API模块设计报告

计科 1404 卢涛 3140102441

**一、 模块概述**

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调用。该模块接受interpreter 提供的解析命令，再根据catalog 的信息，来确定执行规则，并调用record manage，index manage，catalog manage来执行相应的操作，并将结果返回给interpreter 模块。

**二、 主要功能**

**Create table**

Create table 创建表

Catalog.createtable

更新catalog中表的信息

Record.createtable

记录中创建表

**drop table**

drop table 删除表

Catalog.droptable

删除catalog中表的信息

Record.droptable 记录中删除表

Index.dropindex

删除表上所有的index

**create index**  create index 删除表

Catalog.createindex

更新catalog中索引的信息

Index.createindex index中创建索引

**Drop index**

Drop index 删除索引

Catalog.dropindex

删除catalog中索引的信息

Index.dropindex index中删除索引

**select**

select 选择

Catalog.getindexinfo

得到catalog中索引的信息

Index.select record.selelct

如果存在index，则调用B+树索引搜索 如果不存在索引，就逐一搜索

**insert**

insert 插入

Record.select

查询是否满足primary key的要求

Index.createvalue B+树更新节点信息

Record.insetRecord 插入记录

**delete**

delete 删除记录

Record.delete

record中删除记录

**三、 具体实现**

1、当interpreter解析出的命令出现语法错误，或者无法执行命令时，打印出错误信息，例如

|  |
| --- |
| case UNKNOW: //unknown query  cout << "Unkonw query ,please chech your input !" << endl;  break;  case NOTABLE: //table 不存在  cout << "Table " << interfaces.tablename << " is not exist,please check your input!" << endl;  break;  case INSERTSIZERR: //insert的属性数目错误  cout << "The number of insert values is not equal to the number of attributes of the table,please check the table!" << endl;  break; |

2、当命令可以执行时，调用相应模块执行命令，例如：

|  |
| --- |
| case CREATETABLE: //create table  interfaces.tableinfo.attriNum = interfaces.tableinfo.attributes.size();  catalog.createtable(interfaces.tableinfo);  record.createTable(interfaces.tableinfo);  cout << "Table " << interfaces.tableinfo.name << " has been created successfully!" << endl;  break;  case CREATINDEX: //create index  if (!interfaces.tableinfo.attributes[interfaces.indexinfo.columnNum].isUnique){  cout << "Column " << interfaces.tableinfo.attributes[interfaces.indexinfo.columnNum].name << " is not unique." << endl;  break;  }  catalog.createindex(interfaces.indexinfo);  index.createIndex(interfaces.tableinfo, interfaces.indexinfo);  catalog.update(interfaces.indexinfo);  cout << "The index " << interfaces.indexinfo.index\_name << "has been created successfully." << endl;  break; |

3、执行完命令后，返回interpreter是否完成操作，并输出执行结果。

void output(Data datas, Table tableinfo, vector <Attribute> columns);

1. **API类**

class API{

public:

bool isdone; //判断操作是否为 quit，返回给interpreter

void execute(interpreter interfaces); //执行解析后的指令

API(){

isdone = false;

}

};

**五 、部分代码**

**Create index**

|  |
| --- |
| case CREATINDEX: //create index  if (!interfaces.tableinfo.attributes[interfaces.indexinfo.columnNum].isUnique){ //判断unique  cout << "Column " << interfaces.tableinfo.attributes[interfaces.indexinfo.columnNum].name << " is not unique." << endl;  break;  }  catalog.createindex(interfaces.indexinfo); //调用catalog模块  index.createIndex(interfaces.tableinfo, interfaces.indexinfo); //调用index模块  catalog.update(interfaces.indexinfo);  cout << "The index " << interfaces.indexinfo.index\_name << "has been created successfully." << endl;  break; |

**Drop table**

|  |
| --- |
| case DROPTABLE: //DROP table  record.dropTable(interfaces.tableinfo); //删除记录  catalog.droptable(interfaces.tableinfo); //删除 catalog里面的table信息  for (i = 0; i < interfaces.tableinfo.attriNum; i++){  indexinfo = catalog.getindexinfo(interfaces.tableinfo, i);  if (indexinfo.index\_name != "")  index.dropIndex(indexinfo); //删除index里面的索引信息  }  cout << "Table " << interfaces.tableinfo.name << " has been dropped successfully." << endl;  break; |

**其余功能不再一一列出.**