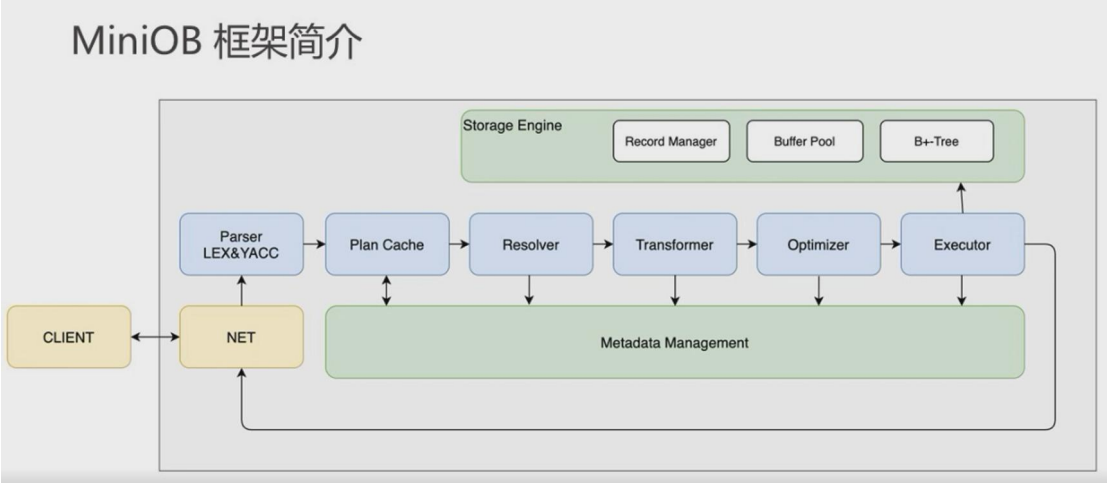
《数据库系统原理》实验报告（5）					
题目：miniob 实验二					
学号	2152118	姓名	史君宝	日期	2023.11.26

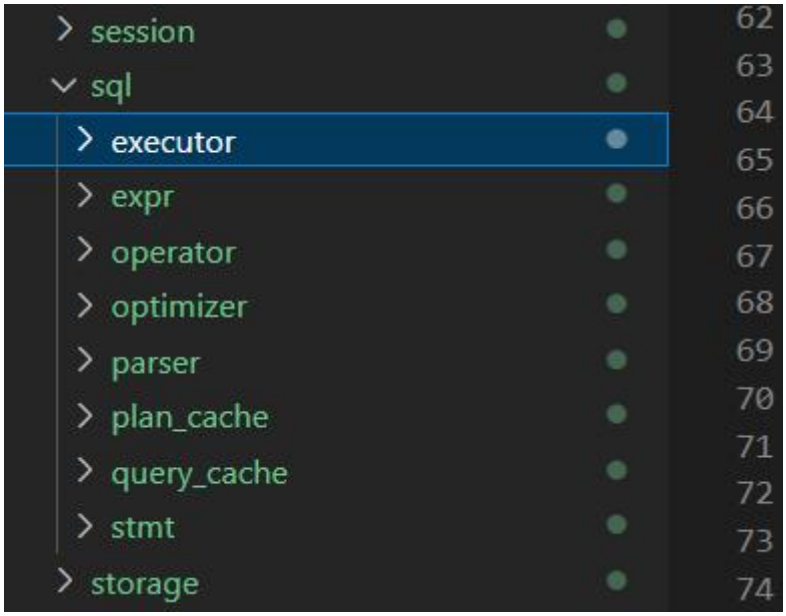
实验环境：基于 miniob 原码的修改，完成 drop_table 的问题。

实验步骤及结果截图：

（1）首先了解 Miniob 的框架：



根据上面的框架，我们在 SQL 部分也能找到对应：



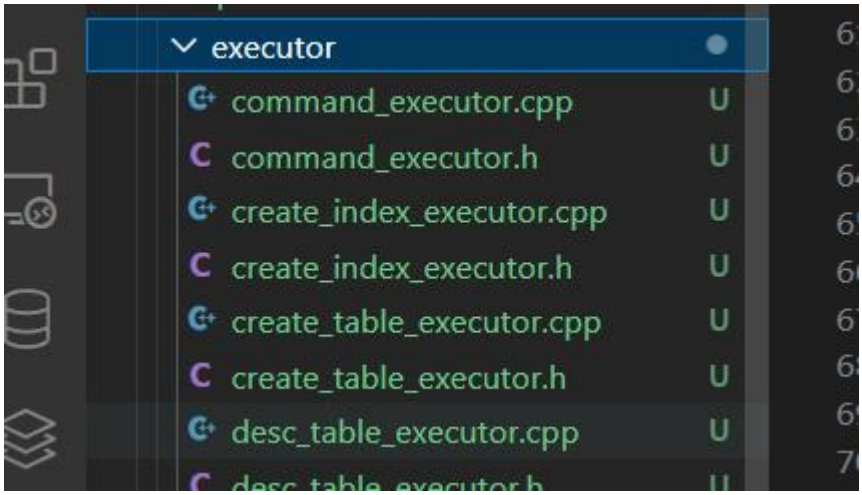
我们观察到上面的代码目录就是按照 Miniob 的架构创建的。

（2）在 miniob 中修改：

在 miniob 中关于 SQL 语句的词法语法的识别，已经完成了。

在中间几个环节只是进行优化，并未完成比较重要的任务。

因此我们需要去看具体的执行阶段：



我们进入 command_executor.cpp 中可以观察到：

```
RC CommandExecutor::execute(SQLStageEvent *sql_event)
{
  Stmt *stmt = sql_event->stmt();

  switch (stmt->type()) {
    case StmtType::CREATE_INDEX: {
      CreateIndexExecutor executor;
      return executor.execute(sql_event);
    } break;

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

这个是对各个 sql 语句进行的一个分发，但是里面并没有对 Drop_table 语句进行实现，我们要具体实现它。

（3）在上面的 command_executor.cpp 中：

```
case StmtType::DROP_TABLE {
  DropTableExecutor executor;
  return executor.execute(sql_event);
} break;
```

我们仿照之前写的代码，可以由上面。

接下来上面执行的具体代码都没有写出来，我们需要进行添加

首先在 executor 文件夹下创建 drop_table_executor.cpp 文件和 drop_table_executor.h
这里我们可以仿照其他的写出具体的代码：

```h

#pragma once

#include "common/rc.h"

class SQLStageEvent;

/**
 * @brief 创建索引的执行器
 * @ingroup Executor
 * @note 创建索引时不能做其它操作。MiniOB当前不完善，没有对一些并发做控制，包括schema的并发。
 */
class DropTableExecutor
{
public:
  DropTableExecutor()          = default;
  virtual ~DropTableExecutor() = default;

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

```cpp

#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 CreateTableExecutor::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<DropTableStm *>(stmt);

  const char *table_name = drop_table_stmt->table_name().c_str();

  RC rc = session->get_current_db()->drop_table(table_name);

  return rc;
}
```

但是需要注意的是，在上述文件中会用到 drop_table_stmt.h 这个文件，这个文件并没有创建，我们要找到 sql 下的 stmt 文件夹，再里面创建 drop_table_stmt.h，drop_table_stmt.cpp 这两个文件。

我们创建这两个文件之后，就可以仿照其他写出：

```h
#pragma once

#include <string>
#include <vector>

#include "sql/stmt/stmt.h"

class Db;

/**
 * @brief 表示创建表的语句
 * @ingroup Statement
 * @details 虽然解析成了stmt，但是与原始的SQL解析后的数据也差不多
 */
class DropTableStmt : public Stmt
{
public:
  DropTableStmt(const std::string &table_name)
      : table_name_(table_name)
  {}
  virtual ~DropTableStmt() = default;

  StmtType type() const override { return StmtType::DROP_TABLE; }

  const std::string                 &table_name() const { return table_name_; }

  static RC drop(Db *db, const CreateTableSqlNode &drop_table, Stmt *&stmt);

private:
  std::string                 table_name_;
};
```

```cpp
#include "sql/stmt/drop_table_stmt.h"
#include "event/sql_debug.h"

RC DropTableStmt::drop(Db *db, const CreateTableSqlNode &drop_table, Stmt *&stmt)
{
  stmt = new DropTableStmt(drop_table.relation_name);
  sql_debug("drop table statement: table name %s", create_table.relation_name.c_str());
  return RC::SUCCESS;
}
```

还要注意的是，在 drop_table_executor.cpp 文件的具体实现中，还需要调用数据库，对数据库进行表的删除，但是在 db.h 中并没有编写对应的代码。
我们需要前往 storage 文件夹下的 db 文件下的 db.h 和 db.cpp 这两个文件，进行修改。

```h
RC create_table(const char *table_name, int attribute_count, const AttrInfoSqlNode *attributes);

RC drop_table(const char *table_name);

Table *find_table(const char *table_name) const;
Table *find_table(int32_t table_id) const;

const char *name() const;
```

```
105  RC Db::drop_table(const char *table_name)
106  {
107    RC rc = RC::SUCCESS;
108
109    Table *table = find_table(table_name);
110    if (table == nullptr) {
111      LOG_WARN("%s not exist.", table_name);
112      return RC::SCHEMA_TABLE_NOT_EXIST;
113    }
114
115    rc = table->drop(table_name);
116    if (rc != RC::SUCCESS) {
117      LOG_ERROR("Failed to drop table");
118      return rc;
119    }
120
121    opened_tables_.erase(table_name);
122    LOG_INFO("Drop table success. table name=%s", table_name);
123
124    return rc;
125  }
```

实验结果:

出现的问题：

在整个实验过程中并未出现比较大的问题，主要问题就是 miniob 数据库中的很多代码需要自己去细细去读，不能很快的掌握，经过查找一些资料就可以知道。这些在上课并未学到，用起来不是特别的得心应手，需要查找一下资料才能进一步完成。

解决方案：

以上出现的问题都不是比较严重的问题，通过在网上查找资料，比如 CSDN、百度等众多的学习工具，就可以顺利的解决了。