中 国 地 质 大 学

本科生课程论文封面

课程名称 ： 数 据 库 课 程 设 计

教师姓名 ： 王媛妮

学生姓名 ： 牟鑫一

学生学号 ： 20161001764

学生专业 ： 计算机科学与技术

所在院系 ： 计算机学院

时　　间 ： 2019 年 7 月

目　　录

一、 需求分析

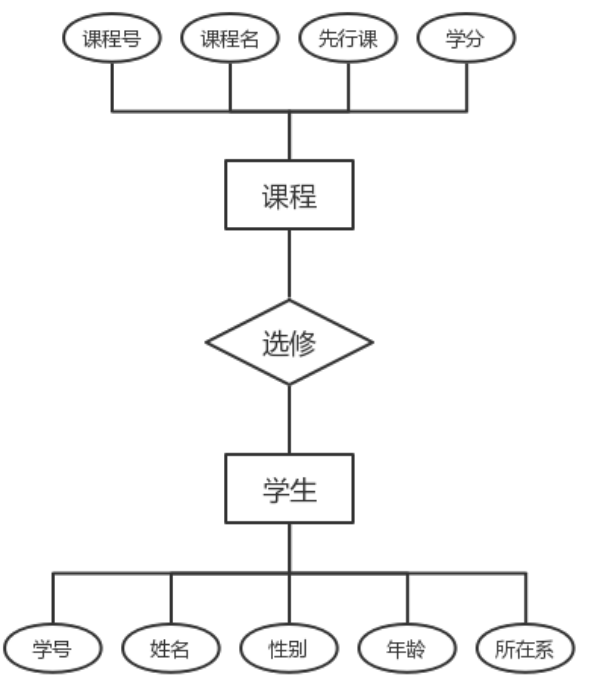
学生有学号、姓名、性别、年龄、所在系等信息，每一门课程有课程号、课程名、先行课、学分等信息，对应每个学生的每门课有一个成绩。

新来的学生，需要登记学生信息，包括分配一个唯一的学号，登记学生的姓名、性别、年龄，为该学生分配其所在的系。学生信息如果有变动，需要更新学生信息。学生毕业之后需要将其从数据库中删除。

新开的一门课程，需要给该课程分配唯一的课程号，登记该课程的课程名，根据专业培养计划为其分配先行课，以及为其分配学分。课程信息变动需要更新课程信息。课程取消之后需要将该课程从数据库中删除。

每个学生的每一门课程有一个成绩，可以根据学生学号以及学生选修的课程号查询这门课程的成绩，可以根据学生的学号查询该学生各门课程的成绩，也可以查询学生的总分。

二、 概念结构设计



三、 逻辑结构设计

1. 学生

创建学生表，包含属性（学号，姓名，性别，年龄，所在系），学号作为唯一能标识学生身份的信息作为学生表的主码，学号为 int 型，姓名、性别以及所在系为 VARCHAR 类型，年龄为 INT(3)型，人的年龄达不到四位数。

```
CREATE TABLE `student` (
  `Sno` int(11) NOT NULL,
  `Sname` varchar(45) DEFAULT NULL,
  `Ssex` varchar(4) DEFAULT NULL,
  `Sage` int(4) DEFAULT NULL,
  `Sdept` char(2) DEFAULT NULL,
  PRIMARY KEY (`Sno`)
)
```

2. 课程

课程表包含属性课程号、课程名、先修课和学分，课程号可以唯一标识一门课程，所以将其作为课程表主码，课程名为 VARCHAR 类型，课程号、先修课以及学分均为 INT 型，学分 INT 型以方便对学分进行计算。

```
CREATE TABLE `course` (
  `Cno` int(2) NOT NULL,
  `Cname` varchar(45) DEFAULT NULL,
  `Cpno` int(2) DEFAULT NULL,
  `Ccredit` int(2) DEFAULT NULL,
  PRIMARY KEY (`Cno`)
)
```

3. 选修

选修关系是学生表和课程表之间的关系，学生表包含学号、课程号和成绩三个属性，根据常识可以知道根据学号和课程号可以唯一的确定某个学生选修的某门课的成绩，所以选修表中将学号和课程号一起作为选修表的主码，同时学号和课程号均是选秀表的外码，成绩为 INT 型，以便于对学生成绩进行计算，可以计算某个学生的总成绩，也可以计算某个学生的平均成绩，或者计算某一门课程的平均成绩。

```
CREATE TABLE `sc` (
  `Sno` int(11) NOT NULL,
  `Cno` int(11) NOT NULL,
  `Grade` int(3) DEFAULT NULL,
  PRIMARY KEY (`Sno`,`Cno`),
  KEY `Cno_idx` (`Cno`),
  CONSTRAINT `Cno` FOREIGN KEY (`Cno`) REFERENCES `course` (`Cno`),
  CONSTRAINT `Sno` FOREIGN KEY (`Sno`) REFERENCES `student` (`Sno`)
)
```

四、 功能分析

1. 学生信息查询

查询所有学生的信息，SQL 语言中只需要一句“SELECT * FROM [表名];”即可，但是嵌入到 C++中复杂不少，教材上的嵌入式 SQL 语句并不能用，或许是使用方法不对，但是即便是网上也很难找到相关的嵌入式 SQL 的实现，所有没法使用，查遍各个网站以及查阅 MySQL 官方文档之后，了解到 MySQL 的 C++实现是通过 C 语言的 API 实现的，当然 C++也能通用，所以这时候的问题就是这些 API 该怎么用了。

具体实现就是通过 MySQL 的 C 语言 API 函数，连接数据库，然后通过 mysql_query() 函数执行“SELECT * FROM student;”。然后查询到的信息并不是像 MySQL 的控制台一样直接以表格形式输出，并且排版美观，查询只会返回一个查询成功或失败的状态，而具体要把数据显示出来就需要通过另外的 API 函数以及循环来遍历查询到的各行上各列的数据，并通过简单的排版将数据展示出来。

Sno	Sname	Ssex	Sage	Sdept
201215121	Ethan	M	20	CS
201215122	William	M	20	CS
201215123	Daniel	W	18	MA
201215125	Alexander	M	19	IS
201912341	Jacob	M	20	IS
201912342	Michael	M	21	CS

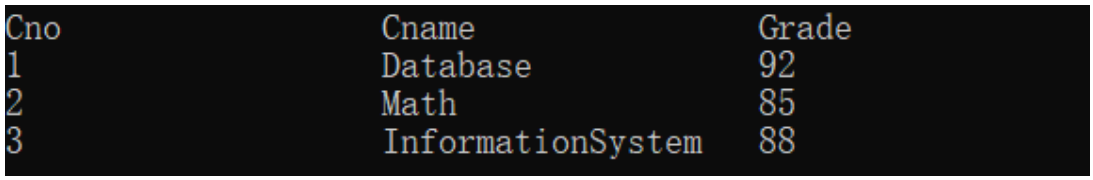
2. 课程信息查询

整体上与上述学生信息查询一致，只需要修改 SQL 语句“SELECT * FROM [表名];”的表名为“course”即可。

Cno	Cname	Cpno	Ccredit
1	Database	5	4
2	Math	NULL	2
3	InformationSystem	1	4
4	OperatingSystem	6	3
5	DataStructure	8	4
6	DataProcessing	NULL	2
7	CalculationMethod	NULL	4
8	C++Primer	2	3
9	Graphics	8	2

3. 学生成绩查询

查询某一个学生的各科成绩，也可以实现对某个学生某门课的成绩的单独查询，但是考虑到一个学生的课程数量并不多，所以直接查询某个学生选修的所有课的成绩。由于查询需要具体到某一个学生，所以需要输入要查询的学生的学号，根据输入的学号直接到 sc 选修表中查询并筛选学号与要查询的学号所一致的学生的各科成绩，并将课程号、课程名以及成绩列表输出。



Cno	Cname	Grade
1	Database	92
2	Math	85
3	InformationSystem	88

4. 学生总分查询

大体与学生成绩查询一致，但不同在于现在我们关心的是学生的总成绩，而不关心某一门课的成绩，所以需要连接学生表和选修表，找出学号与要查询的学号所对应的学生的学号、姓名，通过聚集函数 SUM(Grade)计算学生的总成绩，最终输出学生的学号、姓名和总分。

Sno	Sname	SUM(Grade)
201215122	William	170

5. 学生信息插入

对于有新来的学生的情况，需要将新生的信息添加到数据库中，SQL 中信息插入很简单，但在 C++中要插入数据，如果是常量的话基本上也太难，但要将变量插入到数据库中还是有很大的区别的。

首先，数据插入需要使用 `mysql_query()`函数，其函数定义如下：

```
int mysql_query(MYSQL *mysql, const char *stmt_str)
```

注意到传入的参数为 `const char *`类型的字符串，但是 `const` 就意味着定义之后不能修改，那么怎么把变量整合到一个 `const char *`类型的字符串里就成了最关键的问题。最后想到的解决办法是，先用一个 `string` 类型的变量将 `SQL` 语句和变量连接在一起形成一个完整的 `SQL` 语句，然后再将这个字符串变量通过库函数 `c_str()`转换为 `const char *`类型，如此即可实现变量的插入。

期间还发现了一个很关键的问题，问题的原因就是字符编码，起初我想要插入的人名有中文字符，但是总是无法插入，后来发现，我设置的字符串变量，在通过 `c_str()`函数转换后，遇到中文字符就会自动截断，导致实际传入的字符串不完整，不是一个正确的 `SQL` 语句，所以查询失败，字符编码的问题在很多地方都会遇到，索性将所有信息都改为了英文，可以很好的解决这个问题。

```
输入学生相关信息:
学号: 201912121
姓名: mxy
性别: M
年龄: 21
所在系: CS
数据库已连接
INSERT INTO emp.student (Sno, Sname, Ssex, Sage, Sdept) VALUES (201912121,'mxy','M','21','CS');
INSERT INTO emp.student (Sno, Sname, Ssex, Sage, Sdept) VALUES (201912121,'mxy','M','21','CS');
1 rows affected.
学生mxy的信息已插入
```

Sno	Sname	Ssex	Sage	Sdept
201215121	Ethan	M	20	CS
201215122	William	M	20	CS
201215123	Daniel	W	18	MA
201215125	Alexander	M	19	IS
201912121	mxy	M	21	CS
201912341	Jacob	M	20	IS
201912342	Michael	M	21	CS

6. 课程信息插入

与学生信息插入一致。

```
输入课程相关信息:
课程号: 12
课程名: Science
先行课: 1
学分: 4
数据库已连接
INSERT INTO emp.course (Cno,Cname,Cpno,Ccredit) VALUES (12,'Science',1,4);
INSERT INTO emp.course (Cno,Cname,Cpno,Ccredit) VALUES (12,'Science',1,4);
1 rows affected.
课程Science的信息已插入
```

Cno	Cname	Cpno	Ccredit
1	Database	5	4
2	Math	NULL	2
3	InformationSystem	1	4
4	OperatingSystem	6	3
5	DataStructure	8	4
6	DataProcessing	NULL	2
7	CalculationMethod	NULL	4
8	C++Primer	2	3
9	Graphics	8	2
12	Science	1	4

7. 学生信息删除

学生信息删除是针对某一个学生进行的，而标识某一个学生的属性就是学号，所以需要输入要删除学生的学号，然后根据输入的学号在学生表中查找学号一致的学生进行删除。考虑到对应学号的学生已被删除，那么选修表中也不应该有这个学生的选修信息，要实现这一功能我想到了三种途径：

A. 存储过程

B．触发器

C．C++主语言嵌入 SQL 删除语句

存储过程时最先想到的，功能强大，但是一开始没想到能用主语言调用存储过程，然后也就没想出怎么用存储过程实现该功能；触发器和嵌入式 SQL 也都想到了，但是很明显触发器是一个很好的选择，语法简单，实现起来方便快捷。

后来也想到了存储过程的实现办法了，存储过程创建之后，就相当于一个函数，也可以使用 API 函数来调用存储过程，跟 C++的函数调用很相似。

```
输入要删除的学生的学号：201912121
数据库已连接
string字符串：DELETE FROM emp.student WHERE (Sno = 201912121)
const char*字符串：DELETE FROM emp.student WHERE (Sno = 201912121)
1 rows affected.
学生201912121的信息已删除
```

8. 课程信息删除

与学生信息删除一致。

```
输入要删除课程的课程号：12
数据库已连接
string字符串：DELETE FROM emp.course WHERE (Cno = 12)
const char*字符串：DELETE FROM emp.course WHERE (Cno = 12)
1 rows affected.
课程12的信息已删除
```

9. 学生信息修改

学生信息修改首先需要选定要修改信息的学生，所以同样需要输入要修改信息的学生的学号，通过输入的学号筛选要修改信息的学生，即一个元组，目前数据量较少，并且需求不是很大，所以现在仅针对一个元组的信息进行修改，简单的修改 SQL 语句也是可以实现对多个满足条件的元组进行修改的，即引入通配符。

```
输入要更新信息的学生的学号：201912342
输入要更新的信息：
姓名：Michael
性别：M
年龄：21
所在系：MA
数据库已连接
string字符串：UPDATE emp.student SET Sname = 'Michael',Ssex = 'M',Sage = '21',Sdept = 'MA' WHERE(Sno = '201912342')
const char*字符串：UPDATE emp.student SET Sname = 'Michael',Ssex = 'M',Sage = '21',Sdept = 'MA' WHERE(Sno = '201912342')
1 rows affected.
学生201912342的信息已更新
```

10. 课程信息修改

与学生信息修改一致。

```
输入要更新信息的课程的课程号：6
输入要更新的信息：
课程名：DataProcessing
先行课：5
学分：3
数据库已连接
string字符串：UPDATE emp.course SET Cname = 'DataProcessing',Cpno = '5',Ccredit = '3' WHERE(Cno = '6')
const char*字符串：UPDATE emp.course SET Cname = 'DataProcessing',Cpno = '5',Ccredit = '3' WHERE(Cno = '6')
1 rows affected.
课程6的信息已更新
```

五、 总结感想

首先，这一次课设发现写了界面的同学真的很多，应该说大部分同学都做了界面，而

我就是极少数（说不定是唯一一个）还在使用控制台应用程序的。实在没办法，我确实还不会写界面，这是很大一个问题，所以这个假期要做的一件事就是学会写界面。

然后是关于这次课程设计，我觉得还是学会了很多东西的。课设之前，看了教材上的嵌入式 SQL 编程，觉得真的挺简单，但是真正开始做之后发现，书上那很简单的语法，总之就是用不了，虽然现在也没搞懂那些语法到底是要怎么用的，但至少那些语法很偏，或者很古老吧，几乎没人用，除了教材上，然后才了解到了 MySQL 的 C 语言 API 函数，但是这些 API 函数就不如教材上那样的简单了，并且网上很难找到相关的语法介绍，全英文的 MySQL 官方文档就是最好的介绍；除此之外，也不难发现 C++并不常用来与 MySQL 连接使用，连接 MySQL 的主语言主要是 PHP、Java 和 python，网上也很明显的关于 C++连接 MySQL 的内容很少。

配环境是课设过程中的一大难题，不仅占用了我一半的课设时间，我觉得我也有理由相信环境配置同样占用了大多数别的同学一般的时间。首先是 MySQL 最新版本是第 8 版，然后对应的最高能连接 Visual Studio 2017 版，所以，最新版的 VS2019 并不支持；其次是 MySQL 安装中的一堆组件，最终总结下大概只需要其中五个，一个是必不可少的 MySQL Server，然后是可选的 MySQL Workbench，可选是因为它的功能都能通过 MySQL Server 实现，但 workbench 可以很大程度上减轻操作难度，所以还是装上；然后是连接 VS 用的 MySQL for Visual Studio，应该是必须要装的；C++作为主语言需要装上 Connector/C++，其它语言对应；这个也必须得装 Connector/NET，不懂为什么，但必不可少。然后才是挺复杂的一步，给项目配置环境，每一个新建的项目，要使用 MySQL 的 API 函数都需要对其进行相关的配置，具体操作就是将 MySQL Connector/C++目录下的 include 文件夹路径和 MySQL Server 目录下的 include 文件夹路径添加到项目的 C++包含目录和附加包含目录，将 MySQL Connector/C++目录下的 lib64 文件夹路径和 MySQL Server 目录下的 lib 文件夹路径添加到项目的 C++库目录和附加库目录，将编译模式设为 release x64，然后还需要添加依赖项 libmysql.lib，最后将 MySQL Server 文件夹下的动态链接文件 libmysql.dll 拷贝到系统的动态链接库目录或者项目目录下。很多地方并不是很理解，但是现在即便让我重新配置一遍我也可以很快完成了。

关于嵌入式 SQL，首先 API 函数是首要，其次很关键的一点是变量的处理，简言之 API 函数并没有提供对变量的处理，所以最终对变量的操作的实现其实是通过 C++自身将变量以及字符串连接在一起形成一个完整的 SQL 语句，然后再将其转换为 API 函数能接受的 const char *类型。

整体上，这次课设我觉得我大致了解到了数据库在程序设计中的一些作用，比诉说我们日常使用的各个网站的账号以及密码，对应到数据库中就是一张用户表里的一个元组，我们输入账号和密码，根据账号查找到对应的元组，然后和对应元组中存储的密码进行比对，密码一致即可成功登录，密码不一致即密码错误，登陆失败。感受到了数据库对于数据管理的强大性，以前觉得 C++的数组、链表就可以存储数据，为什么还要数据库，但是如今看来，数据库对数据的操作的确是极其方便快捷的，并且少量的数据用数组、链表尚且还能实现，但是如果是上百万、千万的数据了，这就是数据库管理系统存在的意义了。