**《数据库系统原理》实验报告**

**实验题目：数据类型的使用**

**姓名： 郑德凯** **实验日期： 2024年 11月 19日**

**实验内容及完成情况：**

# 一、实验目的

熟练掌握使用Navicat和Transact-SQL语言两种方法创建、修改和删除表。掌握管理表的有关系统存储过程。

# 实验内容

1.了解并尝试使用MySQL的各数据类型，不作实验操作内容上的硬性要求。

1.1日期和时间数据类型



1.2整型

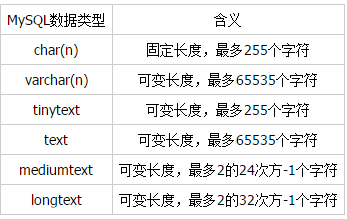


上面定义的都是有符号的，也可以加上unsigned关键字，定义成无符号的类型，那么对应的取值范围就要翻番了，比如：tinyint unsigned的取值范围为0~255。

1.3浮点型



1.4字符串类型



①char（n）和varchar（n）中括号中n代表字符的个数，并不代表字节个数，所以当使用了中文的时候(UTF8)意味着可以插入m个中文，但是实际会占用m\*3个字节。

②同时char和varchar最大的区别就在于char不管实际value都会占用n个字符的空间，而varchar只会占用实际字符应该占用的空间+1，并且实际空间+1<=n。

③超过char和varchar的n设置后，字符串会被截断。

④char在存储的时候会截断尾部的空格，varchar和text不会。

⑤ varchar会使用1-3个字节来存储长度，text不会。

1.5其他类型

① enum(“member1″, “member2″, … “member65535″)：enum数据类型就是定义了一种枚举，最多包含65535个不同的成员。当定义了一个enum的列时，该列的值限制为列定义中声明的值。如果列声明包含NULL属性，则NULL将被认为是一个有效值，并且是默认值。如果声明了NOT NULL，则列表的第一个成员是默认值。

②set(“member”, “member2″, … “member64″)：set数据类型为指定一组预定义值中的零个或多个值提供了一种方法，这组值最多包括64个成员。值的选择限制为列定义中声明的值。

2.了解MySQL数据类型属性，并依次实际操作、验证。

2.1 auto\_increment

auto\_increment能为新插入的行赋一个唯一的整数标识符。为列赋此属性将为每个新插入的行赋值为上一次插入的ID+1。

MySQL要求将auto\_increment属性用于作为主键的列。此外，每个表只允许有一个auto\_increment列。例如：

id smallint not null auto\_increment primary key

2.2 binary

binary属性只用于char和varchar值。当为列指定了该属性时，将以区分大小写的方式排序。与之相反，忽略binary属性时，将使用不区分大小写的方式排序。例如：

hostname char(25) binary not null

2.3 default

default属性确保在没有任何值可用的情况下，赋予某个常量值，这个值必须是常量，因为MySQL不允许插入函数或表达式值。此外，此属性无法用于BLOB或TEXT列。如果已经为此列指定了NULL属性，没有指定默认值时默认值将为NULL，否则默认值将依赖于字段的数据类型。例如：

subscribed enum('0', '1') not null default '0'

2.4 index

如果所有其他因素都相同，要加速数据库查询，使用索引通常是最重要的一个步骤。索引一个列会为该列创建一个有序的键数组，每个键指向其相应的表行。以后针对输入条件可以搜索这个有序的键数组，与搜索整个未索引的表相比，这将在性能方面得到极大的提升。

create table employees

(

id varchar(9) not null,

firstname varchar(15) not null,

lastname varchar(25) not null,

email varchar(45) not null,

phone varchar(10) not null,

index lastname(lastname),

primary key(id)

);

我们也可以利用MySQL的create index命令在创建表之后增加索引：

create index lastname on employees (lastname(7));

这一次只索引了名字的前7个字符，因为可能不需要其它字母来区分不同的名字。因为使用较小的索引时性能更好，所以应当在实践中尽量使用小的索引。

2.5 not null

如果将一个列定义为not null，将不允许向该列插入null值。建议在重要情况下始终使用not null属性，因为它提供了一个基本验证，确保已经向查询传递了所有必要的值。

2.6 null

为列指定null属性时，该列可以保持为空，而不论行中其它列是否已经被填充。记住，null精确的说法是“无”，而不是空字符串或0。

2.7 primary key

primary key属性用于确保指定行的唯一性。指定为主键的列中，值不能重复，也不能为空。为指定为主键的列赋予auto\_increment属性是很常见的，因为此列不必与行数据有任何关系，而只是作为一个唯一标识符。主键又分为以下两种：

①单字段主键: 如果输入到数据库中的每行都已经有不可修改的唯一标识符，一般会使用单字段主键。注意，此主键一旦设置就不能再修改。

②多字段主键: 如果记录中任何一个字段都不可能保证唯一性，就可以使用多字段主键。这时，多个字段联合起来确保唯一性。如果出现这种情况，指定一个auto\_increment整数作为主键是更好的办法。

2.8 unique

被赋予unique属性的列将确保所有值都有不同的值，只是null值可以重复。一般会指定一个列为unique，以确保该列的所有值都不同。例如：

email varchar(45) unique

2.9 zerofill

zerofill属性可用于任何数值类型，用0填充所有剩余字段空间。例如，无符号int的默认宽度是10；因此，当“零填充”的int值为4时，将表示它为0000000004。例如：

orderid int unsigned zerofill not null

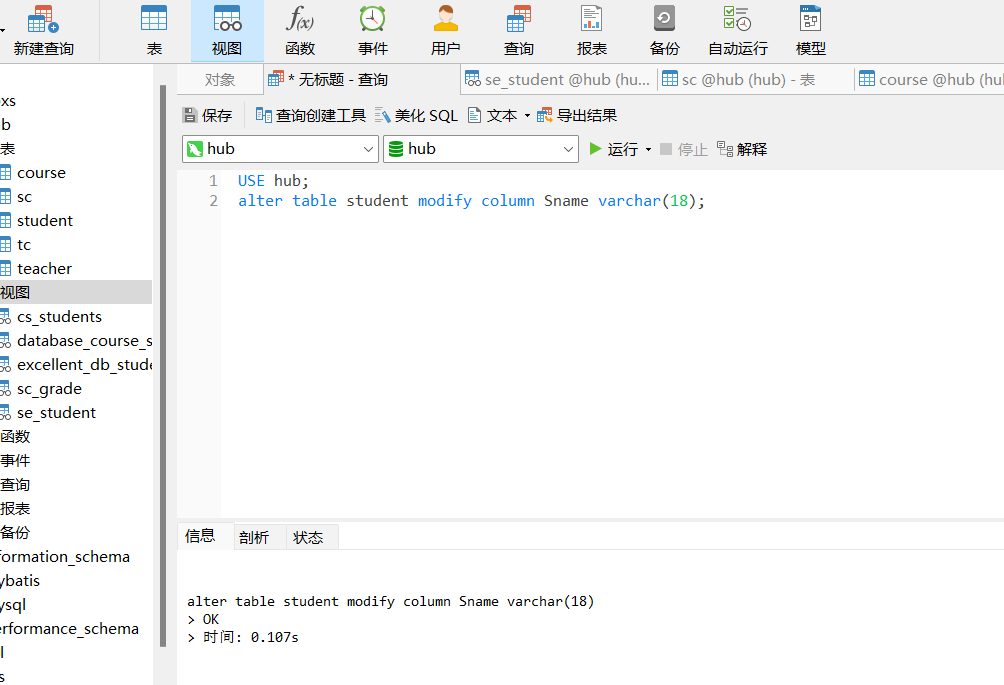
1. 如果要设计一个表用来存放商品的基本信息，包括商品编号、商品名称、品牌商标、型号、产地、生产厂商、生产日期、保质期、进货价格、销售价格和商品图片等。请设计出这个表的表结构，包括表名、列名、列的数据类型和长度等等。

### 表结构：

| **列名** | **数据类型** | **长度/精度** | **是否允许为空** | **主键** | **备注** |
| --- | --- | --- | --- | --- | --- |
| product\_id | VARCHAR | 50 | 否 | 是 | 商品编号，唯一标识商品 |
| product\_name | VARCHAR | 100 | 否 | 否 | 商品名称 |
| brand | VARCHAR | 100 | 是 | 否 | 品牌商标 |
| model | VARCHAR | 50 | 是 | 否 | 型号 |
| origin | VARCHAR | 100 | 是 | 否 | 产地 |
| manufacturer | VARCHAR | 100 | 是 | 否 | 生产厂商 |
| manufacture\_date | DATE | N/A | 是 | 否 | 生产日期 |
| shelf\_life | INT | N/A | 是 | 否 | 保质期（单位：天） |
| purchase\_price | DECIMAL | 10, 2 | 是 | 否 | 进货价格（保留两位小数） |
| sale\_price | DECIMAL | 10, 2 | 是 | 否 | 销售价格（保留两位小数） |
| product\_image | VARCHAR | 255 | 是 | 否 | 商品图片，存储图片URL路径或文件名 |

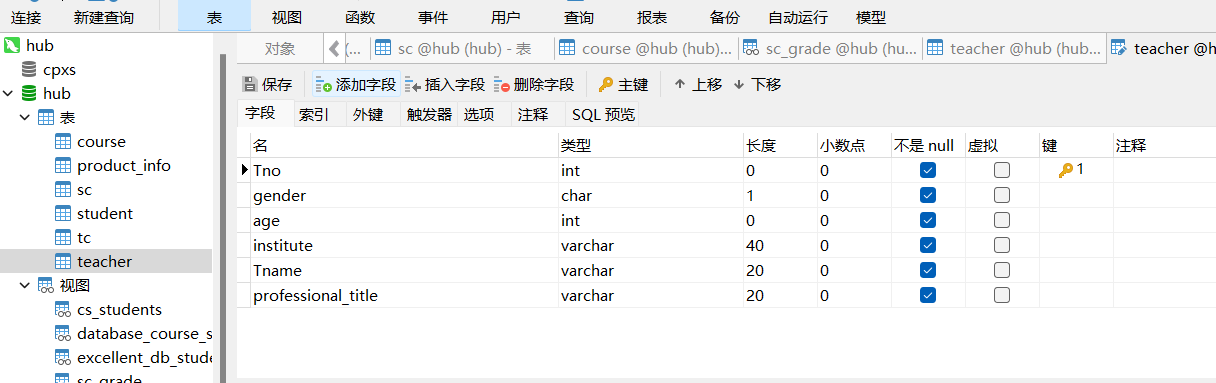


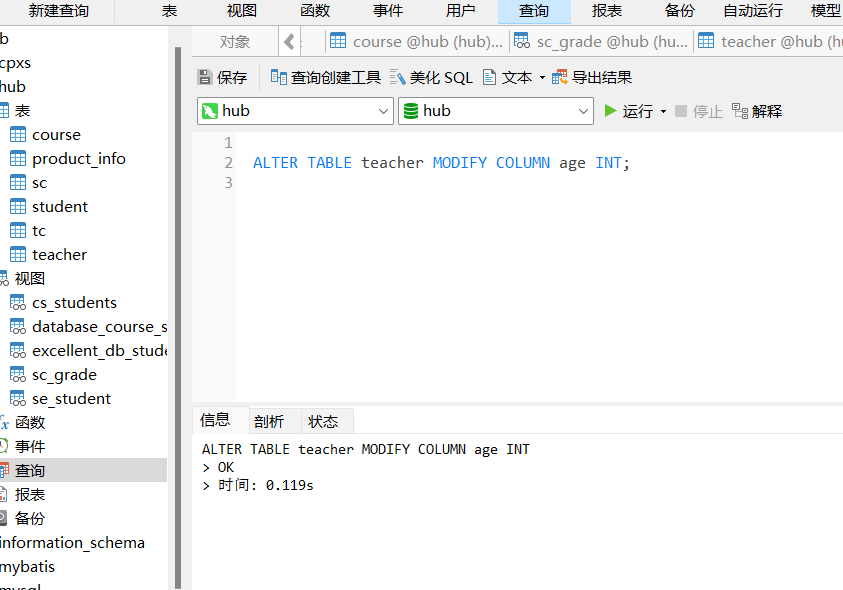
1. 将数据库hub中student表的Sname字段的数据类型改为varchar（18）



# 课后练习题

1. 用Navicat和SQL语言两种方式将数据库hub中teacher表的age字段的数据类型改为int.





2. 在MySQL中建立一个表，有一列为float(5, 3).做以下试验：

①插入12.345，成功则查询该表得到结果；

②插入12.3456，成功则查询该表得到结果；

③插入123.456，成功则查询该表得到结果。

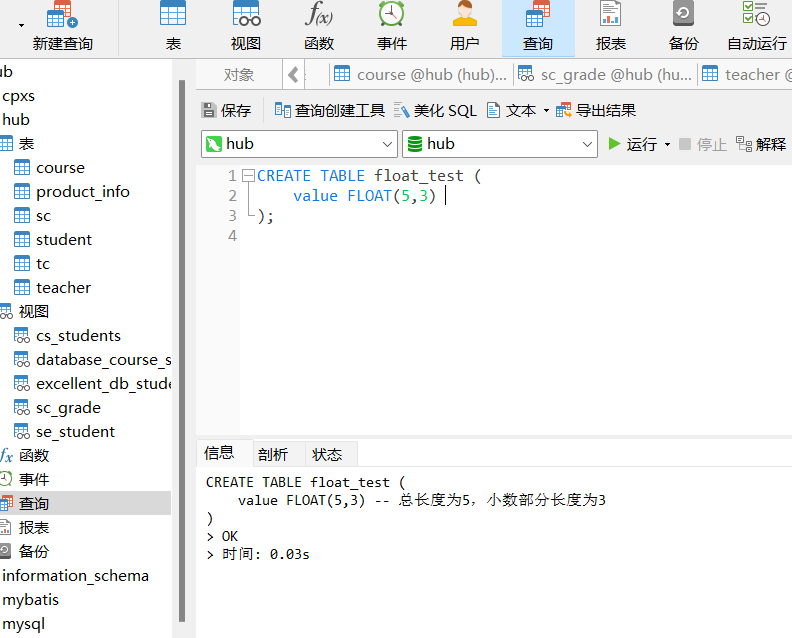
比较三次结果的差异，并分析原因

首先创建表：

CREATE TABLE float\_test (

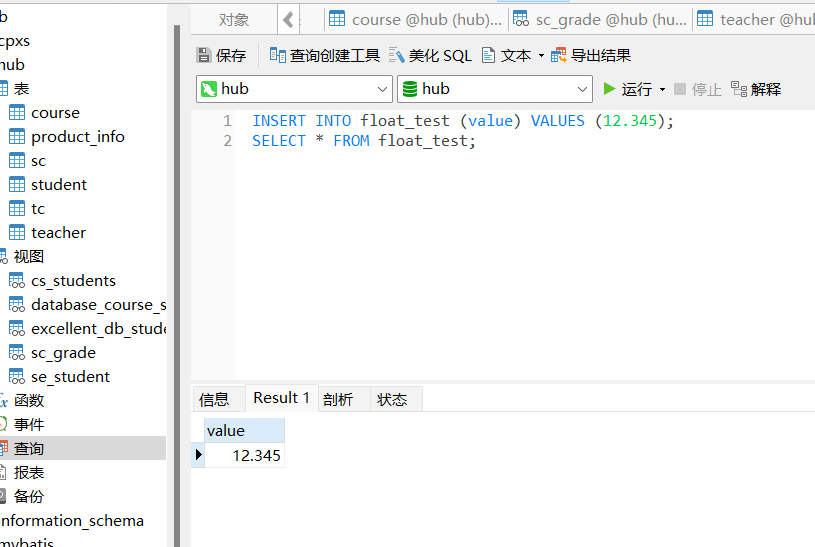
value FLOAT(5,3)

);



试验操作及结果

1. 插入 12.345



结果：

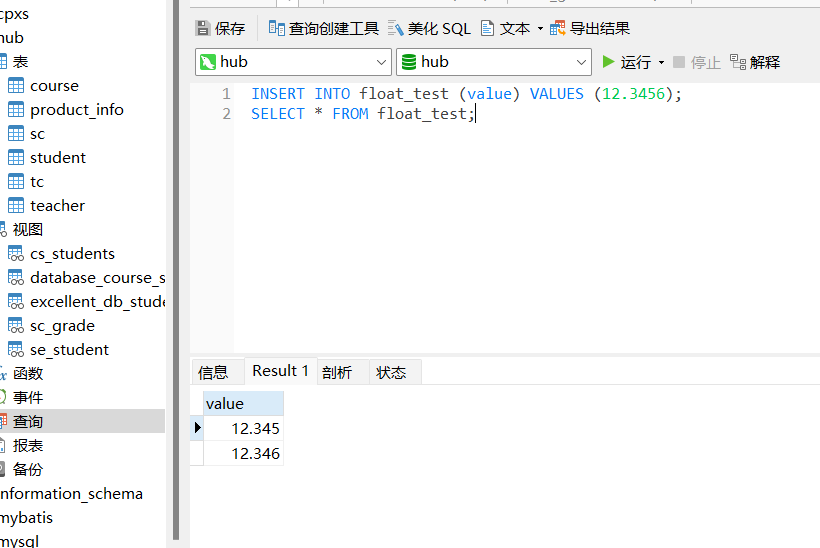
插入成功。

查询结果为 12.345。

原因：

FLOAT(5,3) 的定义表示总共可存储 5位有效数字（包括整数部分和小数点后的小数部分），其中 小数部分占3位。12.345 符合限制（整数部分2位 + 小数点 + 小数部分3位 = 总共5位）。

② 插入 12.3456

结果：

插入成功。

查询结果为 12.346。

原因：

FLOAT(5,3) 限制小数部分为3位，因此插入 12.3456 时，小数部分会自动四舍五入为 12.346。

1. 插入 123.456



结果：

插入失败，出现错误：

原因：

123.456 的整数部分有3位，加上小数点和小数部分总计 6位有效数字，超出了 FLOAT(5,3) 的限制，因此插入失败。

**实验总结：**

本次实验围绕数据库的基本操作展开，重点学习了使用 Navicat 和 Transact-SQL 两种方法进行表的创建、修改、删除以及表管理的相关操作。通过实验，熟练掌握了以下内容：

1. **表的创建与定义**：学会通过图形化工具和 SQL 语句定义表结构，包括字段类型、约束条件等。
2. **表的修改与维护**：理解如何通过两种方法对表结构进行调整，如字段类型的修改、列的添加和删除。
3. **表的删除与清理**：能够使用 DROP TABLE 和 Navicat 删除表，确保数据和表结构的有效管理。
4. **系统存储过程的应用**：学习了管理表的相关系统存储过程，为高效管理数据库提供了可靠手段。

通过本次实验，进一步熟悉了图形化操作和 SQL 脚本的结合使用，为今后更灵活地管理数据库奠定了基础。

**教师评语及成绩**：