数据库系统project报告

2023-2024学年第2学期（CST21118）

|  |  |
| --- | --- |
| 数据库系统project任务书 | |
| 名称 | DROP/ALTER TABLE 设计与实现 |
| 类型 | □验证性 □设计性 综合性 |
| 内容 | 1. 基于MiniOB理解数据库系统的存储原理。 2. 分析MiniOB的Create Table的实现原理。 3. 设计并实现MiniOB的Drop/ALTER Table功能。 |
| 要求 | 1. 设计方案要合理； 2. 能基于该方案完成系统要求的功能； 3. 设计方案有一定的合理性分析。 |
| 任务时间 | 2024年4月16日至2024年5月6日 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 小组成员 | | | | | | |
| 20221879 | | 20225847 | | 20221849 | | 20220999 |
| 姚凡 | | 陈骁睿 | | 杨小艺 | | 冉倬樾 |
| 项目评分表 | | | | | | |
| 序号 | 评分项 | | 比例 | | 得分 | |
| 1 | Project内容完成情况 | | 50% | |  | |
| 2 | 工具熟练度 | | 30% | |  | |
| 3 | 团队协作 | | 20% | |  | |
| 项目总得分： | | | | | | |

课程项目评分标准（总分10分）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 考核内容（权重） | 评分标准 | | | |
| 10 | 8-9 | 6-7 | 0-5（不合格） |
| 1. 完成project的实现（50%） | 按时完成project分析、设计、实现等核心内容。 | 完成project分析、设计、实现等核心内容，但存在少量错误。 | 基本project分析、设计、实现等核心内容，但存在较多错误。 | 未完成project内容。 |
| 2.熟练使用设计工具（30%） | 熟练掌握实验设计相关的软件工具，完成数据库设计、编码等工作。 | 较为掌握实验设计相关的软件工具，完成数据库设计、编码等工作。 | 能够使用实验设计相关的软件工具，完成数据库设计、编码等工作。 | 对于实验设计软件和工具使用不熟练。 |
| 3.团队协作（20%） | 有团队，分工合理，密切协作。 | 有团队，分工合理，有一定协作。 | 有团队，分工不合理，无协作。 | 无团队，无协作。 |

|  |
| --- |
| 1. **小组分工**   陈骁睿：分析MiniOB的Create Table实现原理和画出对应过程及描述。  姚凡：设计Drop Table 和Alter Table的实现方案和设计方案图。  杨小艺：实现Drop Table 和Alter Table的代码并进行功能测试。  冉倬樾：总结并编写实验报告，协助代码实现。 |
| 1. **分析MiniOB的Create Table的实现原理**   建表过程主要用到三个函数，流程如图2-1所示    图2-1  生成创建表的执行器 如图2-2所示    图2-2  1.在execute()函数中  获取当前事件信息（Stmt和Session） 如图2-3、图2-4所示    图2-3    图2-4  断言检查，是CREAT\_TABLE，如图2-5所示    图2-5  获取要创建表的名称，如图2-6所示    图2-6  获取当前数据库的指针，根据表名和表的属性在当前数据库调用creat\_table()创建表如图2-7所示    图2-7  2.在creat\_table()函数中  检查表名是否已存在，如图2-8所示    图2-8  生成相关数据，如图2-9所示    图2-9  调用表对象的creat()方法创建  3.在creat()函数中  检查表名、表id、表的属性是否合法，如图2-10所示    图2-10  利用判断表文件是否已存在，如图2-11所示    图2-11  若不存在，则创建文件如图2-12所示    图2-12  记录元数据，如图2-13所示    图2-13  创建表数据文件，如图2-14所示    图2-14  4.返回到creat\_table()中  将表名加入当前所有打开的表列表中  并返回创建成功，如图2-15所示    图2-15 |
| 1. **设计Drop Table和ALTER Table的实现方案**   **Drop Table:**  **Drop Table设计方案：**  1. 检查表是否存在：在执行 Drop Table 命令之前，需要首先检查待删除的表是否存在于数据库中。  2. 删除相关文件：如果表存在，需要删除相关文件释放该表占用的内存空间，包括t.table文件、t.data文件和关联的索引文件。  3.清空buffer pool：由于buffer pool的存在，在新建表和插入数据的时候，会写入buffer pool缓存。所以drop table，不仅需要删除文件，也需要清空buffer pool ，防止在数据没落盘的时候，再建立同名表，仍然可以查询到数据。  4. 更新元数据：最后，更新系统的元数据信息，从系统中移除对应的表信息。  5.返回结果：向用户返回操作结果，确认表已删除。  **Drop Table具体实现方案：**  利用vscode进行调试；  由一个sql语句查询执行的过程，通过前面的语法解析，语法树生成，执行计划生成等，可以得知最终会有相应的执行器(executor)去执行该sql语句。如图3-1所示    图3-1  根据create\_table\_executor执行器代码中的CreateTableStmt先添加drop\_table对应的stmt，在里面写入描述一个drop table语句的DropTableSqlNode；如图3-2所示    写出drop table的执行器，并在Db类中加入drop table执行器所需要调用的函数drop\_table；        根据创建表代码得知t.table文件所在路径，droptable则将这些文件进行删除操作；根据table::create\_index 的代码得知该表的索引文件路径，drop函数则在文件系统中对这些文件进行删除。  在CommanExecutor::execute中加入Drop Table的对应的case；在stmt.cpp中加入创建删除表对应的case；      **Drop Table设计方案图**：  drop  **Alter Table:**  **Alter Table 实现方案：**  1. 插入字段：  - 解析 Alter Table 语句，提取新增字段的信息。  - 在表结构中添加新字段，可能需要重新分配内存空间。  - 更新磁盘文件和元数据信息，确保持久化存储和系统元数据的一致性。  2. 删除字段：  - 解析 Alter Table 语句，提取待删除字段的信息。  - 从表结构中移除指定字段，可能需要调整内存空间。  - 更新磁盘文件和元数据信息，确保持久化存储和系统元数据的一致性。  3. 修改字段类型：  - 解析 Alter Table 语句，提取需要修改的字段和新的字段类型信息。  - 在表结构中修改指定字段的类型，可能需要进行数据类型转换。  - 更新磁盘文件和元数据信息，确保持久化存储和系统元数据的一致性。   1. 重建索引：根据需要，重建受影响的索引 2. 返回结果：向用户返回操作结果，确认表已修改。   **Altertable具体实现方案：**  在src/observer/sql/parser/lex\_sql.l 添加ALTER、ADD关键字；    在src/observer/sql/parser/yacc\_sql.y 添加相应token；    在src/observer/sql/parser/yacc\_sql.y 添加alter\_add\_column\_stmt;    在command中添加alter\_add\_column\_stmt；    在src/observer/sql/parser/yacc\_sql.y 添加alter\_add\_column\_stmt对应的规则；    在sqlcommandflag中添加对应的flag；在parsedsqlnode类中添加AlterT  AbleAddSqlDode的属性；    最后运行构建脚本 bash gen\_parser.sh  **Alter table设计方案图：**  alter  **SQL语法：**  ALTER TABLE *table\_name*  ADD column\_name datatype  **miniob词法/语法解析：**  词法分析:词法分析是通过 Lex 工具完成的，它将输入的字符序列转换成一系列的标记（tokens）。在src/observer/sql/parser/lex\_sql.l 以及lex\_sql.cpp 文件中定义了这些标记的规则。  语法分析:语法分析是由 Yacc（Bison）工具完成的，它使用 Yacc 文件（src/observer/sql/parser/yacc\_sql.y）中定义的语法规则来构建抽象语法树（AST）。 |
| 1. **Drop Table 、Alter Table的实现代码**   **Drop Table：**  **头文件：**  #pragma once  #include "common/rc.h"  class SQLStageEvent;  /\*\*   \* @brief 创建表的执行器   \* @ingroup Executor   \*/  class DropTableExecutor  {  public:    DropTableExecutor()          = default;    virtual ~DropTableExecutor() = default;    RC execute(SQLStageEvent \*sql\_event);  };  **源文件：**  #include "sql/executor/drop\_table\_executor.h"  #include "common/log/log.h"  #include "event/session\_event.h"  #include "event/sql\_event.h"  #include "session/session.h"  #include "sql/stmt/drop\_table\_stmt.h"  #include "storage/db/db.h"  RC DropTableExecutor::execute(SQLStageEvent \*sql\_event)  {    Stmt    \*stmt    = sql\_event->stmt();    Session \*session = sql\_event->session\_event()->session();    ASSERT(stmt->type() == StmtType::DROP\_TABLE,        "drop table executor can not run this command: %d",        static\_cast<int>(stmt->type()));    DropTableStmt \*drop\_table\_stmt = static\_cast<DropTableStmt \*>(stmt);    const char \*table\_name = drop\_table\_stmt->table\_name().c\_str();    RC rc = session->get\_current\_db()->drop\_table(table\_name);    return rc;  }  **Alter Table：**  头文件：  #pragma once  #include "common/rc.h"  class SQLStageEvent;  /\*\*   \* @brief 创建表的执行器   \* @ingroup Executor   \*/  class AlterModifyTableExecutor  {  public:    AlterModifyTableExecutor()          = default;    virtual ~AlterModifyTableExecutor() = default;    RC execute(SQLStageEvent \*sql\_event);  };  #pragma once  #include "common/rc.h"  class SQLStageEvent;  /\*\*   \* @brief 创建表的执行器   \* @ingroup Executor   \*/  class AlterAddTableExecutor  {  public:    AlterAddTableExecutor()          = default;    virtual ~AlterAddTableExecutor() = default;    RC execute(SQLStageEvent \*sql\_event);  };  源文件：  #include "sql/executor/alter\_modify\_table\_executor.h"  #include "common/log/log.h"  #include "event/session\_event.h"  #include "event/sql\_event.h"  #include "session/session.h"  #include "sql/stmt/alter\_modify\_table\_stmt.h"  #include "storage/db/db.h"  RC AlterModifyTableExecutor::execute(SQLStageEvent \*sql\_event)  {    Stmt    \*stmt    = sql\_event->stmt();    Session \*session = sql\_event->session\_event()->session();    ASSERT(stmt->type() == StmtType::ALTER\_MODIFY\_TABLE,        "alter modify table executor can not run this command: %d",        static\_cast<int>(stmt->type()));    AlterModifyTableStmt \*alter\_modify\_table\_stmt = static\_cast<AlterModifyTableStmt \*>(stmt);    const char \*table\_name = alter\_modify\_table\_stmt->table\_name().c\_str();    const char \*attri\_old = alter\_modify\_table\_stmt->attr\_old().c\_str();    const char \*attri\_new = alter\_modify\_table\_stmt->attr\_new().c\_str();     RC rc = session->get\_current\_db()->alter\_modify\_table(table\_name, attri\_old,attri\_new);    return rc;  }  #include "sql/executor/alter\_add\_table\_executor.h"  #include "common/log/log.h"  #include "event/session\_event.h"  #include "event/sql\_event.h"  #include "session/session.h"  #include "sql/stmt/alter\_add\_table\_stmt.h"  #include "storage/db/db.h"  RC AlterAddTableExecutor::execute(SQLStageEvent \*sql\_event)  {    Stmt    \*stmt    = sql\_event->stmt();    Session \*session = sql\_event->session\_event()->session();    ASSERT(stmt->type() == StmtType::ALTER\_ADD\_TABLE,        "alter add table executor can not run this command: %d",        static\_cast<int>(stmt->type()));    AlterAddTableStmt \*alter\_add\_table\_stmt = static\_cast<AlterAddTableStmt \*>(stmt);    const char \*table\_name = alter\_add\_table\_stmt->table\_name().c\_str();    const int attribute\_count = static\_cast<int>(alter\_add\_table\_stmt->attr\_infos().size());       RC rc = session->get\_current\_db()->alter\_add\_table(table\_name,attribute\_count, alter\_add\_table\_stmt->attr\_infos().data());    return rc;  }  SQL语法解析：  在src/observer/sql/parser/lex\_sql.l 添加ALTER、ADD关键字    在src/observer/sql/parser/yacc\_sql.y 添加相应token    src/observer/sql/parser/yacc\_sql.y 添加    添加到command    添加对应规则    其中AlterTableAddSqlNode等要先在src/observer/sql/parser/parse\_defs.h 定义    添加对应的flag    在类里面加上AlterAddTableSqlNode属性    进入src/observer/sql/parser运行构建脚本 bash gen\_parser.sh |
| 1. **功能测试**   实现了show table、create table、drop table、alter\_modify、alter\_add等功能。 |
| 1. **总结**   在这次实验中，我们获益匪浅。我们第一次接触到一个较为规范的开发项目，并在开发中提高了自己的代码能力，也对SQL语句的实现和drop table和alter table的执行原理理解更充分，同时锻炼了我们上机实操的能力。我们了解了一种新的数据库系统，学习了对表各种操作的设计和代码实现，进一步拓展了知识面。在实验过程中，我们小组组员团结协作，分工明确，共同完成了每个实验，增强了我们交流配合和团队合作的能力。  在实验过程中，我们也遇到了一些问题和不足： SQL语言掌握不够熟练：虽然我们已经掌握了SQL语言的基本语法和常用命令，但在实际操作中，我们发现自己在编写复杂查询和存储过程时还存在一定的困难。 数据库设计能力有待提高：在数据库设计过程中，我们在需求分析、概念设计和逻辑设计等方面还有待提高。我们需要更加深入地了解用户需求，学会使用更加先进的数据库设计工具和方法。 总之，本学期的数据库实验课程让我们收获颇丰。我们将继续努力学习和实践，不断提高自己的数据库技术水平，为未来的职业发展打下坚实的基础。 |