数据库系统设计-MiniSQL

API模块详细设计报告

黄昭阳 3130000696 - 11/6/15

API模块总体设计

一、实验要求：

API模块是整个系统的核心，其主要功能为提供执行SQL语句的接口，供Interpreter层调用。该接口以Interpreter层解释生成的命令内部表示为输入，根据Catalog Manager提供的信息确定执行规则，并调用Record Manager、Index Manager和Catalog Manager提供的相应接口进行执行，最后返回执行结果给Interpreter模块。

二、实验完成情况：

完美实现实验要求功能，并将部分Interpreter层功能转移到API模块，Interpreter层最终只是以命令解释器的形态存在而不会参与任何的主体程序流程的工作，API将负责所有的业务逻辑而不是只作为接口的提供模块。

API将接受来自Record Manager、Index Manager、Interpreter三个模块的委托，完成整个MiniSQL各模块之间的信息交互。同时提供一个start接口，用来运行MiniSQL，在输入命令“quit;”后MiniSQL将结束运行。

模块整体详细分析

一、API整体结构分析和代码介绍：

```
class API:public SQLCommandInterpreterDelegate,public
recordDelegate,public indexDelegate    //继承来自各个模块的委托
{
private:
    catalogManager* catalog;  //持有一个catalog来访问有效信息
    string getInputString();  //从用户输入处得到一个命令
    void solveSelectFrom(SQLCommandSelectFrom* command);
    void solveInsertInto(SQLCommandInsertInto* command);
    void solveDeleteFrom(SQLCommandDeleteFrom* command);
  //分别用来处理这三个命令的函数
    void printTuple(SQLTuple* tuple);
  //SelectFrom获得了用户需要的记录后，通过此函数将他们输出

    //basic delegates
    virtual SQLTable* getTableInfo(string* tableName);
    virtual SQLIndex* getIndexInfo(string* indexName);
    virtual string* getRecordFileName(string* tableName);
    virtual string* getIndexFileName(string* indexName);
  //这是basic delegate中的相关函数，用来获取catalog中的Table和Index的相关信息

    //record delegates
    virtual int selectedTupleFromRecord(vector<SQLTuple*> tuple);
    virtual int deleteIndex(string* indexName,SQLData* data);
    virtual int insertIndex(string* indexName,SQLData* data,int
recordAddress);
  //这是Record delegate中的相关函数，用来完成Record Manager交给API的委托，具体函
数功能将在总体设计报告中得到详尽的阐述

    //index delegates
    virtual int selectedRecordAddressFromIndex(string*
tableName,vector<int>& recordAddress,vector<SQLCondition*>& condition);
    virtual int deleteRecordAddressFromIndex(string*
tableName,vector<int>& recordAddress,vector<SQLCondition*>& condition);
  //这是Index delegate中的相关函数，用来完成Index Manager交给API的委托，具体函数
功能将在总体设计报告中得到详尽的阐述

public:
    void start();
  //这个接口用来运行MiniSQL，执行此函数后，API将不断进入等待用户输入->执行用户命令的
循环中，直到用户输入”quit;“命令MiniSQL将结束运行

};
```

二、API使用方法：

API使用方法十分简单，实例化API后直接调用start即可，以下为使用实例。

```
API* api = new API();
api->start();
```

模块关键函数详细介绍与分析

1. start函数：

```
void API::start()
{
    SQLCommand* command;
    bufferManager::initialMemory();  //初始化bufferManager
    catalog = new catalogManager(new string("/Users/drinkingcoder/
Documents/university/Database Design/MiniSQL-final/lib/catalog"));
  //创建catalog实例，获取MiniSQL的相关信息
    SQLCommandInterpreter::setDelegate(this);
    indexManager::setDelegate(this);
    recordManager::setDelegate(this);
 //设置相关委托
    while(1)  //进入死循环，不断读取用户输入，执行命令
    {
        command =
SQLCommandInterpreter::transferStringToCommand(getInputString());
  //读取用户命令，通过interpreter解析用户命令
        if(command == NULL)  //用户命令解析失败，提示用户
            cout << endl << "Failed!" << endl;
        else {              //用户命令解析成功，进入命令执行阶段
            switch(command->commandType)
            {
                case QUIT:  //quit命令，直接退出函数
                    delete catalog;
                    bufferManager::quitBufferManager();
                    delete command;
                    return;
                case CREATE_TABLE:  //创建表单
                    catalog->createTable(new
SQLTable(*((SQLCommandCreateTable*)command)->table));
                    delete command;
                    break;
                case DROP_TABLE:    //删除表单
                    catalog->dropTable(new
string(*((SQLCommandDropTable*)command)->tableName));
                    delete command;
                    break;
                case CREATE_INDEX:  //创建索引
                    catalog->createIndex(new
SQLIndex(*((SQLCommandCreateIndex*)command)->index));
                    recordManager::createIndex(new
SQLIndex(*((SQLCommandCreateIndex*)command)->index));
                    delete command;
                    break;
```

```
                case DROP_INDEX:    //删除索引
                    catalog->dropIndex(new
string(*((SQLCommandDropIndex*)command)->indexName));
                    delete command;
                    break;
                case SELECT_FROM:    //选择命令
                    solveSelectFrom((SQLCommandSelectFrom*)command);
                    break;
                case INSERT_INTO:    //插入命令
                    solveInsertInto((SQLCommandInsertInto*)command);
                    break;
                case DELETE_FROM:    //删除命令
                    solveDeleteFrom((SQLCommandDeleteFrom*)command);
                    break;
                default:
                    break;
            }
        }
    delete catalog;  //删除catalog
    bufferManager::quitBufferManager();
  //退出对bufferManager的使用，所有缓存都将被写入文件
    }
}
```

分析：API的start函数被调用后将对MiniSQL的全局进行初始化并进行准备工作，如catalog和bufferManager，委托的设置等等，然后进入MiniSQL的业务流程。用户输入"quit;"命令后，MiniSQL运行解释，此时API将负责所有善后工作，删除catalog的实例，退出bufferManager的使用。

API模块总结

1. 通过API模块的整体实现，我对MiniSQL的业务流程有了一个整体的深刻的理解，对于数据库系统的运行方式有了详细的了解，大大加深了我对于数据库系统的学习。

2. API作为整个MiniSQL系统的核心，需要负责所有模块之间的信息交流，为了降低底层模块之间的耦合度，简化程序架构的复杂度，以便于整个项目的推进和实现，这里采用委托的方式将所有任务都通过API来调度，而不是底层模块之间的直接交流。这次API模块的实现，大大加深了我对于MVVM设计模式和面向对象设计理念的理解，对于委托的使用也变得更为娴熟。

3. 此次MiniSQL实验的完成，我实现了除Record Manager模块之外的五个模块的任务，并架构了整个MiniSQL实验的接口设计，使我对MiniSQL的整体结构有了一个完整的彻底的了解，了解了一条命令从用户输入到最终输出结果给用户每一个环节的每一个细节，对于其中发生的任何一个问题都了如指掌，收获十分之大。