# 第一章 绪论

随着计算机技术的迅速发展，各种程序渗透到不同的各行各业，成为几乎为人们生活中不可或缺的工具。大数据已成为时代的潮流，几乎各种数据的处理都会用的计算机程序，简单的则是使用EXCEL这种现成的强大的工具，但是有很多时候这种现成的工具很难满足我们多样化的数处理需求，所以我们需要使用一些高级语言来编写。随着软件设计模式的不断完善，前后端分离以降低模块之间的依赖性，是规范的必须需求，数据与程序分开存储也称为一种流行的趋势。数据与程序分离可以很好地保护数据不被意外的修改或者删除，也不用过多地考虑程序崩溃所带来的数据丢失，将数据存储至远端mysql服务器内，可以方便快速地获取到所需要的数据。

* 1. 研究背景与意义

随着大数据技术的快速发展和进步，数据对于软件的分析能力和分析速度有很大的需求。不同的算法和数据库处理对于程序有着巨大的影响。

C语言是面向过程的语言，常用来编写操作系统，用于计算机软件的底层，仅次于汇编语言，由于缺乏面向对象的支持，一般人使用C语言来编写用户层层序是很困难的。C++语言是从C语言发展过来的，是一门面向对象的语言，它继承了C语言的优势，同时也添加了三个主要的内容：Oriented-Object class,Template,STL. C++面向对象是对C语言的进一步完善，这使得C++在编写大型程序和面向对象的程序时显得十分简单。C++相对与C语言提供了更多的库函数，用户可以轻松方便地使用动态数组vector和双向链表list，以及其他更高级的数据结构和迭代器，这些现成的数据结构可以使得我们更快速地编写程序，而不用自己去实现。

MySQL 是开放源代码的数据库，任何人都可以获取该数据库的源代码。这就使得任何人都可以修正 MySQL 的缺陷，并且任何人都能以任何目的来使用该数据库。MySQL 是一款可以自由使用的数据库。

MySQL 不仅可以在 Windows 系列的操作系统上运行，还可以在 UNIX、Linux 和 Mac OS 等操作系统上运行。因为很多网站都选择 UNIX、Linux 作为网站的服务器，所以 MySQL 的跨平台性保证了其在 Web 应用方面的优势。虽然微软公司的 SQL Server 数据库是一款很优秀的商业数据库，但是其只能在 Windows 系列的操作系统上运行。方便小巧MySQL 数据库使用标准的SQL语句，学习和使用都有很多资料可以查阅。而且没有复杂的配置等问题。占用的空间相对较小。因此，MySQL 数据库的跨平台性是一个很大的优势。我们所做的学生信息管理系统的数据库可以轻松部署到各种各样的平台，拥有很强的可以执行。

# 第二章 程序的设计

介绍了程序的总体设计思路与设计过程。

首先创建主函数以及其头文件，main.cpp和main.h，这是程序的主体用于调用其他类和方法。其次创建管理类management.h和management.cpp，这个类用于管理学生的各种信息，比如添加学生，查找学生，修改学生和删除学生信息等。然后创建Person类以及其头文件,Person.cpp和Person.h，该类用于描述人的信息。最后创建Student类和其头文件，student.cpp,student.h，该类用于描述学生信息继承于Person类，其中包含许多方法。

* 1. 源代码如下

主函数程序如下:

#include <iostream>

#include <list>

#include "management.h"

int main(int argc, char \*argv[])

{

int input;

string name, gender;

Student stu;

Management management("student management");

while (1) {

management.welcome();

cout << "输入你的选项:" << endl;

cin >> input;

switch (input)

{

case 1:

cout << "名字:" << endl;

cin >> name;

cout << "性别:" << endl;

cin >> gender;

stu.setName(name);

stu.setGender(gender);

stu.setScore();

management.addStudent(stu);

cout << "添加成功" << endl;

break;

case 2:

cout << "输入要删除的名字:" << endl;

cin >> name;

management.delStudentByName(name);

cout << "删除成功" << endl;

break;

case 3:

cout << "输入要修改的名字:" << endl;

cin >> name;

management.modifyStudentByName(name);

cout << "修改成功" << endl;

break;

case 4:

cout << "请输入要查询的名字:" << endl;

cin >> name;

management.queryStudentByName(name);

cout << "查询成功" << endl;

break;

cout << "6:) 保存数据到mysql服务器" << endl;

cout << "7:) 从mysql服务器读取数据" << endl;

case 5:

management.showStat();

break;

case 6:

management.saveToMysqlServer();

break;

case 7:

management.readFromMysqlServer();

break;

case 8:

exit(1);

}

}

}

主函数主要生成一个Management对象来管理学生信息，然后使用switch and case来判断用户所输入的选项，最后根据用户所输入的选项执行对应的方法。

Management.cpp代码如下:

#include <iostream>

#include "management.h"

Management::Management(string name)

{

this->name = name;

}

Management::~Management()

{

// delete this;

}

void Management::welcome() {

cout << "1:) 添加学生" << endl;

cout << "2:) 删除学生" << endl;

cout << "3:) 修改学生" << endl;

cout << "4:) 查询学生" << endl;

cout << "5:) 统计" << endl;

cout << "6:) 保存数据到mysql服务器" << endl;

cout << "7:) 从mysql服务器读取数据" << endl;

cout << "8:) 退出" << endl;

}

int Management::addStudent(Student stu) {

this->students.push\_back(stu);

}

int Management::delStudentByName(string name) {

for (list<Student>::iterator it = this->students.begin(); it != this->students.end();

it++) {

if ((\*it).getName() == name) {

this->students.erase(it);

}

}

}

int Management::modifyStudentByName(string name) {

string newName;

string newGender;

for (list<Student>::iterator it = this->students.begin(); it != this->students.end();

it++) {

if ((\*it).getName() == name) {

cout << "成员已找到" << endl;

cout << "请输入新的名字:" << endl;

cin >> newName;

(\*it).setName(newName);

cout << "请输入新的性别:" << endl;

cin >> newGender;

(\*it).setGender(newGender);

cout << "修改成功";

}

}

}

int Management::queryStudentByName(string name) {

for (list<Student>::iterator it = this->students.begin(); it != this->students.end();

it++) {

if ((\*it).getName() == name) {

cout << "成员已找到" << endl;

cout << "姓名:" << (\*it).getName() << endl;

cout << "性别:" << (\*it).getGender() << endl;

cout << "成绩(语文，数学，英语):" << endl;

(\*it).printScore();

}

}

}

int Management::showStat() {

}

int Management::saveToMysqlServer() {

// Management management;

try {

sql::Driver \*driver;

sql::Connection \*con;

sql::Statement \*stmt;

sql::ResultSet \*res;

/\* Create a connection \*/

driver = get\_driver\_instance();

con = driver->connect("tcp://localhost:3306", "test", "testtest");

/\* Connect to the MySQL test database \*/

con->setSchema("test\_database");

stmt = con->createStatement();

for (std::list<Student>::iterator it = this->students.begin();

it != this->students.end(); ++it) {

string name;

string gender;

int yuwenScore, shuxueScore, yingyuScore;

name = it->getName();

gender = it->getGender();

yuwenScore = it->score[0];

shuxueScore = it->score[1];

yingyuScore = it->score[2];

char buf[1024];

char buff[100];

sprintf(buff, "insert into test\_table values( \"%s\", \"%s\", %d, %d, %d);", name.c\_str(), gender.c\_str(), yuwenScore, shuxueScore, yingyuScore);

std::string sqlString = buff;

cout << "sqlstring:" << sqlString << endl;

stmt->execute(sqlString);

}

cout << "保存成功" << endl;

delete stmt;

delete con;

} catch (sql::SQLException &e) {

cout << "# ERR: SQLException in " << \_\_FILE\_\_;

cout << "(" << \_\_FUNCTION\_\_ << ") on line " << \_\_LINE\_\_ << endl;

cout << "# ERR: " << e.what();

cout << " (MySQL error code: " << e.getErrorCode();

cout << ", SQLState: " << e.getSQLState() << " )" << endl;

}

cout << endl;

}

int Management::readFromMysqlServer() {

// Management management;

try {

sql::Driver \*driver;

sql::Connection \*con;

sql::Statement \*stmt;

sql::ResultSet \*res;

/\* Create a connection \*/

driver = get\_driver\_instance();

con = driver->connect("tcp://localhost:3306", "test", "testtest");

/\* Connect to the MySQL test database \*/

con->setSchema("test\_database");

stmt = con->createStatement();

res = stmt->executeQuery("SELECT \* from test\_table");

for (std::list<Student>::iterator it = this->students.begin(); it != this->students.end();it++) {

}

while (res->next()) {

string name;

string gender;

int yuwenScore, shuxueScore, yingyuScore;

Student stu;

name = res->getString(1);

gender = res->getString(2);

stu.setName(name);

stu.setGender(gender);

stu.score[0] = res->getInt(3);

stu.score[1] = res->getInt(4);

stu.score[2] = res->getInt(5);

this->students.push\_back(stu);

}

cout << "远程获取数据成功" << endl;

delete res;

delete stmt;

delete con;

} catch (sql::SQLException &e) {

cout << "# ERR: SQLException in " << \_\_FILE\_\_;

cout << "(" << \_\_FUNCTION\_\_ << ") on line " << \_\_LINE\_\_ << endl;

cout << "# ERR: " << e.what();

cout << " (MySQL error code: " << e.getErrorCode();

cout << ", SQLState: " << e.getSQLState() << " )" << endl;

}

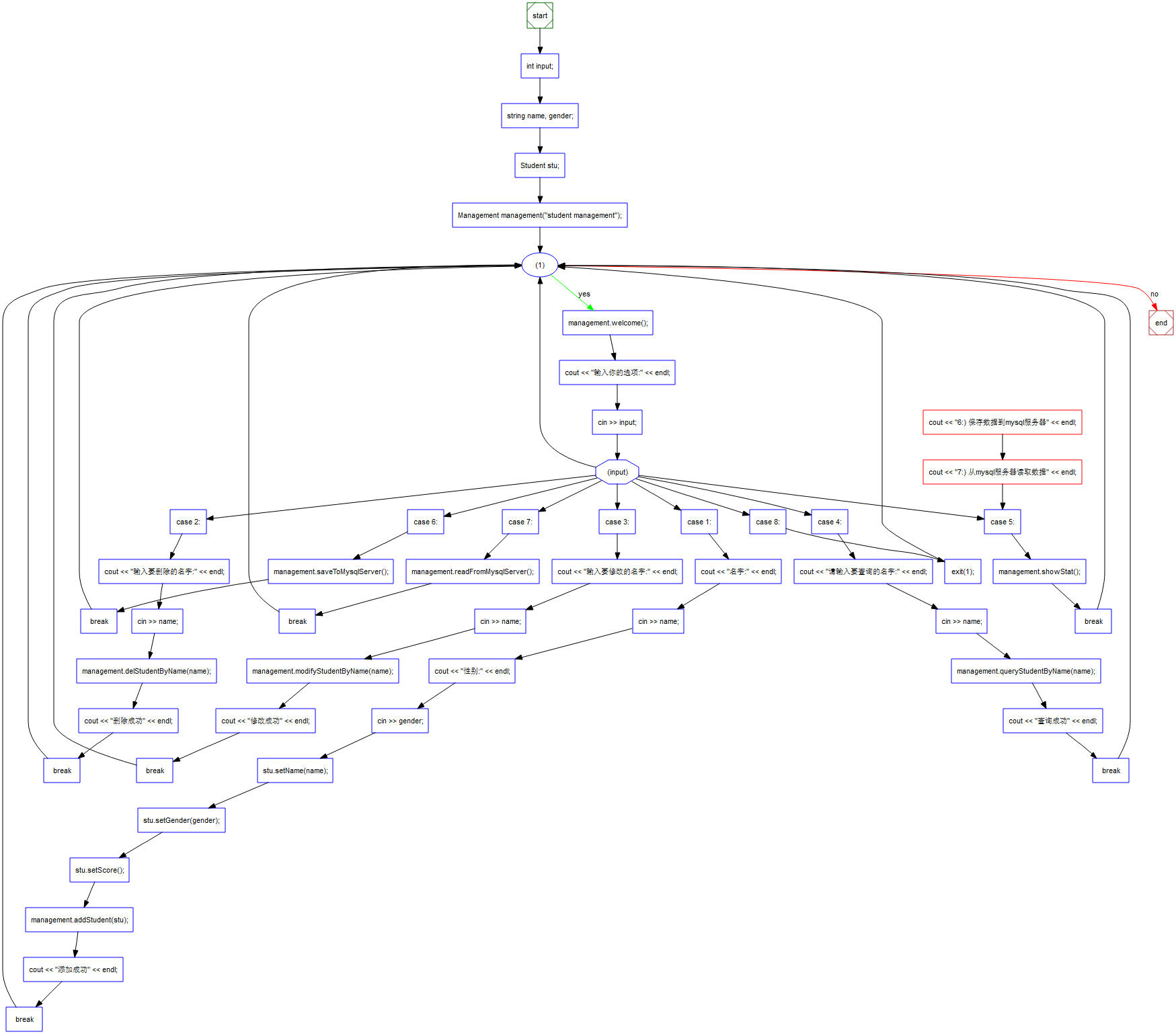
cout << endl;

}

Management.cpp实现了Management类的所有方法。该文件定义了Management的构造函数和析构函数，welcome()函数打印菜单信息，addStudent()添加学生到list链表, delStudentByName()从list链表删除学生, queryStudentByName()从list链表查询数据, modifyStudentByName()函数修改学生数据。saveToMysqlserver()函数将数据保存到远程的mysql服务器，readFromMysqlServer()从远程Mysql服务器获取数据并存入内存。

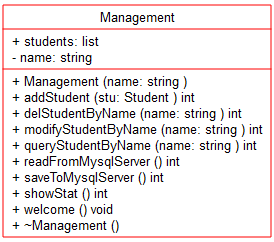
* 1. 流程图和类图

main函数流程图:

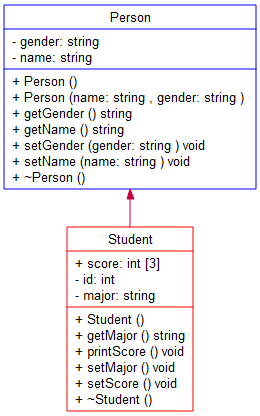


具体类图如下:

management类

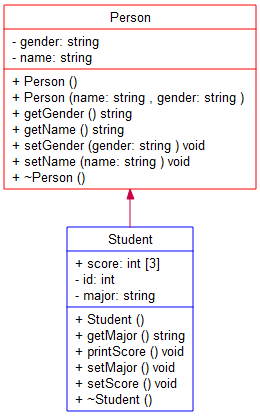


student类



Student类继承于Person类，继承了getName(),getGender(),setName(),setGender()方法。

Person类



# 第三章 程序运行结果

* 1. Mysql服务器

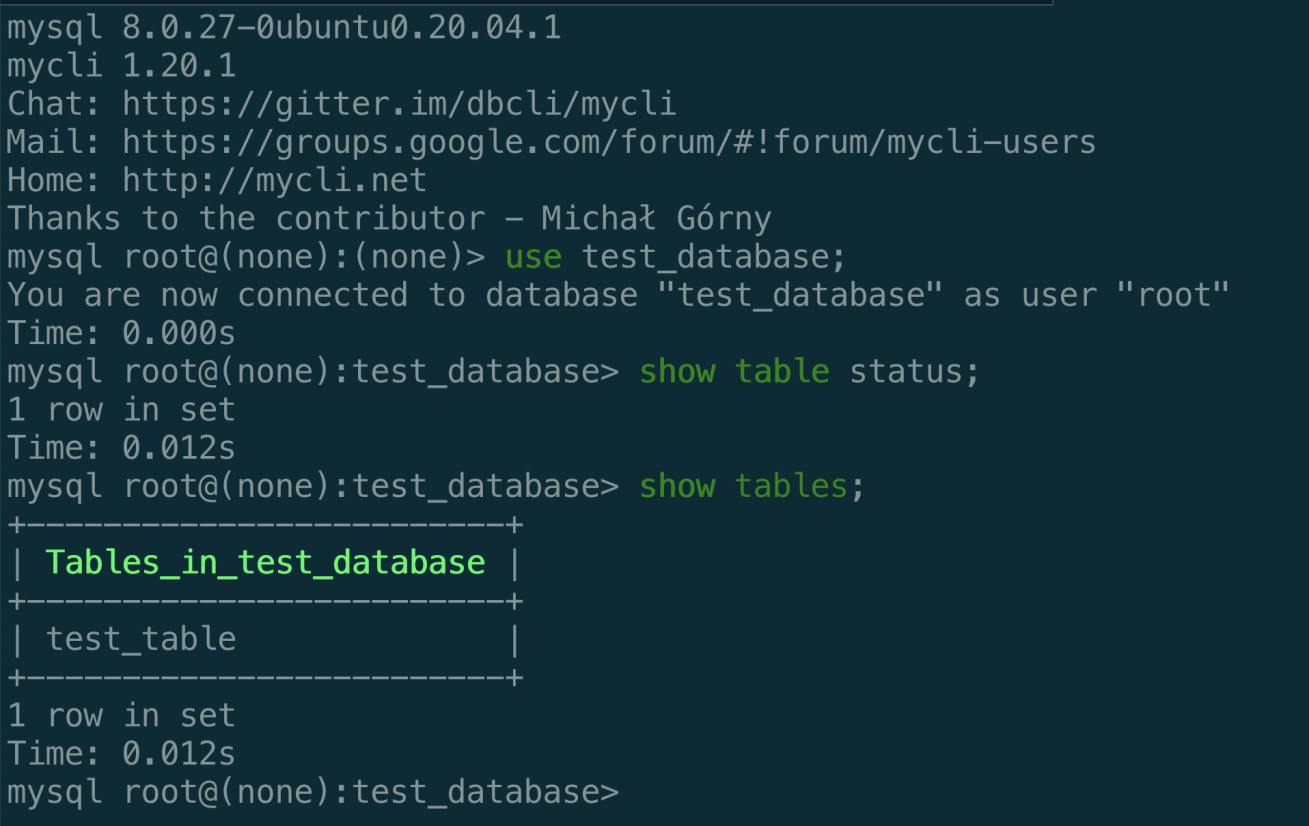


图3-1 不同平台的勒索程序

这里创建了数据库test\_database，和测试数据表test\_table;

Sql语句如下:

Create database test\_database;

create table test\_table (name varchar(20) , gender varchar(20), yuwen

Score int, shuxueScore int, yingyuScore int);

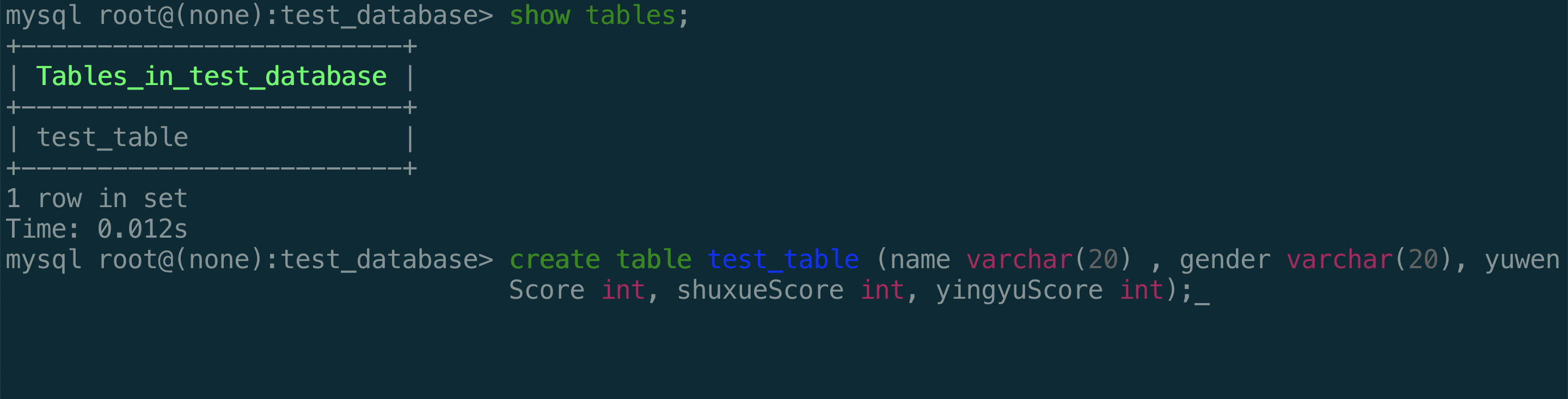


图3-1 创建数据库

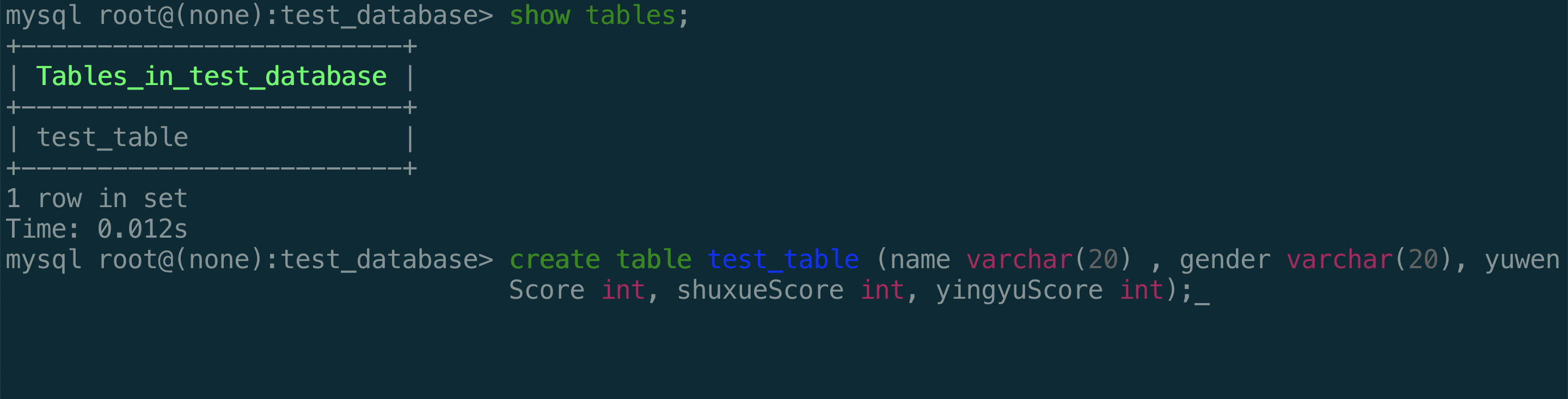


图3-2 创建数据表

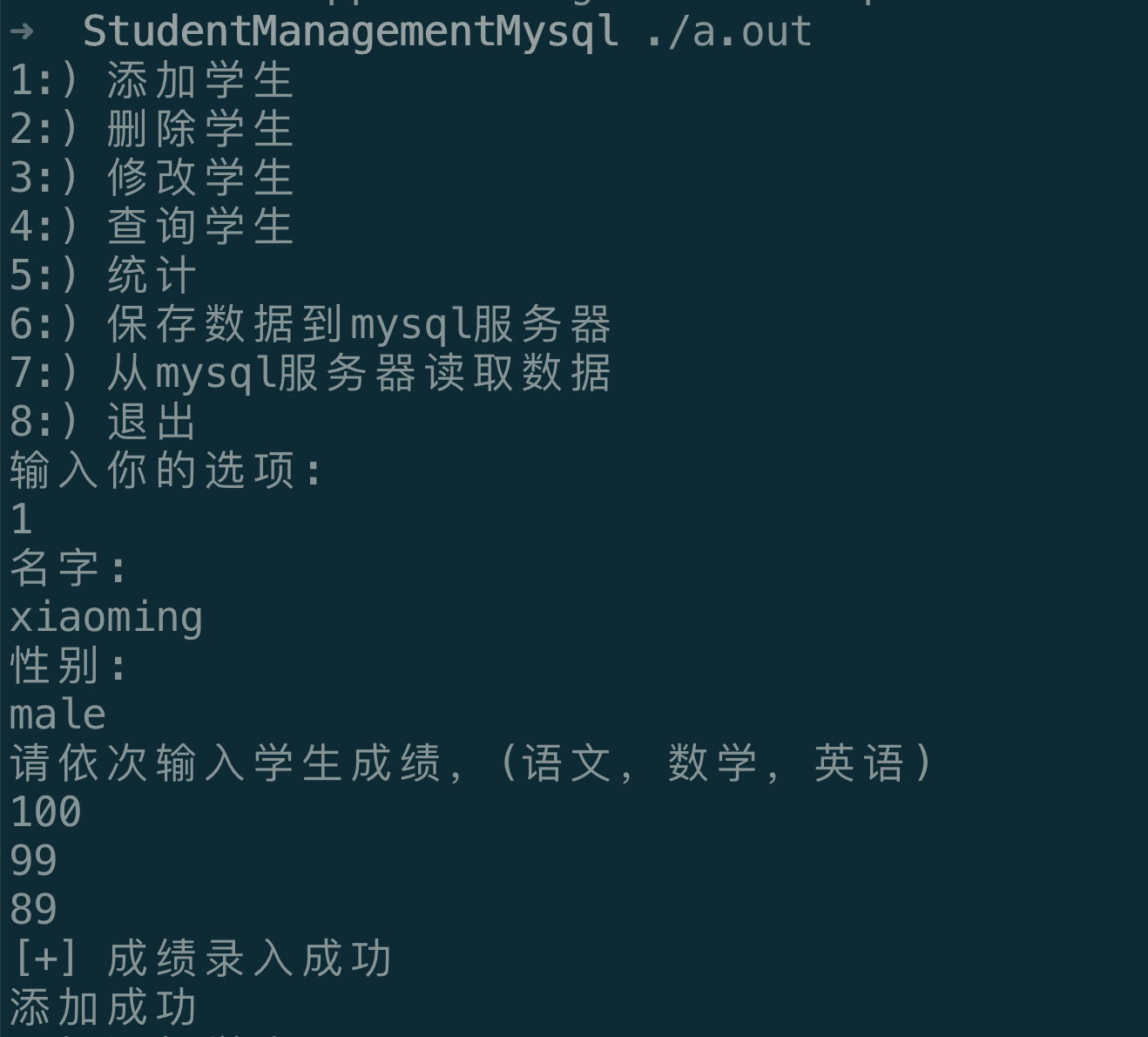


图3-3 添加学生信息

保存数据到服务器，可以看到生成的mysql插入语句是正确的

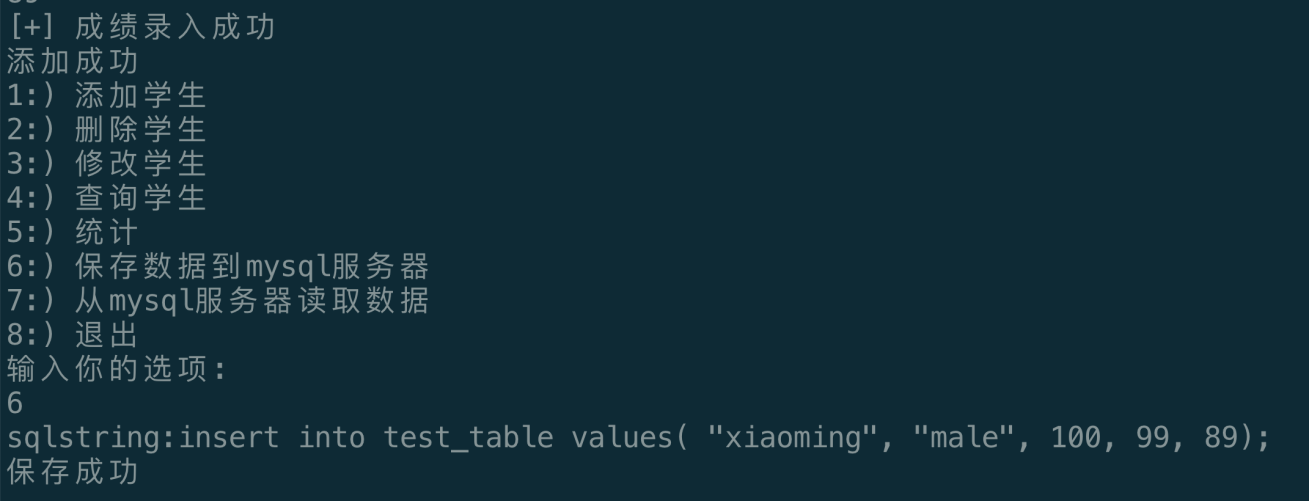


图3-4 保存数据到服务器

使用mycli进入数据库，使用select \* from test\_table where name=”xiaoming”;进行选择查询

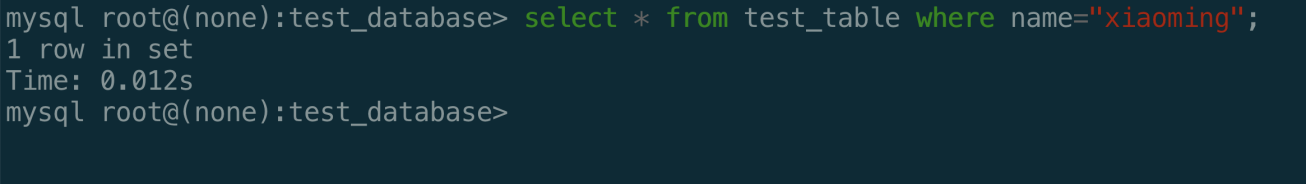


图3-5 使用select查询

可以看到数据库的数据与刚才程序插入的数据一致，证明保存服务器接口是正确的

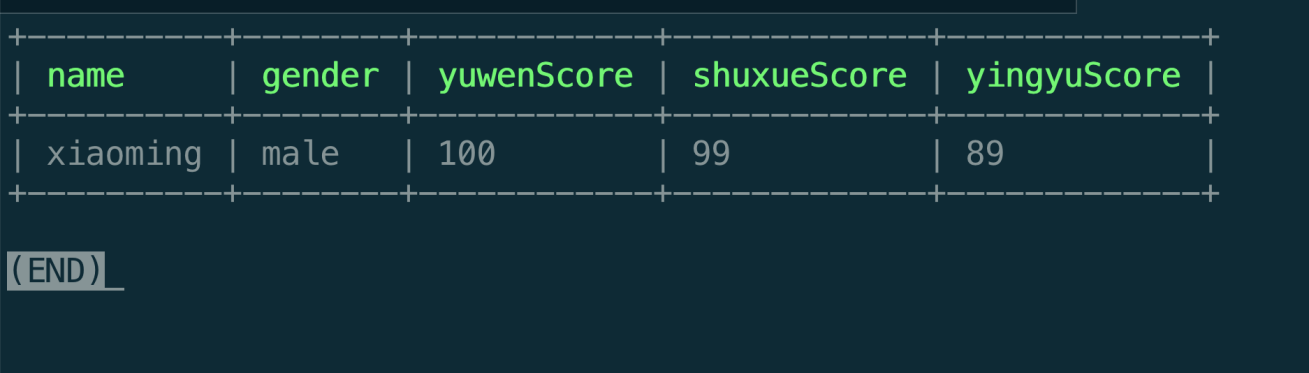
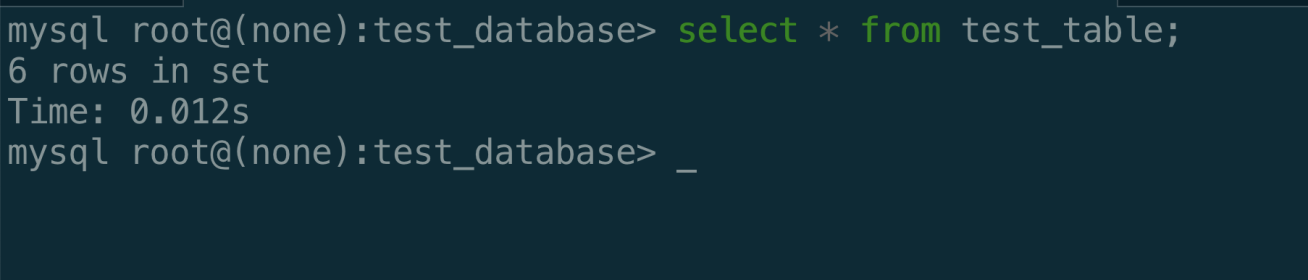


图3-6 使查询结果

程序使用readFromMysqlServer（）从远程Mysql服务器上面读取数据



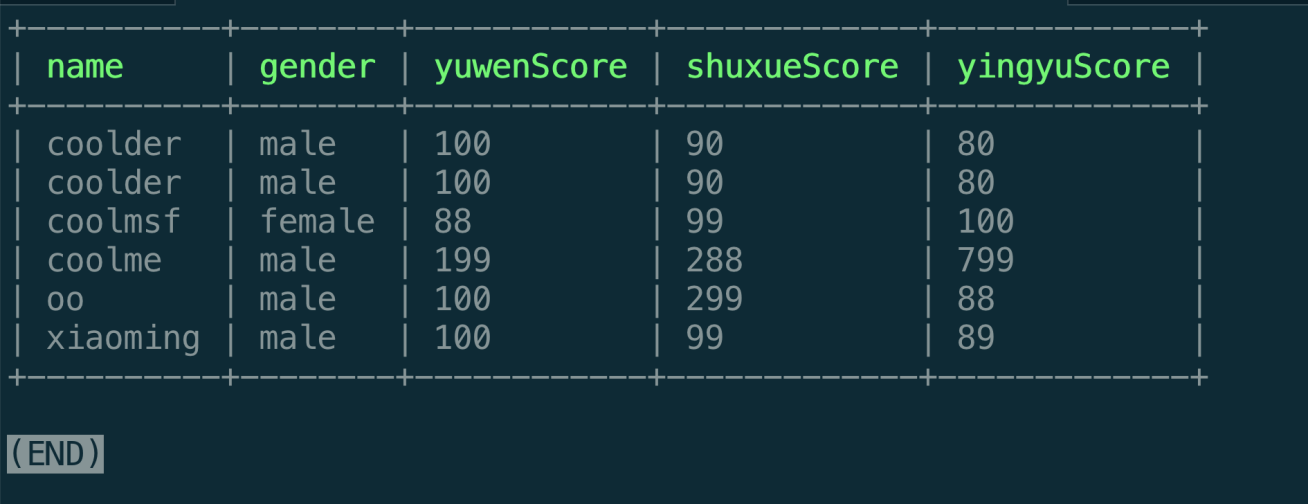
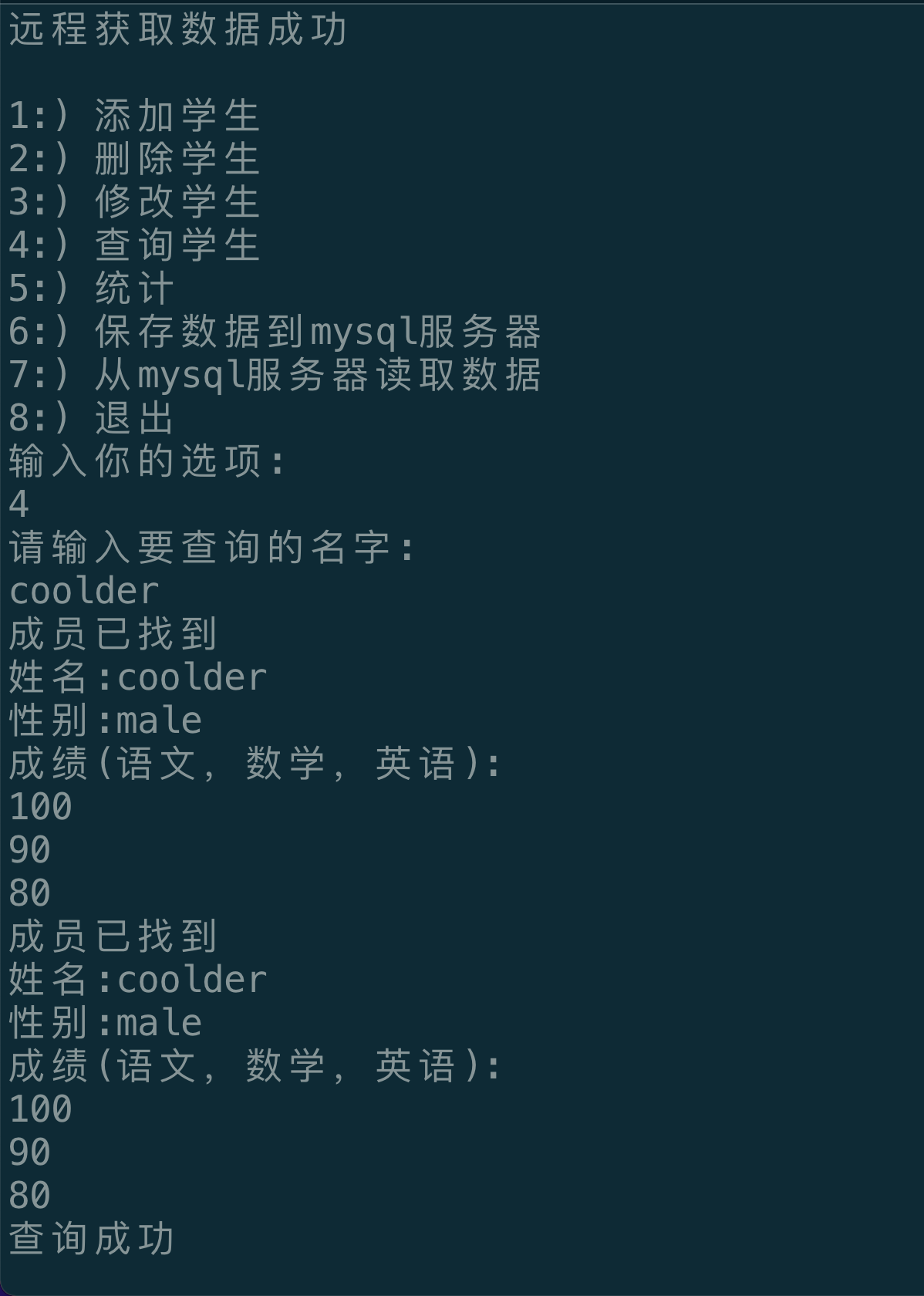


图3- 7 3-8使查询结果

可以看到远程Mysql数据库中test\_table表中的数据如下。



可以看到我们成功获取到了名字为coolder的数据。

* 1. 详细分析

本程序的关键思路如下，首先使用list链表进行添加和删除学生数据信息。当用户使用saveToMysqlServer()函数时，使用迭代器遍历list链表，然后根据链表的数据，如姓名，性别，构造相应的说sql语句，代码如下:

for (std::list<Student>::iterator it = this->students.begin(); it != this->students.end(); it++) {

string name;

string gender;

string sqlString;

int yuwenScore, shuxueScore, yingyuScore;

name = it->getName();

gender = it->getGender();

yuwenScore = it->score[0];

shuxueScore = it->score[1];

yingyuScore = it->score[2];

sqlString = std::format("insert into test\_table values( {}, {}, {}, {}, {});",

name, gender, yuwenScore, shuxueScore, yingyuScore);

stmt->execute(sqlString);

}

使用std::format或者使用C语言的sprintf函数，进行格式化字符串的构造，如果是用的sprintf的话，还需要将C语言字符串char \*,转化为C++的string类型，最后执行execute方法运行sql语句。对于从Mysql服务器获取数据也是类似的，代码如下:

int Management::readFromMysqlServer() {

// Management management;

try {

sql::Driver \*driver;

sql::Connection \*con;

sql::Statement \*stmt;

sql::ResultSet \*res;

/\* Create a connection \*/

driver = get\_driver\_instance();

con = driver->connect("tcp://localhost:3306", "test", "testtest");

/\* Connect to the MySQL test database \*/

con->setSchema("test\_database");

stmt = con->createStatement();

res = stmt->executeQuery("SELECT \* from test\_table");

for (std::list<Student>::iterator it = this->students.begin(); it != this->students.end();it++) {

}

while (res->next()) {

string name;

string gender;

int yuwenScore, shuxueScore, yingyuScore;

Student stu;

name = res->getString(1);

gender = res->getString(2);

stu.setName(name);

stu.setGender(gender);

stu.score[0] = res->getInt(3);

stu.score[1] = res->getInt(4);

stu.score[2] = res->getInt(5);

this->students.push\_back(stu);

}

cout << "远程获取数据成功" << endl;

delete res;

delete stmt;

delete con;

}

使用connector的executeQuery方法获取远程返回的查询数据，然后使用getString和getInt获取并保存相应数据，使用Student类生成一个stu对象， 然后从Mysql服务器获取的数据保存到stu对象中， 将保存的数据使用list的push\_back()方法将数据插入到list链表末尾。最后使用delete关键字，释放之前分配的内存，以防止内存泄漏。数组模板定义了成员函数 begin() 和 end()，分别返回指向第一个元素和最后一个元素的下一个位置的随机访问迭代器。如前面章节所述，随机访问迭代器具有最多的功能，能使用它进行全部的操作。迭代器iterator 提供了一种一般化的方法对顺序或关联容器类型中的每个元素进行连续访问。例如，假设iter为任意容器类型的一个iterator，则++iter 表示向前移动迭代器使其指向容器的下一个元素，而\*iter 返回iterator 指向元素的值，每种容器类型都提供一个begin()和一个end()成员函数。begin()返回一个iterator 它指向容器的第一个元素。end()返回一个iterator 它指向容器的末元素的下一个位置通过迭代器，我们可以用相同的方式来访问、遍历容器。不同容器提供自己的迭代器，所以不同迭代器具有不同的能力。

我们可以在循环中显式地使用迭代器来设置 容器的值。迭代器对象是由 array 对象的成员函数 begin() 和 end() 返回的。使用 auto 时小需要担心迭代的实际类型，除非你要自己考虑，在本例中它们是 std::list<unsigned int,19>::iterator 类型，这意味着 iterator 类型被定义在 list<> 类型中。可以看出，可以像使用普通指针那样上使用迭代器对象。在保存了元素值后，使用后缀 ++ 运算符对 first 进行自增。当 first 等于 end 时，所有的元素都被设完值，循环结束。当迭代器指向容器中的一个特定元素时，它们没有保留任何关于容器本身的信息，所以我们无法从迭代器中判断，它是指向 array容器还是指向 vector 容器。容器中的一段元素可以由迭代器指定，这让我们有了对它们使用算法的可能。我们通过list的迭代器轻松地遍历所有成员。

对于选择vector或者list我们也有考虑。vector类似于C语言中的数组，它维护一段连续的内存空间，具有固定的起始地址，因而能非常方便地进行随机存取，即 [] 操作符，但因为它的内存区域是连续的，所以在它中间插入或删除某个元素，需要复制并移动现有的元素。此外，当被插入的内存空间不够时，需要重新申请一块足够大的内存并进行内存拷贝。值得注意的是，vector每次扩容为原来的两倍，对小对象来说执行效率高，但如果遇到大对象，执行效率就低了。

list类似于C语言中的双向链表，它通过指针来进行数据的访问，因此维护的内存空间可以不连续，这也非常有利于数据的随机存取，因而它没有提供 [] 操作符重载。

deque类似于C语言中的双向队列，即两端都可以插入或者删除的队列。queue支持 [] 操作符，也就是支持随机存取，而且跟vector的效率相差无几。它支持两端的操作：push\_back,push\_front,pop\_back,pop\_front等，并且在两端操作上与list的效率也差不多。或者我们可以这么认为，deque是vector跟list的折中。

# 第四章 程序重难点分析

* 1. 数据库的安装与创建:

我使用的是Ubuntu 18.04，使用apt install mysql-server进行mysql服务器的安装。

因为使用root进行远程登陆十分危险，所以这里创建一个名为test的mysql用户。代码如下:

CREATE USER 'test'@'localhost' IDENTIFIED BY 'testtest';

使用一下代码加权限:

GRANT ALL PRIVILEGES ON database\_name.\* TO test@'localhost';

C++对于mysql数据库的接口:

按照mysql官方的说法，安装mysql connector使用apt就可以安装了，mysql的接口也可以在官方文档看到。

sql::Connection代表到数据库的连接，可以通过sql::Driver来创建。sql::mysql::get\_mysql\_driver\_instance()方法用于获取sql::Driver，通过调用sql::Driver::connect方法来创建sql::Connection对象。

Driver类重载了connect方法，一个接收数据库地址的url、用户名和密码的字符串，后一个接收一个map，map中以key/value的形式包含数据库地址、用户名与密码。使用TCP/IP连接到MySql服务器的url字符串的格式如下："tcp://[hostname[:port]][/schemaname]"。例如：tcp://127.0.0.1:5555/some\_scehma。hostname和端口号是可选的，如果省略，默认是127.0.0.1与3306。如果hostname为"localhost"，会被自动转换为"127.0.0.1"。schemaname也是可选的，如果连接字符串中没有设置schema，需要在程序中通过Connection::setSchema方法手动设置。

在unix系统上，可以通过UNIX domain socket连接运行在本地的MySQL服务，连接字符串格式为："unix://path/to/unix\_socket\_file"，例如：unix:///tmp/mysql.sock。

在windows系统上，可以以命名管道的方式连接到运行在本地的MySQL数据库，连接字符串格式为："pipe://path/to/the/pipe"。MySQL服务必须启动允许命名管道连接，可以在启动MySQL服务器的时候，通过--enable-named-pipe命令行选项来启动该功能。如果没有通过--socket=name选项设置命名管道的名称，系统默认使用MySQL。在windows上，管道的名称是区别大小写的。

如果你不清楚要执行的是像select这样的查询语句还是像update/insert/delete这样的操作语句，可以使用execute方法。对于查询语句，execute()返回True，然后通过getResultSet方法获取查询的结果；对于操作语句，它返回False，通过getUpdateCount方法获取受影响记录的数量。

在一些特殊的情况下，单条SQL语句（如执行存储过程），可能会返回多个结果集 和/或 受影响的记录数量。如果你不想忽略这些结果，通过getResultSet或getUpdateCount方法第一个结果后，再通过getMoreResults()来获取其他的结果集。\