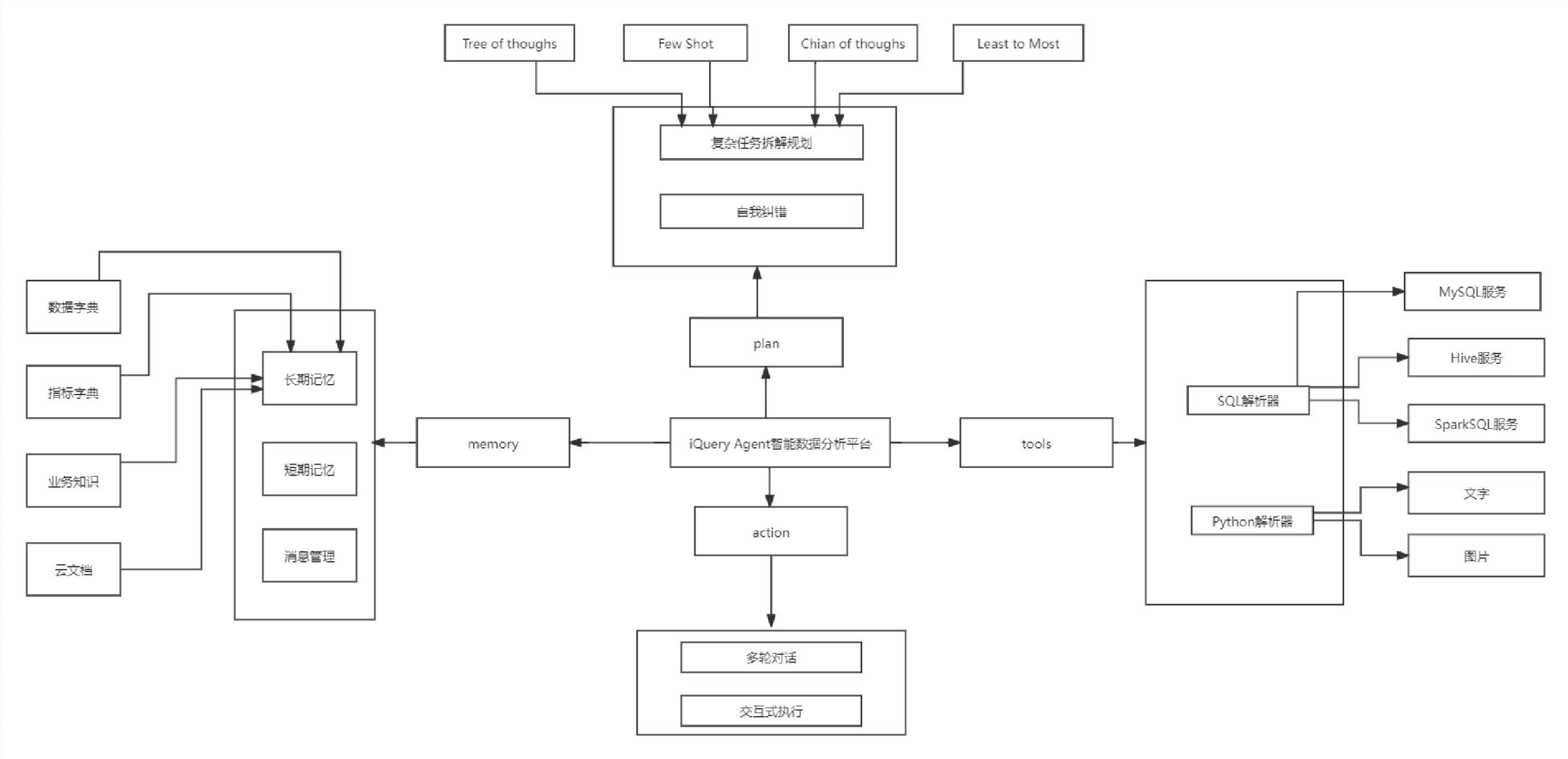


13、iQuery Agent落地之Action



⑥ 百亿级数据分析平台项目总结

01、项目总结

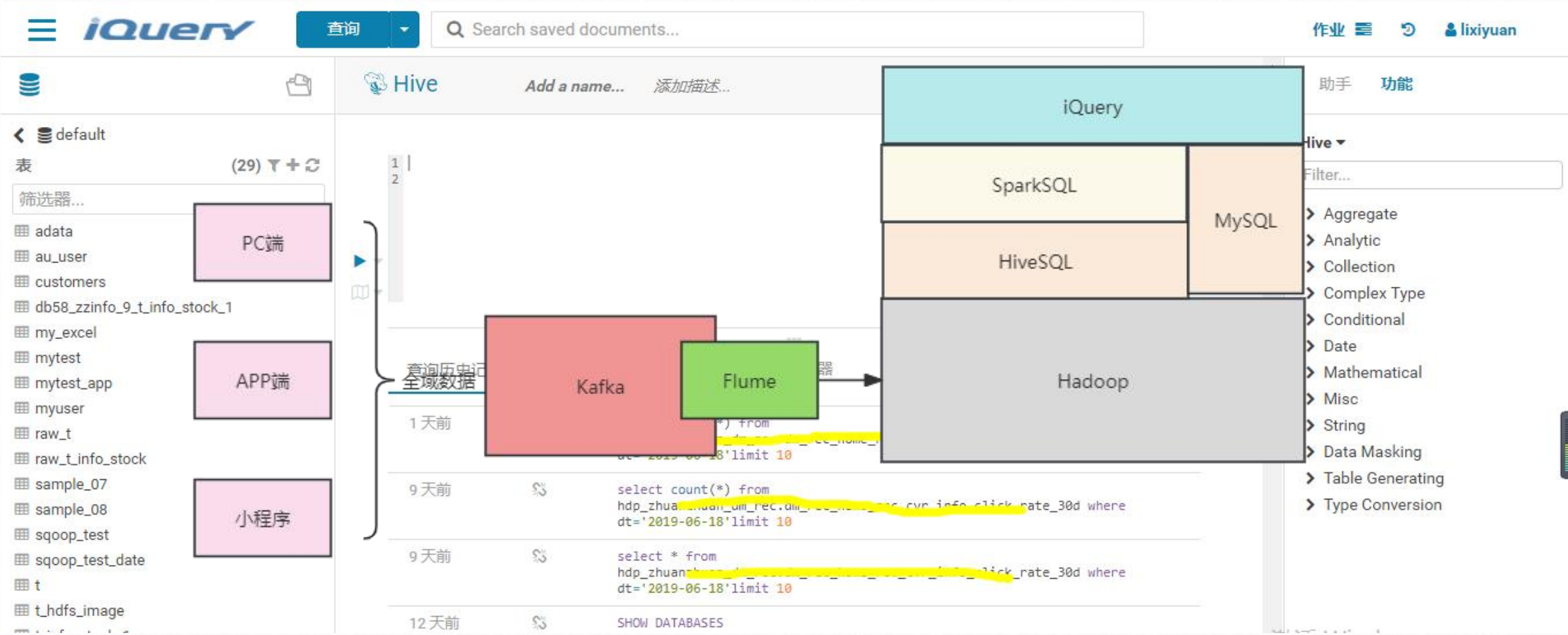


02、项目总结-支持HiveSQL

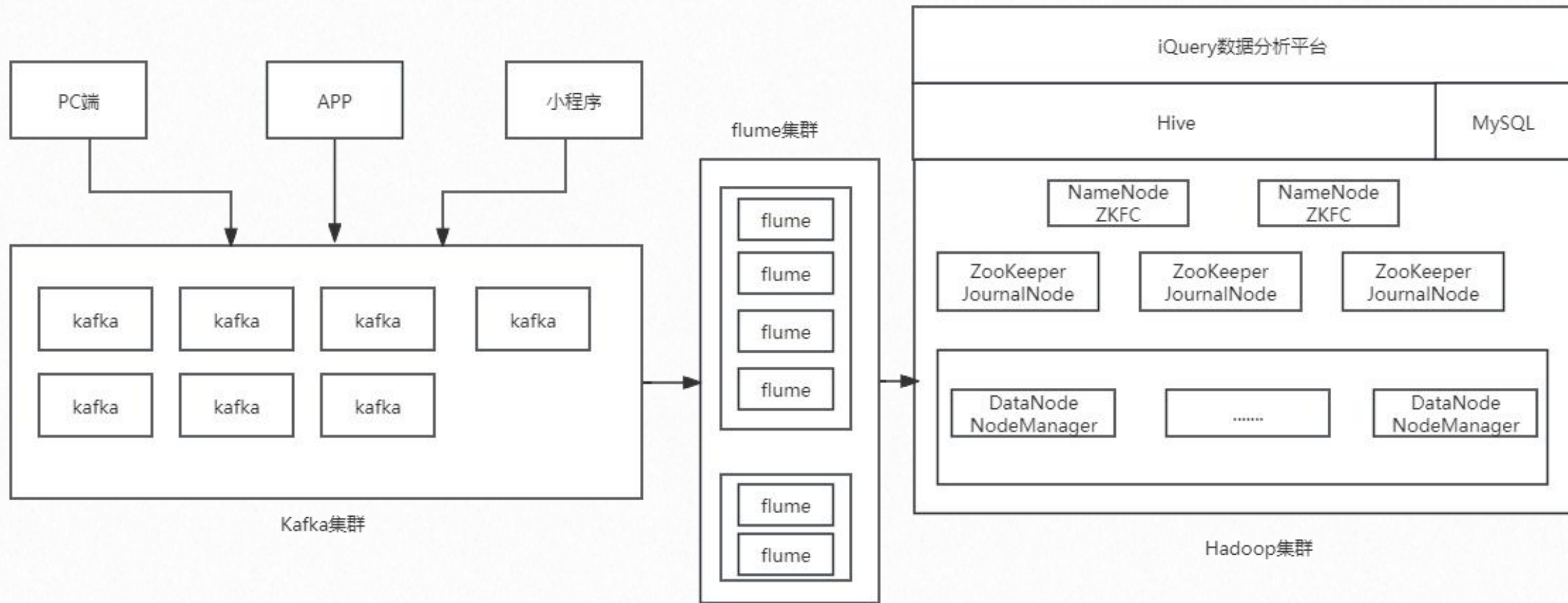
🔗 gpt-3.5-turbo ▾

```
def sql_inter(sql_query):  
    """  
    用于获取iquery数据库中各张表的有关相关信息，\  
    核心功能是将输入的SQL代码传输至iquery数据库所在的Hive环境中进行运行，\  
    并最终返回SQL代码运行结果。需要注意的是，本函数是借助pyhive来连接Hive数据库。  
    :param sql_query: 字符串形式的SQL查询语句，用于执行对Hive中iquery数据库中各张表进行查询，并获得各表中的各类相关信息  
    :return: sql_query在Hive中的运行结果。  
    """  
  
    connection = hive.Connection(  
        host='your_hive_host', # Hive数据库地址  
        port=10000, # 默认Hive端口  
        username='your_username', # Hive用户名  
        database='your_database' # Hive数据库名  
    )  
  
    try:  
        with connection.cursor() as cursor:  
            # SQL查询语句  
            sql = sql_query  
            cursor.execute(sql)  
  
            # 获取查询结果  
            results = cursor.fetchall()  
  
        finally:  
            connection.close()  
  
    return json.dumps(results)
```

03、项目总结-支持百亿级数据分析



04、百亿级数据分析平台大数据架构设计



05、百亿级数据分析平台大数据架构设计

Kafka集群资源评估

每天集群需要承载10亿数据请求，一天24小时，对于网站，晚上12点到凌晨8点这8个小时几乎没多少数据。使用二八法则估计，也就是80%的数据（8亿）会在其余16个小时涌入，而且8亿的80%的数据（6.4亿）会在这16个小时的20%时间（3小时）涌入。

QPS计算公式= $640000000 \div (3 \times 60 \times 60) = 6$ 万，
故高峰期集群需要要抗住每秒6万的并发

每天10亿数据，每个请求10kb，也就是9T的数据。如果保存3副本， $9 \times 3 = 27$ T，保留最近5天的数据。故需要 $27 \times 5 = 135$ T

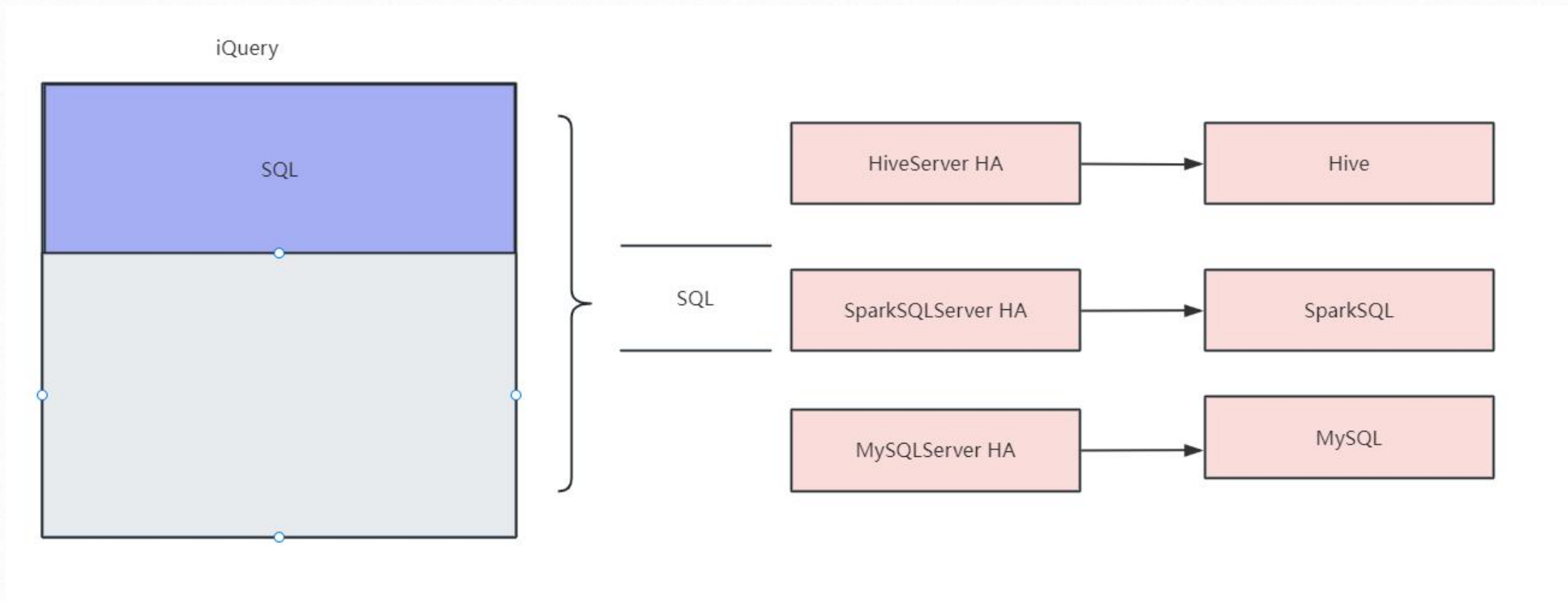
如果资源充足，让高峰期QPS控制在集群能承载的总QPS的30%左右，故目集群能承载的总QPS为20万左右才是安全的，根据经验一台物理机能支持4万QPS是没问题的，所以从QPS的角度讲，需要物理机5台。再考虑上消费者请求，增加1.5倍，需要7-8台服务器。

10亿写请求，6w/s的吞吐量，9T的数据，7台物理机
硬盘：11（SAS） * 1T，7200转
内存：128GB，JVM分配6G，剩余的给os cache
CPU：16核/32核
网络：万兆网卡更佳

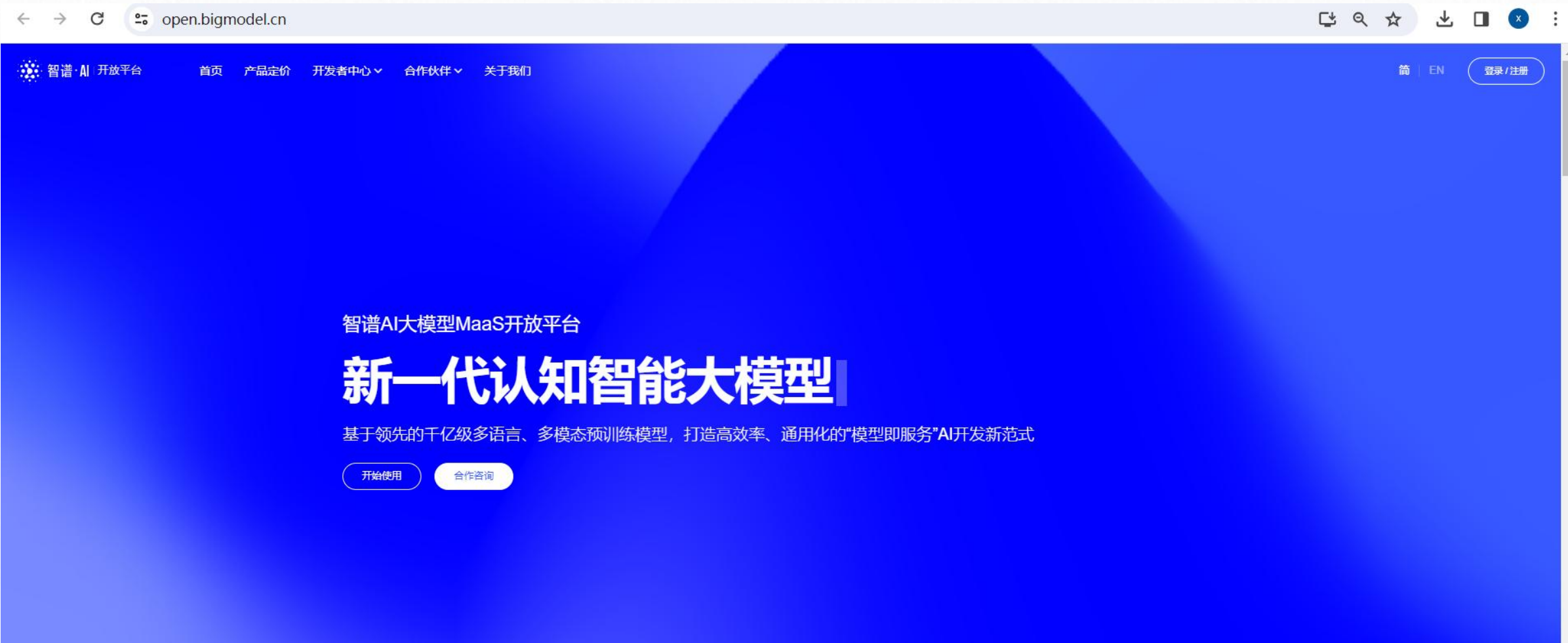
Hadoop集群资源评估

1. 每天9T，副本数为3，一年需要的存储资源： $9 \times 3 \times 365 = 9855$ T
 2. 数据需要进行加工（建模）： $9855 \times 2.5 = 24637$ T
 3. 数据增速是每年50%， $24637 \times (1.5) = 36956$ T
 4. 磁盘只能存到80%，故需要46195T的存储空间
 5. 压缩比，按50%估算，故需要存储23076T
- 机器配置：32cpu core, 128G内存，11 * 7T
故： $27717 / 77 = 300$ 台服务器

06、百亿级数据分析平台设计



07、项目总结-支持GLM4大模型



大家辛苦了

关注视频号：玄姐谈AGI
助力数字化人才提升 AIGC 能力



玄姐谈 AGI



扫一扫二维码，关注我的视频号