实验3：数据定义的实现

邹兆年，哈尔滨工业大学计算学部，znzou@hit.edu.cn

https://gitee.com/HIT-DB/hit-db-class-rucbase-lab

一、实验目的

1. 掌握Rucbase元数据管理的实现方法。
2. 掌握Rucbase数据定义语句的实现方法。

二、相关知识

1. 元数据管理
2. 数据定义语言

三、实验内容

本实验包括2项任务。

任务1：元数据管理实现

补全`SmManager`类，实现元数据管理功能，使Rucbase能够支持下列语句和命令：

- `CREATE TABLE`语句
- `DROP TABLE`语句
- `show tables`命令
- `desc`命令

具体完成如下任务。

（1）阅读代码

阅读`src/system`目录下的代码。

- `src/system/sm_meta.h`
- `src/system/sm_manager.h`
- `src/system/sm_manager.cpp`

理解`ColMeta`类、`TabMeta`类和`DbMeta`类的设计。

理解`SmManager`类的设计。

（2）实现`SmManager::open_db`函数

函数声明：

```
void SmManager::open_db(const std::string& db_name);
```

功能：打开数据库。参数的含义参考代码注释。

实现：参考代码注释。

实现打开数据库的功能前需要先了解Rucbase的使用方法。启动Rucbase服务端程序的方法如下：

```
./bin/rmdb <database_name>
```

`<database_name>`代表数据库的名称。如果该数据库存在，则存在一个名为`<database_name>`的目录，该目录中存储了该数据库的所有相关文件；如果该数据库不存在，则Rucbase会创建一个名为`<database_name>`的目录。

`SmManager::open_db`函数的基本实现逻辑如下：找到数据库的同名目录，读取元数据文件获取该数据库的元数据，打开所有表文件和索引文件，并将文件句柄记录在`SmManager::fhs_`和`SmManager::ihs_`中。可以参考`SmManager::create_db`函数的实现。

（3）实现`SmManager::close_db`函数

函数声明：

```
void SmManager::close_db();
```

功能：关闭数据库。

实现：参考代码注释。基本实现逻辑如下：

1. 使用缓冲区管理器将该数据库每个文件在缓冲池中的脏页刷写到磁盘。
2. 关闭该数据库的所有文件。

（4）实现`SmManager::drop_table`函数

函数声明：

```
void SmManager::drop_table(const std::string& tab_name, Context* context);
```

功能：删除表。参数的含义参考代码注释。

实现：参考代码注释。基本实现逻辑如下：

1. 使用缓冲区管理器将该表文件在缓冲池中的脏页刷写到磁盘。
2. 关闭该表文件。
3. 删除与该表相关的表文件和索引文件。
4. 更新元数据。

参考`SmManager::create_table`函数函数的实现。

（5）单元测试

单元测试代码在文件`src/test/query/query_unit_test.py`中。

执行下列命令，进行单元测试。

```
cd src/test/query
python query_unit_test.py basic_query_test1.sql
```

（6）注意事项

不允许修改任何公有函数的声明。

四、考核方法

1. 实验完成情况（80%）。根据单元测试通过情况和代码理解情况给分。
2. 实验报告（20%）。根据实验报告的完整性、科学性和规范性评分。