一、mysql 查询的五种子句

where 子句（条件查询）：按照“条件表达式”指定的条件进行查询。

group by 子句（分组）：按照“属性名”指定的字段进行分组。group by 子句通常和 count()、sum()等聚合函数一起使用。

having 子句（筛选）：有 group by 才能 having 子句，只有满足“条件表达式”中指定的条件的才能够输出。

order by 子句（排序）：按照“属性名”指定的字段进行排序。排序方式由“asc”和“desc”两个参数指出，默认是按照“asc”来排序，即升序。

limit（限制结果集）。

1、where——基础查询

where 常用运算符：

运算符	说明
比较运算符	
<	小于
<=	小于或等于
=	等于
!= 或 <>	不等于
>=	大于等于
>	大于

in	在某集合内
between	在某范围内
逻辑运算符	
not 或 !	逻辑非
or 或 ||	逻辑或
and 或 &&	逻辑与

2、group by 分组

“Group By”从字面意义上理解就是根据“By”指定的规则对数据进行分组，所谓的分组就是将一个“数据集”划分成若干个“小区域”，然后针对若干个“小区域”进行数据处理。

```
select 类别, sum(数量) as 数量之和
from A
group by 类别
```

注：group by 语句中 select 指定的字段必须是“分组依据字段”，其他字段若想出现在 select 中则必须包含在聚合函数中。

mysql 中五种常用的聚合函数：

（1）max(列名)：求最大值。

（2）min(列名)：求最小值。

（2）sum(列名)：求和。

（4）avg(列名)：求平均值。

（5）count(列名)：统计记录的条数。

3、having

having 子句可以让我们筛选成组后的各种数据，where 子句在聚合前先筛选记录，也就是说作用在 group by 和 having 子句前。而 having 子句在聚合后对组记录进行筛选。

示例：

```
select 类别, sum(数量) as 数量之和 from A
group by 类别
having sum(数量) > 18
```

示例：Having 和 Where 的联合使用方法

```
select 类别, SUM(数量)from A
where 数量 >8
group by 类别
having SUM(数量) >10
```

where 和 having 的区别：

作用的对象不同。WHERE 子句作用于表和视图，HAVING 子句作用于组。

WHERE 在分组和聚集计算之前选取输入行（因此，它控制哪些行进入聚集计算）， 而 HAVING 在分组和聚集之后选取分组的行。因此，WHERE 子句不能包含聚集函数； 因为试图用聚集函数判断那些行输入给聚集运算是没有意义的。 相反，HAVING 子句总是包含聚集函数。（严格说来，你可以写不使用聚集的 HAVING 子句， 但这样做只是白费劲。同样的条件可以更有效地用于 WHERE 阶段。）

在上面的例子中，我们可以在 WHERE 里应用数量字段来限制，因为它不需要聚集。 这样比在 HAVING 里增加限制更加高效，因为我们避免了为那些未通过 WHERE 检查的行进行分组和聚集计算。

综上所述：

having 一般跟在 group by 之后，执行记录组选择的一部分来工作的。where 则是执行所有数据来工作的。

再者 having 可以用聚合函数，如 having sum(qty)>1000

例子：where + group by + having + 函数 综合查询

练习表：

查询创建工具 查询编辑器

```
1  SELECT * from tb_stu;
```

信息 结果1 状态

id	name	subject	score
1	张三	数学	90
2	张三	语文	50
3	张三	地理	40
4	李四	政治	45
5	李四	语文	55
6	王五	政治	30
7	王五	语文	70

查询出两门及两门以上不及格者的平均成绩(注意是所有科目的平均成绩)

错误情况 1：题意理解错误，理解成查出不及格科目的平均成绩。

查询创建工具 查询编辑器

```
1  SELECT name,avg(score) from tb_stu WHERE score<60 GROUP BY name HAVING count(1)>=2;
```

信息 结果1 状态

name	avg(score)
张三	45
李四	50

错误情况 2：count()不正确，SQL 错误。

查询创建工具 查询编辑器

```
1  SELECT name,avg(score),count(score<60) AS 'count' from tb_stu
2  GROUP BY name HAVING count >=2;
```

信息 结果1 状态

name	avg(score)	count
张三	60	3
李四	50	2
王五	50	2

count(a),无论 a 是什么，都只是数一行；count 时，每遇到一行，就数一个 a，跟条件无关！

正解：count(score<60)达不到想要的结果，并不是条件的问题，而是无论 count()里的表达式是什么都会数一行。score<60 返回 1 或 0；所以可以用 sum(score<60)来计算不及格的科目数！

查询创建工具 查询编辑器

```
1  SELECT name,avg(score),sum(score<60) AS 'gk' from tb_stu
2  GROUP BY name HAVING gk >=2;
```

信息 结果1 状态

name	avg(score)	gk
张三	60	2
李四	50	2

4、order by 排序

（1）order by price //默认升序排列

（2）order by price desc //降序排列

（3）order by price asc //升序排列，与默认一样
（4）order by rand() //随机排列，效率不高

5、limit

limit [offset,] N
offset 偏移量，可选，不写则相当于 limit 0,N
N 取出条目

示例：取价格第 4-6 高的商品

```
select good_id,goods_name,goods_price from goods order by good_price desc limit 3,3;
```

总结：

select 子句顺序

子句	说明	是否必须使用
select	要返回的列或表示式	是
form	从中检索数据的表	仅在从表选择数据时使用
where	行级过滤	否
group by	分组说明	仅在按组计算聚集时使用
having	组级过滤	否
order by	输出排序顺序	否
limit	要检索的行数	否

二、mysql 子查询

1、where 型子查询（把内层查询结果当作外层查询的比较条件）

（1）查询 id 最大的一件商品(使用排序+分页实现)

```
SELECT goods_id,goods_name,shop_price FROM goods ORDER BY goods_id DESC LIMIT 1;
```

（2）查询 id 最大的一件商品(使用 where 子查询实现)

```
SELECT goods_id,goods_name,shop_price FROM goods WHERE goods_id = (SELECT MAX(goods_id) FROM goods);
```

（3）查询每个类别下 id 最大的商品(使用 where 子查询实现)

```
SELECT goods_id,goods_name,cat_id,shop_price FROM goods WHERE goods_id IN (SELECT MAX(goods_id) FROM goods GROUP BY cat_id);
```

2、from 型子查询(把内层的查询结果当成临时表，供外层 sql 再次查询。查询结果集可以当成表看待。临时表要使用一个别名。)

（1）查询每个类别下 id 最大的商品(使用 from 型子查询)

```
 SELECT goods_id,goods_name,cat_id,shop_price FROM
(SELECT goods_id,goods_name,cat_id,shop_price FROM goods ORDER BY cat_id ASC,goods_id DESC) AS tmp
GROUP BY cat_id;
```

子查询查出的结果集看第二张图，可以看到每个类别的第一条的商品 id 都为该类别下的最大值。然后将这个结果集作为一张临时表，巧妙的使用 group by 查询出每个类别下的第一条记录，即为每个类别下商品 id 最大。

```
mysql> #使用from子查询，子查询结果集相当于一张临时表
mysql>
mysql> SELECT goods_id,goods_name,cat_id,shop_price FROM
    ->     (SELECT goods_id,goods_name,cat_id,shop_price FROM goods ORDER BY cat_id ASC,goods_id DESC) AS tmp
    -> GROUP BY cat_id;
+----------+-----------------------------+--------+------------+
| goods_id | goods_name                  | cat_id | shop_price |
+----------+-----------------------------+--------+------------+
|       16 | 恒基伟业G101                |      2 |     823.33 |
|       32 | 诺基亚N85                   |      3 |    3010.00 |
|       18 | 夏新T5                      |      4 |    2878.00 |
|       23 | 诺基亚N96                   |      5 |    3700.00 |
|        7 | 诺基亚N85原装立体声耳机HS-82 |      8 |     100.00 |
|        6 | 胜创KINGMAX内存卡           |     11 |      42.00 |
|       26 | 小灵通/固话20元充值卡       |     13 |      19.00 |
|       30 | 移动20元充值卡              |     14 |      18.00 |
|       28 | 联通50元充值卡              |     15 |      45.00 |
+----------+-----------------------------+--------+------------+
9 rows in set (0.00 sec)
```

```
mysql> #子查询结果
mysql>
mysql> SELECT goods_id,goods_name,cat_id,shop_price FROM goods ORDER BY cat_id ASC,goods_id DESC;
+----------+--------------------------------+--------+------------+
| goods_id | goods_name                     | cat_id | shop_price |
+----------+--------------------------------+--------+------------+
|       16 | 恒基伟业G101                   |      2 |     823.33 |
|       32 | 诺基亚N85                      |      3 |    3010.00 |
|       31 | 摩托罗拉E8                     |      3 |    1337.00 |
|       24 | P806                           |      3 |    2000.00 |
|       22 | 多普达Touch HD                 |      3 |    5999.00 |
|       21 | 金立 A30                       |      3 |    2000.00 |
|       20 | 三星BC01                       |      3 |     280.00 |
|       19 | 三星SGH-F258                   |      3 |     858.00 |
|       17 | 夏新N7                         |      3 |    2300.00 |
|       15 | 摩托罗拉A810                   |      3 |     788.00 |
|       13 | 诺基亚5320 XpressMusic         |      3 |    1311.00 |
|       12 | 摩托罗拉A810                   |      3 |     983.00 |
|       11 | 索爱C702c                      |      3 |    1300.00 |
|       10 | 索爱C702c                      |      3 |    1328.00 |
|        9 | 诺基亚E66                      |      3 |    2298.00 |
|        8 | 飞利浦9@9v                     |      3 |     399.00 |
|       18 | 夏新T5                         |      4 |    2878.00 |
|       14 | 诺基亚5800XM                   |      4 |    2625.00 |
|        1 | KD876                          |      4 |    1388.00 |
|       23 | 诺基亚N96                      |      5 |    3700.00 |
|        7 | 诺基亚N85原装立体声耳机HS-82   |      8 |     100.00 |
|        4 | 诺基亚N85原装充电器            |      8 |      58.00 |
|        3 | 诺基亚原装5800耳机             |      8 |      68.00 |
|        6 | 胜创KINGMAX内存卡              |     11 |      42.00 |
|        5 | 索爱原装M2卡读卡器             |     11 |      20.00 |
|       26 | 小灵通/固话20元充值卡          |     13 |      19.00 |
|       25 | 小灵通/固话50元充值卡          |     13 |      48.00 |
|       30 | 移动20元充值卡                 |     14 |      18.00 |
|       29 | 移动100元充值卡                |     14 |      90.00 |
|       28 | 联通50元充值卡                 |     15 |      45.00 |
|       27 | 联通100元充值卡                |     15 |      95.00 |
+----------+--------------------------------+--------+------------+
31 rows in set (0.00 sec)
```

3、exists 型子查询（把外层 sql 的结果，拿到内层 sql 去测试，如果内层的 sql 成立，则该行取出。内层查询是 exists 后的查询。）

（1）从类别表中取出其类别下有商品的类别(如果该类别下没有商品，则不取出)[使用 where 子查询]

```
SELECT c.cat_id,c.cat_name FROM category c WHERE c.cat_id IN (SELECT g.cat_id FROM goods g GROUP BY g.cat_id);
```

（2）从类别表中取出其类别下有商品的类别(如果该类别下没有商品，则不取出)[使用 exists 子查询]

```
SELECT c.cat_id,c.cat_name FROM category c WHERE EXISTS (SELECT 1 FROM goods g WHERE g.cat_id = c.cat_id);
```

exists 子查询，如果 exists 后的内层查询能查出数据，则表示存在；为空则不存在。

```
mysql> #使用exists子查询
mysql>
mysql> SELECT c.cat_id,c.cat_name FROM category c WHERE EXISTS (SELECT 1 FROM goods g WHERE g.cat_id = c.cat_id);
+--------+-------------------+
| cat_id | cat_name          |
+--------+-------------------+
|      2 | CDMA手机          |
|      3 | GSM手机           |
|      4 | 3G手机            |
|      5 | 双模手机          |
|      8 | 耳机              |
|     11 | 读卡器和内存卡    |
|     13 | 小灵通/固话充值卡 |
|     14 | 移动手机充值卡    |
|     15 | 联通手机充值卡    |
+--------+-------------------+
9 rows in set (0.00 sec)
```

三、连接查询

学习连接查询，先了解下"笛卡尔积"，看下百度给出的解释：

在数学中，两个集合*X*和*Y*的**笛卡尔积**（Cartesian product），又称**直积**，表示为*X* × *Y*，第一个对象是*X*的成员而第二个对象是*Y*的所有可能有序对的其中一个**成员**。

假设集合A={a, b}，集合B={0, 1, 2}，则两个集合的笛卡尔积为{(a, 0), (a, 1), (a, 2), (b, 0), (b, 1), (b, 2)}。

类似的例子有，如果A表示某学校学生的集合，B表示该学校所有课程的集合，则A与B的笛卡尔积表示所有可能的选课情况。A表示所有声母的集合，B表示所有韵母的集合，那么A和B的笛卡尔积就为所有可能的汉字全拼。

在数据库中，一张表就是一个集合，每一行就是集合中的一个元素。表之间作联合查询即是作笛卡尔乘积，比如 A 表有 5 条数据，B 表有 8 条数据，如果不作条件筛选，那么两表查询就有 5 X 8 = 40 条数据。

先看下用到的测试表基本信息：我们要实现的功能就是查询商品的时候，从类别表将商品类别名称关联查询出来。

行数：类别表 14 条，商品表 4 条。

```
mysql> #查询类别表行数
mysql>
mysql> SELECT COUNT(1) FROM category;
+----------+
| COUNT(1) |
+----------+
|       14 |
+----------+
1 row in set (0.00 sec)

mysql> #查询商品表行数
mysql>
mysql> SELECT COUNT(1) FROM mingoods;
+----------+
| COUNT(1) |
+----------+
|        4 |
+----------+
1 row in set (0.00 sec)
```

结构：商品表和类别表都有一个 cat_id

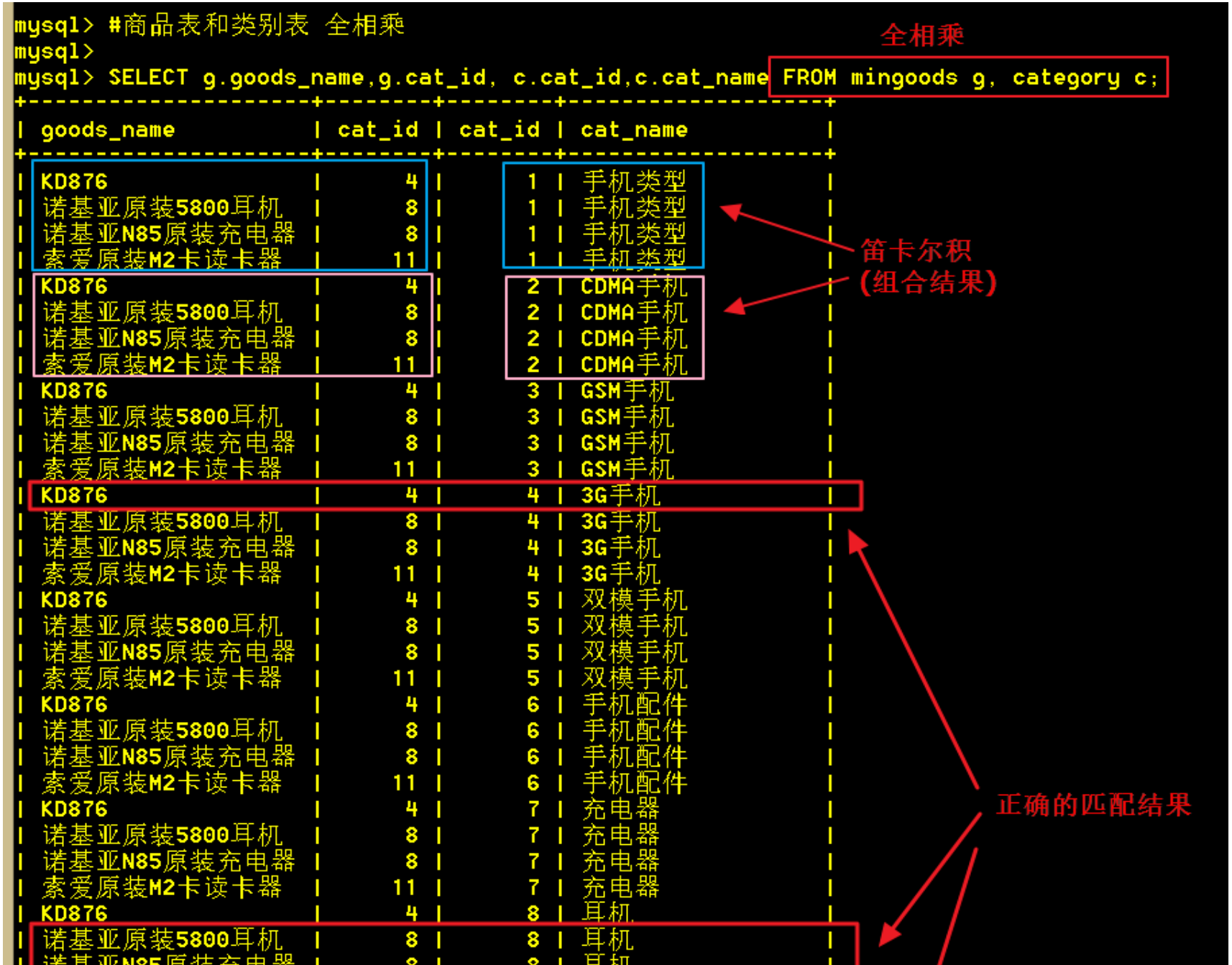
```
mysql> #查询类别表结构
mysql>
mysql> DESC category;
+-----------+----------+------+-----+---------+----------------+
| Field     | Type     | Null | Key | Default | Extra          |
+-----------+----------+------+-----+---------+----------------+
| cat_id    | int(11)  | NO   | PRI | NULL    | auto_increment |
| cat_name  | char(20) | NO   |     |         |                |
| parent_id | int(11)  | YES  |     | NULL    |                |
+-----------+----------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> #查询商品表结构
mysql>
mysql> DESC mingoods;
+--------------+-----------------------+------+-----+---------+----------------+
| Field        | Type                  | Null | Key | Default | Extra          |
+--------------+-----------------------+------+-----+---------+----------------+
| goods_id     | mediumint(8) unsigned | NO   | PRI | NULL    | auto_increment |
| cat_id       | smallint(5) unsigned  | NO   |     | 0       |                |
| goods_sn     | varchar(60)           | NO   |     |         |                |
| goods_name   | varchar(120)          | NO   |     |         |                |
| click_count  | int(10) unsigned      | NO   |     | 0       |                |
| goods_number | smallint(5) unsigned  | NO   |     | 0       |                |
| market_price | decimal(10,2) unsigned| NO   |     | 0.00    |                |
| shop_price   | decimal(10,2) unsigned| NO   |     | 0.00    |                |
| is_best      | tinyint(1) unsigned   | NO   |     | 0       |                |
| is_new       | tinyint(1) unsigned   | NO   |     | 0       |                |
| is_hot       | tinyint(1) unsigned   | NO   |     | 0       |                |
+--------------+-----------------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)
```

1、全相乘(不是全连接、连接查询)，全相乘是作笛卡尔积

两表全相乘，就是直接从两张表里查询；从查询的截图看出，总共查出了 4 X 14 = 56 条记录，这些记录是笛卡尔乘积的结果，即两两组合；

但我们要的是每个商品信息显示类别名称而已，这里却查出了 56 条记录，其中有 52 条记录都是无效的数据，**全相乘的查询效率低**。

```
SELECT goods_id,goods_name,cat_name FROM mingoods,category;
```


mysql> #商品表和类别表 全相乘
mysql>
mysql> SELECT g.goods_name,g.cat_id, c.cat_id,c.cat_name FROM mingoods g, category c;
全相乘
goods_name | cat_id | cat_id | cat_name
KD876 | 4 | 1 | 手机类型
诺基亚原装5800耳机 | 8 | 1 | 手机类型
诺基亚N85原装充电器 | 8 | 1 | 手机类型
索爱原装M2卡读卡器 | 11 | 1 | 手机类型
KD876 | 4 | 2 | CDMA手机
诺基亚原装5800耳机 | 8 | 2 | CDMA手机
诺基亚N85原装充电器 | 8 | 2 | CDMA手机
索爱原装M2卡读卡器 | 11 | 2 | CDMA手机
KD876 | 4 | 3 | GSM手机
诺基亚原装5800耳机 | 8 | 3 | GSM手机
诺基亚N85原装充电器 | 8 | 3 | GSM手机
索爱原装M2卡读卡器 | 11 | 3 | GSM手机
KD876 | 4 | 4 | 3G手机
诺基亚原装5800耳机 | 8 | 4 | 3G手机
诺基亚N85原装充电器 | 8 | 4 | 3G手机
索爱原装M2卡读卡器 | 11 | 4 | 3G手机
KD876 | 4 | 5 | 双模手机
诺基亚原装5800耳机 | 8 | 5 | 双模手机
诺基亚N85原装充电器 | 8 | 5 | 双模手机
索爱原装M2卡读卡器 | 11 | 5 | 双模手机
KD876 | 4 | 6 | 手机配件
诺基亚原装5800耳机 | 8 | 6 | 手机配件
诺基亚N85原装充电器 | 8 | 6 | 手机配件
索爱原装M2卡读卡器 | 11 | 6 | 手机配件
KD876 | 4 | 7 | 充电器
诺基亚原装5800耳机 | 8 | 7 | 充电器
诺基亚N85原装充电器 | 8 | 7 | 充电器
索爱原装M2卡读卡器 | 11 | 7 | 充电器
KD876 | 4 | 8 | 耳机
诺基亚原装5800耳机 | 8 | 8 | 耳机
笛卡尔积
(组合结果)
正确的匹配结果

如果在两张表里有相同字段，做联合查询的时候，要区别表名，否则会报错误(模糊不清)。

```
SELECT goods_name,cat_id,cat_name FROM mingoods,category;
```

```
mysql> #如果在两张表里有相同字段，做联合查询的时候，要区别表名，否则会报错误
mysql>
mysql> SELECT goods_name,cat_id,cat_name FROM mingoods,category;
ERROR 1052 (23000): Column 'cat_id' in field list is ambiguous
```

添加条件，使两表关联查询，这样查出来就是商品和类别一一对应了。虽然这里查出来 4 条记录，但是全相乘效率低，全相乘会在内存中生成一个非常大的数据(临时表)，因为有很多不必要的数据。

如果一张表有 10000 条数据，另一张表有 10000 条数据，两表全相乘就是 100W 条数据，是非常消耗内存的。而且，全相乘不能好好的利用索引，因为全相乘生成一张临时表，临时表里是没有索引的，大大降低了查询效率。

```
SELECT g.goods_name,g.cat_id AS g_cat_id, c.cat_id AS c_cat_id, c.cat_name FROM mingoods g, category c WHERE
g.cat_id = c.cat_id;
```

```
mysql> #添加条件，使两表关联查询
mysql>
mysql> SELECT g.goods_name,g.cat_id AS g_cat_id, c.cat_id AS c_cat_id, c.cat_name FROM mingoods g, category c WHERE g.cat_id = c.cat_id;
+--------------------+----------+----------+----------------+
| goods_name         | g_cat_id | c_cat_id | cat_name       |
+--------------------+----------+----------+----------------+
| KD876              |        4 |        4 | 3G手机         |
| 诺基亚原装5800耳机 |        8 |        8 | 耳机           |
| 诺基亚N85原装充电器 |        8 |        8 | 耳机           |
| 索爱原装M2卡读卡器 |       11 |       11 | 读卡器和内存卡 |
+--------------------+----------+----------+----------------+
4 rows in set (0.00 sec)
```

2、左连接查询 left join ... on ...

语法：

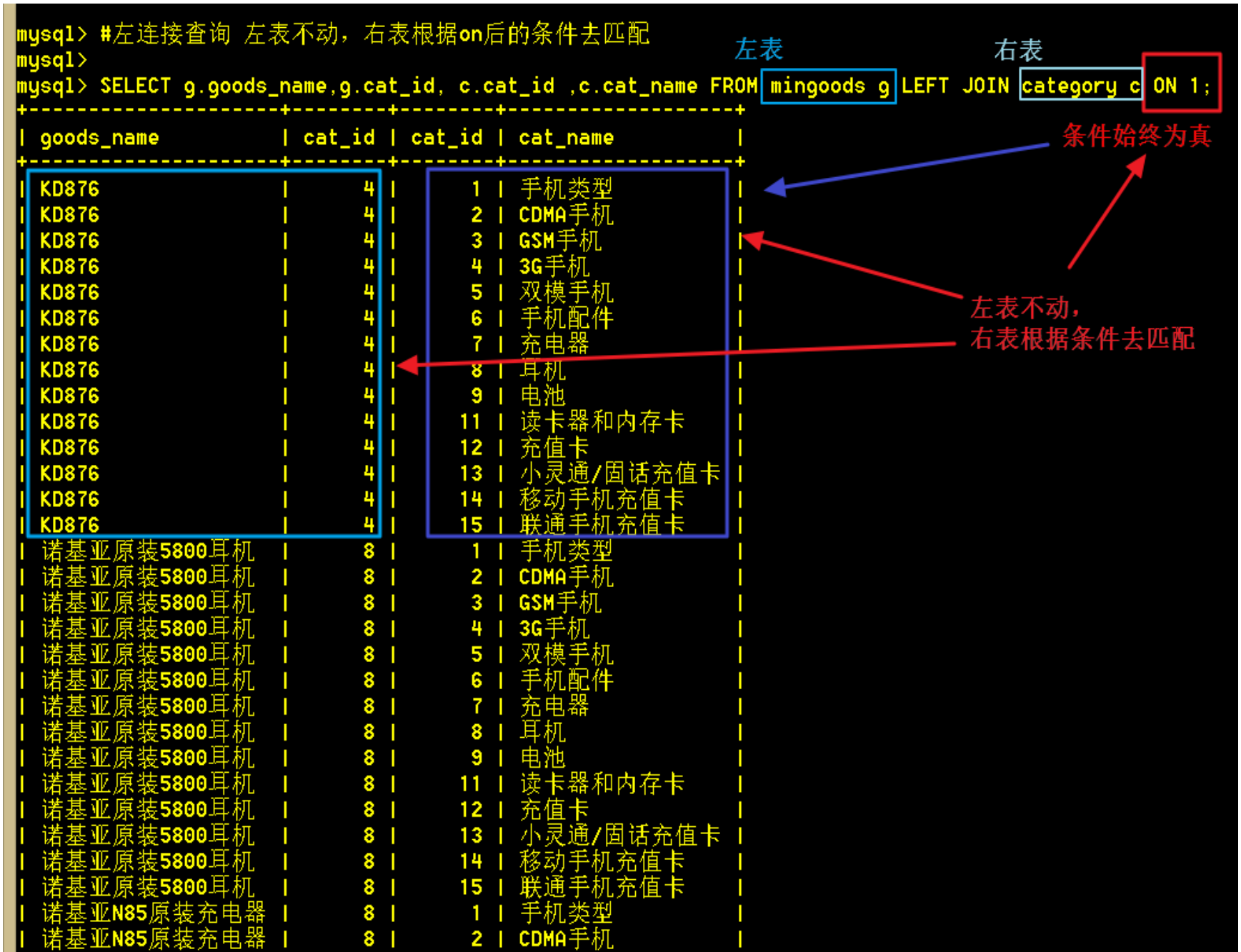
```
select A.filed, [A.filed2, .... ,] B.filed, [B.filed4...,] from <left table> as A  left join <right table> as B on <expression>
```

假设有 A、B 两张表，左连接查询即 A 表在左不动，B 表在右滑动，A 表与 B 表通过一个关系来关联行，B 表去匹配 A 表。

2.1、先来看看 on 后的条件恒为真的情况

```
SELECT g.goods_name,g.cat_id, c.cat_id ,c.cat_name FROM mingoods g LEFT JOIN category c ON 1;
```

跟全相乘相比，从截图可以看出，总记录数仍然不变，还是 4 X 14 = 56 条记录。但这次是商品表不动，类别表去匹配，因为每次都为真，所以将所有的记录都查出来了。左连接，其实就可以看成左表是主表，右表是从表。


mysql> #左连接查询 左表不动，右表根据on后的条件去匹配
mysql>
mysql> SELECT g.goods_name,g.cat_id, c.cat_id ,c.cat_name FROM mingoods g LEFT JOIN category c ON 1;
左表
右表
条件始终为真
左表不动，
右表根据条件去匹配
goods_name	cat_id	cat_id	cat_name
KD876	4	1	手机类型
KD876	4	2	CDMA手机
KD876	4	3	GSM手机
KD876	4	4	3G手机
KD876	4	5	双模手机
KD876	4	6	手机配件
KD876	4	7	充电器
KD876	4	8	耳机
KD876	4	9	电池
KD876	4	11	读卡器和内存卡
KD876	4	12	充值卡
KD876	4	13	小灵通/固话充值卡
KD876	4	14	移动手机充值卡
KD876	4	15	联通手机充值卡
诺基亚原装5800耳机	8	1	手机类型
诺基亚原装5800耳机	8	2	CDMA手机
诺基亚原装5800耳机	8	3	GSM手机
诺基亚原装5800耳机	8	4	3G手机
诺基亚原装5800耳机	8	5	双模手机
诺基亚原装5800耳机	8	6	手机配件
诺基亚原装5800耳机	8	7	充电器
诺基亚原装5800耳机	8	8	耳机
诺基亚原装5800耳机	8	9	电池
诺基亚原装5800耳机	8	11	读卡器和内存卡
诺基亚原装5800耳机	8	12	充值卡
诺基亚原装5800耳机	8	13	小灵通/固话充值卡
诺基亚原装5800耳机	8	14	移动手机充值卡
诺基亚原装5800耳机	8	15	联通手机充值卡
诺基亚N85原装充电器	8	1	手机类型
诺基亚N85原装充电器	8	2	CDMA手机

2.2 、根据 cat_id 使两表关联行

```
SELECT g.goods_name,g.cat_id,c.cat_id,c.cat_name FROM mingoods g LEFT JOIN category c ON g.cat_id = c.cat_id;
```

使用左连接查询达到了同样的效果，但是不会有其它冗余数据，查询速度快，消耗内存小，而且使用了索引。左连接查询效率相比于全相乘的查询效率快了 10+倍以上。

左连接时，mingoods 表(左表)不动，category 表(右表)根据条件去一条条匹配，虽说 category 表也是读取一行行记录，然后判断 cat_id 是否跟 mingoods 表的相同，但是，左连接使用了索引，cat_id 建立了索引的话，查询速度非常快，所以整体效率相比于全相乘要快得多，全相乘没有使用索引。

```
mysql> SELECT g.goods_name,g.cat_id,c.cat_id,c.cat_name FROM mingoods g LEFT JOIN category c ON g.cat_id = c.cat_id;
+--------------------+--------+--------+----------------+
| goods_name         | cat_id | cat_id | cat_name       |
+--------------------+--------+--------+----------------+
| KD876              |      4 |      4 | 3G手机         |
| 诺基亚原装5800耳机 |      8 |      8 | 耳机           |
| 诺基亚N85原装充电器 |      8 |      8 | 耳机           |
| 索爱原装M2卡读卡器 |     11 |     11 | 读卡器和内存卡 |
+--------------------+--------+--------+----------------+
4 rows in set (0.00 sec)
```

2.3、查询出第四个类别下的商品，要求显示商品名称

```
SELECT g.goods_name,g.cat_id,c.cat_name,g.shop_price FROM goods g LEFT JOIN category c ON g.cat_id = c.cat_id WHERE g.cat_id = 4;
```

```
mysql> SELECT g.goods_name,g.cat_id,c.cat_name,g.shop_price FROM goods g LEFT JOIN category c ON g.cat_id = c.cat_id WHERE g.cat_id = 4;
+--------------+--------+----------+------------+
| goods_name   | cat_id | cat_name | shop_price |
+--------------+--------+----------+------------+
| KD876        |      4 | 3G手机   |    1388.00 |
| 诺基亚5800XM |      4 | 3G手机   |    2625.00 |
| 夏新T5       |      4 | 3G手机   |    2878.00 |
+--------------+--------+----------+------------+
3 rows in set (0.00 sec)
```

2.4 、对于左连接查询，如果右表中没有满足条件的行，则默认填充 NULL。

```
SELECT g.goods_name,g.cat_id AS g_cat_id, c.cat_id AS c_cat_id,c.cat_id FROM mingoods g LEFT JOIN mincategory c ON g.cat_id = c.cat_id;
```

```
mysql> #类别表没有cat_id=8这条记录，左连接时，右表的值默认填充NULL
mysql>
mysql> SELECT g.goods_name,g.cat_id AS g_cat_id, c.cat_id AS c_cat_id,c.cat_id FROM mingoods g LEFT JOIN mincategory c ON g.cat_id = c.cat_id;
+--------------------+----------+----------+--------+
| goods_name         | g_cat_id | c_cat_id | cat_id |
+--------------------+----------+----------+--------+
| KD876              |        4 |        4 |      4 |
| 诺基亚原装5800耳机 |        8 |     NULL |   NULL |
| 诺基亚N85原装充电器 |        8 |     NULL |   NULL |
| 索爱原装M2卡读卡器 |       11 |       11 |     11 |
+--------------------+----------+----------+--------+
4 rows in set (0.00 sec)
```

3、右连接查询 right join ... on ...

语法：

```
select A.field1,A.field2,..., B.field3,B.field4  from <left table> A right join <right table> B on <expression>
```

右连接查询跟左连接查询类似，只是右连接是以右表为主表，会将右表所有数据查询出来，而左表则根据条件去匹配，如果左表没有满足条件的行，则左边默认显示 NULL。左右连接是可以互换的。

```
SELECT g.goods_name,g.cat_id AS g_cat_id,  c.cat_id AS c_cat_id,c.cat_name FROM mingoods g RIGHT JOIN mincategory c
ON g.cat_id = c.cat_id;
```

```
mysql> #使用右连接查询，右表所有数据查询出来，左表去匹配记录，如果没有则返回NULL
mysql>
mysql> SELECT g.goods_name,g.cat_id AS g_cat_id,  c.cat_id AS c_cat_id,c.cat_name FROM mingoods g RIGHT JOIN mincategory c ON g.cat_id = c.cat_id;
+--------------------+----------+----------+------------------+
| goods_name         | g_cat_id | c_cat_id | cat_name         |
+--------------------+----------+----------+------------------+
| KD876              |        4 |        4 | 3G手机           |
| NULL               |     NULL |        5 | 双模手机         |
| NULL               |     NULL |        3 | GSM手机          |
| 索爱原装M2卡读卡器 |       11 |       11 | 读卡器和内存卡   |
| NULL               |     NULL |       15 | 联通手机充值卡   |
| NULL               |     NULL |        2 | CDMA手机         |
| NULL               |     NULL |       13 | 小灵通/固话充值卡 |
+--------------------+----------+----------+------------------+
7 rows in set (0.00 sec)
```

查出了7条数据

4、内连接查询 inner join ... on ...

语法：

```
select A.field1,A.field2,.., B.field3, B.field4 from <left table> A inner join <right table> B on <expression>
```

内连接查询，就是取左连接和右连接的交集，如果两边不能匹配条件，则都不取出。

```
SELECT g.goods_name,g.cat_id, c.* from mingoods g INNER JOIN mincategory c ON g.cat_id = c.cat_id;
```

```
mysql> #内连接是取左连接和右连接的交集，如果不满足条件则两边都不取出
mysql>
mysql> SELECT g.goods_name,g.cat_id, c.* from mingoods g INNER JOIN mincategory c ON g.cat_id = c.cat_id;
+--------------------+--------+--------+----------------+-----------+
| goods_name         | cat_id | cat_id | cat_name       | parent_id |
+--------------------+--------+--------+----------------+-----------+
| KD876              |      4 |      4 | 3G手机         |         1 |
| 索爱原装M2卡读卡器 |     11 |     11 | 读卡器和内存卡 |         6 |
+--------------------+--------+--------+----------------+-----------+
2 rows in set (0.00 sec)
```

5、全连接查询 full join ... on ...

语法：

```
select ... from <left table> full join <right table> on <expression>
```

全连接会将两个表的所有数据查询出来，不满足条件的为 NULL。

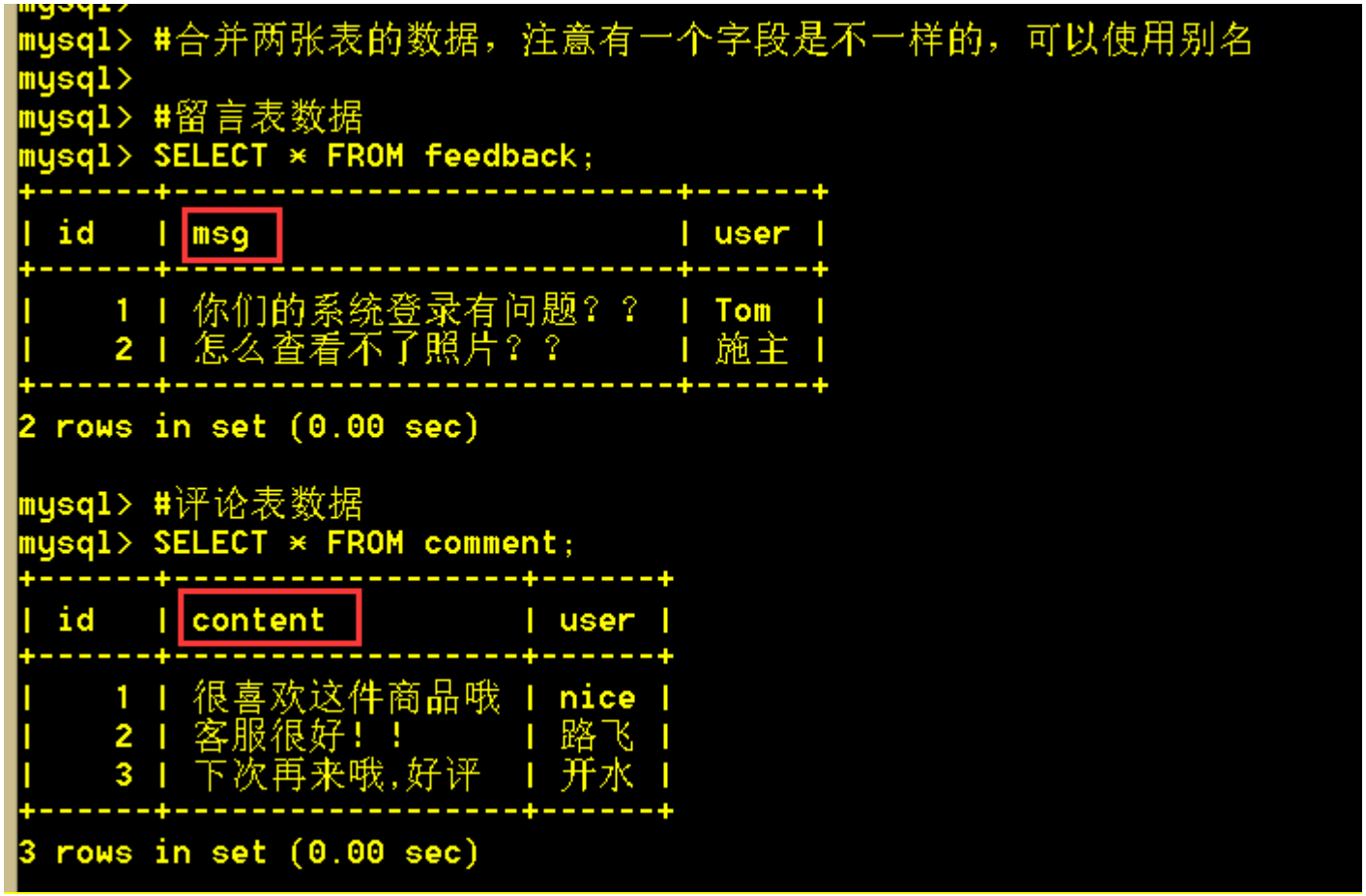
全连接查询跟全相乘查询的区别在于，如果某个项不匹配，全相乘不会查出来，全连接会查出来，而连接的另一边则为 NULL。

6、联合查询 union

语法：

```
select A.field1 as f1, A.field2 as f2 from <table1> A union (select B.field3 as f1, field4 as f2 from <table2> B)
```

union 是求两个查询的并集。union 合并的是结果集，不区分来自于哪一张表，所以可以合并多张表查询出来的数据。



6.1、将两张表的数据合并查询出来

```
SELECT id, content, user FROM comment UNION (SELECT id, msg AS content, user FROM feedback);
```

```
mysql> #将两张表查询出来的数据合并
mysql>
mysql> SELECT id, content, user FROM comment UNION (SELECT id, msg AS content, user FROM feedback);
+------+--------------------------+------+
| id   | content                  | user |
+------+--------------------------+------+
|    1 | 很喜欢这件商品哦         | nice |
|    2 | 客服很好！！             | 路飞 |
|    3 | 下次再来哦,好评          | 开水 |
|    1 | 你们的系统登录有问题？？ | Tom  |
|    2 | 怎么查看不了照片？？     | 施主 |
+------+--------------------------+------+
5 rows in set (0.00 sec)
```

union查询时，后面的表头以第一个查询的表头为准

6.2、union 查询，列名不一致时，以第一条 sql 语句的列名对齐

```
SELECT id, content, user FROM comment UNION (SELECT id, msg, user FROM feedback);
```

```
mysql> #如果两张表的表头不一致 会以第一张表的表头为准 ，并对齐栏目
mysql>
mysql> SELECT id, content, user FROM comment UNION (SELECT id, msg, user FROM feedback);
+------+--------------------------+------+
| id   | content                  | user |
+------+--------------------------+------+
|    1 | 很喜欢这件商品哦         | nice |
|    2 | 客服很好！！             | 路飞 |
|    3 | 下次再来哦,好评          | 开水 |
|    1 | 你们的系统登录有问题？？ | Tom  |
|    2 | 怎么查看不了照片？？     | 施主 |
+------+--------------------------+------+
5 rows in set (0.00 sec)
```

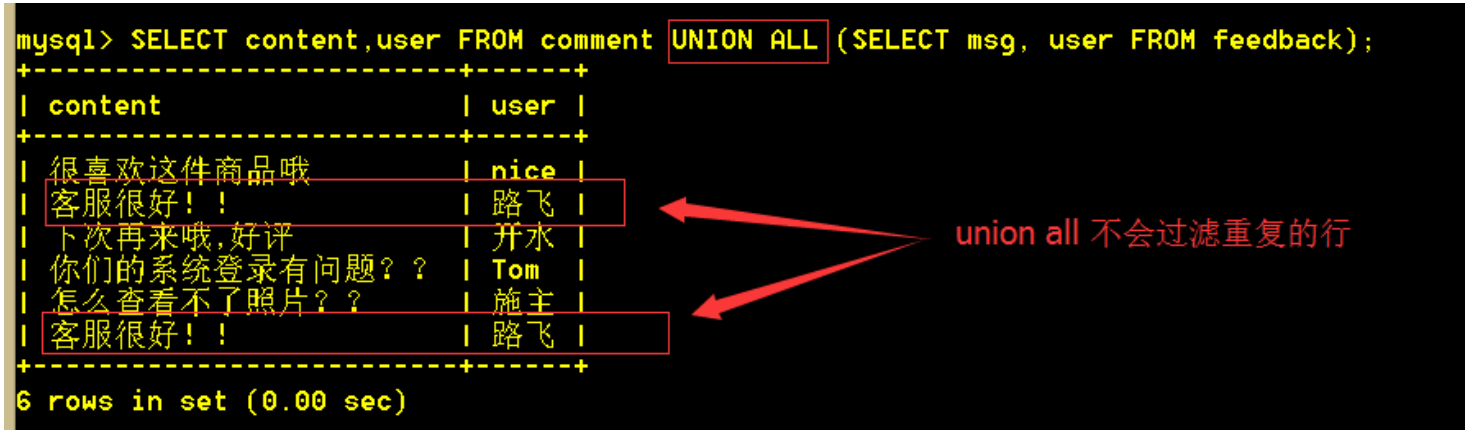
6.3、使用 union 查询会将重复的行过滤掉

```
SELECT content,user FROM comment UNION (SELECT msg, user FROM feedback);
```

```
mysql> #union 会过滤掉重复的行数据
mysql>
mysql> SELECT * FROM feedback;
+------+--------------------------+------+
| id   | msg                      | user |
+------+--------------------------+------+
|    1 | 你们的系统登录有问题？？ | Tom  |
|    2 | 怎么查看不了照片？？     | 施主 |
|    3 | 客服很好！！             | 路飞 |
+------+--------------------------+------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM comment;
+------+------------------+------+
| id   | content          | user |
+------+------------------+------+
|    1 | 很喜欢这件商品哦 | nice |
|    2 | 客服很好！！     | 路飞 |
|    3 | 下次再来哦,好评  | 开水 |
+------+------------------+------+
3 rows in set (0.00 sec)

mysql> #union
mysql>
mysql> SELECT content,user FROM comment UNION (SELECT msg, user FROM feedback);
+--------------------------+------+
| content                  | user |
+--------------------------+------+
| 很喜欢这件商品哦         | nice |
| 客服很好！！             | 路飞 |
| 下次再来哦,好评          | 开水 |
| 你们的系统登录有问题？？ | Tom  |
| 怎么查看不了照片？？     | 施主 |
+--------------------------+------+
5 rows in set (0.00 sec)
```

6.4、使用 union all 查询所有，重复的行不会被过滤

```
SELECT content,user FROM comment UNION ALL (SELECT msg, user FROM feedback);
```



6.5、union 查询，如果列数不相等，会报列数不相等错误

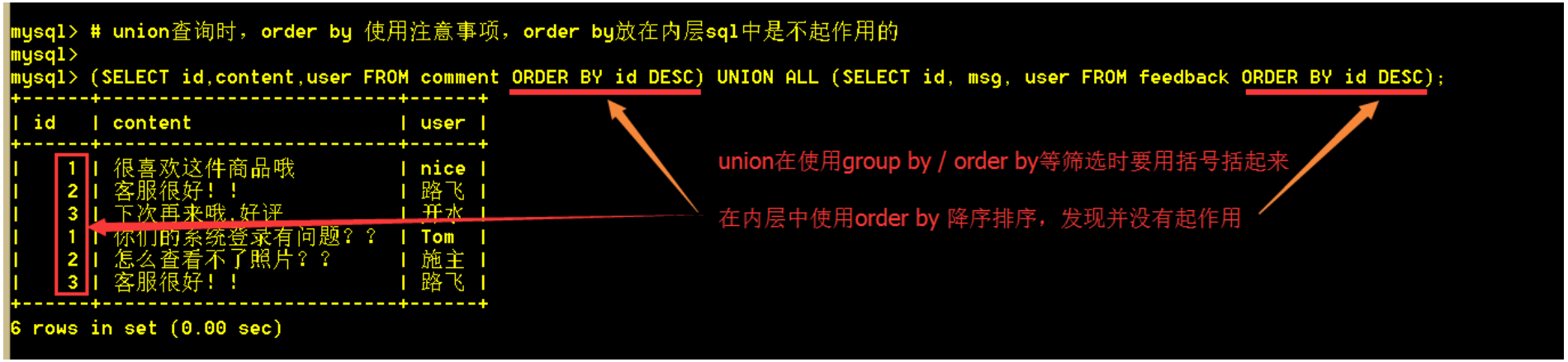
6.6、union 后的结果集还可以再做筛选

```
SELECT id,content,user FROM comment UNION ALL (SELECT id, msg, user FROM feedback) ORDER BY id DESC;
```

```
mysql> #union 后的结果集还可以再做筛选
mysql>
mysql> SELECT id,content,user FROM comment UNION ALL (SELECT id, msg, user FROM feedback) ORDER BY id DESC;
+------+--------------------------+------+
| id   | content                  | user |
+------+--------------------------+------+
|    3 | 客服很好！！             | 路飞 |
|    3 | 下次再来哦,好评          | 开水 |
|    2 | 客服很好！！             | 路飞 |
|    2 | 怎么查看不了照片？？     | 施主 |
|    1 | 很喜欢这件商品哦         | nice |
|    1 | 你们的系统登录有问题？？ | Tom  |
+------+--------------------------+------+
6 rows in set (0.00 sec)
```

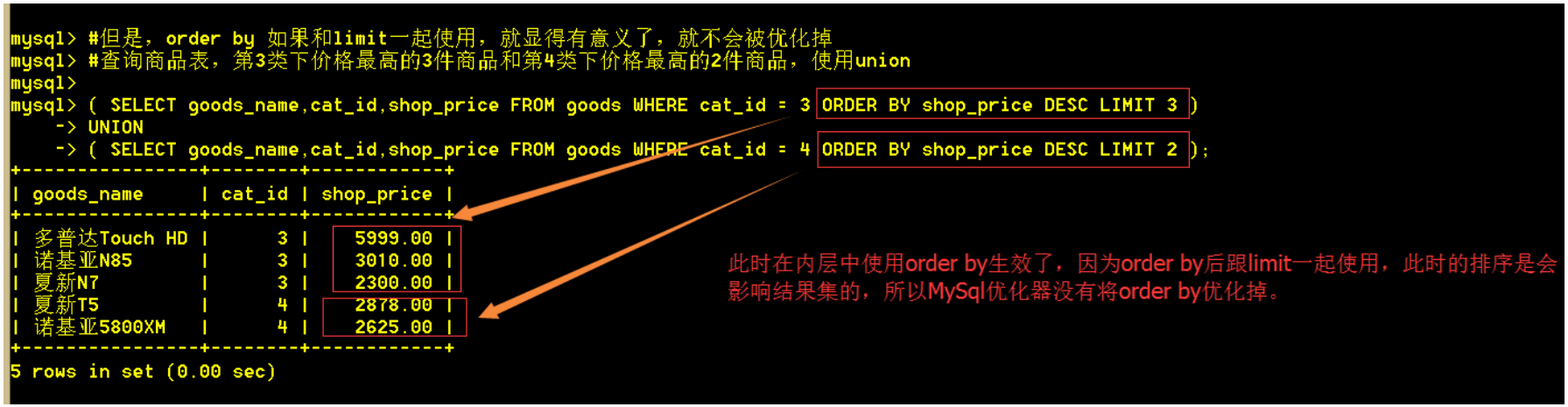
union 查询时，order by 放在内层 sql 中是不起作用的；因为 union 查出来的结果集再排序，内层的排序就没有意义了；因此，内层的 order by 排序，在执行期间，被 mysql 的代码分析器给优化掉了。

```
(SELECT id,content,user FROM comment ORDER BY id DESC) UNION ALL (SELECT id, msg, user FROM feedback ORDER BY id DESC);
```



order by 如果和 limit 一起使用，就显得有意义了，就不会被优化掉。

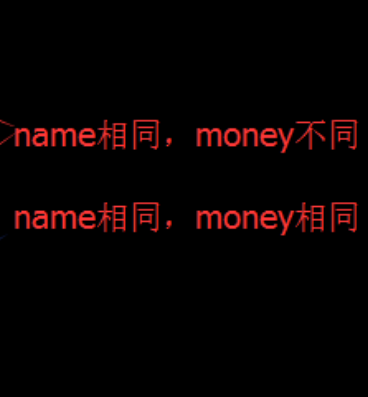
```
( SELECT goods_name,cat_id,shop_price FROM goods WHERE cat_id = 3 ORDER BY shop_price DESC LIMIT 3 )
UNION
( SELECT goods_name,cat_id,shop_price FROM goods WHERE cat_id = 4 ORDER BY shop_price DESC LIMIT 2 );
```

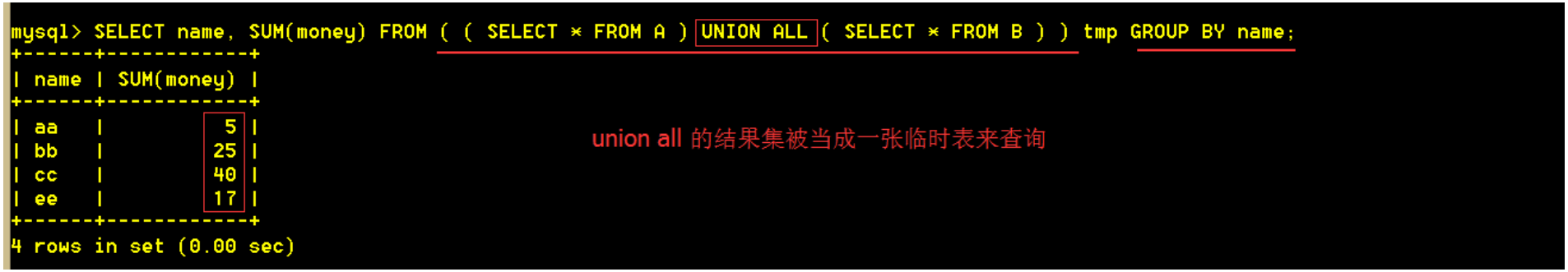


6.7、练习

```
SELECT name, SUM(money) FROM ( ( SELECT * FROM A ) UNION ALL ( SELECT * FROM B ) ) tmp GROUP BY name;
```

```
mysql> #两张表合并查询，如果名字相同的就将money相加，而不是去重
mysql>
mysql> #查看A表数据
mysql> SELECT * FROM A;
+------+-------+
| name | money |
+------+-------+
| aa   |     5 |
| bb   |    10 |
| cc   |    20 |
+------+-------+
3 rows in set (0.00 sec)

mysql> #查看B表数据
mysql> SELECT * FROM B;
+------+-------+
| name | money |
+------+-------+
| bb   |    15 |
| cc   |    20 |
| ee   |    17 |
+------+-------+
3 rows in set (0.00 sec)
```





连接查询总结：

1、在数据库中，一张表就是一个集合，每一行就是集合中的一个元素。连接查询即是作笛卡尔积，比如 A 表有 1W 条数据，B 表有 1W 条数据，那么两表查询就有 1W X 1W = 100W 条数据

2、如果在两张表里有相同字段，做联合查询的时候，要区别表名，否则会报错误(ambiguous 模糊不清)

3、全相乘效率低，全相乘会在内存中生成一个非常大的数据(临时表)，因为有很多不必要的数据。

如果一张表有 10000 条数据，另一张表有 10000 条数据，两表全相乘就是 100W 条数据，是非常消耗内存的。

而且，全相乘不能好好的利用索引，因为全相乘生成一张临时表，临时表里是没有索引的，大大降低了查询效率。

4、左连接查询时，以左表为主表，会将左表所有数据查询出来；左表不动，右表根据条件去一条条匹配，如果没有满足条件的记录，则右边返回 NULL。

右连接查询值，以右表为主表，会将右表所有数据查询出来，右表不动，左表则根据条件去匹配，如果左表没有满足条件的行，则左边返回 NULL。

左右连接是可以互换的：A left join B == B right join A (都是以 A 为主表) 。

左右连接既然可以互换，出于移植兼容性方面的考虑，尽量使用左连接。

5、连接查询时，虽说也是读取一行行记录，然后判断是否满足条件，但是，连接查询使用了索引，条件列建立了索引的话，查询速度非常快，所以整体效率相比于全相乘要快得多，全相乘是没有使用索引的。

使用连接查询，查询速度快，消耗内存小，而且使用了索引。连接查询效率相比于全相乘的查询效率快了 10+倍以上。

6、内连接查询，就是取左连接和右连接的**交集**，如果两边不能匹配条件，则都不取出。
7、MySql 可以用 union(联合查询)来查出左连接和右连接的**并集**。

union 查询会过滤重复的行，union all 不会过滤重复的行。

union 查询时，union 之间的 sql 列数必须相等，列名以第一条 sql 的列为准；列类型可以不一样，但没太大意义。

union 查询时，order by 放在内层 sql 中是不起作用的；因为 union 查出来的结果集再排序，内层的排序就没有意义了；因此，内层的 order by 排序，在执行期间，被 mysql 的代码分析器给优化掉了。

但是，order by 如果和 limit 一起使用，就显得有意义了，会影响最终结果集，就不会被优化掉。order by 会根据最终是否会影响结果集而选择性的优化。

注：union 和 union all 的区别，union 会去掉重复的记录，在结果集合并后悔对新产生的结果集进行排序运算，效率稍低，union all 直接合并结果集，如果确定没有重复记录，建议使用 union all。

8、 LEFT JOIN 是 LEFT OUTER JOIN 的缩写，同理，RIGHT JOIN 是 RIGHT OUTER JOIN 的缩写；JOIN 是 INNER JOIN 的缩写。

关联查询

1、使用 join 关键字关联查询

（1）、内连接（inner join）

连接两张表，连接条件使用 on 关键字，内连接只会显示匹配的数据记录。

eg:查询学生姓名、科目、分数

```
select a.name 姓名,b.subject 科目,b.score 分数 from student a inner join score b on a.id = b.sid;
```

	姓名	科目	分数
	周芷若	高等数学	76
	周芷若	计算机组成原理	85
	周芷若	货币银行学	79
	张三丰	高等数学	63
	张三丰	计算机组成原理	70
	张三丰	货币银行学	68
	张三	高等数学	85
	张三	计算机组成原理	76
	张三	货币银行学	76

（2）、左连接（left join）

返回左表中所有记录以及右表中符合连接条件的所有记录。

eg: 使用左连接查询学生姓名、科目、分数

```
select a.name 姓名,b.subject 科目,b.score 分数 from student a left join score b on a.id = b.sid;
```

	姓名	科目	分数
	周芷若	高等数学	76
	周芷若	计算机组成原理	85
	周芷若	货币银行学	79
	张三丰	高等数学	63
	张三丰	计算机组成原理	70
	张三丰	货币银行学	68
	张三	高等数学	85
	张三	计算机组成原理	76
	张三	货币银行学	76
	王芷晴	(NULL)	(NULL)
	张晴	(NULL)	(NULL)
	王芷	(NULL)	(NULL)
		(NULL)	(NULL)

（3）、右连接（right join）

返回右表中所有记录以及左表中符合连接条件的所有记录。

eg:使用右连接查询学生姓名、科目、分数

```
select a.name 姓名,b.subject 科目,b.score 分数 from student a right join score b on a.id = b.sid;
```

	姓名	科目	分数
	周芷若	高等数学	76
	周芷若	计算机组成原理	85
	周芷若	货币银行学	79
	张三丰	高等数学	63
	张三丰	计算机组成原理	70
	张三丰	货币银行学	68
	张三	高等数学	85
	张三	计算机组成原理	76
	张三	货币银行学	76
	(NULL)	(NULL)	55
	(NULL)	(NULL)	(NULL)

注：内外连接区别：内连接只会显示匹配的数据记录，外连接例如左连接会把左边表中所有记录显示出来，即使在右边表中没有匹配记录也会显示左表的数据，右连接反之。

2、使用表和表之间相同 id 关联查询

这种关联方式和内连接一样，只会显示出匹配的数据

```
select a.name 姓名,b.subject 科目,b.score 分数 from student a,score b where a.id = b.sid;
```

	姓名	科目	分数
	周芷若	高等数学	76
	周芷若	计算机组成原理	85
	周芷若	货币银行学	79
	张三丰	高等数学	63
	张三丰	计算机组成原理	70
	张三丰	货币银行学	68
	张三	高等数学	85
	张三	计算机组成原理	76
	张三	货币银行学	76