

《数据库系统原理》实验报告（5）					
题目：MiniOB 实验 2					
学号	2154312	姓名	郑博远	日期	2023.11.22
<p>实验环境：</p> <p>硬件配置：联想小新 Pro 14ACH 2021</p> <p>CPU：AMD Ryzen 7 5800H with Radeon Graphics</p> <p>操作系统：Windows11</p> <p>编辑器：VSCode</p>					
<p>实验步骤及结果截图：</p> <p>本次实验中，我选择了 drop_table 语句进行实现。</p> <p>1. 首先,增加 drop_table 对应的 statement 语句.在“src\observer\sql\stmt\”下增加“drop_table_stmt.h”与 “drop_table_stmt.cpp” 文件，对应代码如下：</p> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <pre>// Created by Boyuan Zheng on 2023.11.22  #include "drop_table_stmt.h" #include "event/sql debug.h"  RC DropTableStmt::create(Db *db, const DropTableSqlNode &amp;drop_table, Stmt *&amp;stmt) {     stmt = new DropTableStmt(drop_table.relation_name);     sql debug("drop table statement: table name %s", drop_table.relation name.c str());     return RC::SUCCESS; }  // Created by Boyuan Zheng on 2023.11.22  #pragma once  #include &lt;string&gt; #include &lt;vector&gt;  #include "sql/stmt/stmt.h"  class Db;  class DropTableStmt : public Stmt { public:     DropTableStmt(const std::string &amp;table_name) : table_name_(table_name) {}     virtual ~DropTableStmt() = default;      StmtType type() const override { return StmtType::DROP TABLE; }      const std::string &amp;table_name() const { return table_name_; }      static RC create(Db *db, const DropTableSqlNode &amp;drop_table, Stmt *&amp;stmt);  private:     std::string table_name_; </pre> </div>					

```
};
```

## 2. 在同目录下的 stmt.cpp 的 create\_stmt 函数中，添加对应的 case:

```
...
    case SCF_CREATE_TABLE: {
        return CreateTableStmt::create(db, sql_node.create_table, stmt);
    }

    // Added by Boyuan Zheng on 2023.11.22
    case SCF_DROP_TABLE: {
        return DropTableStmt::create(db, sql_node.drop_table, stmt);
    }

    case SCF_DESC_TABLE: {
        return DescTableStmt::create(db, sql_node.desc_table, stmt);
    }

    ...
```

## 3. 其次，在“src\observer\sql\executor\”下为 drop\_table 添加对应的执行器。对应代码如下:

```
// Created by Boyuan Zheng on 2023.11.22
#pragma once
#include "common/rc.h"

class SQLStageEvent;

/**
 * @brief 删除表的执行器
 * @ingroup Executor
 */
class DropTableExecutor
{
public:
    DropTableExecutor() = default;
    virtual ~DropTableExecutor() = default;

    RC execute(SQLStageEvent *sql_event);
};
```

```
// Created by Boyuan Zheng on 2023.11.22

#include "drop_table_executor.h"

#include "session/session.h"
#include "common/log/log.h"
#include "storage/table/table.h"
#include "sql/stmt/drop_table_stmt.h"
#include "event/sql_event.h"
#include "event/session_event.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);
RC rc =
session->get_current_db()->drop_table(drop_table_stmt->table_name().c_str());

return rc;
}

```

4. 类似地，在同目录下的“command\_executor.cpp”中添加 DROP\_TABLE 对应的 case（同时在文件头添加#include "sql/executor/drop\_table\_executor.h"，新增对应头文件）：

```

...
case StmtType::CREATE_TABLE: {
    CreateTableExecutor executor;
    return executor.execute(sql_event);
} break;

// Added by Boyuan Zheng on 2023.11.22
case StmtType::DROP_TABLE: {
    DropTableExecutor executor;
    return executor.execute(sql_event);
} break;

case StmtType::DESC_TABLE: {
    DescTableExecutor executor;
    return executor.execute(sql_event);
}
...

```

5. 在“src\observer\storage\db\”文件夹下，在“db.cpp”与“db.h”文件中分别加入 database 的 droptable 实现，代码如下：

```

RC Db::drop_table(const char *table_name)
{
    RC rc = RC::SUCCESS;

    // 检查表是否存在
    Table *table = find_table(table_name);
    if(table == nullptr){
        LOG_WARN("Failed to find table %s.", table_name);
        return RC::SCHEMA_TABLE_NOT_EXIST;
    }

    // 删除表
    rc = table->drop(table_name);
    if(rc != RC::SUCCESS) {
        LOG_ERROR("Failed to drop table %s.", table_name);
        return rc;
    }

    // 从打开的表中删除

```

```
opened tables .erase(table name);
LOG_INFO("Drop table success. table name=%s.", table_name);

return rc;
}
```

```
// new added
RC drop_table(const char *table_name);
```

6. 在“src\observer\storage\table\”文件夹下，“table.cpp”与“table.h”文件中分别加入删除 table 具体操作的实现。代码如下：

```
RC Table::drop(const char *table name)
{
    RC rc = RC::SUCCESS;
    PersistHandler persistHandler;

    std::string data file = base_dir + "/" + table name + ".data";
    std::string table file = base_dir + "/" + table name + ".table";
    rc = persistHandler.remove_file(data_file.c_str());
    if(rc != RC::SUCCESS)
        return rc;
    rc = persistHandler.remove_file(table_file.c_str());
    if(rc != RC::SUCCESS)
        return rc;

    for(auto index : indexes ){
        std::string index_file = base_dir_ + "/" + table_name + "-" +
index->index_meta().name() + ".index";
        rc = persistHandler.remove_file(index_file.c_str());
        if(rc != RC::SUCCESS)
            return rc;
    }

    rc = data buffer pool ->close file();
    if(rc != RC::SUCCESS)
        return rc;

    record handler ->close();

    return rc;
}
```

```
/**
 * 删除一个表（新增）
 * @param name 表名
 */
RC drop(const char *name);
```

7. Oceanbase 官网评测结果如下，能够通过 drop-table 的官方测试样例：



drop-table	10	10	-
------------	----	----	---

出现的问题：

- 开始时没有在 miniOB 的 test 分支上进行修改，而是在 main 分支上修改，导致出现了一些不在比赛范围内的问题。

解决方案：

- 比赛中推荐在 miniOB 的 test 分支上增加代码，因此最终 clone 了 test 分支的代码进行修改。