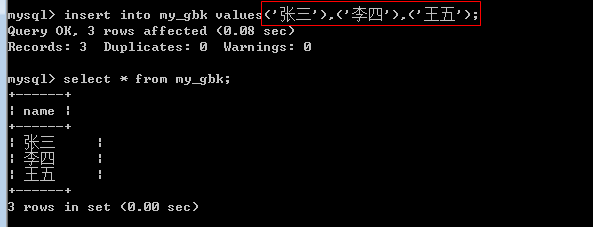
**高级数据操作**

**新增数据**

**多数据插入**

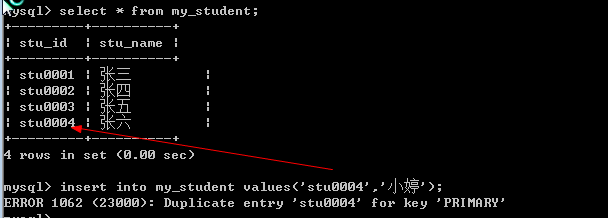
只要写一次insert指令，但是可以直接插入多条记录

基本语法：insert into 表名 [(字段列表)] values(值列表), (值列表)…;



**主键冲突**

主键冲突：在有的表中，使用的是业务主键（字段有业务含义），但是往往在进行数据插入的时候，又不确定数据表中是否已经存在对应的主键。

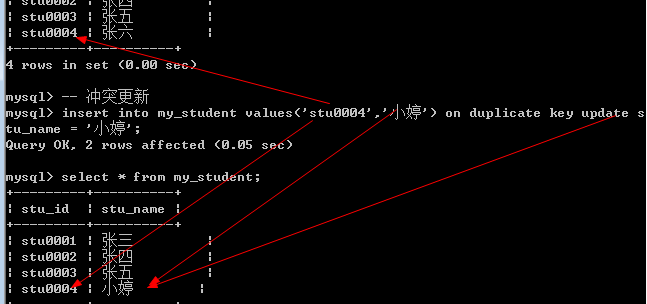


主键冲突的解决方案：

1、 主键冲突更新：

类似插入数据语法，如果插入的过程中主键冲突，那么采用更新方法。

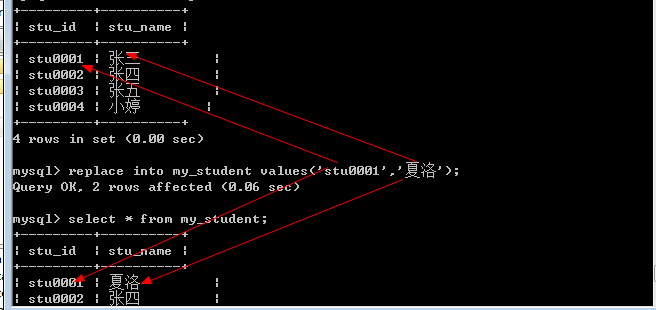
Insert into 表名 [(字段列表)] values(值列表) on duplicate key update 字段 = 新值;



2、 主键冲突替换：

当主键冲突之后，干掉原来的数据，重新插入进去。

Replace into 表名 [(字段列表)] values(值列表);

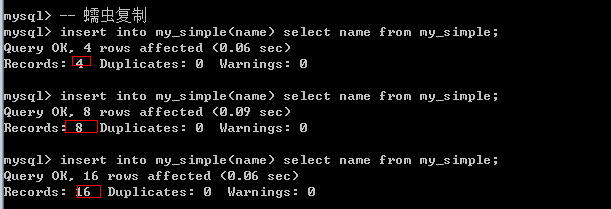


**蠕虫复制**

蠕虫复制：一分为二，成倍的增加。从已有的数据中获取数据，并且将获取到的数据插入到数据表中。

基本语法：

Insert into 表名 [(字段列表)] select \*/字段列表 from 表;



注意：

1、 蠕虫复制通常是重复数据，没有太大业务意义：可以在短期内快速增加表的数据量，从而可以测试表的压力，还可以通过大量数据来测试表的效率（索引）

2、 蠕虫复制虽好，但是要注意主键冲突。

**更新数据**

1、 在更新数据的时候，特别要注意：通常一定是跟随条件更新

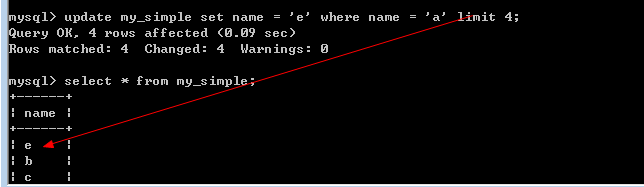
Update 表名 set 字段名 = 新值 where 判断条件;

2、 如果没有条件，是全表更新数据。但是可以使用limit 来限制更新的数量；

Update 表名 set 字段名 = 新值 [where 判断条件] limit 数量;

改变4个a变成e

Update my\_simple set name = ‘e’ where name = ‘a’ limit 4;

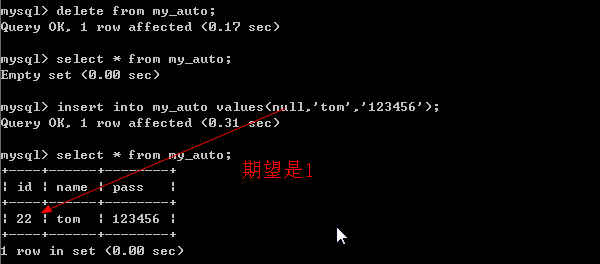


**删除数据**

1、 删除数据的时候尽量不要全部删除，应该使用where进行 判定；

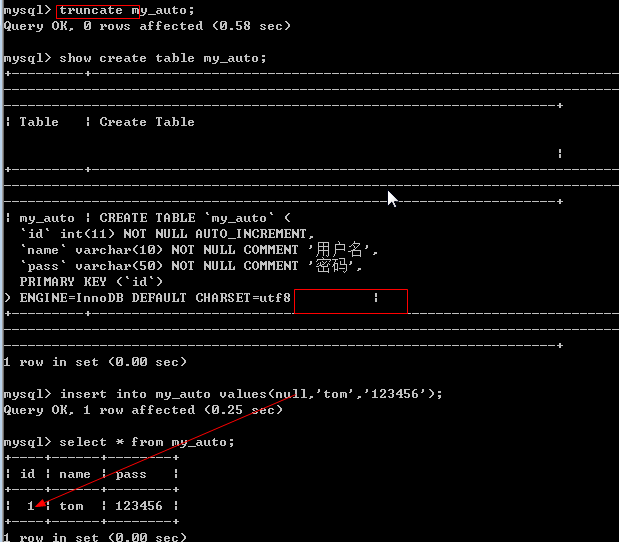
2、 删除数据的时候可以使用limit来限制要删除的具体数量

Delete删除数据的时候无法重置auto\_increment



Mysql有一个能够重置表选项中的自增长的语法;

Truncate 表名; == drop - create



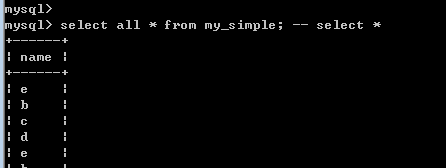
**查询数据**

完整的查询指令：

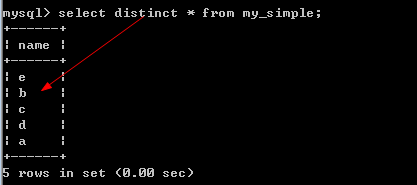
Select select选项 字段列表 from 数据源 where条件 group by分组 having条件 order by排序 limit限制;

Select选项：系统该如何对待查询得到的结果

All：默认的，表示保存所有的记录

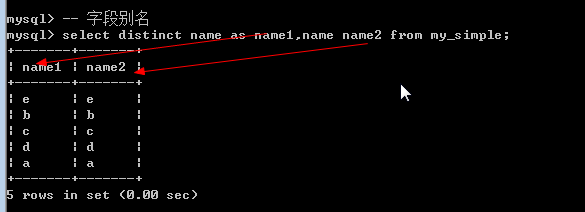


Distinct：去重，去除重复的记录，只保留一条（所有的字段都相同）



字段列表：有的时候需要从多张表获取数据，在获取数据的时候，可能存在不同表中有同名的字段，需要将同名的字段命名成不同名的：别名 alias

基本语法：字段名 [as] 别名

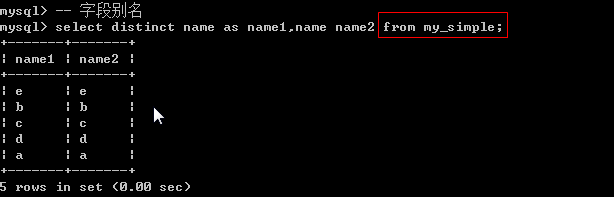


**From数据源**

From是为前面的查询提供数据：数据源只要是一个符合二维表结构的数据即可。

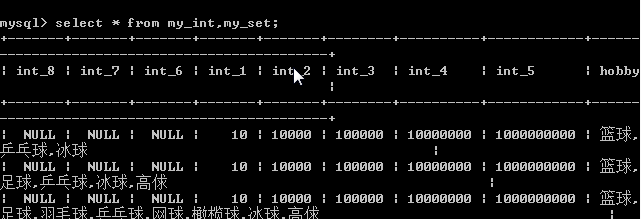
**单表数据**

From 表名;



**多表数据**

从多张表获取数据，基本语法：from 表1,表2…



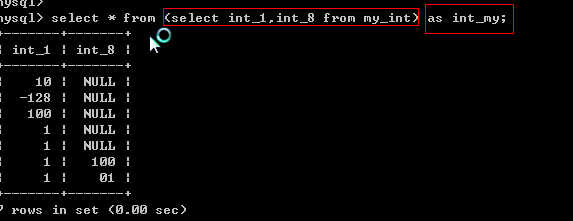
结果：两张表的记录数相乘，字段数拼接

本质：从第一张表取出一条记录，去拼凑第二张表的所有记录，保留所有结果。得到的结果在数学上有一个专业的说法：笛卡尔积，这个结果出了给数据库造成压力，没有其他意义：应该尽量避免出现笛卡尔积。

**动态数据**

From后面跟的数据不是一个实体表，而是一个从表中查询出来得到的二维结果表（子查询）。

基本语法：from (select 字段列表 from 表) as 别名;



**Where子句**

Where字句：用来从数据表获取数据的时候，然后进行条件筛选。

数据获取原理：针对表去对应的磁盘处获取所有的记录（一条条），where的作用就是在拿到一条结果就开始进行判断，判断是否符合条件：如果符合就保存下来，如果不符合直接舍弃（不放到内存中）

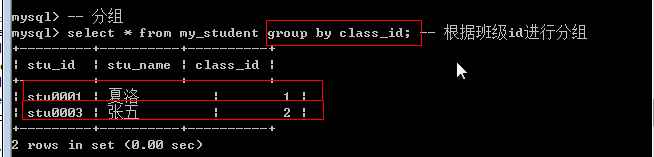
Where是通过运算符进行结果比较来判断数据。

**Group by子句**

Group by表示分组的含义：根据指定的字段，将数据进行分组：分组的目标是为了统计

**分组统计**

基本语法： group by 字段名;



Group by是为了分组后进行数据统计的，如果只是想看数据显示，那么group by没什么含义：group by将数据按照指定的字段分组之后，只会保留每组的第一条记录。

利用一些统计函数：（聚合函数）

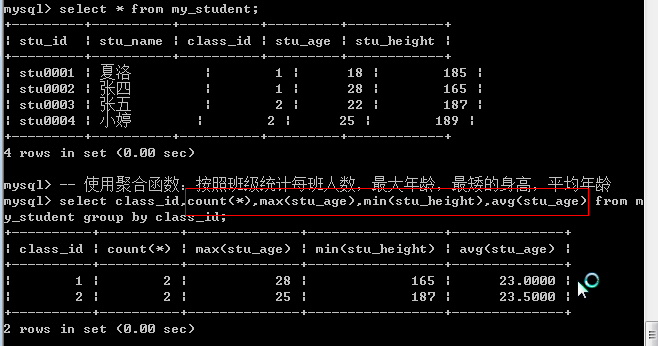
count()：统计每组中的数量，如果统计目标是字段，那么不统计为空NULL字段，如果为\*那么代表统计记录

avg()：求平均值

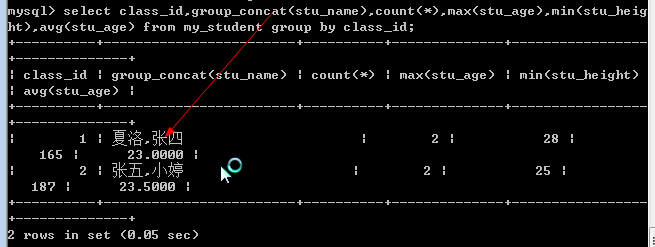
sum()：求和

max()：求最大值

min()：求最小值



Group\_concat()：为了将分组中指定的字段进行合并（字符串拼接）



**多分组**

将数据按照某个字段进行分组之后，对已经分组的数据进行再次分组

基本语法：group by 字段1,字段2; //先按照字段1进行排序，之后将结果再按照字段2进行排序，以此类推。



**分组排序**

Mysql中，分组默认有排序的功能：按照分组字段进行排序，默认是升序

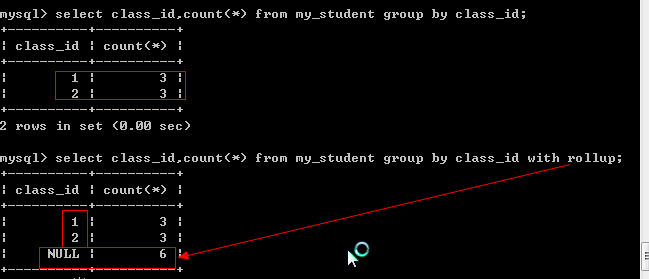
基本语法：group by 字段 [asc|desc]，字段 [asc|desc]



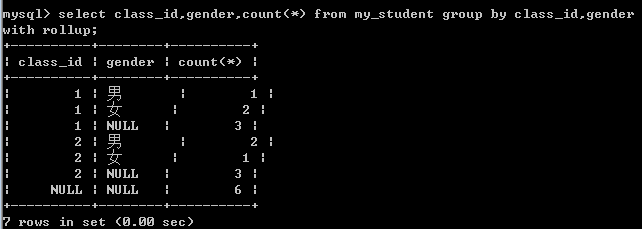
**回溯统计**

当分组进行多分组之后，往上统计的过程中，需要进行层层上报，将这种层层上报统计的过程称之为回溯统计：每一次分组向上统计的过程都会产生一次新的统计数据，而且当前数据对应的分组字段为NULL。

基本语法：group by 字段 [asc|desc] with rollup;



多分组回溯统计



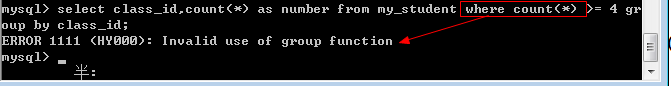
**Having子句**

Having的本质和where一样，是用来进行数据条件筛选。

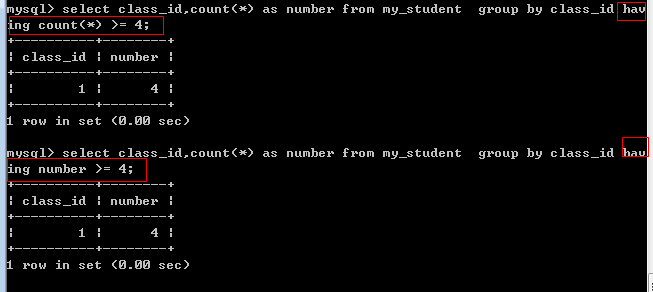
1、 Having是在group by子句之后：可以针对分组数据进行统计筛选，但是where不行

查询班级人数大于等于4个以上的班级

Where不能使用聚合函数：聚合函数是用在group by分组的时候，where已经运行完毕



Having在group by分组之后，可以使用聚合函数或者字段别名（where是从表中取出数据，别名是在数据进入到内存之后才有的）

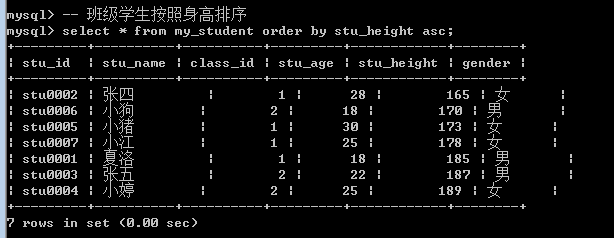


强调：having是在group by之后，group by是在where之后：where的时候表示将数据从磁盘拿到内存，where之后的所有操作都是内存操作。

**Order by子句**

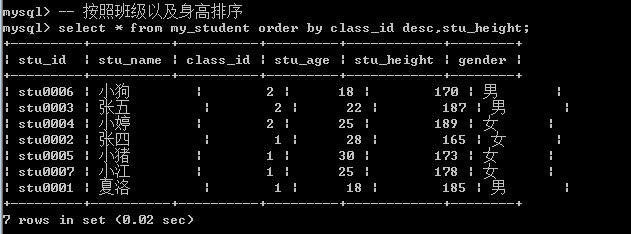
Order by排序：根据校对规则对数据进行排序

基本语法：order by 字段 [asc|desc]; //asc升序，默认的



Order by也可以像group by一样进行多字段排序：先按照第一个字段进行排序，然后再按照第二个字段进行排序。

Order by 字段1 规则,字段2 规则;



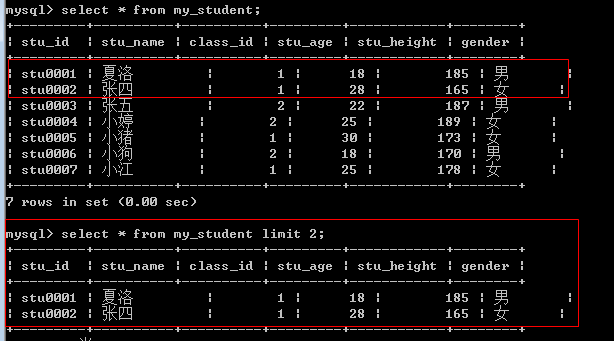
**Limit子句**

Limit限制子句：主要是用来限制记录数量获取

**记录数限制**

纯粹的限制获取的数量：从第一条到指定的数量

基本语法： limt 数量;



Limit通常在查询的时候如果限定为一条记录的时候，使用的比较多：有时候获取多条记录并不能解决业务问题，但是会增加服务器的压力。

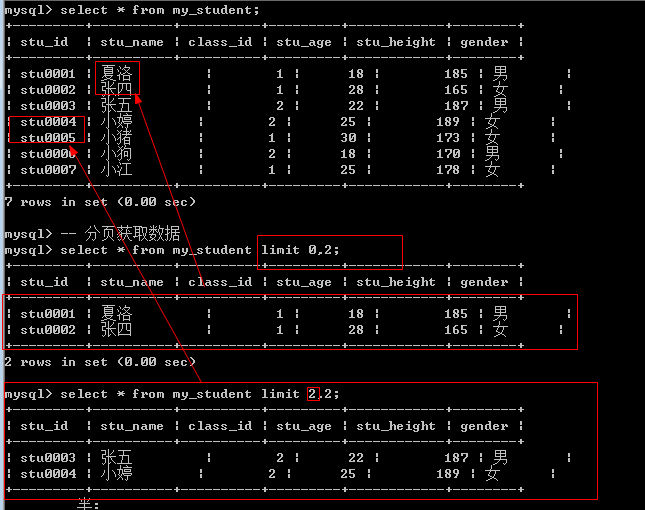
**分页**

利用limit来限制获取指定区间的数据。

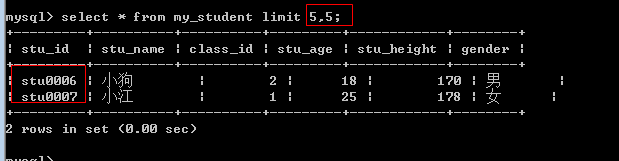
基本语法：limit offset,length; //offset偏移量：从哪开始，length就是具体的获取多少条记录

Mysql中记录的数量从0开始

Limit 0,2; 表示获取前两条记录



注意：limit后面的length表示最多获取对应数量，但是如果数量不够，系统不会强求

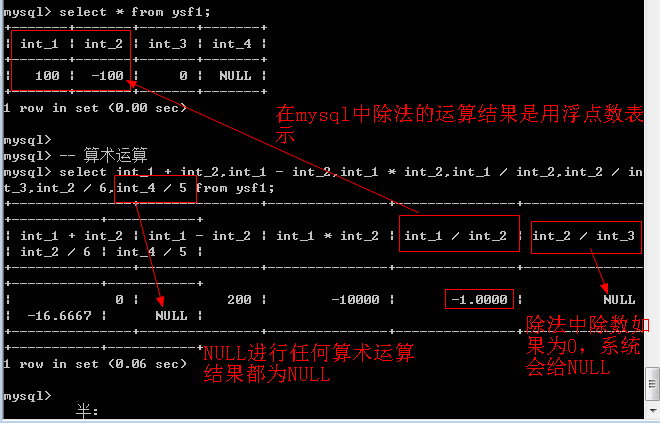


**查询中的运算符**

**算术运算符**

+、-、\*、/、%

基本算术运算：通常不在条件中使用，而是用于结果运算（select 字段中）



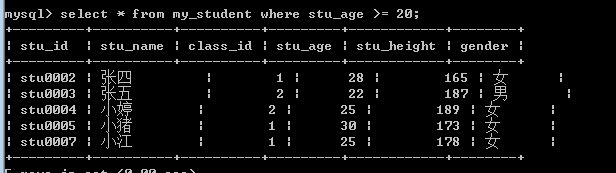
**比较运算符**

>、>=、<、<=、=、<>

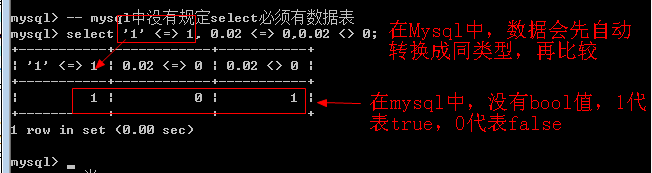
通常是用来在条件中进行限定结果

=：在mysql中，没有对应的 ==比较符号，就是使用=来进行相等判断

<=>：相等比较

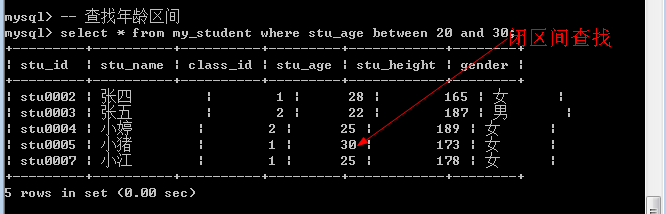


特殊应用：就是在字段结果中进行比较运算

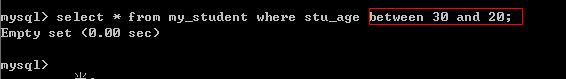


在条件判断的时候，还有有对应的比较运算符：计算区间

Between 条件1 and 条件2;



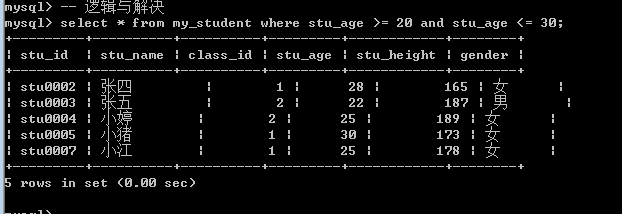
Between中条件1必须小于条件2，反过来不可以



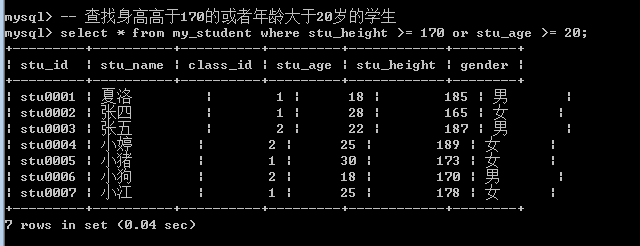
**逻辑运算符**

and、or、not

and：逻辑与



or：逻辑或

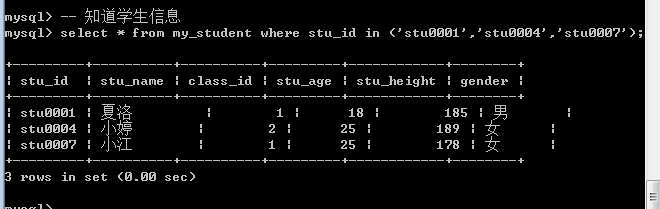


not：逻辑非

**In运算符**

In：在什么里面，是用来替代=，当结果不是一个值，而是一个结果集的时候

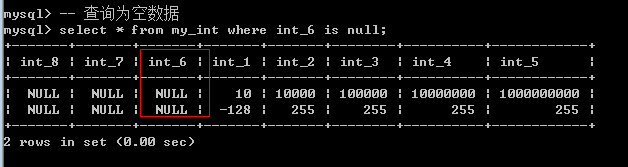
基本语法： in (结果1,结果2,结果3…)，只要当前条件在结果集中出现过，那么就成立



**Is运算符**

Is是专门用来判断字段是否为NULL的运算符

基本语法：is null / is not null



**Like运算符**

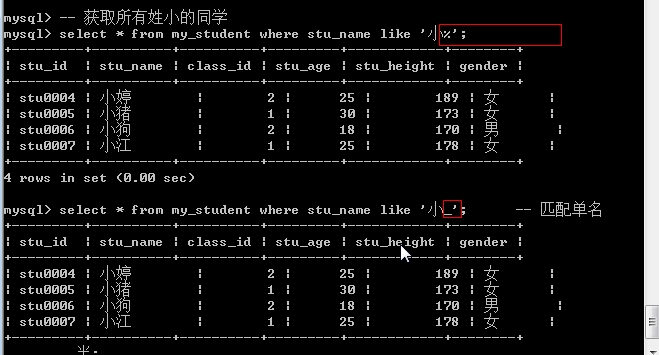
Like运算符：是用来进行模糊匹配（匹配字符串）

基本语法：like ‘匹配模式’;

匹配模式中，有两种占位符：

\_：匹配对应的单个字符

%：匹配多个字符



**联合查询**

**基本概念**

联合查询是可合并多个相似的选择查询的结果集。等同于将一个表追加到另一个表，从而实现将两个表的查询组合到一起，使用谓词为UNION或UNION ALL。

联合查询：将多个查询的结果合并到一起（纵向合并）：字段数不变，多个查询的记录数合并。

**应用场景**

1、 将同一张表中不同的结果（需要对应多条查询语句来实现），合并到一起展示数据

男生身高升序排序，女生身高降序排序

2、 最常见：在数据量大的情况下，会对表进行分表操作，需要对每张表进行部分数据统计，使用联合查询来讲数据存放到一起显示。

QQ1表获取在线数据

QQ2表获取在线数据 ---》将所有在线的数据显示出来

**基本语法**

基本语法：

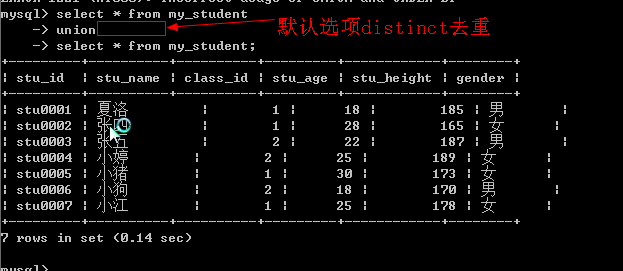
Select 语句

Union [union 选项]

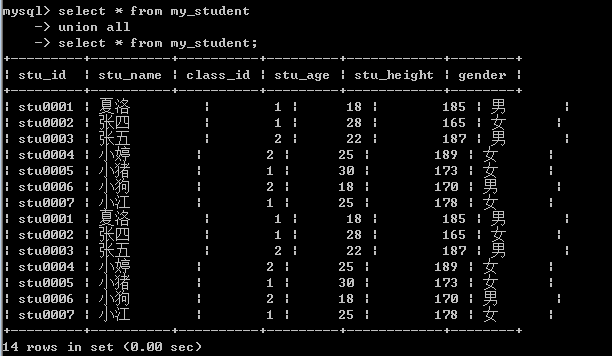
Select 语句;

Union选项：与select选项基本一样

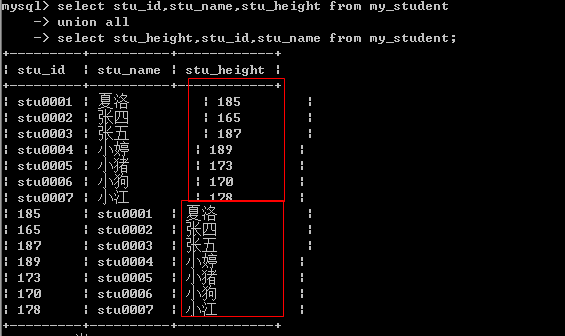
Distinct：去重，去掉完全重复的数据（默认的）



All：保存所有的结果

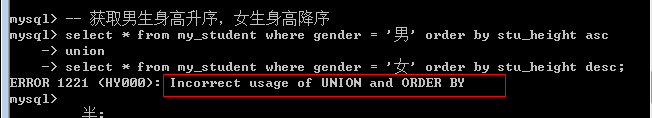


注意细节：union理论上只要保证字段数一样，不需要每次拿到的数据对应的字段类型一致。永远只保留第一个select语句对应的字段名字。

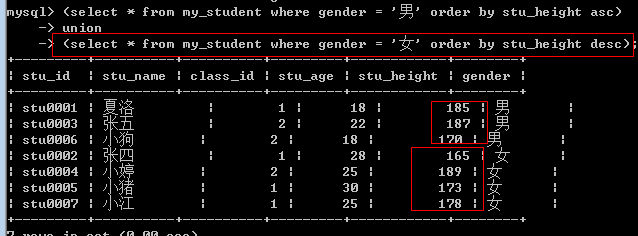


**Order by的使用**

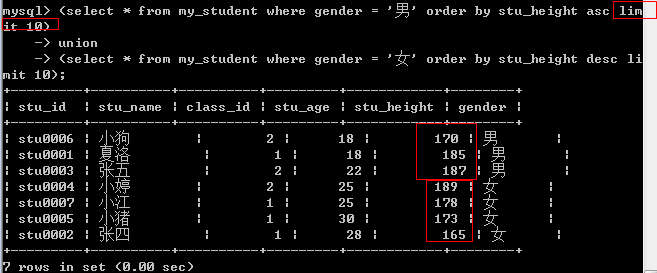
1、 在联合查询中，如果要使用order by，那么对应的select语句必须使用括号括起来



正确的语法：加上括号



2、 orderby在联合查询中若要生效，必须配合使用limit：而limit后面必须跟对应的限制数量（通常可以使用一个较大的值：大于对应表的记录数）



**连接查询**

连接查询：将多张表连到一起进行查询（会导致记录数行和字段数列发生改变）

**连接查询的意义**

在关系型数据库设计过程中，实体（表）与实体之间是存在很多联系的。在关系型数据库表的设计过程中，遵循着关系来设计：一对一，一对多和多对多，通常在实际操作的过程中，需要利用这层关系来保证数据的完整性。

**连接查询分类**

连接查询一共有以下几类：

交叉连接

内连接

外连接：左外连接（左连接）和右外连接（右连接）

自然连接

**交叉连接**

交叉连接：将两张表的数据与另外一张表彼此交叉

**原理**

1、 从第一张表依次取出每一条记录

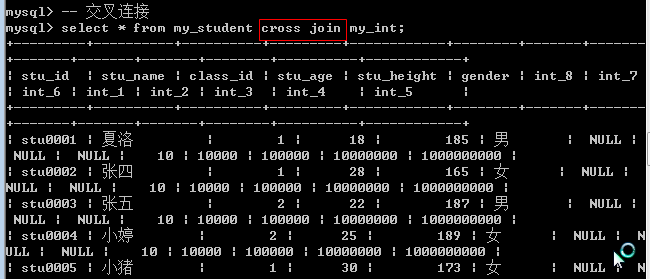
2、 取出每一条记录之后，与另外一张表的全部记录挨个匹配

3、 没有任何匹配条件，所有的结果都会进行保留

4、 记录数 = 第一张表记录数 \* 第二张表记录数；字段数 = 第一张表字段数 + 第二张表字段数（笛卡尔积）

**语法**

基本语法：表1 cross join 表2;



**应用**

交叉连接产生的结果是笛卡尔积，没有实际应用。

本质：from 表1,表2;

**内连接**

内连接：inner join，从一张表中取出所有的记录去另外一张表中匹配：利用匹配条件进行匹配，成功了则保留，失败了放弃。

**原理**

1、 从第一张表中取出一条记录，然后去另外一张表中进行匹配

2、 利用匹配条件进行匹配：

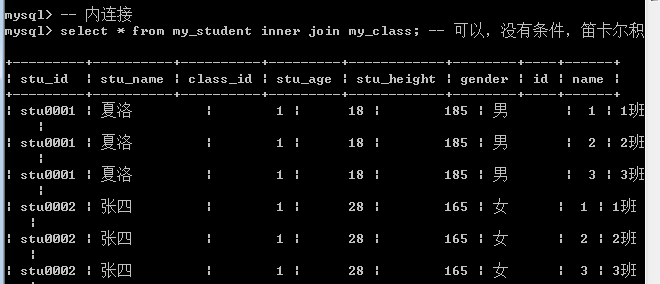
2.1 匹配到：保留，继续向下匹配

2.2 匹配失败：向下继续，如果全表匹配失败，结束

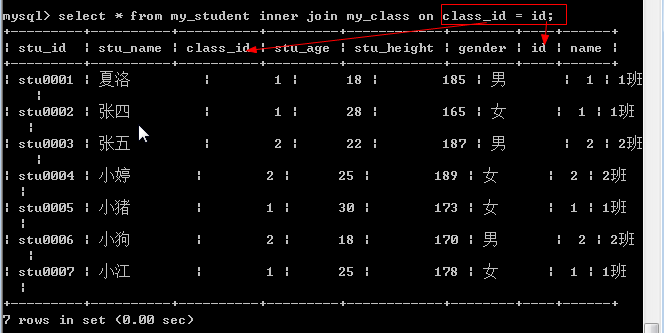
**语法**

基本语法：表1 [inner] join 表2 on 匹配条件;

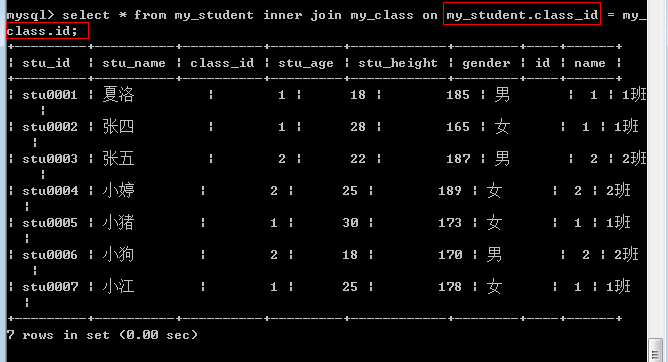
1、 如果内连接没有条件（允许），那么其实就是交叉连接（避免）



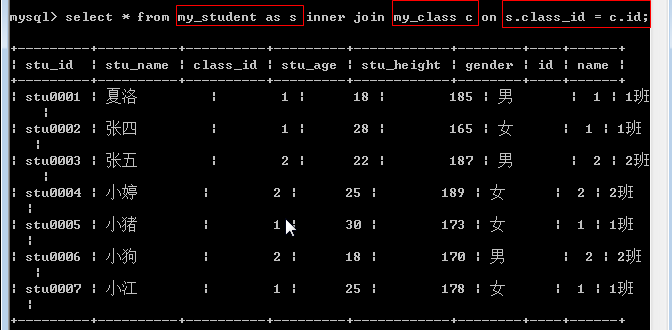
2、 使用匹配条件进行匹配



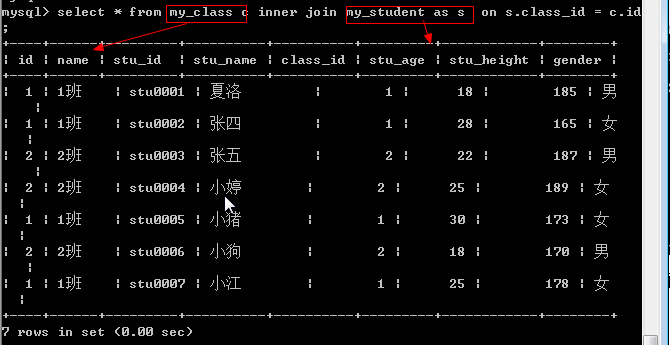
3、 因为表的设计通常容易产生同名字段，尤其是ID，所以为了避免重名出现错误，通常使用表名.字段名，来确保唯一性



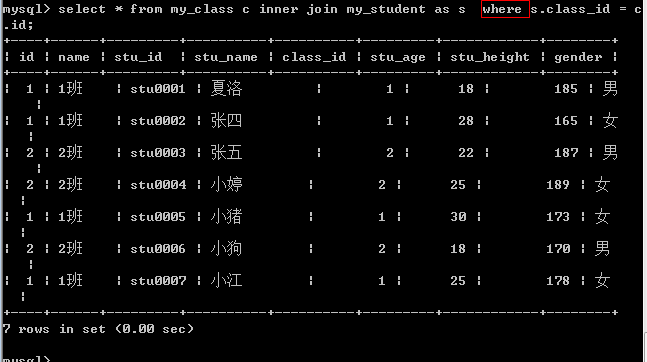
4、 通常，如果条件中使用到对应的表名，而表名通常比较长，所以可以通过表别名来简化



5、 内连接匹配的时候，必须保证匹配到才会保存



6、 内连接因为不强制必须使用匹配条件（on）因此可以在数据匹配完成之后，使用where条件来限制，效果与on一样（建议使用on）



**应用**

内连接通常是在对数据有精确要求的地方使用：必须保证两种表中都能进行数据匹配。

**外连接**

外链接：outer join，按照某一张表作为主表（表中所有记录在最后都会保留），根据条件去连接另外一张表，从而得到目标数据。

外连接分为两种：左外连接（left join），右外连接（right join）

左连接：左表是主表

右连接：右表是主表

**原理**

1、 确定连接主表：左连接就是left join左边的表为主表；right join就是右边为主表

2、 拿主表的每一条记录，去匹配另外一张表（从表）的每一条记录

3、 如果满足匹配条件：保留；不满足即不保留

4、 如果主表记录在从表中一条都没有匹配成功，那么也要保留该记录：从表对应的字段值都未NULL

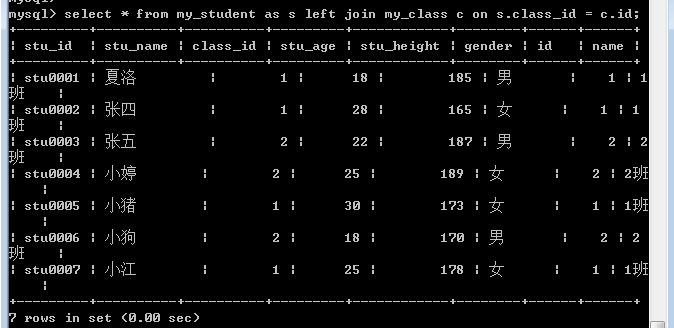
**语法**

基本语法：

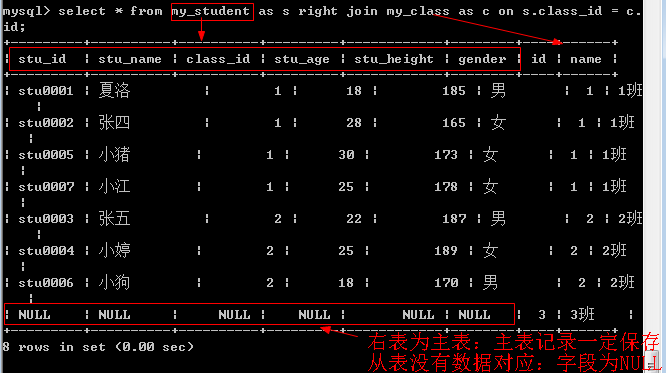
左连接：主表 left join 从表 on 连接条件;

右连接：从表 right join 主表 on连接条件;

左连接对应的主表数据在左边；右连接对应的主表数据在右边：



右连接查看数据



特点

1、 外连接中主表数据记录一定会保存：连接之后不会出现记录数少于主表（内连接可能）

2、 左连接和右连接其实可以互相转换，但是数据对应的位置（表顺序）会改变

**应用**

非常常用的一种获取的数据方式：作为数据获取对应主表以及其他数据（关联）

**Using关键字**

是在连接查询中用来代替对应的on关键字的，进行条件匹配。

**原理**

1、 在连接查询时，使用on的地方用using代替

2、 使用using的前提是对应的两张表连接的字段是同名（类似自然连接自动匹配）

3、 如果使用using关键字，那么对应的同名字段，最终在结果中只会保留一个。

**语法**

基本语法：表1 [inner,left,right] join 表2 using(同名字段列表); //连接字段

