**Xi’an Jiaotong University**

**课 程 实 验 报 告**

实验名称： RAG(DB+LLM)

姓名: 王顺平

学号: 2223211946

专业班级: 计试2201

指导教师: 任雪斌

报告日期: 2025.5.21

****

# 一、实验目的

1) 了解RAG的定义和实现过程；

2) 体会数据库与大模型结合的应用实践；

3) 学会在本地部署一个专属的AI知识库。

# 二、实验内容

完成给定的关于RAG的题目练习。

以本课程中的一个章节为主题（如：关系模型、事务管理等），收集至少 20 篇相关文档，完成以下步骤：

此次实验的选中的章节是第六章存储部分。

1）数据预处理：对相关文档进行清洗、分割段落等。

2）数据库部署：在 opengauss 中创建专属表，导入数据并建立向量索引。

3）系统集成：编写代码，实现 “用户提问→向量检索→上下文拼接→模型生成” 全流程。

4）测试评估：设计 6 个测试用例（问题），比较RAG与直接LLM的回答质量，撰写 500 字以上的实验总结。

# 实验主要步骤

**搜集文档：**  
在网络上搜集二十篇有关存储与数据库索引等相关的文档。例如：。

**预处理：**

当搜集到足够多的文档后，进行预处理，将文档内容进行清洗，统一格式，提取有效文本。

例如图片中的文档就预处理成：

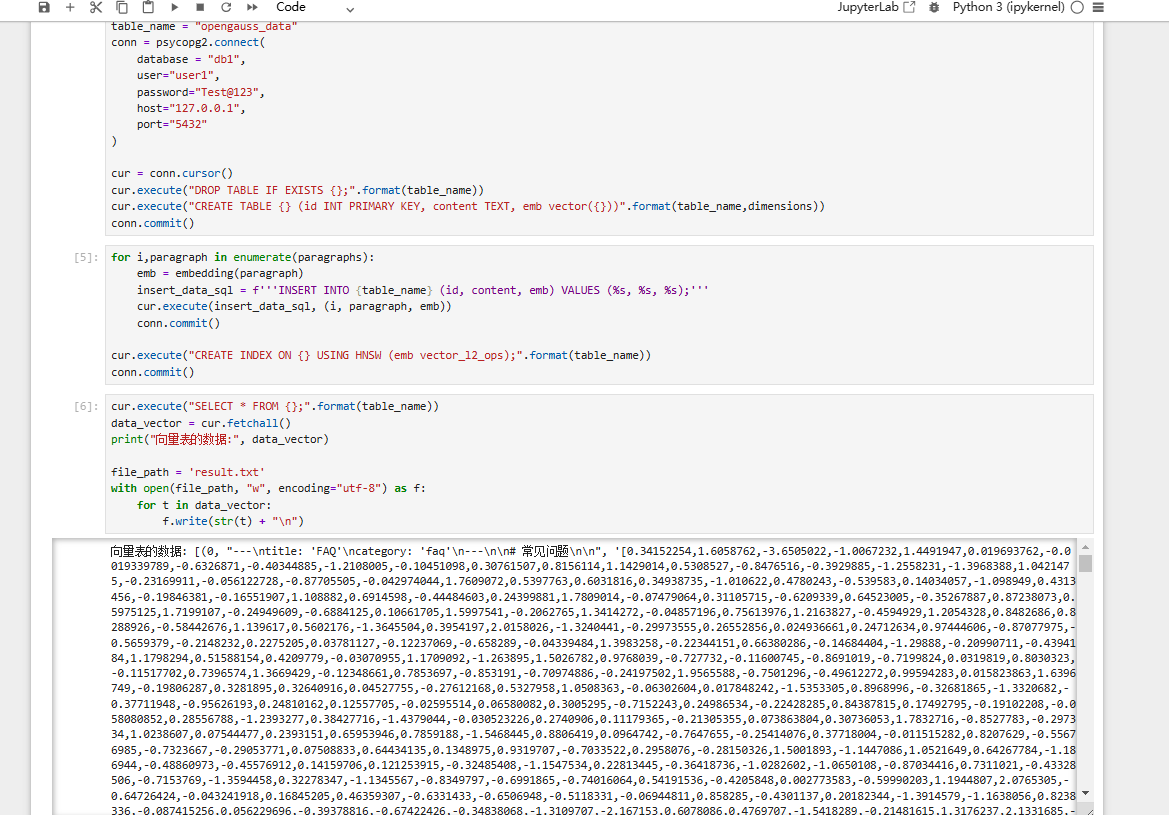
**## 47.为什么用B+树而不用B树，B+树有什么优势？**

（1）IO代价更低。B+树由于非叶子节点中不存放data，因此可以存放更多的索引值（单个大节点的容量固定，每个小单位size变小了），从而使得树的高度更低，磁盘IO次数更少。

（2）查询效率稳定。B+树由于所有data都放在叶子节点中，因此每次查询都要走完整的根节点到叶子节点的路径，所有查询的路径长度相同，查询效率更加稳定。

（3）更利于范围查询。B+树叶子节点之间有指针，注意是双向的指针，更利于范围查询。

**数据库部署：**

接着，进行数据库部署。在程序中，分割段落，在 opengauss 中创建专属表，导入数据并建立向量索引。将向量表的内容打印到result.txt文件中。  


**系统集成：**

编写代码，具体见ipynb文件。

**测试评估：**设计六个问题，分别是：  
1.磁盘按磁头设计有哪些分类？

2.具有变长字段的记录以什么格式存储？

3.指针混写中，被钉住的记录与块该怎么理解？

4.B+tree与B-tree的主要区别在于什么？

5.用通俗易懂的话介绍聚集索引和非聚集索引。

6.为什么用B+树而不用B树，B+树有什么优势？

并比较RAG和LLM的回答质量。  
  
**结果：**

问题4成功使用了数据库中的语料文件，生成了更准确的回答。

1. **B+tree与B-tree的主要区别在于什么？**使用语料文件中的
2. 1.节点存储内容的不同‌：

B+tree中，非叶节点只存储关键字和子节点的引用，不保存物理数据；所有的物理数据都保存在叶子节点中‌‌。

B-tree中，每个节点都存储关键字和对应的物理数据。

2.树的高度差异‌：

由于B+tree的非叶节点不存储数据，只存储关键字和索引，因此每个节点可以容纳更多的元素，这导致B+tree的树高比B-tree小。

由于B-tree中，每个节点都存储关键字和对应的物理数据，导致每个节点存放的元素更少，树的高度也更高。

3.叶子节点的特性‌：

B+tree的叶子节点是顺序排列的，并且相邻节点具有顺序引用关系，形成一个有序的链表，这支持高效的范围查询和顺序遍历‌‌。

B-tree的叶子节点虽然也位于同一层，但并不具备这种顺序引用关系‌‌。

4.查询性能的差异‌：

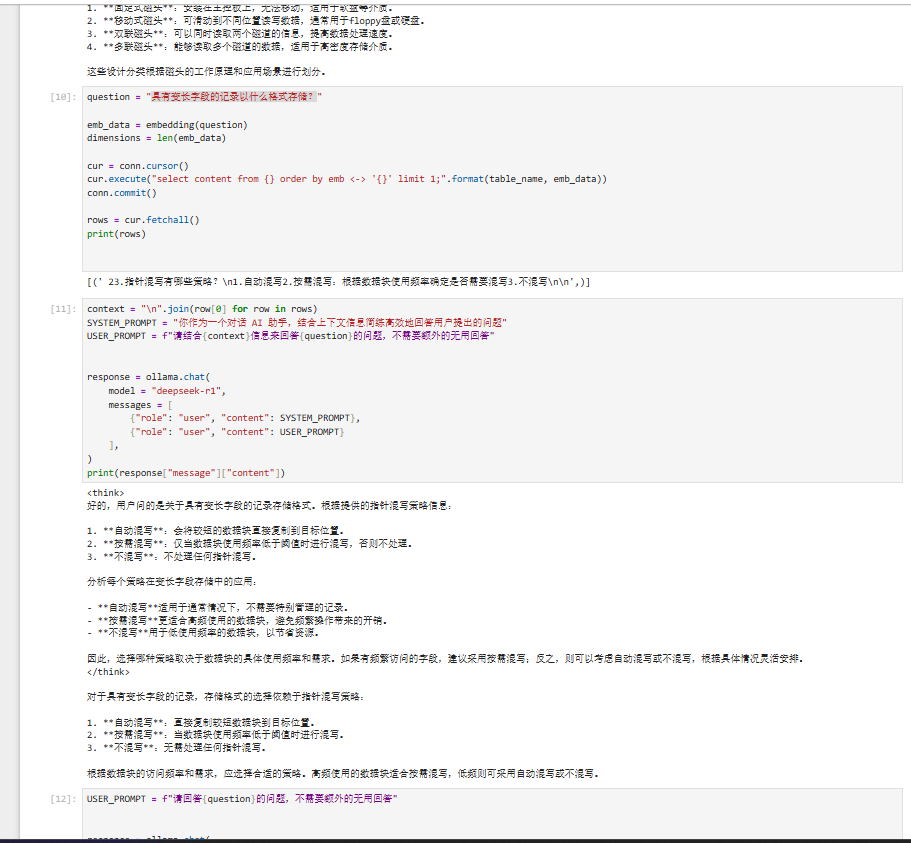
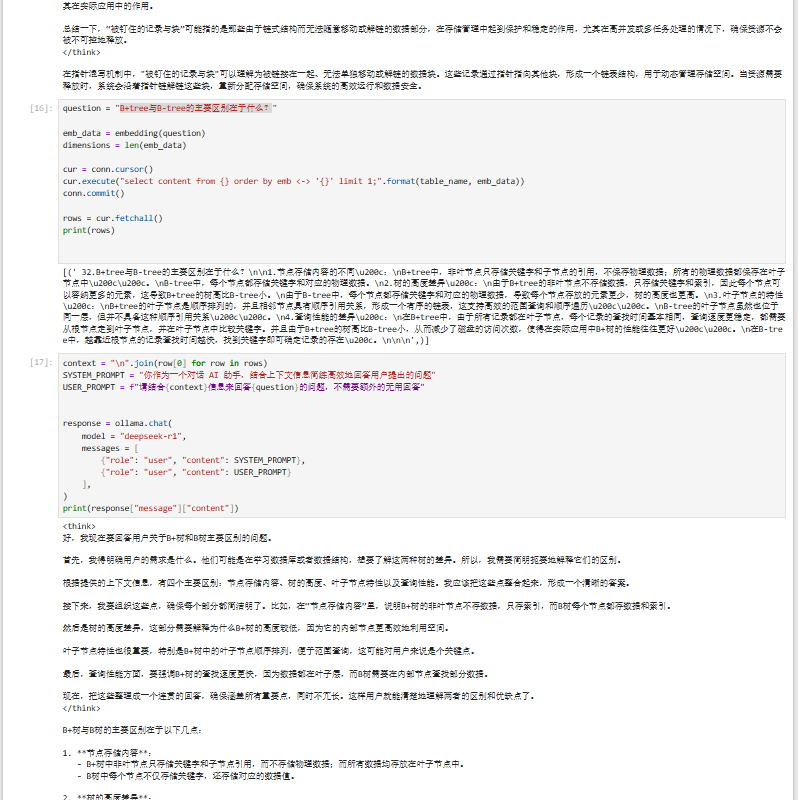
在B+tree中，由于所有记录都在叶子节点，每个记录的查找时间基本相同，查询速度更稳定，都需要从根节点走到叶子节点，并在叶子节点中比较关键字。并且由于B+tree的树高比B-tree小，从而减少了磁盘的访问次数，使得在实际应用中B+树的性能往往更好‌‌。

在B-tree中，越靠近根节点的记录查找时间越快，找到关键字即可确定记录的存在‌。

回答更准确。

设计的六个问题只有第四个问题的回答使用到了数据库的相关资料，回答效果更出色，其余问题RAG改善效果不明显。

其中第一个、第三个问题的回答效果甚至明显下降。



# 实验问题以及解决过程

1. 起初提示数据库权限缺失。

解决方案：发现是db1的schema没有指定，将public schema的全部权限赋予到user1就解决了问题。

1. 生成到第五个问题突然报错。

解决方案：发现是电脑内存不足，关闭后台软件解决。

# 五、实验总结

RAG系统在本次测试中成功使得一个问题回答更精确，其余问题回答效果未能得到明显改善。  
分析问题：

问题4搜索到了匹配的文本，结合上下文，使得回答更精确。体现了RAG系统在消除幻觉等方面的优势，得到了验证。

但实验中的RAG系统仍然存在问题需要得到解决。在数据库已有相关问题的情况下，六个问题只有一个问题正确匹配到了原问题，向量检索很失败。这其中的潜在原因有很多。可能是由于文本分割策略不当导致的，段落信息过长，而文本问题所占太小，导致向量标征模糊。但这不是实验参与者可以改变的，若精简段落，又会导致回答效果不佳。所以考虑其他原因：检索策略缺陷，例如没有设置相似度阈值，导致匹配到了低质量效果，没能选择高效索引类型等等。

但是，即使是能够保证正确匹配，实验中的RAG系统仍然是具有缺陷的。  
 在本次测试样例的设计中，六个测试样例其实已经在语料文件中有完全覆盖了，这样的情况下，仍会出现匹配失败导致效果不佳的情况，而实际运用时，不可能保证所有样例都能有对应的语料文件覆盖。在这种情况下，实验中的RAG系统会强行匹配一条不相关的信息。而大模型的PROMPT是结合匹配到的上下文来回答。在这种情况下，回答效果不仅可能不会有改善，甚至会得到下降。（例如第一个测试样例的回答），强行使用上下文的信息进行回答，导致结果出现严重偏差。

因此，我们可以得到结论：实验中的简易RAG系统，只有当语料文件极小，

且question均被语料文件覆盖的情况下，才能得到明显改善的回答效果。除此之外，语料文件过大，question不被覆盖时，实验中的RAG系统回答效果可能不会有明显改善。

要想改善这种情况，必须使用更先进的向量检索策略，提高文本匹配成功率，并且添加匹配度阈值，当匹配到的文本相似度小于阈值时，放弃结合匹配到的文本进行回答，转而直接回答。这样才能保证RAG系统的效果是良性的。

而在实际的RAG系统中，RAG 通过接入外部数据库，实时调取最新数据，确保回答时效性。针对幻觉问题：RAG生成内容前先检索权威信源（如学术数据库、政府官网），通过 “证据锚定” 强制大模型基于检索结果输出，大幅降低幻觉率。而在本次实验中，问题4成功使用了数据库中的语料文件，生成了更准确的回答，因此可以认为成功验证了RAG在消除幻觉等方面的优势，实验是成功的。

通过本次实验，我掌握了RAG技术核心流程：从文本预处理、向量数据库部署到检索生成全链路实现；深入理解了数据库与大模型的协同机制，通过本地构建AI知识库，验证了RAG在专业领域问答中减少幻觉、提升准确性的优势，为垂直场景应用奠定实践基础。