MySQL数据库之SQL高级查

目录：

一、SQL中运算符及常用函数

（一）、SQL常用运算符：

1、算术运算符

- 算术运算符（+ 、 - 、 * 、 / 或者div 、% 或者mod）
- 算术运算符包括：+ 、 - 、 * 、 / 或者div 、% 或者mod。作用分别是加法、减法、乘法、除法、取余数。例如：
- mysql> select 0.1+0.2, 0.1-0.2 , 0.1*0.2, 0.1/0.2,1%2;

2、比较运算符：

比较运算符中有以下六类：

- 比较数值大小的：= 、<>或!= 、< 、 > 、 <= 、 > 、>= ；
- 指定范围的：between and 、 not between and；
- 指定集合的：in 、 not in ；
- 是否为空值的：is null 、 is not null ；
- 通配符匹配字符的：like 、not like
- 正则表达式匹配字符的：regexp 或 rlike。

3、逻辑运算符：

- 逻辑运算符（AND 、OR 、 NOT）
- 逻辑运算符又称为布尔运算符，用来确认表达式的真和假。MySQL支持四种逻辑运算符，包括：NOT或! ，AND或&&，OR或||， XOR。作用分别是逻辑非、逻辑与、逻辑或、逻辑异或。在实际工作中常用的为前三个，XOR

一般不使用，这里不做讲解。

- 例如，查询注册会员表中email不为空，电话号码不为空，年龄大于30或小于20岁的男性会员的信息。

（二）、SQL常用函数：

- count()
- length()
- min()
- max()

二、DQL语句高级查询

1） 条件查询
2） 不重复记录查询（distinct）
3） 排序和限制查询（order by 、 limit）
4） 分组聚合查询（group by 、 count() sum() avg() max() min()）
5） 多表连接查询(inner join left join right join ）
6） 子查询(in not in exists any all)
7） 记录联合查询(union , union all)
8） 给表和字段起别名

1、条件查询：

实际工作中，不加条件的查询语句是很少的。用户几乎很少去查询所有的记录，而都是根据需求设置查询条件，由限定条件查询出符合要求的数据。实现方法是使用WHERE条件表达式。因为运算符是构成WHERE 表达式的必要成分，这也就是为什么必须要先学习MySQL运算符的原因了。

语法规则如下：

SELECT …… WHERE 条件表达式

WHERE条件表达式会充分运用各种比较运算符，而多个条件之间还可以使用or，and等逻辑运算符进行多条件的查询。设置的条件越多，查询语句的限制就会越多，能够满足所有条件的记录就会越少。

示例代码详见上一节，“比较运算符”。

2、查询不重复的记录：
实际工作中，经常会需要将表中某字段的重复记录去掉后再显示出来，实现方法是使用distinct关键字。distinct关键字能消除相同的记录。“distinct”的意思是“有区别的，与其它不同的”。
语法规则如下：
SELECT DISTINCT 字段名称

例如，在新闻表中，有时候会重复插入相同标题、相同出处、相同新闻类型的信息，那么在显示的时候就需要去除这些重复的信息。
下面的例子是：显示新闻表中标题、新闻来源、新闻类型都不重复的所有信息。

```
mysql> SELECT  distinct  title  , source  , typeid  FROM tb_news;
+----------------------------+----------+--------+
| title                      | source   | typeid |
+----------------------------+----------+--------+
|菲律宾示威者抗议日本猎杀海豚         |新华网      |      4 |
|全民抢入手!超级娱乐S60诺记N81近底线       |太平洋      |      5|
|俄罗斯承诺将援南奥塞梯每人2000美元         | sohu.com |      6 |
|菲律宾示威者抗议日本猎杀海豚          |新华网     |      2 |
+----------------------------+----------+--------+
4 rows in set (0.03 sec)
```

【备注：】上面的例子中，虽然有两条的标题和新闻来源都一样，但是因为新闻类型不同，所以两条都显示出来了，如果连新闻类型，也就是typeid也一样，那么只显示其中一条了。

3、排序和限制查询：
从表中查询出来的记录可能是无序的，或者排列顺序不是用户所期望的。为了使得查询结果的顺序满足用户要求，可以使用ORDER BY来对记录进行排序。
语法规则如下：
SELECT … ORDER BY 字段名称［ASC | DESC］

ASC表示升序排列，DESC表示降序排列。默认情况下，按照ASC方式排序。

ORDER BY后面可以跟多个不同的排序字段，并且每个排序字段可以有不同的排序方式。

语法规则如下：
SELECT * FROM TABLE_NAME [WHERE ...] [ORDER BY field1 [DESC|ASC] , field2 [DESC|ASC]...];

对于排序后的记录，如果希望只显示一部分，而不是全部，那么可以使用LIMIT关键字来实现。
LIMIT语法规则如下：
SELECT * FROM TABLE_NAME [WHERE ...] [ORDER BY ...] [LIMIT OFFSET , ROW_COUNT];
LIMIT关键字有两种使用方式：
1. 不指定OFFSET偏移量，用法为：LIMIT 记录数；
2. 指定OFFSET偏移量，用法为： LIMIT 偏移量 , 记录数。
LIMIT和ORDER BY关键字联合使用，解决了页面数据分页展示问题。

例如，查询会员注册表中，最后注册的5名会员的部分信息。

```
mysql> SELECT id , username , age , email FROM tb_user ORDER BY id DESC LIMIT 0,5;
+----+--------------+-----+--------------------------+
| id | username     | age | email                    |
+----+--------------+-----+--------------------------+
| 42 | Steven       |  17 | 14455588@qq.com          |
| 41 | zhangyouqiang |  22 | liqiang@sina.com         |
| 39 | xiangjunwang |  32 | wangxiangjun2008@gmail.com |
| 38 | zeqiwang     |  18 | zeqi@qq.com              |
| 37 | shaoqi       |  30 | shaoqi@163.com           |
+----+--------------+-----+--------------------------+
5 rows in set (0.00 sec)
```

4、分组聚合查询：

实际工作中，经常会需要对表中数据按组统计总数量，按照组查其中的最大、最小或平均值，按照组求其总和等等操作。GROUP BY 关键字可以将查询结果按照某个字段或多个字段进行分组。
GROUP BY语法规则如下：
SELECT * FROM TABLE_NAME [WHERE ...] [GROUP BY 字段名] [HAVING 表达式] [WITH ROLLUP];

1) HAVING关键字表示要对分类后的结果再进行条件过滤；
2) WITH ROLLUP是可选语法，表明是否对分类聚合后的结果进行再汇总。如果使用该关键字，则会在所有记录的最后增加一条记录，该记录是上面所有记录的总和。

GROUP BY可以单独使用，但是单独使用时，只能查询出每个分组的一条记录。这样使用的意义不大，因此，一般在使用聚合函数的时候才使用GROUP BY关键字。
常用聚合函数有：COUNT()、SUM() 、MAX()、MIN()、AVG()。
1) COUNT()用来统计记录的总条数；
2) SUM()用来计算某个字段的符合条件的所有值的总和；
3) MAX()用来查询某字段的符合条件的所有值中的最大值；
4) MIN()用来查询某字段的符合条件的所有值中的最小值；
5) AVG()用来计算某字段的符合条件的所有值的平均值。

运用聚合函数的语法如下：

```
SELECT [field1, field2 , field3...] FUNCTION_NAME
FROM TABLE_NAME
[WHERE where_condition]
[GROUP BY field1 , field2...]
[HAVING where_condition]
[WITH ROLLUP]
```

聚合函数一般情况下都跟GROUP BY一起使用。在MYSQL5.1官方手册中就将聚合函数称为GROUP BY函数。因为聚合函数通常都是用来计算某一组数据的总和、最大值、最小值、平均值。

【备注：】使用聚合函数的时候，不一定非要跟GROUP BY 关键字一起使用。例如统计表中所有的记录数时，就可以直接使用COUNT()，统计某个表中所有记录中的最大值、最小值、平均值和总和时就不用GROUP BY关键字。但是反之，GROUP BY 关键字一般都在使用聚合函数的时候才更有意义。

例1，查询用户注册表中男性和女性各有多少注册用户。

```
mysql> SELECT sex , COUNT(*) FROM tb_user GROUP BY sex;
+-----+----------+
| sex | COUNT(*) |
+-----+----------+
|   0 |        1 |
|   1 |       14 |
+-----+----------+
2 rows in set (0.00 sec)
```

例1中，性别为0（女）的有1位，性别为1（男）的有14位。
从例1中看到查询出来的字段有sex和count(*)，count(*)看起来不够直观，最后用一个更加直观的名字来表示这个列。为了解决这个问题，可以为字段取别名。

MySQL中为字段取别名的语法规则如下：
字段［AS］别名

因此例1可更改为：
查询用户注册表中男性和女性各有多少注册用户。

```
mysql> SELECT sex AS 性别，COUNT(*) AS 总数 FROM tb_user GROUP BY sex;
+-----+----------+
| 性别 | 总数 |
+-----+----------+
|   0 |        1 |
|   1 |       14 |
+-----+----------+
2 rows in set (0.00 sec)
```

例2，查询用户注册表中男性和女性各有多少注册用户，以及注册会员的总数。
分析：只需对例1稍加修改，增加上WITH ROLLUP即可。

mysql> SELECT sex AS 性别，COUNT(*）AS 总数 FROM tb_user GROUP BY sex WITH ROLLUP;

```
+-----+----------+
| 性别 | 总数 |
+-----+----------+
|   0 |        1 |
|   1 |       14 |
| NULL |       15 |
+-----+----------+
3 rows in set (0.00 sec)
```

【备注：】注意WITH ROLLUP的用法，在所有记录的最后加上了一条记录，该记录是上面所有记录的总和。

例3，查询会员注册表中男性和女性用户的平均年龄。

mysql> SELECT sex AS 性别，AVG(age）AS 平均年龄 FROM tb_user GROUP BY sex;

```
+-----+----------+
| 性别 | 平均年龄 |
+-----+----------+
|   0 |  22.0000 |
|   1 |  28.5000 |
+-----+----------+
2 rows in set (0.03 sec)
```

例4，查询会员注册表中男性和女性会员的最大年龄。

```
mysql> SELECT  sex AS 性别，MAX(age）AS  最大年龄  FROM   tb_user
GROUP  BY  sex ;
+-----+----------+
| 性别 | 最大年龄 |
+-----+----------+
|   0 |       22 |
|   1 |       32 |
+-----+----------+
2 rows in set (0.00 sec)
```

例5，以新闻表为例。新闻有各种新闻类型，每种新闻类型下都有很多新闻信息。
统计新闻信息表中每种新闻类型各有多少条新闻。

```
mysql> SELECT typeid AS新闻类型, COUNT(*) AS  新闻总条数 FROM tb_news
GROUP BY typeid;
+--------+----------+
| 新闻类型 | 新闻总条数 |
+--------+----------+
|      1 |       50 |
|      2 |       21|
|      3 |        2 |
|      4 |       15 |
|      5 |       12 |
+--------+----------+
5 rows in set (0.00 sec)
```

例6，以订单明细表为例。
通过订单明细表，统计每种商品售出的总数。（只要下订单即可，不论是否发货或到货）
分析：商品明细表中有字段是goods_id，这个是商品的唯一标识id，有一个字段

是goods_count，也就是每个订单中购买该商品的数量。那么只要按照goods_id分组，然后统计goods_count的总和，那么就可以达到查询目的。

mysql> SELECT goods_id AS 商品id，SUM(goods_count）AS 销售总量 FROM tb_orderlist GROUP BY goods_id;

```
+----------+-----------------+
|商品id |销售总量|
+----------+-----------------+
|       2 |              7 |
|       3 |              2 |
|       4 |              4 |
+----------+-----------------+
3 rows in set (0.00 sec)
```

例7，对例6稍加修改。统计新闻信息表中新闻类型的总条数大于20条的新闻类型。

mysql> SELECT typeid AS 新闻类型, COUNT(*) AS 新闻总条数 FROM tb_news GROUP BY typeid HAVING 新闻总条数>20;

```
+--------+----------+
| 新闻类型 | 新闻总条数 |
+--------+----------+
|      1 |       50 |
|      2 |       21|
+--------+----------+
2 rows in set (0.00 sec)
```

备注：本例中查询出的记录是在例5的基础上再次进行了一次过滤，也就是将新闻总条数中小于20的都过滤掉了。从本例可以看出，HAVING是对聚合后的结果再进行条件的过滤。那么HAVING 和WHERE区别在哪里呢?

【备注：】HAVING和WHERE的区别：

HAVING是对聚合后的结果再进行条件的过滤，而WHERE是在聚合前就对记录进行过滤，如果逻辑允许，我们尽可能用WHERE先过滤记录，这样因为结果集减小，将对聚合的效率大大提高，最后再根据逻辑判断是否还需要用HAVING进行

再次过滤。

5、多表连接查询：
实际工作中，有时候需要查询的信息在多个表中，仅仅通过查询一个表，无法一次性获取到所有需要的信息。于是就需要使用多表的连接查询。
当多个表中存在相同意义的字段时，就可以通过该字段来连接这几个表。连接查询就是将两个或两个以上的表按某个条件连接起来，从中选取需要的数据。连接查询是同时查询两个或两个以上的表的时候使用的。
　　连接查询又分为内连接和外连接。后者又分为左连接和右连接。而内连接时最常用的连接查询。

（1）．　内连接：内连接仅选出两张表中互相匹配的记录，而外连接可以选出其它不匹配的记录。
内连接语法规则如下：
SELECT 字段1 , 字段2 ... FROM 表名1 , 表名2 ...
WHERE 表名1.字段 = 表名2.字段;

内连接语法规则2（该语法规则是根据外连接语法规则来的）：
SELECT 字段1 , 字段2 ... FROM 表名1
INNER JOIN 表名2
ON 表名1.字段 = 表名2.字段;

　　为了能一次性查出多个表的信息出来，就要将多个表连接起来，而多个表能连接起来的关键就在于这些表中有表示相同意义的字段。找到这个字段，将其作为WHERE查询条件，就可以实现内连接查询。
例如查询新闻表，可以查询到新闻id、新闻标题、新闻来源、新闻类型id、新闻内容等等。虽然能查出新闻的类型id，但是这个类型id代表的是什么新闻，却不能一下子知道，只能再去查新闻类型表，才能知道每个类型id代表的中文含义。
那么能不能就查询一次，一下子就能得到新闻id、新闻标题、新闻来源、新闻类型的名称、新闻内容这些信息呢?
能！这就要通过表连接查询，把新闻表和新闻类型表连接起来。那么连接的关键是什么呢？新闻表（tb_news）中有个typeid字段，代表新闻的类型id，这个值正好是新闻类型表(tb_newstype)中的id值 ， 也就是“

tb_news.typeid=tb_newstype.id ”。于是连接点就找到了。

例1，查询新闻表中最新发布的一条新闻，要求有新闻id、新闻标题、新闻来源、新闻类型名称。

```
mysql> SELECT  tb_news.id ，title ，source , typename  FROM  tb_news ，
tb_newst
ype  WHERE tb_news.typeid=tb_newstype.id  LIMIT 1;
+----+---------------+--------------+----------+
| id | title         | source       | typename |
+----+---------------+--------------+----------+
| 56 | 菲律宾示威者抗议日本猎杀海豚 | 新华网 | 国内新闻     |
+----+---------------+--------------+----------+
1 row in set (0.00 sec)
```

从例1中看到，查询新闻id的时候要写成tb_news.id。因为将两个表连接起来查询时，两个表中都有id字段，那么到底要显示那个表的id呢，就需要在字段前写上“表名称.”。如果字段比较多，或者表名称比较长，那么在查询中直接使用表名很不方便，这个时候可以为表取一个别名。

MySQL中为表取别名的语法规则如下：

表名 表的别名

因此例1可更改为：

查询新闻表中最新发布的一条新闻，要求有新闻id、新闻标题、新闻来源、新闻类型名称。

```
mysql> SELECT  n.id ，title ，source , typename  FROM  tb_news  n ，
tb_newstype   t   WHERE n.typeid=t.id  LIMIT 1;
+----+---------------+--------------+----------+
| id | title         | source       | typename |
+----+---------------+--------------+----------+
| 56 | 菲律宾示威者抗议日本猎杀海豚 | 新华网 | 国内新闻     |
+----+---------------+--------------+----------+
```

1 row in set (0.00 sec)

通过修改后例1可以看到，新闻表tb_news 的别名是n ，新闻类型表tb_newstype 的别名是t 。 有了表别名，那么在写WHERE条件的时候，只用写“表别名1.字段=表别名2.字段”。而且展示字段列表的时候，只用写“表别名.字段”即可。

例2，对例1稍加改动，增加一个查询条件。
查询新闻表中新闻类型为3的，最新发布的一条新闻，要求有新闻id、新闻标题、新闻来源、新闻类型名称。

```
mysql> SELECT  n.id ,  title ,  source , typename ,typeid  FROM  tb_news  n ,  tb_newstype   t   WHERE n.typeid=t.id  AND  n.typeid=3  LIMIT 1;
+----+---------------+--------------+----------+
| id | title         | source       | typename |
+----+---------------+--------------+----------+
| 56 | 菲律宾示威者抗议日本猎杀海豚 | 新华网 | 国内新闻     |
+----+---------------+--------------+----------+
1 row in set (0.00 sec)
```

例2是让大家明白，如果在内连接查询后，还有其它的查询条件，只要在WHERE子句中增加AND表达式即可。

例3，用内连接查询的另一种写法实现。
查询新闻表中新闻类型为3的，最新发布的一条新闻，要求有新闻id、新闻标题、新闻来源、新闻类型名称。

```
mysql> SELECT  n.id ,  title ,  source , typename  FROM  tb_news  n  INNER  JOIN  tb_newstype   t   ON  n.typeid=t.id  WHERE  n.typeid=3  LIMIT 1;
+----+---------------+--------------+----------+
| id | title         | source       | typename |
+----+---------------+--------------+----------+
| 56 | 菲律宾示威者抗议日本猎杀海豚 | 新华网 | 国内新闻     |
+----+---------------+--------------+----------+
1 row in set (0.00 sec)
```

（2）.　外连接：外连接分左连接和右连接。
1）　左连接：包含所有的左边表中的记录，甚至是右边表中没有和它匹配的记录。
2）　右连接：包含所有的右边表中的记录，甚至是左边表中没有和它匹配的记录。

外连接语法规则如下：
SELECT 字段1 , 字段2 ... FROM 表名1
LEFT JOIN | RIGHT JOIN 表名2
ON 表名1.字段 = 表名2.字段;

LEFT JOIN表示左连接，RIGHT JOIN表示右连接，ON后面接的是连接条件。
通过左连接查询，可以查询出“表1”中所有记录，而“表2”中只能查出跟“表1”匹配的记录，其它的记录无法显示。
通过右连接查询，可以查询出“表2”中所有记录，而“表1”中只能查出跟“表2”匹配的记录，其它的记录无法显示。

例4，用左连接查询实现。
查询新闻表中最新发布的5条新闻，要求有新闻id、新闻标题、新闻来源、新闻类型名称。

```
mysql> SELECT n.id , title , source , typename FROM tb_news n LEFT JOIN
tb_newstype t ON n.typeid=t.id LIMIT 5;
+----+----------------------------+--------------+----------+
| id | title                      | source       | typename |
+----+----------------------------+--------------+----------+
| 56 |菲律宾示威者抗议日本猎杀海豚              |新华网 | 国内新闻     |
| 58 |超强S60智能怪物LG KT610评测|太平洋          | 手机新闻     |
| 59 |全民抢入手!超级娱乐S60诺记N81近底线     |太平洋          |手机新闻     |
| 61 |俄罗斯承诺将援南奥塞梯每人2000美元       | sohu.com     |国际新闻     |
| 62 |美国宣布向格鲁吉亚提供10亿美元援助       | sina.com         | NULL     |
```

```
+----+----------------------------+--------------+----------+
```
5 rows in set (0.00 sec)

使用左连接查询后，出现一条新闻的新闻类型为NULL。这是因为新闻表中id为62的新闻的typeid为0，而在新闻类型表中没有id为0的数据。这个时候两个表中有不匹配的记录。因为左连接查询，可以查询出“表1”（例3中就是新闻表）中所有记录，所以id为62的记录虽然在新闻类型表中无匹配记录，也被显示出来了。一定要注意跟内连接区别。

例5，用右连接查询实现。
查询新闻表中最新发布的5条新闻，要求有新闻id、新闻标题、新闻来源、新闻类型名称。

```
mysql> SELECT  n.id ,  title ,  source , typename  FROM  tb_news  n  RIGHT
JOIN
  tb_newstype   t   ON  n.typeid=t.id   limit  5;
+------+----------------------------+--------------+----------+
| id   | title                      | source       | typename |
+------+----------------------------+--------------+----------+
| 56 |菲律宾示威者抗议日本猎杀海豚              |新华网 | 国内新闻     |
| 58 |超强S60智能怪物LG KT610评测|太平洋           | 手机新闻     |
| 59 |全民抢入手!超级娱乐S60诺记N81近底线     |太平洋           |手机新闻     |
| 61 |俄罗斯承诺将援南奥塞梯每人2000美元       | sohu.com      |国际新闻     |
| NULL | NULL                     | NULL         |国内娱乐     |
+------+----------------------------+--------------+----------+
```
5 rows in set (0.00 sec)

使用右连接查询后，出现一条新闻信息为NULL， 而新闻类型为“国内娱乐”的记录。而此时新闻表中id为62的新闻也不显示了。因为右连接查询，可以查询出“表2”（本例中就是新闻类型表）中所有记录， 因此虽然暂时没有“国内娱乐”类型的新闻，但是该记录也被显示出来了。 此外，因为“表1”中只显示跟“表2”匹配的记录，因为id为62的记录跟“表2”不匹配，所以不再显示。一定要注意跟内连接区别。

6、子查询（嵌套查询）：
实际工作中，需要将一个表的结果显查询出来，然后作为另一个SELECT语句的条件。这就要用到子查询。子查询就是将一个查询语句嵌套在另一个查询语句中。内层查询语句的查询结果，可以作为外层查询语句的条件。
子查询，也叫做内查询。虽然有些参考书中，将子查询和嵌套查询作为两种查询方式来讲，但是其实嵌套查询就是子查询。所以子查询，又叫做内查询或者嵌套查询。

通过子查询，可以实现多表之间的查询。

子查询中可以包含比较运算符：= 、!= 、< 、 > 、 <= 、 > 、>= 、IN 、NOT IN， 还可以包括：ANY 、SOME 、ALL 、 EXISTS 、NOT EXISTS等关键字。

子查询和表连接查询都可以实现多表之间的查询。某些情况下，子查询可以转化为表连接查询。但是在查询效率上会有所不同，需要大家逐步积累和总结。
（在mysql4.1版本前不支持子查询，需要用表连接查询来实现子查询的功能。表连接很多情况下用于优化子查询。

子查询可总结为以下七类：
- 利用内层查询语句派生出新表的子查询；
- 利用内层查询语句作为表达式的子查询；
- 带常规比较运算符的子查询。包括的运算符有：= 、<>或!= 、< 、 > 、 <= 、 > 、>=等 ；
- 带IN关键字的子查询；
- 带EXISTS关键字的子查询；
- 带ANY或SOME关键字的子查询；
- 带ALL关键字的子查询。

（1）． 利用内层查询语句派生出新表的子查询。
语法结构如下：
SELECT 字段1 , 字段2 ... FROM （SELECT子查询语句）[AS] 别名;

从语法规则上可以看出，平时FROM后跟的是表名，而此时后面跟另一个SELECT子查询语句。这里将内层查询语句派生出一个新表，然后给这个派生出的新表起个别名。

例1，统计新闻信息表中新闻类型的总条数大于20条的新闻类型。
细心的同学会发觉，这个例子就是“4.2.4分组聚合查询”中的例7。那么这个例子又跟子查询有什么关系呢？好，我们先回顾一下4.2.4分组聚合查询例7的SQL语句的写法：

```
mysql> SELECT typeid AS  新闻类型, COUNT(*) AS  新闻总条数 FROM
tb_news  GROUP BY  typeid  HAVING  新闻总条数>20;
+--------+----------+
| 新闻类型 | 新闻总条数 |
+--------+----------+
|      1 |        50 |
|      2 |        21|
+--------+----------+
2 rows in set (0.00 sec)
```

大家能看到，显示出两列，“新闻类型”和“新闻总条数”。如果我只想要得到符合条件的“新闻类型”，不要显示“新闻总条数”可以做到呢？第二列是由聚合函数COUNT(*)产生的，如果不要第二列，就不可能有语句后面的HAVING条件。到底如何做呢？利用子查询就可以做到。

例2，统计新闻信息表中新闻类型的总条数大于20条的新闻类型的id。仅仅显示新闻类型id一列，不要显示新闻条数。

```
mysql> SELECT typeid FROM（select typeid , COUNT(*) FROM tb_news
GROUP BY  typeid HAVING count(*) >20 ）AS  a;
+--------+
| typeid |
+--------+
|      1 |
|      2 |
+--------+
2 rows in set (0.00 sec)
```

大家能看到例2跟例1结果上的差异，一个有两个字段，一个只显示一个字段。这就是利用内层查询语句派生出新表的子查询的效果。例2的SQL语句中记

得给派生出的新表起别名，其中AS关键字可以使用，也可以不用。

（2）．　利用内层查询语句作为表达式的子查询。

在“4.2.4分组聚合查询”的例3中有这样的查询例子。查询会员注册表中男性和女性用户的平均年龄。

```
mysql> SELECT sex  AS 性别，AVG(age）AS  平均年龄　FROM　tb_user
GROUP  BY  sex;
+-----+----------+
| 性别 | 平均年龄 |
+-----+----------+
|　0 |  22.0000 |
|　1 |  28.5000 |
+-----+----------+
2 rows in set (0.00 sec)
```

查询结果是如上，两行数据，两列字段。这样的显示结果不够直观，能不能显示的更加直观些？如果查询要求是就显示成一行数据，两列字段，该如何实现呢？接下来的例3就是利用内层查询语句作为表达式的方式来实现的。

例3，查询会员注册表中男性和女性用户的平均年龄。要求显示成一行两列。

```
mysql> SELECT (SELECT AVG(age) FROM tb_user WHERE sex=1) AS 男会员平均年龄 , (SELECT AVG(age) FROM tb_user WHERE sex=0）AS  女会员平均年龄 ;
+-----+----------+
|男会员平均年龄| 女会员平均年龄 |
+-----+----------+
|　28.5000　|　22.0000　|
+-----+----------+
2 rows in set (0.00 sec)
```

（3）．　带常规比较运算符的子查询。

包括的运算符有：= 、<>或!= 、< 、 > 、 <= 、 > 、>=等。

例4， 查询注册会员表中年龄最大的男性会员的注册信息。

```
mysql> SELECT id , username , age , sex  FROM tb_user WHERE  age =
(SELECT MAX(age)  FROM  tb_user  WHERE  sex=1);
+----+----------+-----+-----+
| id | username | age | sex |
+----+----------+-----+-----+
|  1 | 王向军   |  32 |   1 |
+----+----------+-----+-----+
1 row in set (0.00 sec)
```

例4中的子查询语句返回结果只有一个值，所以可以使用“age=”直接赋值，如果子查询语句的返回结果是一组记录，又该如何呢？答案是可以借助带IN关键字的子查询。

（4）． 带IN关键字的子查询。

稍微修改例2，将查询条件更改为：

例5，查询新闻总数大于100条的那几个类型的新闻，只需要显示最后发布的5条即可。

分析：新闻总数大于100条的那几个新闻类型有哪些呢？通过例2就可以获取到他们的typeid，那么在新闻表中查找typeid是这些的新闻不就可以了吗？

```
mysql> SELECT * FROM tb_news WHERE typeid  IN
(SELECT typeid FROM (SELECT typeid , COUNT(*) FROM tb_news GROUP BY
typeid HAVING COUNT(*) >0 ) AS a);
```

例5中使用了关键字IN。因为其子查询语句返回的不是一个typeid，而是一组typeid。所以不能使用“=”，就只能使用IN关键字。

（5）． 带EXISTS关键字的子查询。

EXISTS表示存在，使用EXISTS关键字时，内层查询语句不返回查询记录。只返回true或者false。当返回true，外层查询语句将进行查询，当返回false的时候，

外层查询语句不查询或查询不出任何记录。
以新闻表表和新闻类型表为例。新闻类型表中有id为1、2、3、4、5这5种新闻类型，相应在新闻表中也会存在typeid为这5种的新闻。某一日，用户将新闻类型表中的id为5的记录删除了，而新闻表中typeid为5的数据未同时删除。那么新闻表中的typeid为5的数据就“无家可归”，因此在显示在网页上的时候也不应该显示。可是typeid为5的记录明明又在新闻表中存在着，如何让它不显示呢?
为了解决以上的问题，可以用表连接查询中的内连接。因为内连接是两个表中的记录都完全匹配的时候才能查询出来。除了内连接外，还有一种做法就是利用带EXISTS关键字的子查询。
EXISTS子查询语法结构如下：
SELECT 字段1 , 字段2 ... FROM 表名
WHERE［ where条件语句 ］［ AND ］
EXISTS （SELECT子查询语句）；
如果查询的时候外层查询语句没有额外的where条件，那么WHERE 后就直接连EXISTS，否则就需要写where条件语句和AND关键字，再连上EXISTS。

例6，查询新闻表中新闻类型id为5的新闻信息，要求是类型id必须在新闻类型表中真实存在，如果不存在，就不要查询出任何记录。

```
mysql> SELECT * FROM tb_news WHERE typeid=5 AND EXISTS (SELECT id
FROM tb_newstype WHERE id=5);
Empty set (0.00 sec)
```

例6中，因为新闻类型表中id为5的记录已经被删除了。所以查询结果为空。

某些情况下，子查询可以转化为表连接查询来实现。例6即是如此。转换为内连接后的SQL语句如下：

```
mysql> SELECT * FROM tb_news n , tb_newstype t WHERE n.typeid = t.id
AND n.typeid=5 ;
Empty set (0.00 sec)
```

【备注：】EXISTS关键字与前面几种子查询语句都不一样。使用EXISTS关键字时，内层查询语句只返回true和false。而前面几种子查询，其内层查询语句都会返回查询到的记录。

（6）.　带ANY或SOME关键字的子查询和带ALL关键字的子查询。

实际工作中，使用ANY 、SOME 、 ALL的情形比较少。所以仅仅作为了解，知道子查询中可以有这三个关键字即可。

7、记录联合查询：
某些情况下，需要将几个SELECT语句查询出来的结果合并起来显示。这个时候需要用表联合来实现。进行合并操作使用UNION和UNION ALL关键字来实现。

记录联合语法规则如下：
SELECT 语句1
UNION | UNION ALL
SELECT 语句2
UNION | UNION ALL
SELECT 语句3
UNION | UNION ALL ...
SELECT 语句n;

例如：从注册会员表中查出年龄最大和最小的所有会员的信息。

```
mysql> SELECT id , username , age , sex  FROM tb_user WHERE  age =
(SELECT MAX(age)  FROM  tb_user )
UNION
SELECT id , username , age , sex  FROM tb_user WHERE  age = (SELECT
MIN(age)  FROM  tb_user );
+----+----------+-----+-----+
| id | username | age | sex |
+----+----------+-----+-----+
|  1 | 王向军    |  32 |   1 |
| 28 |郑建勋      |  17 |   1 |
| 42 | Steven     |  17 |   1 |
```

```
+----+----------+-----+-----+
```
3 rows in set (0.00 sec)

如果不使用记录联合，那么一次性获得的要么是年龄最大的会员的信息，要么就是年龄最小的会员的信息。而使用UNION后，这些信息都可以同时显示出来。

【备注：】UNION和UNION ALL的区别：
UNION ALL是把结果集直接合并在一起，而UNION是将UNION ALL后的结果进行一次DISTINCT处理，消除其中所有的重复记录。