|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **《数据库系统原理》实验报告（）** | | | | | |
| **题目：MINIOB实验一** | | | | | |
| 学号 | 2152809 | 姓名 | 曾崇然 | 日期 | 2023-11-9 |
| **实验环境：从docer仓库中拉取的oceanbase/miniob镜像，在运行的容器中从github上克隆miniob的源代码，进行编译和运行** | | | | | |
| **实验步骤及结果截图：**  **1.创建miniob环境**  **a)克隆代码**    **b)编译运行**      **2.创建表**      **3.插入数据**    **4.展示学号姓名**    **5.修改成绩**    **结果：没有修改成功**    **原因：**miniob是一个用于学习的的数据库，其虽然实现了update功能的词法和语法解析，但是没有实现其对应的执行，也就是说update并没有实际的执行  **6.删除记录**      **7.代码分析和理解**  **选取功能：create table**  **Miniob的框架如下：**    **具体到create table：**  **a)**client接收输入的命令，create table Scores(id int,name chars(10),score floats)通过网络模块传递到服务端(observer)  **b)**服务端接收到该sql语句后进行处理，在session\_stage.cpp的handle\_sql中定义了处理的流程            **c)**cache\_stage目前是直接跳过了    **d)**parser\_stage对输入的create语句进行词法和语法解析，将分析的结果如语句类型，属性等放入对应的数据结构中，传递给下一个阶段  **c)**resolve\_stage 根据语法解析出来的结果来创建对应的执行语句，将其放入sql\_event中交给下一阶段，具体为：  先获取当前数据库：    创建执行语句：是根据语句的类型(create)将其分发给具体的创建函数进行创建    修改事件：    **d)**optimize\_stage create语句在这个阶段没有优化，直接返回了    **e)**execute\_stage 首先判断有没有物理执行计划，create table没有：    然后根据执行语句去进行执行    具体的执行如下：  根据语句类型分发给具体的执行器：    在对应的execute函数中进入到create\_table的执行：    create\_table函数：  检查表名：    执行创建：    Table\_create函数：这是和存储和文件打交道的创建的物理操作包括以下部分：创建文件，记录元数据等，至此，数据表成功创建。  **f)**语句执行的结果通过net模块传递到客户端进行显示 | | | | | |
| **出现的问题：**  代码的阅读，尽管miniob是一个微型的数据库，但是其代码量依然很大，同时项目结构较为复杂，类很多，让人阅读和分析起来找不到方向。 | | | | | |
| **解决方案：**  **1.查阅资料：**我去观看了官方网站的视频和文档，对miniob的代码结构有了基础的了解  **2.使用chatGPT：**对于阅读出现困难的部分我使用了chatGPT来进行辅助分析  **3.使用gdb调试：**通过gdb调试逐步观察一条sql语句的执行步骤，并深度到对应的代码部分进行理解，这让我对整个项目有了更多的理解 | | | | | |