### 数据库课程体系

在PHP阶段，将数据库分为三个阶段：

基础阶段：6天，mysql数据库的基本操作（增删改查），以及一些高级操作（视图，触发器，函数，存储过程等），和PHP操作mysql数据库

优化阶段：如何搭建真实的环境系统，如服务器集群，负载均衡

### 数据库基础

1. 什么是数据库？

数据库：database,存储数据的仓库

数据库：高效的存储和处理数据的介质（介质主要是两种：磁盘和内存）

1. 数据库的分类？

数据库基于存储介质的不同：进行了分类，分为两类：关系型数据库（SQL）和非关系型数据库（NoSQL,不是关系型的数据库都叫做非关系型数据库）

1. 不同的数据库阵营中的产品有哪些？

关系型数据库

大型：Oracle,DB2

中型：SQL-SERVER,Mysql等

小型：access等

非关系型数据库：memcached,mongodb,redis(同步到磁盘)

1. 两种数据库阵营的区别？

关系型数据库：安全（保存磁盘基本不可能丢失），容易理解，比较浪费空间（二维表）

非关系型数据库：效率高，不安全（断电丢失）

#### 关系型数据库

1. 什么是关系型数据库？

关系型数据库：是一种建立在关系模型（数学模型）上的数据库。

关系模型：一种所谓建立在关系上的模型，关系模型包含三个方面：

数据结构：数据存储的问题，二维表（有行有列）

操作指令集合：所以SQL语句

完整性约束：表内数据约束（字段与字段），表与表之间约束（外键）

1. 关系型数据库的设计？

关系型数据库：从需要存储的数据需求中分析，如果是一类数据（实体）应该设计成一张二维表，表是由（字段名：用来规定数据的名字）和数据部分组成（实际存储的数据单元）

二维表：行和列

|  |  |  |
| --- | --- | --- |
| 表头 | 字段名1 | 字段名2 |
| 数据单元 | 数据1 | 数据2 |

以实际案例来进行处理：分析一个教学系统，讲师负责教学，教学生，在教室教学生。

①**,**找出系统中存在的实体：讲师表，学生表，班级表

②找出实体中应该存在的数据信息

讲师：姓名，性别，年龄，工资

学生：姓名，性别，学号，学科

班级：班级名字，教室编号

关系型数据库：维护实体内部，实体与实体之间的联系

实体内部联系：每个学生都有姓名，性别，学号，学科信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 性别 | 学号 | 学科 | 年龄 |
| 陈明 | 男 | Itcast0001 | PHP | 20 |
| 陈红 | 女 | Itcast0002 | PHP |  |
| 王威 |  | Itcast00003 | UI |  |

第二行的所有字段，都是在描述陈明这个学生（内部联系）；第二列只能放性别（内部约束）

关系型数据库的特点之一：如果表中对应的某个字段没有值（数据），但是系统依然要分配空间：关系型数据库比较浪费空间

实体与实体之间的联系：每个学生肯定属于某个班级，每个班级一定有多个学生（一对多）

学生表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 性别 | 学号 | 学科 | 年龄 |
| 陈明 | 男 | Itcast0001 | PHP | 20 |
| 陈红 | 女 | Itcast0002 | PHP |  |
| 王威 |  | Itcast00003 | UI |  |

班级表

|  |  |
| --- | --- |
| 班级表 | 教室编号 |
| PHP0801 | B205 |
| PHP0710 | A203 |

解决方案：在学生表中增加一个班级字段来指向班级（必须能够唯一的找到一个班级信息）

实体内部联系：每个学生都有姓名，性别，学号，学科信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 性别 | 学号 | 学科 | 年龄 | 班级信息 |
| 陈明 | 男 | Itcast0001 | PHP | 20 | PHP0801 |
| 陈红 | 女 | Itcast0002 | PHP |  | PHP0801 |
| 王威 |  | Itcast00003 | UI |  | PHP0801 |

学生实体与班级实体的关联关系：实体与实体之间的关系

#### 关键字说明

数据库：database

数据库系统：DBS(Database System):是一种虚拟系统，将多种内容关联起来的称呼

DBS = DBMS + DB

DBMS:Database Management System，数据管理系统，专门管理数据库

DBA：Database Administrator，数据库管理员

行/记录：row/record，本质是一个东西：都是指表中的一行（一条记录）：行是从结构角度出发，记录是从数据角度出发

列/字段：column/field,本质是一个东西

#### SQL

SQL:Structured Query Language，结构化查询语言（数据以查询为主：99%是在进行查询操作）

SQL分为三个部分

DDL：Data Definition Language，数据定义语言，用来维护存储数据结构（数据库，表），代表指令：create,drop,alter等

DML:Data Manipulation Language，数据操作语言，用来对数据进行操作（数据表中的内容），代表指令：insert,delete,update等：其中DML内部又单独进行了一个分类：DQL(Data Query Language：数据查询语言，如select)

DCL:Data Control Language，数据控制语言，主要是负责权限管理（用户），代表指令：grant,revoke等

SQL是关系型数据库的操作指令，SQL是一种约束，但不强制（类似W3C）：不同的数据库产品（如Oracle,mysql）可能内部会有一些细微区别。

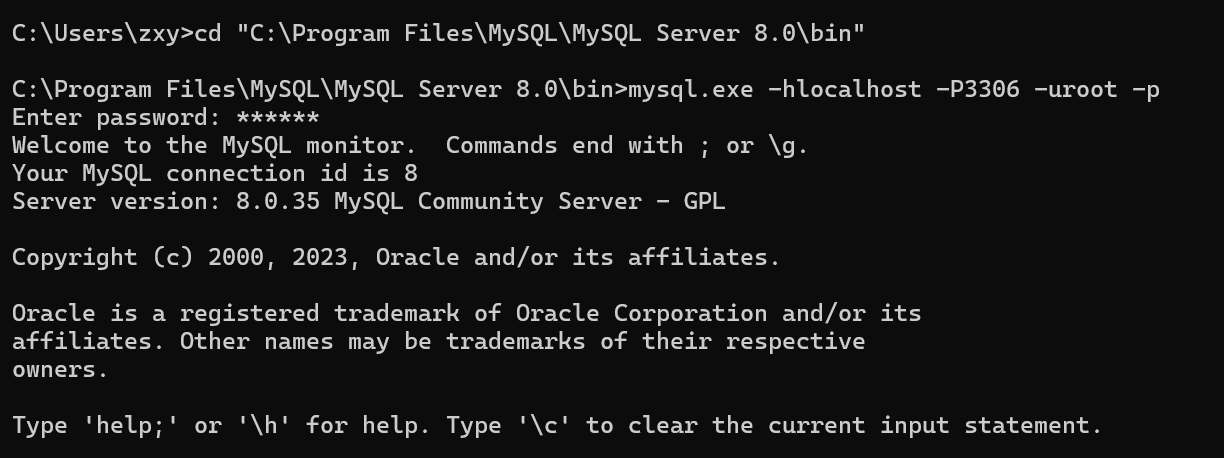
#### Mysql数据库

Mysql数据库是一种c/s结构的软件：客户端/服务端，若想访问服务器必须通过客户端（服务器一直运行，客户端在需要使用的时候运行）。

交互方式

1. 客户端连接认证：连接服务器，认证身份：mysql.exe –hpup

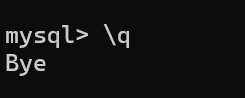
cd "C:\Program Files\MySQL\MySQL Server 8.0\bin"



1. 发送SQL指令
2. 服务器接收SQL指令：处理SQL指令：返回操作结果
3. 客户端接收结果：显示结果



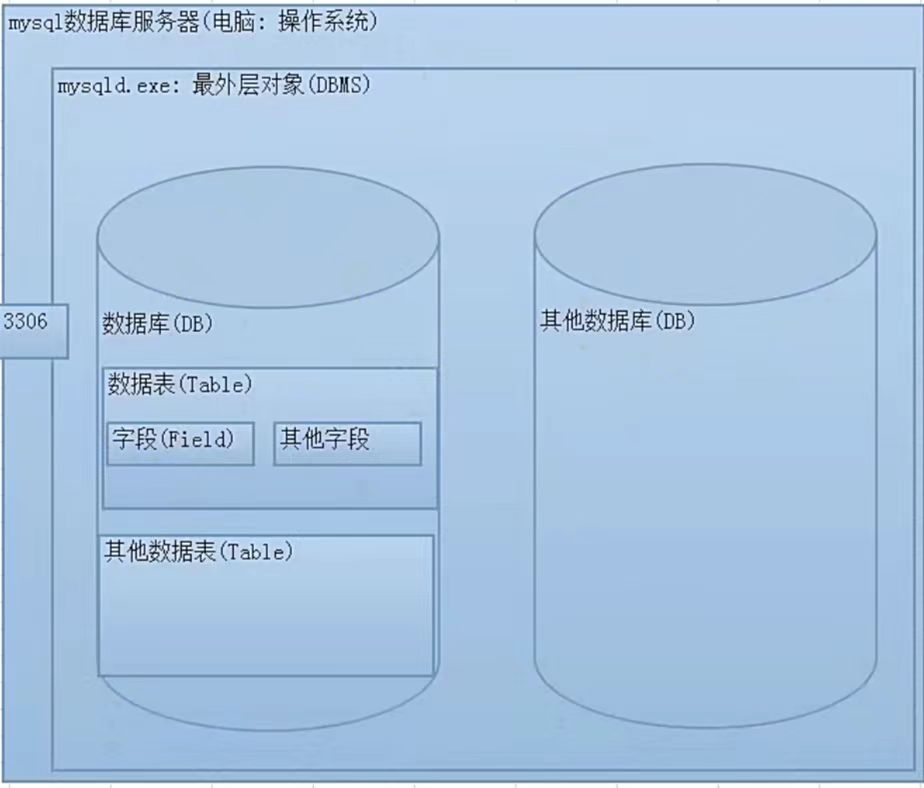
1. 断开连接（释放资源：服务器并发限制）



#### Mysql服务器对象

没有办法完全了解服务内部的内容：只能粗略的去分析数据库服务器的内部的结构。

将mysql服务器内部对象分为了四层：系统（DBMS） ——> 数据库（DB） ——> 数据表（Table） ——> 字段（field）



### SQL基本操作

基本操作：CRUD

将SQL的基本操作根据操作对象进行分类，分为三类：库操作，表操作（字段），数据操作

#### 库操作

对数据库的增删改查。

#### 新增数据库

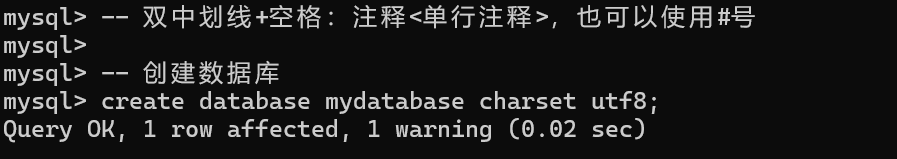
基本语法

Create database 数据库名字[库选项]；

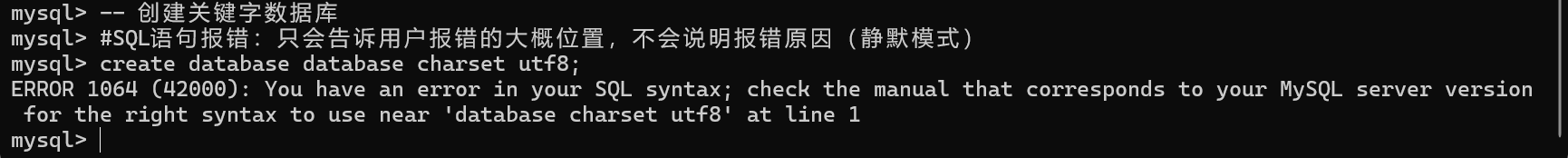
库选项：用来约束数据库，分为两个选项

字符集设定：charset/character set 具体字符集（数据存储的编码格式）：常用字符集：GBK和UTF8

校对集设定：collate具体校对集（数据比较规则）

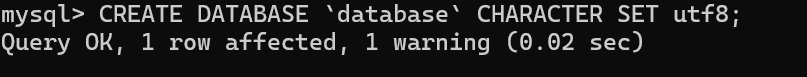


其中：数据库名字不能用关键字（已被使用的字符）或者保留（将来可能会用到的）

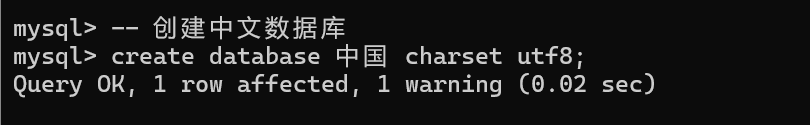


如果非要使用关键字或者保留字，那么必须使用反引号（esc键下面在英文状态下的输出：’）



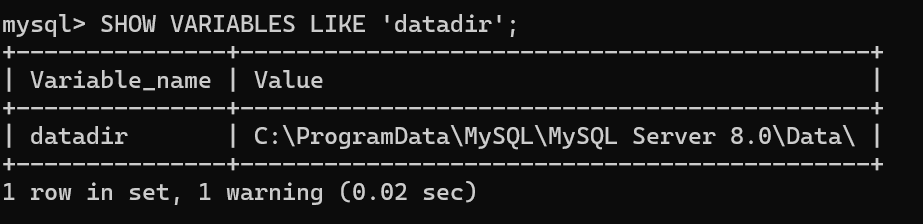


中文数据库是可以的，但是有前提条件：保证服务器能够识别（建议不用）



当创建数据库的SQL语句执行之后，发生了什么？

1. 在数据库系统中，增加了对应的数据库信息
2. 会在保存 数据的文件夹下：Data目录，创建一个对应数据库名字的文件夹



1. 每个数据库下都有一个opt文件：保存了库选项

#### 查看数据库

1. 查看所有数据库：show databases;

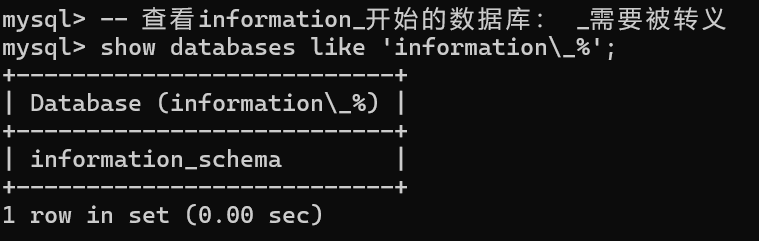


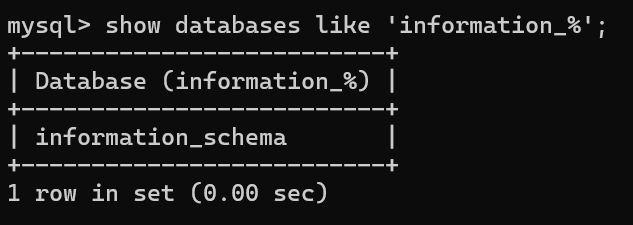
1. 查看指定部分的数据库：模糊查询

Show databases like ‘pattern’; --pattern是匹配模式

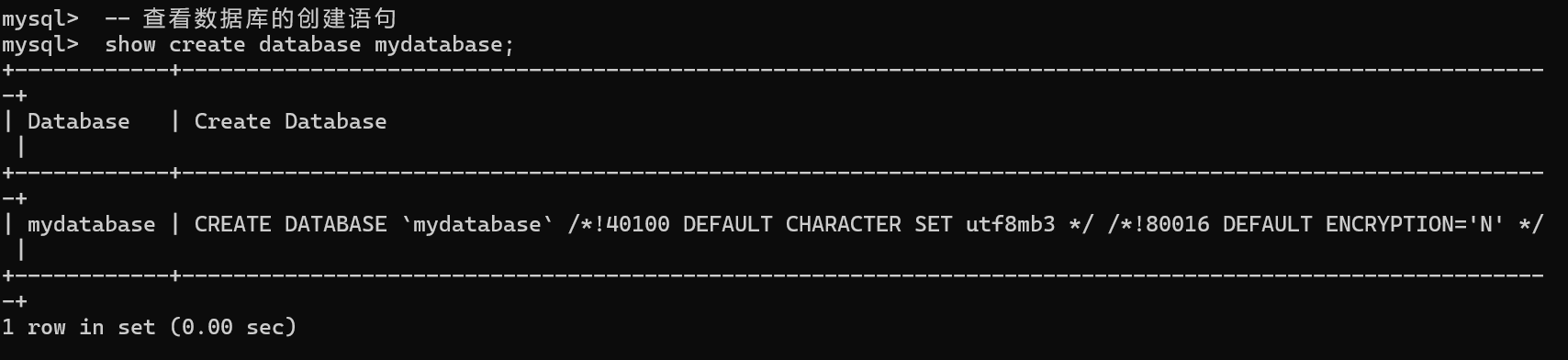
%：表示匹配多个字符

\_：表示匹配单个字符





1. 查看数据库的创建语句:show create database数据库名字；



数据库在执行SQL语句之前会优化SQL；系统保存的结果是优化的结果

#### 更新数据库

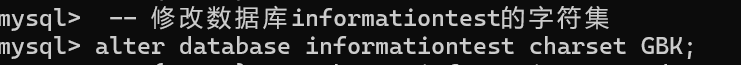
数据库名字不可以修改

数据库的修改仅限选项：字符集和校对集（校对集依赖字符集）

Alter database数据库名字[库选项]；

Charset/character set [=] 字符集

Collate校对集



#### 删除数据库

所有的操作中：删除是最简单

Drop database数据库名字；



当删除数据库语句执行之后，发生了什么？

1. 在数据库内部看不到对应的数据库



1. 在对应的数据库存储的文件夹内：数据库名字对应的文件夹也被删除（级联删除：里面的数据表全部删除）

注意：数据库的删除不是闹着玩的，不要随意删除，应该先进行备份后操作。（删除不可逆）

### 表操作

表与字段是密不可分的。

#### 新增数据表

Create table [if not exists] 表名（

字段名字 数据类型，

字段名字 数据类型 --最后一行不需要逗号

）[表选项];

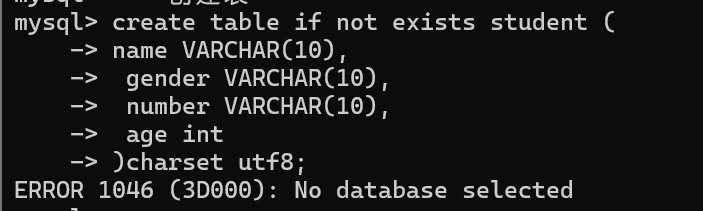
If not exists: 如果表名不存在，那么就创建，否则不执行创建代码：检查功能

表选项：控制表的表现

字符集：charset/character set 具体字符集； --保证表中数据存储的字符集

校对集：collata 具体校对集

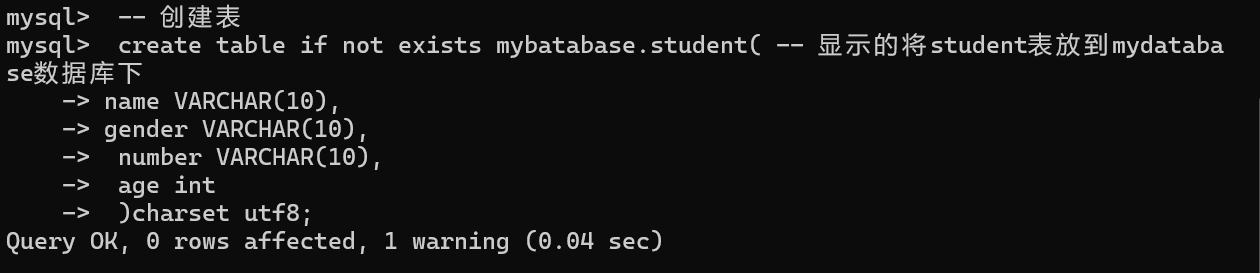
存储引擎：engine具体的存储引擎（innodb和myisam）



任何一个表的设计都必须指定数据库

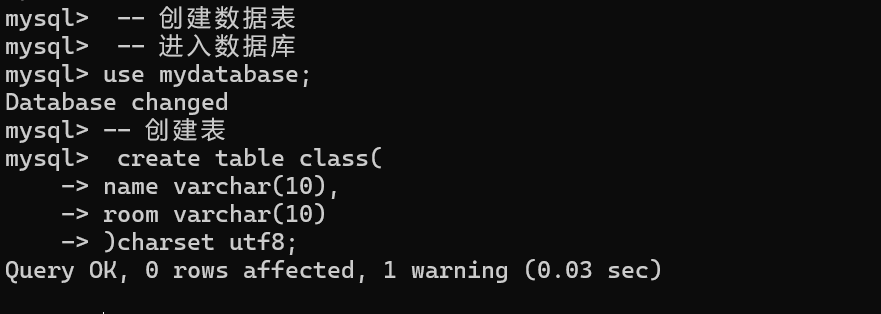
方案1：显示的指定所属的数据库

Create table 数据库.表名()； --将当前数据表创建到指定的数据库下



方案2隐式的指定表所属数据库：先进入到某个数据库环境，然后这样创建的表自动归属到某个指定的数据库。

进入数据库环境：use数据库名字



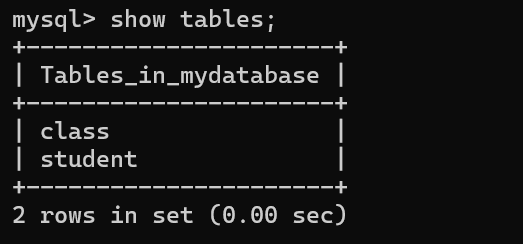
当创建数据表的SQL指令执行之后，到底发生了什么？

1. 指定数据库下已经存在对应的表
2. 在数据库对应的文件夹下会产生对应表的结构文件（跟存储引擎有关系）

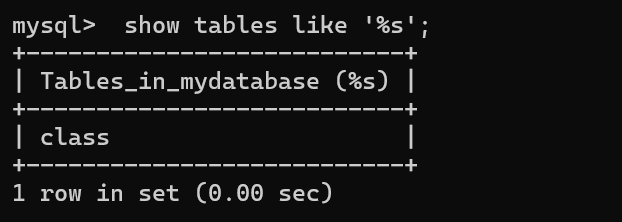
#### 查看数据表

数据库能查看的方式，表都可以查看。

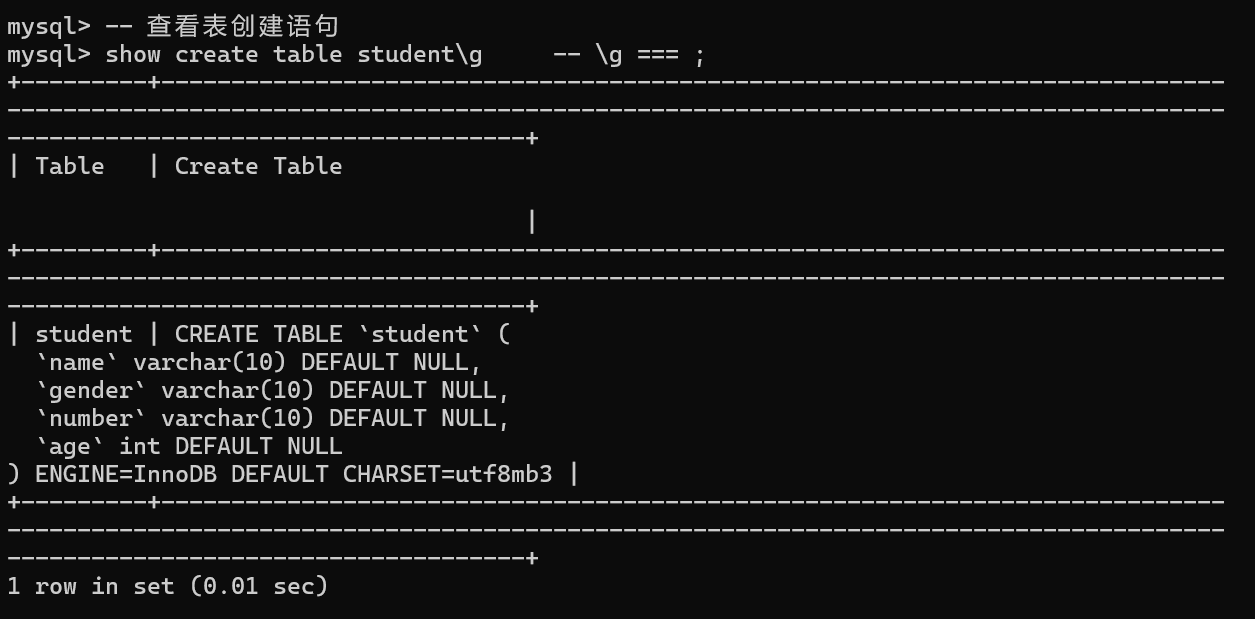
1.查看所有表：show tables;



2.查看部分表：模糊匹配：show tables like ‘pattern’

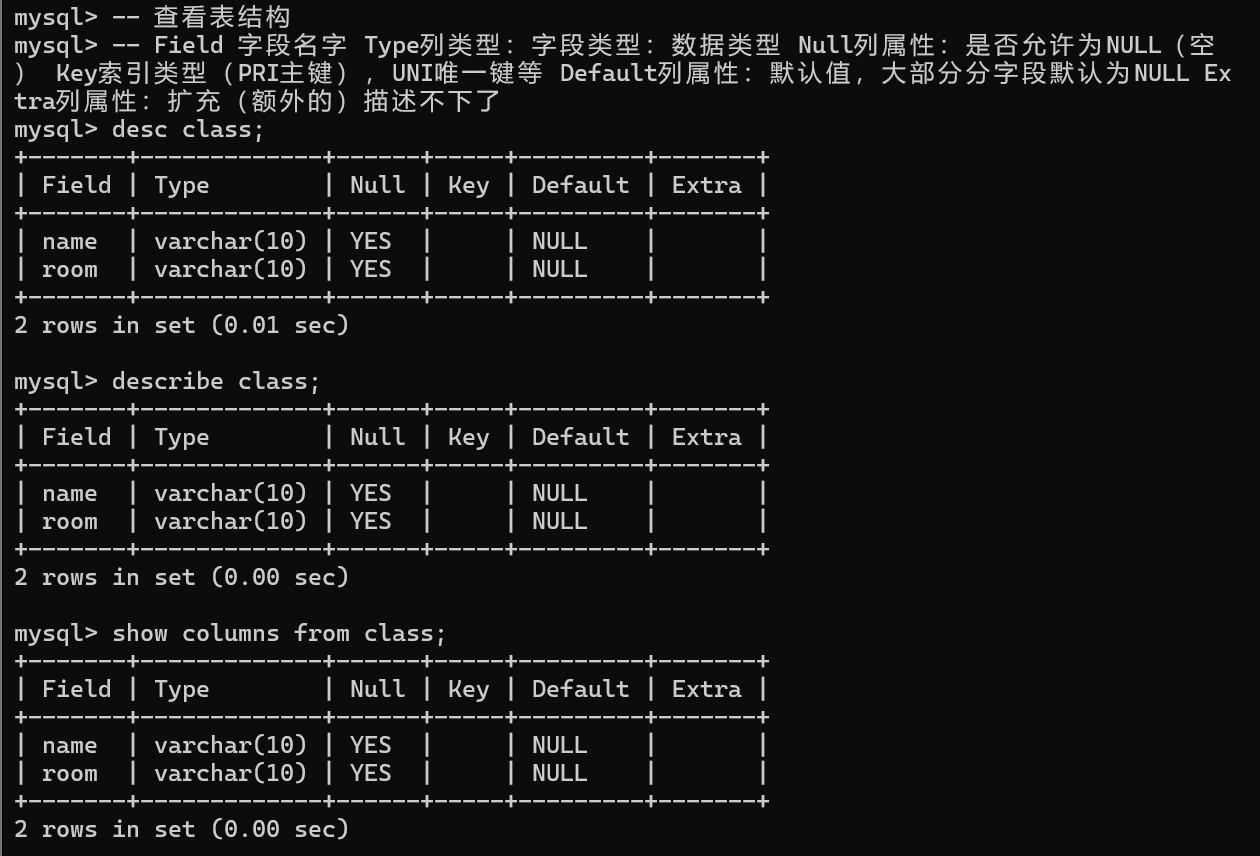


1. 查看表的创建语句：show create table 表名；



1. 查看表结构：查看表中的字段信息

Desc/describe/show columns from 表名;



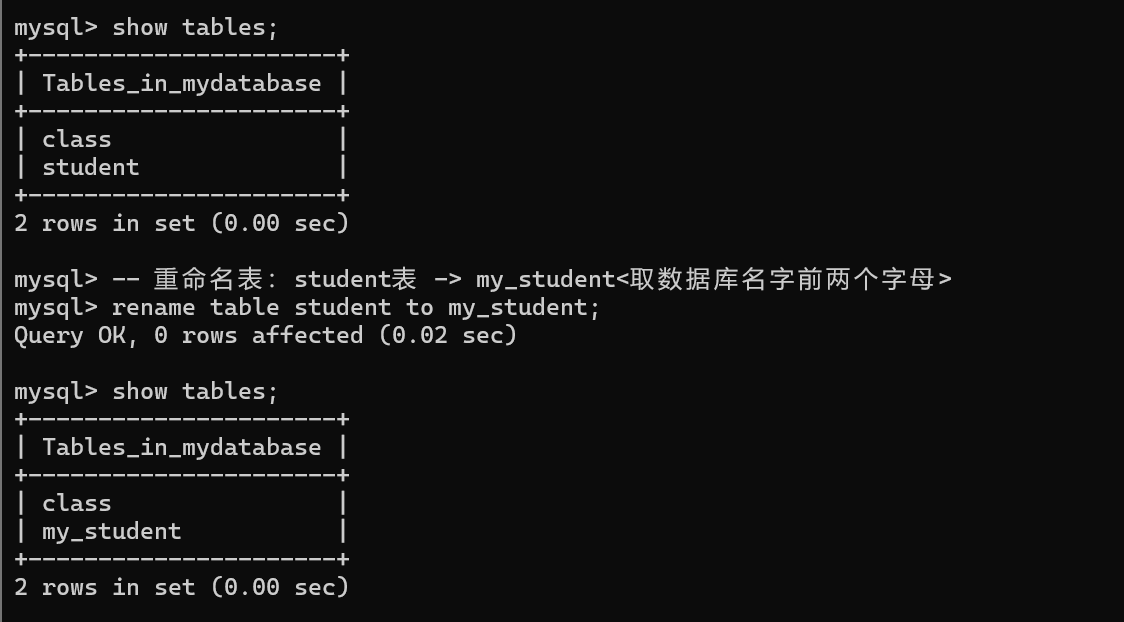
#### 修改数据表

表本身存在，还包含字段：表的修改分两个部分：修改表本身和修改字段

#### 修改表本身

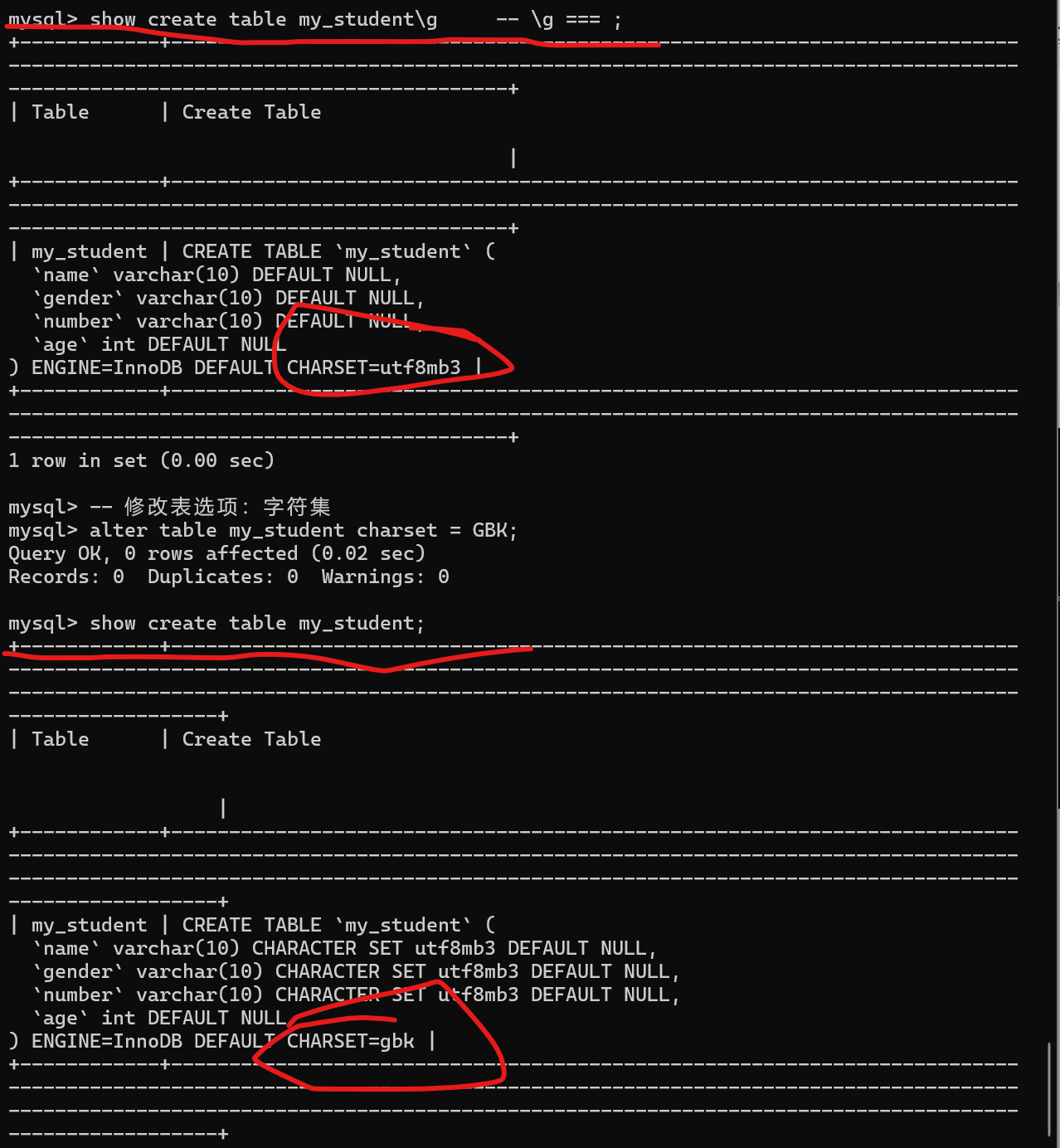
表本身可以修改：表名和表选项

修改表名：rename table 老表名 to 新表名;



修改表选项：字符集，校对集和存储引擎

Alter table 表名 表选项 [=] 值;



#### 修改字段

字段操作很多：新增，修改，重名，删除

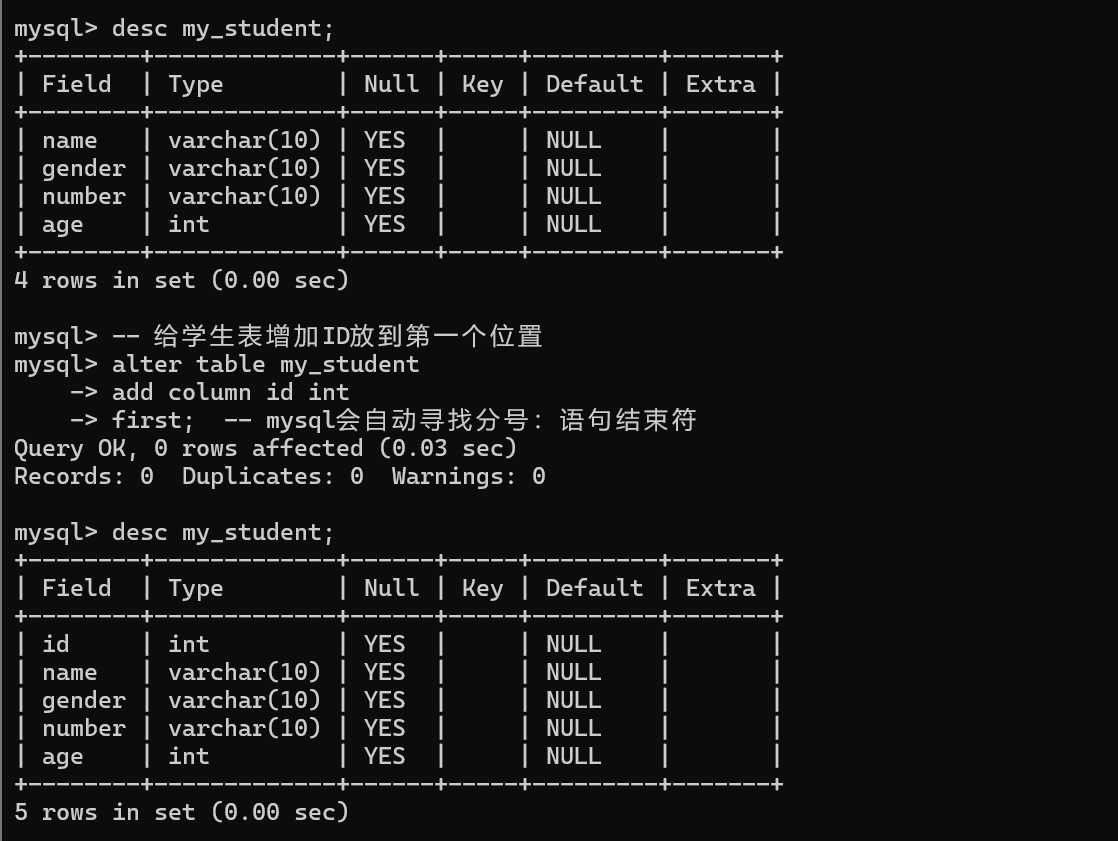
新增字段

Alter table 表名 add[column]字段名 数据类型 [列属性][位置];

位置：字段名可以存放表中的任意位置

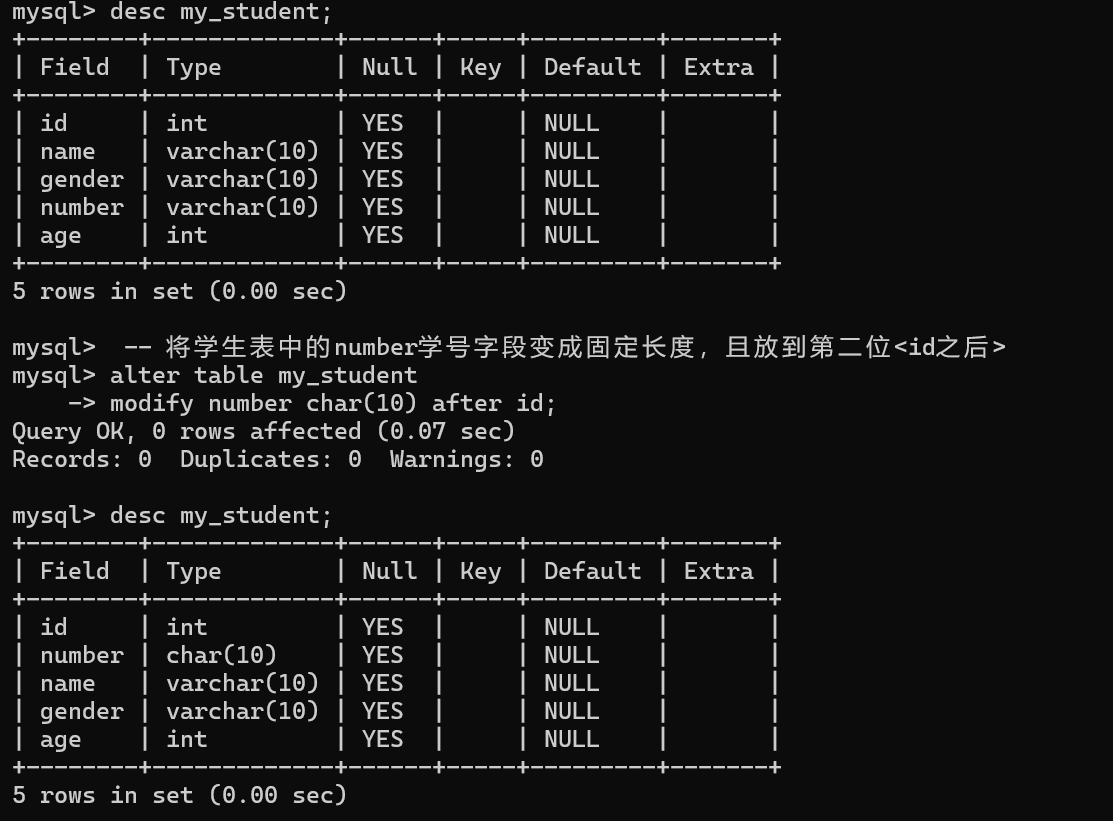
First:第一个位置

After:在哪个字段之后:after 字段名;默认的是在最后一个字段之后



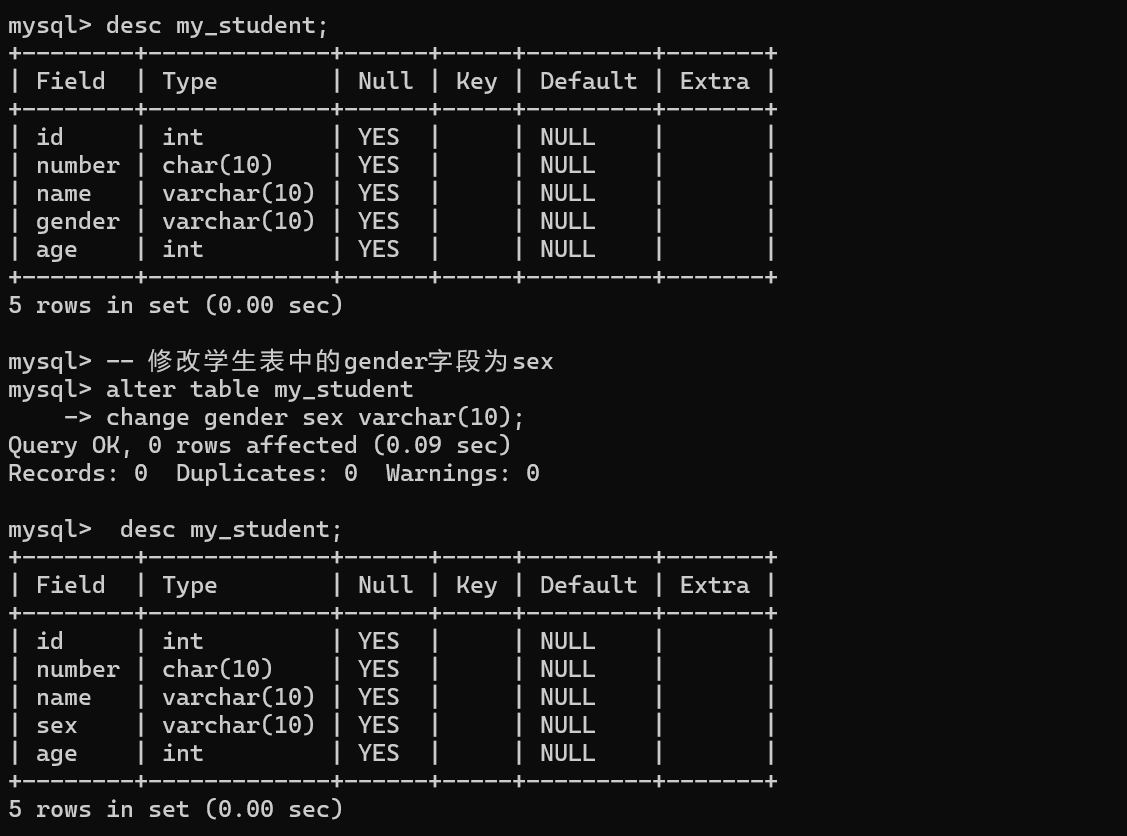
修改字段:修改通常是修改属性或者数据类型

Alter table 表名 modify 字段名 数据类型 [属性][位置];



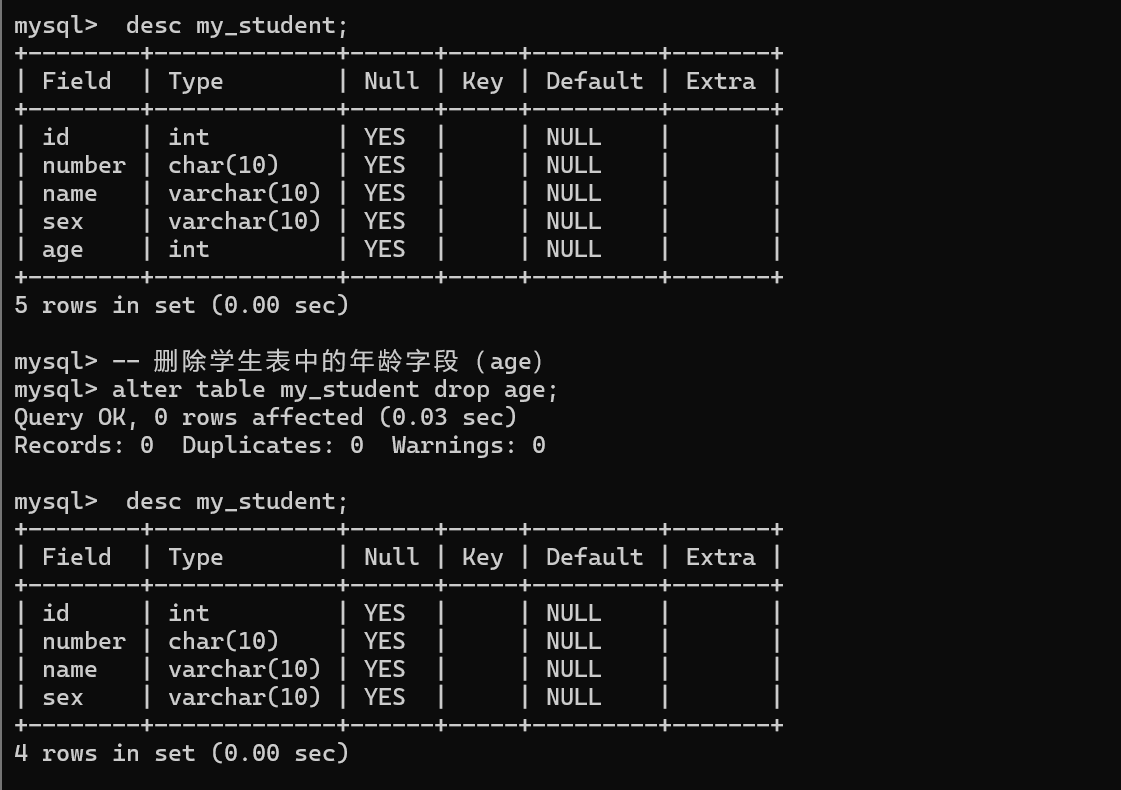
重命名字段

Alter table 表名 change 旧字段 新字段名 数据类型 [属性][位置];



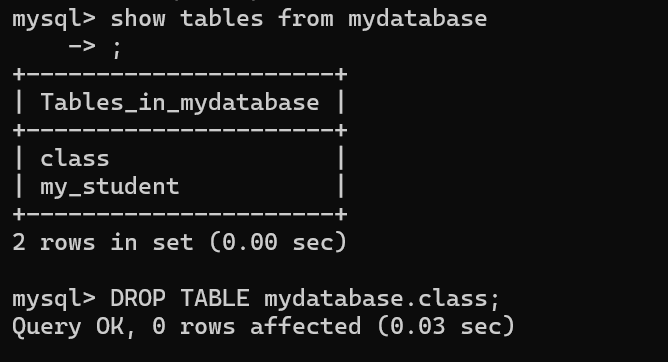
删除字段

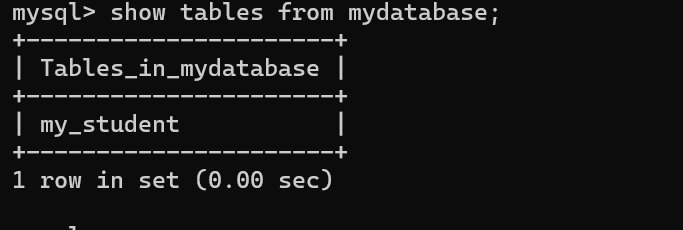
Alter table 表名 drop 字段名;



#### 删除数据表

Drop table 表名1，表名2...; --可以一次性删除多张表





当删除数据表的指令执行之后发生了什么？

1. 在表空间中，没有了指定的表（数据也没有了）
2. 在数据库对应的文件夹下，表对应的文件（与存储引擎有光）也会被删除

注意：删除有危险，操作需谨慎（不可逆）

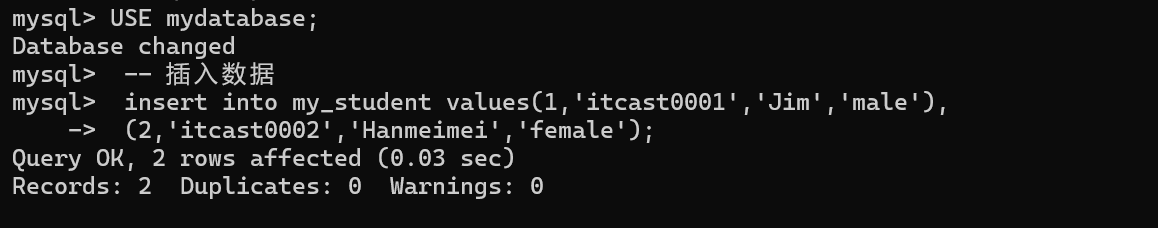
### 数据操作

#### 新增数据

有两种方案

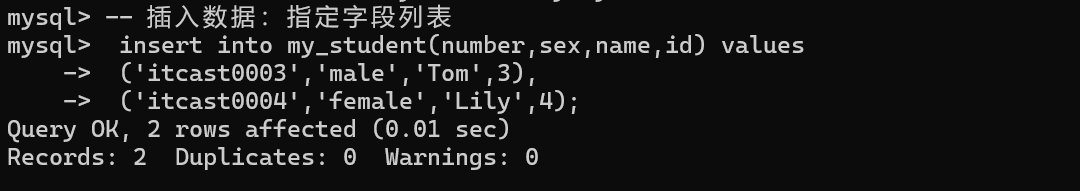
方案1：给全表字段插入数据，不需要指定字段列表；要求数据的值出现的顺序必须与表中设计的字段出现的顺序一致：都需要使用引号（建议是单引号）包裹

Insert into 表名 values(值列表)[,(值列表)]; --可以一次插入多条记录



方案2：给部分字段插入数据，需要选定字段列表：字段列表出现的顺序与字段的顺序无关；第三但是值列表的顺序必须与选定的字段的顺序一致。

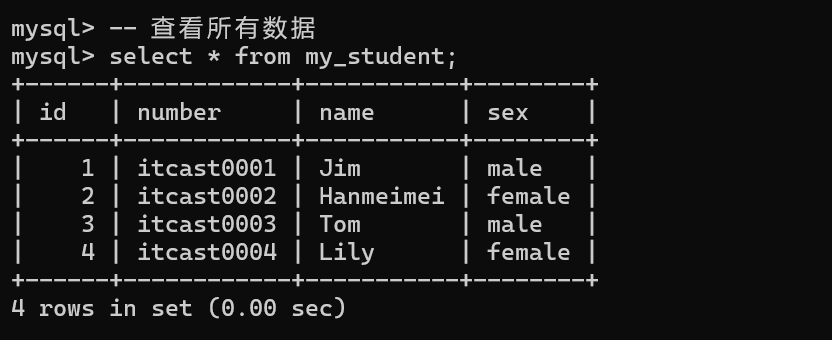
Insert into 表名（字段列表) values(值列表)[,(值列表)];



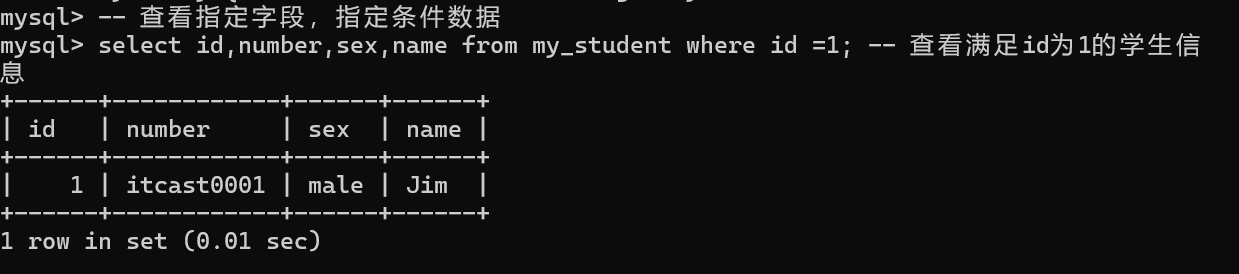
#### 查看数据

Select \*/字段列表 from 表名 [where条件];

查看所有数据

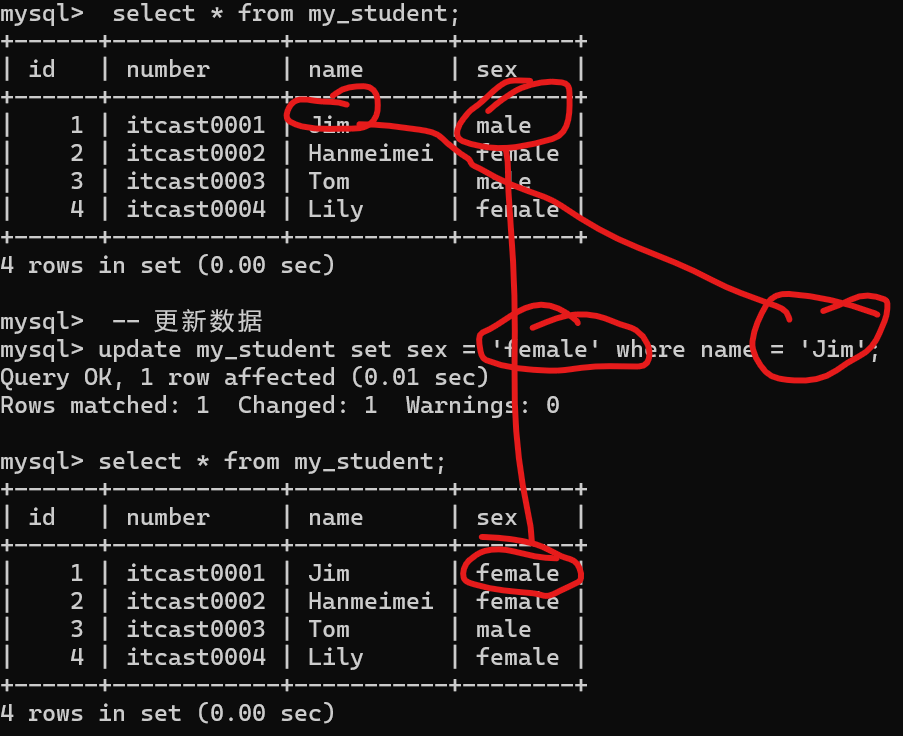


查看指定字段，指定条件的数据。

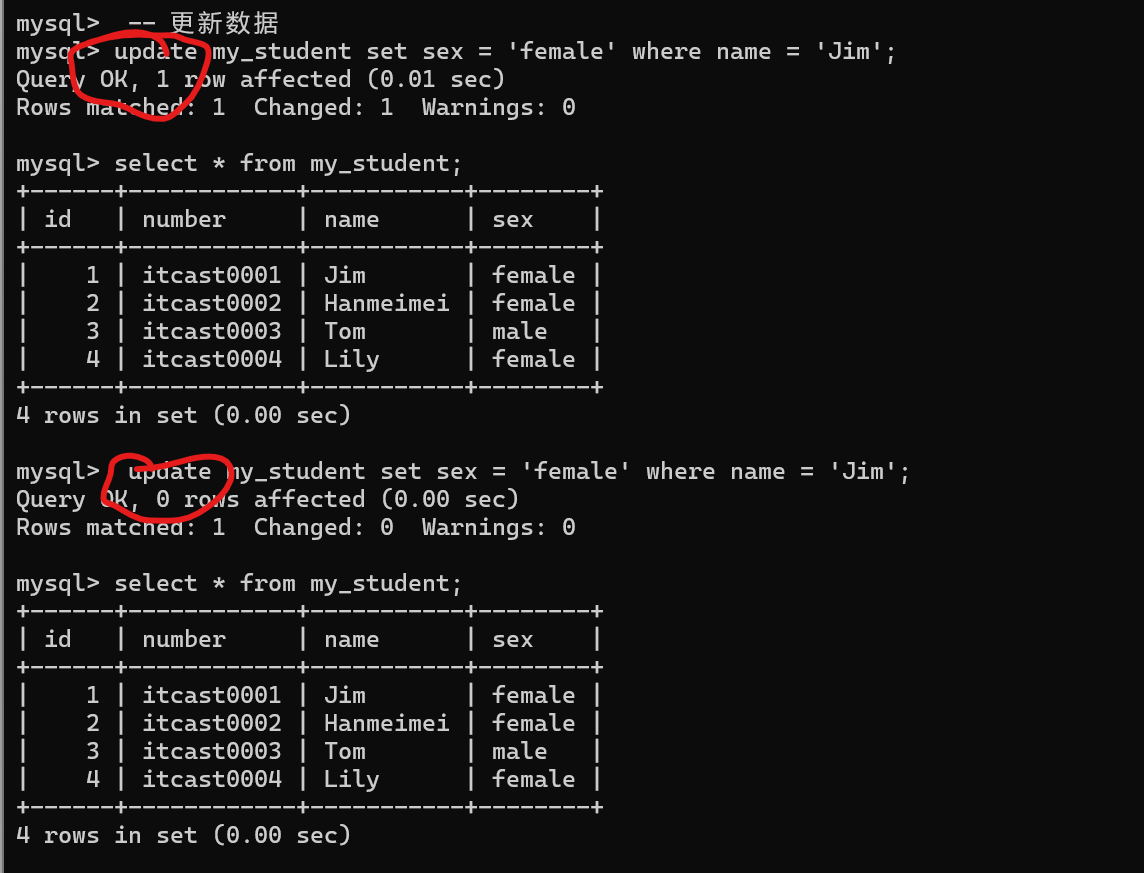


#### 更新数据

Updata 表名 set 字段 = 值 [where条件]; --建议都有where: 要不是更新全部



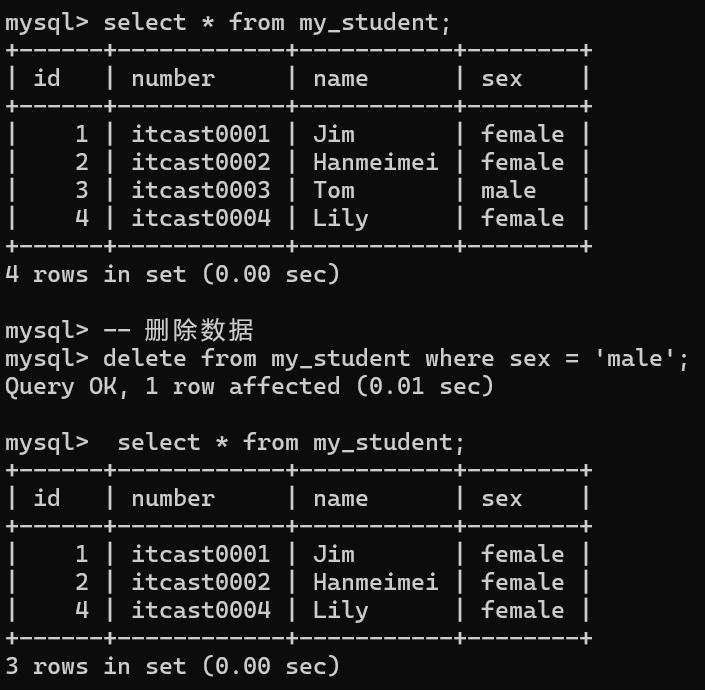
更新不一定会成功：如没有真正要更新的数据



#### 删除数据

删除是不可逆的：谨慎删除

Delete from 表名 [where条件];

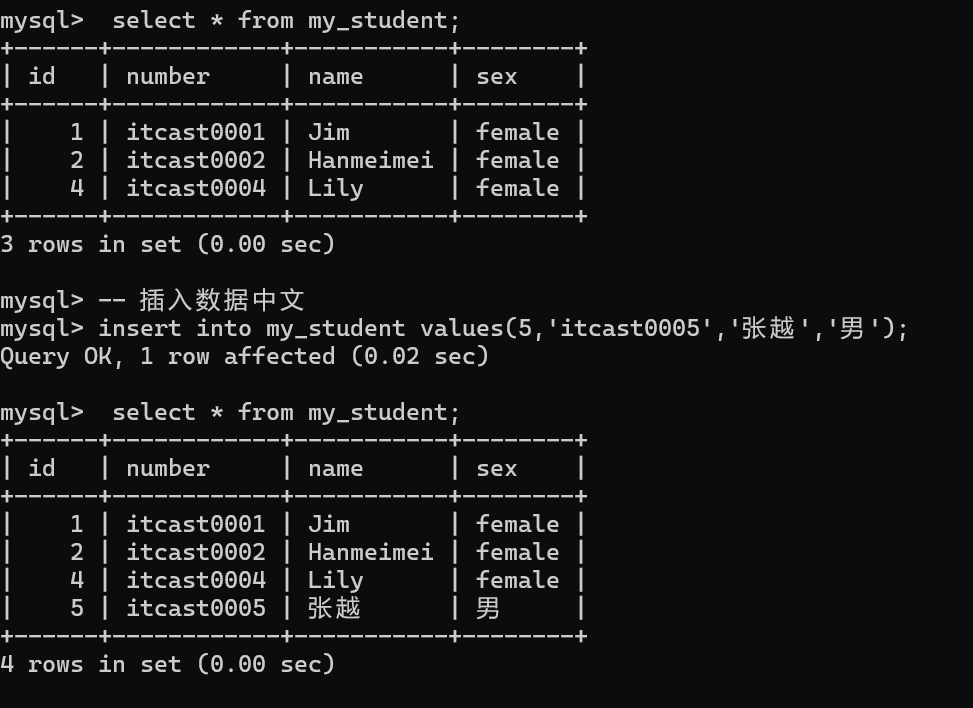


### 中文数据问题

中文数据问题本质是字符据问题

计算机只识别二进制：人类更多是识别符号：需要有个二进制与字符的对应关系（字符集）

客户端向服务端插入中文数据（这个版本可以）



如果失败了：

原因：\xD5\xC5\xBD代表的是“张越”在当前编码（字符集）下对应的二进制编码转换成的十六进制：两个汉字 => 四个字节（GBK）

报错：服务器没有识别对应的四个字节：服务器认为数据是UTF8,一个汉字有三个字节：

读取三个字节转换汉字（失败），剩余的再读三个字节（不够）：最终失败。

所有数据库服务器认为（表现）的一些特性都是通过服务器的变量来保存：系统先读取自己的变量，看看应该怎么表现。

//查看服务器到底识别哪些字符集

Show character set;

基本上：服务器是万能，什么字符集都支持

//既然服务器识别这么多：总有一种是服务器默认的跟客户端打交道的字符集

Show variable like ‘character\_set%’

问题根源：客户端数据只能是GBK，而服务器认为是UTF8：矛盾产生

解决问题：改变服务器，默认的接收字符集为GBK;

Set character\_set\_client = gbk

插入中文的效果

查看数据效果：依旧是乱码（我的电脑不会，只是看如果出现这种问题应该怎么办？）

原因：数据来源是服务器，解析数据是客户端（客户端只识别GBK：只会两个字节一个汉字）：但是事实服务器给的数据却是UTF8，三个字节一个汉字：乱码

解决方案：修改服务器给客户端的数据字符集为GBK

Set character\_set\_results = GBK;

查看数据效果