

数据库实践实验报告

实验四：自然语言与数据库交互

姓名：邱吉尔 学号：10235101533

完成日期：2025 年 5 月 9 日

一、实验目的

通过开源项目 DB-GPT 和外部大模型，尝试利用大模型进行数据管理和数据库交互，完成作业任务。

二、任务说明

- 使用第一次实验提供的 college 数据库测试 LLM 的 SQL 编写与处理能力。
- 报告需要记录：
 - 项目搭建与运行过程；
 - 至少 10 个通过 LLM 进行数据库交互的对话示例，建议场景具有复杂性、趣味性和连续多轮对话特点。

三、环境搭建过程

1. 克隆 DB-GPT 项目代码：

```
1 git clone https://github.com/eosphoros-ai/DB-GPT.git
```

直接使用此网址 git 到本地因为会网络连接不稳定或 SSL 问题报错：

error: RPC failed; curl 56 OpenSSL SSL_read: OpenSSL/3.2.3: error:0A000119:SSL

遂使用 SSH 进行 clone：git clone git@github.com:eosphoros-ai/dbgpts.git

2. 创建并激活 Conda 虚拟环境：

安装 Anaconda 后尝试按照所给 docx 文档配置了两步，发现 DB-GPT 已经更新至 0.7.0 版本，项目文件与 docx 文件中有所不同，故后面的配置步骤皆是按照官方文档进行，与 docx 文件的步骤有所不同

3. 安装 uv：

因为 DB-GPT 是 monorepo 架构，相较于 pip，uv 支持并行依赖安装，可自动关联项目目录和环境并且内置 uv venv，可一键创建虚拟环境，因此使用 uv 进行包管理在 powershell 中运行：

```
1 pip install uv
```

然后将 uv 所在目录添加到环境变量中

4. 依赖安装：

```
1 uv sync --all-packages --frozen --extra "base" --extra "proxy_openai" --extra "rag"
  --extra "storage_chromadb" --extra "dbgpts"
```

5. 进入虚拟环境：

在终端中运行

```
1 .venv/Scripts/activate
```

即可进入虚拟环境

6. 下载所需依赖：

执行

```
1 pip install dashscope
```

7. 申请通义千问的使用 API：

按照 docx 文档一步步来即可成功申请，这里就不过多赘述了

8. 修改配置文件：

- (a) 找到 configs 目录下的 dbgpt-proxy-tongyi.toml 文件，配置通义 api 调用，可以在该文件下找到此代码：

```
1 api_key = "${env:DASHSCOPE_API_KEY}"
```

因而在用户环境变量中新增变量，变量名为：DASHSCOPE_API_KEY，变量值即是上一步申请到的通义 API

- (b) 接着可以看到文件中还有如下两条代码：

```
1 provider = "${env:LLM_MODEL_PROVIDER:proxy/tongyi}"
2 provider = "${env:EMBEDDING_MODEL_PROVIDER:proxy/tongyi}"
```

我先是像上一步一样配置了环境变量，却在启动项目时发现了报错：找不到相应环境变量，后来搜索发现 `${env:LLM_MODEL_PROVIDER:proxy/tongyi}` 并不是标准 TOML 语法，而是一些框架扩展支持的一种“环境变量注入 + 默认值”的写法

因而采取了另一种简单粗暴的方法：直接写死 provider 的值，将上述代码改成如下代码：

```
1 provider = "proxy/tongyi"
2 provider = "proxy/tongyi"
```

9. 启动服务：

```
1 uv run python packages/dbgpt-app/src/dbgpt_app/dbgpt_server.py --config
  configs/dbgpt-proxy-tongyi.toml
```

如果想略过项目配置检查环节，加快启动速度，可以运行：

```
1 uv run --no-project python packages/dbgpt-app/src/dbgpt_app/dbgpt_server.py --config
  configs/dbgpt-proxy-tongyi.toml
```

10. 打开浏览器访问项目主页：<http://localhost:5670>

11. 配置 SQL，将模型与数据库连接，这里我用了第一次实验用到的 college 数据库：

添加数据源

* 数据库类型

MySQL

* host ?

127.0.0.1

* port ?

3306

* user ?

root

* database ?

college

driver ?

mysql+pymysql

password ?

.....

pool_size ?

5

max_overflow ?

10

pool_timeout ?

30

pool_recycle ?

3600

pool_pre_ping ?

☒

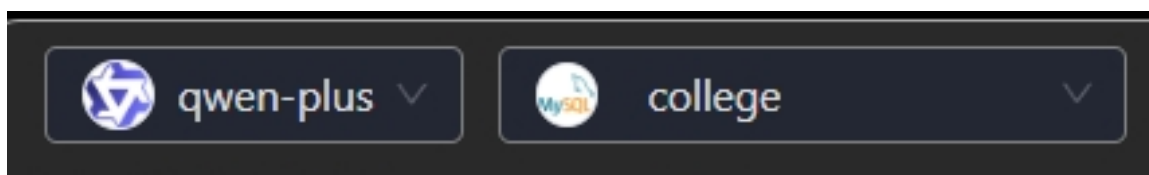
描述

请输入描述

取消

提交

发现 Chat Data 出现如下两个内容，说明已经配置完成：



四、测试问题示例

1. 查询所有有导师的学生数量。
2. 查询所有学生的姓名和所在系名。
3. 找出所有学分大于等于 3 的课程。
4. 查询所有学生的 ID 及其所选课程总数。
5. 查询开设在 “Comp. Sci.” 系的所有课程编号和课程名。
6. 统计每个系开设的课程数量。
7. 找出修了课程数量最多的学生的学号和姓名。
8. 找出所有没有选修任何课程的学生。
9. 查询每门课的选课人数，按人数降序排列。
10. 查询每个学生的平均成绩。
11. 找出所有教授 “数据库系统概论 (Database System Concepts)” 课程的教师姓名。
12. 找出所有课程成绩为全班最高分的学生姓名和成绩。
13. 查询所有与 id 为 1000 的学生选修课程相同的其他学生姓名。
14. 找出至少选修了 id 为 1000 的学生所选所有课程的学生（集合包含）。
15. 找出至少教授过两个不同课程的教师姓名。

六、实验过程截图与 SQL 对话样例

1. 查询所有有导师的学生数量。



最终执行结果为 2000，与实验结果一致

2. 查询所有学生的姓名和所在系名。

dbgpt 查询所有学生的姓名和所在系名。

用户需要查询所有学生的姓名和所在系名，可以通过student表中的name和dept_name字段获取信息。

Data SQL

```
SELECT
  `name`,
  `dept_name`
FROM
  `student`
LIMIT
  50;
```

用户需要查询所有学生的姓名和所在系名，可以通过student表中的name和dept_name字段获取信息。

Data SQL

name	dept_name
Manber	Civil Eng.
Zelty	Mech. Eng.
Duan	Civil Eng.
Colin	Civil Eng.
Mediratta	Geology
Rzecz	Comp. Sci.
Hilberg	Psychology
Ugarte	Pol. Sci.
Grosch	Astronomy
Kieras	Physics

< 1 2 3 4 5 >

3. 找出所有学分大于等于 3 的课程。

dbgp

找出所有学分大于等于3的课程。



用户需要查询学分大于等于3的课程，可以通过course表中的credits字段进行筛选。

Data

SQL

```
SELECT
  *
FROM
  `course`
WHERE
  `credits` >= 3
LIMIT
  50;
```



用户需要查询学分大于等于3的课程，可以通过course表中的credits字段进行筛选。

Data

SQL

course_id	title	dept_name	credits
101	Diffusion and Phase Transformation	Mech. Eng.	3
105	Image Processing	Astronomy	3
123	Differential Equations	Mech. Eng.	3
127	Thermodynamics	Geology	3
130	Differential Geometry	Physics	3
133	Antidisestablishmentarianism in Modern America	Biology	4
137	Manufacturing	Finance	3
139	Number Theory	English	4
158	Elastic Structures	Cybernetics	3
169	Marine Mammals	Elec. Eng.	3

<

1

2

3

4

5

>

4. 查询所有学生的 ID 及其所选课程总数。

dbgpt 查询所有学生的ID及其所选课程总数。

用户需要查询所有学生的ID及其所选课程总数，可以通过统计takes表中每个学生ID的记录数来实现。

Data SQL

```
SELECT
  `ID`,
  COUNT(`course_id`) AS total_courses
FROM
  `takes`
GROUP BY
  `ID`
LIMIT
  50;
```

用户需要查询所有学生的ID及其所选课程总数，可以通过统计takes表中每个学生ID的记录数来实现。

Data SQL

ID	total_courses
1000	13
10033	22
10076	14
1018	22
10204	12
10267	11
10269	10
10454	18
10481	17
10527	21

< 1 2 3 4 5 >

5. 查询开设在“Comp. Sci.”系的所有课程编号和课程名。

dbgpt 查询开设在“Comp. Sci.”系的所有课程编号和课程名

用户需要查询开设在“Comp. Sci.”系的所有课程编号和课程名，可以通过course表中的dept_name字段进行筛选。

Data SQL

```
SELECT
  `course_id`,
  `title`
FROM
  `course`
WHERE
  `dept_name` = 'Comp. Sci.'
LIMIT
  50;
```

Data SQL


course_id	title
274	Corporate Law
276	Game Design
284	Topology
359	Game Programming
539	International Finance
571	Plastics
584	Computability Theory
647	Service-Oriented Architectures
747	International Practicum
949	Japanese

< 1 >

6. 统计每个系开设的课程数量。

dbgpt

统计每个系开设的课程数量




用户需要统计每个系开设的课程数量，可以通过course表中的dept_name字段进行分组并计数。

Data

SQL

```
SELECT
  `dept_name`,
  COUNT(`course_id`) AS course_count
FROM
  `course`
GROUP BY
  `dept_name`
LIMIT
  50;
```



用户需要统计每个系开设的课程数量，可以通过course表中的dept_name字段进行分组并计数。

Data

SQL

dept_name	course_count
Accounting	12
Astronomy	10
Athletics	9
Biology	7
Civil Eng.	10
Comp. Sci.	10
Cybernetics	20
Elec. Eng.	8
English	8
Finance	14

<

1

2

>

7. 找出修了课程数量最多的学生的学号和姓名。

dbgpt 找出修了课程数量最多的学生的学号和姓名。

用户需要找出修了课程数量最多的学生的学号和姓名。可以通过统计takes表中每个学生ID的记录数，并与student表进行连接以获取学生的姓名。

Data SQL

```
SELECT
  s.ID,
  s.name,
  COUNT(t.course_id) AS course_count
FROM
  student s
  JOIN takes t ON s.ID = t.ID
GROUP BY
  s.ID,
  s.name
ORDER BY
  course_count DESC
LIMIT
  1;
```

Data SQL

ID	name	course_count
12078	Knutson	27

< 1 >

8. 找出所有没有选修任何课程的学生。

dbgpt 找出所有没有选修任何课程的学生。

用户需要找出所有没有选修任何课程的学生。可以通过查找在student表中存在但不在takes表中的学生ID来实现。

Data SQL

```
SELECT
  s.ID,
  s.name
FROM
  student s
WHERE
  s.ID NOT IN(
    SELECT DISTINCT
      t.ID
    FROM
      takes t
  )
LIMIT
  50;
```



9. 查询每门课的选课人数，按人数降序排列。

dbgpt 查询每门课的选课人数，按人数降序排列。

用户需要查询每门课的选课人数，并按人数降序排列。可以通过统计takes表中每个course_id的记录数来实现。

Data SQL

```
SELECT
  course_id,
  COUNT(ID) AS enrollment_count
FROM
  takes
GROUP BY
  course_id
ORDER BY
  enrollment_count DESC
LIMIT
  50;
```

用户需要查询每门课的选课人数，并按人数降序排列。可以通过统计takes表中每个course_id的记录数来实现。

Data SQL

course_id	enrollment_count
362	980
105	634
867	626
237	613
468	610
400	610
200	591
408	591
960	590
735	585

< 1 2 3 4 5 >

10. 查询每个学生的平均成绩。

查询每个学生的平均成绩(gpa)，要按照学分权重来计算

用户需要查询每个学生的平均成绩(gpa)，并且要按照学分权重来计算。这需要结合takes表中的grade字段，gpa表中的grade_point字段以及course表中的credits字段。

Data [SQL](#)

```
SELECT
  t.ID,
  SUM(g.grade_point * c.credits) / SUM(c.credits) AS weighted_avg_gpa
FROM
  takes t
  JOIN gpa g ON t.grade = g.grade
  JOIN course c ON t.course_id = c.course_id
GROUP BY
  t.ID
LIMIT
  50;
```

用户需要查询每个学生的平均成绩(gpa)，并且要按照学分权重来计算。这需要结合takes表中的grade字段，gpa表中的grade_point字段以及course表中的credits字段。

Data [SQL](#)

ID	weighted_avg_gpa
40558	2.670732
40481	2.998113
40457	2.868
40371	3.25
4034	3.21875
40303	2.693939
4015	3.194444
40116	3.597143
40080	3.325
40059	3.202857

< 1 2 3 4 5 >

11. 找出所有教授“数据库系统概论 (Database System Concepts)”课程的教师姓名。

dbgp 找出所有教授“数据库系统概论 (Database System Concepts)”课程的教师姓名，通过teaches和course表关联

用户需要找出所有教授“数据库系统概论 (Database System Concepts)”课程的教师姓名。这可以通过teaches表和course表进行关联，找到对应的课程，并通过instructor表获取教师姓名。

Data SQL

```
SELECT DISTINCT
  i.name
FROM
  teaches t
JOIN course c ON t.course_id = c.course_id
JOIN instructor i ON t.ID = i.ID
WHERE
  c.title = 'Database System Concepts'
LIMIT
  50;
```

Data SQL

暂无数据

验证后发现确实没有教授数据库的教师

12. 找出所有课程成绩为全班最高分的学生姓名和成绩。

dbgp 找出所有课程成绩为全班最高分的学生姓名和成绩，成绩排名从高到低是A, A-,B+,B-,C+,C-,D,D-,F

用户需要找出所有课程成绩为全班最高分的学生姓名和成绩，并按照成绩排名从高到低是A, A-,B+,B-,C+,C-,D,D-,F。这可以通过takes表找到每个课程的最高成绩，再与学生表关联获取学生姓名，同时利用gpa表来确定成绩排名。

Data SQL

```
SELECT
  s.name,
  t.grade
FROM
  takes t
JOIN (
  SELECT
    course_id,
    MAX(grade_point) AS max_grade_point
  FROM
    takes t
  JOIN gpa g ON t.grade = g.grade
  GROUP BY
    course_id
) mgp ON t.course_id = mgp.course_id
JOIN gpa g ON t.grade = g.grade
AND g.grade_point = mgp.max_grade_point
JOIN student s ON t.ID = s.ID
ORDER BY
  FIELD(
    t.grade,
    'A',
    'A-',
    'B+',
    'B-',
    'C+',
    'C-',
    'D+',
    'D-',
    'F'
  ) ASC
LIMIT
  50;
```

用户需要找出所有课程成绩为全班最高的学生姓名和成绩，并按照成绩排名从高到低是A-, A-, B+, B+, C+, C+, D, D-, F。这可以通过takes表找到每个课程的最高成绩，再与student表关联获取学生姓名，同时利用gpa表来确定成绩排名。

Data SQL

name	grade
Grosch	A+
Wingb	A+
Diab	A+
Ramaswamy	A+
Curutchet	A+
Visr	A+
Paniez	A+
Kinney	A+
Ä-zel	A+
Neuhold	A+

< 1 2 3 4 5 >

13. 查询所有与 id 为 1000 的学生选修课程相同的其他学生姓名。

Chat Data native_app chat_with_db_execute

查询所有与id为1000的学生选修课程相同的其他学生姓名,意思是选的每一门课程都为id1000的学生相同

用户需要查询所有与id为1000的学生选修课程完全相同的其他学生姓名。这意味着这些学生的选课列表应该和id为1000的学生完全一致，既不能多也不能少。

Data SQL

```
SELECT DISTINCT
s2.ID,
s2.name
FROM
student s2
WHERE
NOT EXISTS (
SELECT
t1.course_id
FROM
takes t1
WHERE
t1.ID = '1000'
AND NOT EXISTS (
SELECT
*
FROM
takes t2
WHERE
t2.ID = s2.ID
AND t2.course_id = t1.course_id
)
)
AND NOT EXISTS (
SELECT
t2.course_id
FROM
takes t2
WHERE
t2.ID = s2.ID
AND NOT EXISTS (
SELECT
*
FROM
takes t1
WHERE
t1.ID = '1000'
AND t1.course_id = t2.course_id
)
)
AND s2.ID != '1000';
```




14. 找出至少选修了 id 为 1000 的学生所选所有课程的学生。

dbgp 找出至少选修了id为1000的学生所选所有课程的学生，只要选的课中有一门被id为1000的学生选了即可

用户需要找出至少选修了id为1000的学生所选所有课程的学生，只要选的课中有一门被id为1000的学生选了即可。这可以通过比较takes表中course_id字段来实现。

Data SQL

```
SELECT DISTINCT
  s.ID,
  s.name
FROM
  student s
JOIN takes t ON s.ID = t.ID
WHERE
  t.course_id IN (
    SELECT
      course_id
    FROM
      takes
    WHERE
      ID = '1000'
  )
AND s.ID != '1000'
LIMIT
  50;
```

Data SQL

ID	name
1018	Colin
10481	Grosch
10556	Reed
10663	Okaf
108	Dhav
10917	Morales
11101	Glaho
11195	Shiang
11441	Betho
11510	Rakoj

< 1 2 3 4 5 >

15. 找出至少教授过两个不同课程的教师姓名。

The screenshot shows a database query interface with a dark theme. At the top, a prompt in Chinese asks to find the names of teachers who have taught at least two different courses. Below the prompt, there are tabs for 'Data' and 'SQL'. The 'SQL' tab is selected, displaying the following SQL query:

```
SELECT DISTINCT
  i.name
FROM
  instructor i
JOIN (
  SELECT
    ID
  FROM
    teaches
  GROUP BY
    ID
  HAVING
    COUNT(DISTINCT course_id) >= 2
) t ON i.ID = t.ID
LIMIT
  50;
```

Below the SQL query, the 'Data' tab is selected, showing a table with the results. The table has a single column named 'name' and lists the following names:

name
Lembr
Wieland
DAgostino
Kean
Atanassov
Gustafsson
Bourrier
Bondi
Morris
Tung

At the bottom right of the results table, there are navigation controls: a left arrow, a blue square with the number '1', a right arrow, and a '2'.

七、总结与反思

- 本次实验帮助我了解了如何使用大模型实现自然语言到 SQL 的转换。
- 模型表现良好，但在问题描述不清或语境复杂时可能生成错误 SQL。
- 后续可尝试多轮对话优化提问策略，引导模型生成更优 SQL。

- 大模型完成基础的 sql 查询，以及有简单聚集、排序的查询毫无问题
- 对于有常识要求的查询需要一定的引导，比如平均 gpa 需要通过学分权重来计算（问题 10），gpa 的高低是从 A-F 排序而不是直接用字符串比较（问题 12）
- 对于高级查询（有嵌套查询、连接、子查询等），我们在提问时需要对这个查询本身有一定的思路，从而进行一些引导，否则大模型很难给出正确查询，比如问题 14：找出至少选修了 id 为 1000 的学生所选所有课程的学生，提问时需要告诉大模型这句话的意思是：只要选的课中有一门被 id 为 1000 的学生选了即可，否则大模型很可能会误解我们的意思从而给出错误查询