详细文档中的添加与修改

1.背景部分增加

本项目的主要命令都参考mysql完成。MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 最流行的关系型数据库管理系统，在 WEB 应用方面MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件之一。MySQL是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，它分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

2.命名规范中修改

1==compare改为compare==1

3.系统模块中增加

|  |  |  |
| --- | --- | --- |
| LexicalAnalysis句法分析模块 | | 传入参数 |
| Analysis() | 从输入流中取出主要命令，调用各命令对参数的分析 |  |
| CommandHelp() | 当输入出现错误时对用户提供帮助 | int调用的帮助项目序号 |
| Use() | 提供USE命令的句法分析 |  |
| Back() | 提供BACK命令的句法分析 |  |
| CreateDataBase() | 提供CREATE DATABASE命令的句法分析 |  |
| CreateTable() | 提供CREATE TABLE命令的句法分析 |  |
| Inster() | 提供INSTER命令的句法分析 |  |
| Update() | 提供UPDATE命令的句法分析 |  |
| Delete() | 提供DELETE命令的句法分析 |  |
| Select() | 提供SELECT命令的句法分析 |  |
| Add() | 提供ADD命令的句法分析 |  |
| Save() | 提供SAVE命令的句法分析 |  |

|  |  |  |
| --- | --- | --- |
| FilesProcessing句法分析模块 | | 传入参数 |
| ChangePath () | 根据输入参数提取出文件路径 |  |
| CreateFolder () | 创建文件夹（即库） | string传入文件路径 |
| DeleteFolde() | 删除文件夹 | string传入文件路径 |
| Rename() | 更改名称 | string原名称，string现名称 |
| WriteTxt() | 将记录写入txt文件中 | string表名，vector记录内容 |
| ReadTxt() | 从txt中读出记录 | string表名，vector记录内容 |

4.接口设计中的增加

用户接口：

本项目中Analysis()是从控制台获取用户输入并且进行分析的主要函数，他从输入中提取到关键命令后将调用函数对参数进行进一步提取。而后CreateDataBase()，CreateTable()等函数将在提取完参数后调用核心处理模块。进而完成用户接口的功能

外部接口：

CreateFolder ()，DeleteFolde()，Rename()等文件及文件夹的创建删除等功能。

WriteTxt()和ReadTxt()函数主要负责完成内存和磁盘的同步

5.目录项目中增加源代码一项

files\_processing .h

#pragma once

//文件处理类的定义

#include<fstream>

#include<iostream>

#include<string>

#include<vector>

#include<direct.h>

using namespace std;

class FilesProcessing

{

public:

static void ChangePath(string path\_name = "");

static void CreateFolder(string dir\_name);

static void DeleteFolder(string dir\_name);

static void Rename(string pre\_name, string new\_name);

static void WriteTxt(string name, vector<string> &record);

static void ReadTxt(string name, vector<string> &record);

};

lexical\_analysis.h

#pragma once

//词法分析类的定义

#include<iostream>

#include<string>

#include<regex>

using namespace std;

class LexicalAnalysis

{

public:

LexicalAnalysis(string str);

~LexicalAnalysis();

void Analysis(); //判断是否合法，并归类命令

void CommandHelp(int n); //显示输入非法并提供命令格式帮助

void Use();

void Back();

void CreateDataBase();

void CreateTable();

void Inster();

void Update();

void Delete();

void Select();

void Add();

void Save();

private:

string mStr; //要处理的字符串

string mKeyWord; //关键字

vector<string> mParameter; //参数表

bool mCorrect; //输入是否合法

};

files\_processing .cpp

#include"files\_processing .h"

void FilesProcessing::ChangePath(string path\_name)

{

if (path\_name.empty()) {

char path[\_MAX\_DIR];

\_getcwd(path, \_MAX\_DIR);

string path0 = path;

int tmp = path0.find\_last\_of("/\\");

path\_name = path0.substr(0, tmp);

}

\_chdir(path\_name.c\_str());

}

void FilesProcessing::CreateFolder(string dir\_name)

{

\_mkdir(dir\_name.c\_str());

}

void FilesProcessing::DeleteFolder(string dir\_name)

{

\_rmdir(dir\_name.c\_str());

}

void FilesProcessing::Rename(string pre\_name, string new\_name)

{

rename(pre\_name.c\_str(), new\_name.c\_str());

}

void FilesProcessing::WriteTxt(string name, vector<string> &record)

{

ofstream ptxt(name, ios::out);

if (!ptxt.is\_open())

cout << "写入文件出错！！" << endl;

int len = record.size();

for (int i = 0; i < len; i++)

ptxt << record[i] << endl;

ptxt.close();

}

void FilesProcessing::ReadTxt(string name, vector<string> &record)

{

ifstream gtxt(name, ios::in);

if (!gtxt.is\_open())

cout << "读取文件出错！！" << endl;

string tmp;

while (getline(gtxt, tmp))

record.push\_back(tmp);

gtxt.close();

}

lexical\_analysis.cpp

#include"files\_processing .h"

void FilesProcessing::ChangePath(string path\_name)

{

if (path\_name.empty()) {

char path[\_MAX\_DIR];

\_getcwd(path, \_MAX\_DIR);

string path0 = path;

int tmp = path0.find\_last\_of("/\\");

path\_name = path0.substr(0, tmp);

}

\_chdir(path\_name.c\_str());

}

void FilesProcessing::CreateFolder(string dir\_name)

{

\_mkdir(dir\_name.c\_str());

}

void FilesProcessing::DeleteFolder(string dir\_name)

{

\_rmdir(dir\_name.c\_str());

}

void FilesProcessing::Rename(string pre\_name, string new\_name)

{

rename(pre\_name.c\_str(), new\_name.c\_str());

}

void FilesProcessing::WriteTxt(string name, vector<string> &record)

{

ofstream ptxt(name, ios::out);

if (!ptxt.is\_open())

cout << "写入文件出错！！" << endl;

int len = record.size();

for (int i = 0; i < len; i++)

ptxt << record[i] << endl;

ptxt.close();

}

void FilesProcessing::ReadTxt(string name, vector<string> &record)

{

ifstream gtxt(name, ios::in);

if (!gtxt.is\_open())

cout << "读取文件出错！！" << endl;

string tmp;

while (getline(gtxt, tmp))

record.push\_back(tmp);

gtxt.close();

}

6.目录中增加测试用例一项

CREATE 学生 （学号 INT,姓名 CHAR,性别 CHAR,生日 DATA,年龄 INT）;

USE 学生:

INSERT

VALUES ("0001", "张三", "男", #1996-1-1#, "20");

INSERT

VALUES ("0001", "张三", "男", #1996-1-1#, "20");（设置两组完全相同的数据）

INSERT

VALUES (0001, "张三", "男", #1996-1-1#, "20");（没加引号，报错）

INSERT

VALUES ("0002", "张四", "女", #1996-1-1#, "20");

INSERT

VALUES ("0003", "张五", "男", #1985-1-1#, "21");

INSERT

VALUES ("0004", "张六", "男", #1995-1-1#, "21");

INSERT

VALUES ("0005", "张七", "女", #1995-1-1#, "21");

INSERT

VALUES ("0006", "张八", "男", #1995-1-1#, "21");

INSERT (学号,姓名,性别)

VALUES ("0007", "李四", "女");

UPDATE SET 年龄= 31 WHERE 姓名="张五";

SELECT \*;（输出所有记录，输出格式没给没办法写预测输出）

SELECT 姓名,性别;（输出姓名，性别两列）

SELECT 姓名,性别；（分号不同）

SELECT 姓名,性别 WHERE 年龄<20；

ADD 成绩 INT;

ADD 成绩 INT;（添加相同表头）

ADD 成绩 IN;（没有的数据类型，报错）

ADD 1 CHAR;

BACK;

CREAT 老师（工号 INT,姓名 CHAR）;

USE 老师；

SELECT \*;（输出空表）

BACK；

USE 学生;

CREATE;（层级不对，应该报错）

INSERT (雇员号,姓名,性别)

VALUES ("0002", "王宏", "男");（表头没有雇员号，应该报错）

UPDATE SET 出生日期= #1960-1-11# WHERE 姓名="张磊";（找不到符合条件的人，应该报错）

DELETE WHERE 学号 = "0002";

SELECT \*;(应该输出中没有2号学生)

SAVE;

(关闭程序再打开)

USE 学生；

SELECT \*;（输出应该和关闭前不变）