MySQL SELECT语句笔记

## MySQL中，SELECT的基本语法形式如下：

SELECT 属性列表

FROM 表名和视图列表

[WHERE 条件表达式]

[GROUP BY 属性名1][having 条件表达式2]

[ORDER BY 属性名2][ASC|DESC]

其中：

|  |  |
| --- | --- |
| 参数名 | 含义 |
| 属性列表 | 表示需要查询的字段名 |
| 表名和视图列表 | 表示从此处指定的表或者视图中查询数据 |
| 条件表达式1 | 表示指定查询条件 |
| 属性名1 | 指按该字段中的数据进行分组 |
| 条件表达式2 | 表示满足该表达式的数据才能输出 |
| 属性名2 | 指按该字段中的数据进行排序，排序方式有“ASC”，“DESC”两个参数指出 |
| ASC | 表示按照升序的顺序进行排序，这是默认参数，可不写 |
| DESC | 表示按照降序的顺序进行排序。 |

如果有WHERE语句，就按照“条件表达式1”指定的条件进行查询，如果没有WHERE字句，就查询所有记录。

如果有GROUP BY子句，就按照属性名1指定的字段进行分组。如果GROUP BY字句后面带着having关键字，那么只有满足“条件表达式2”中指定的条件才可以输出。GROUP BY字句同城和count（），sum（）等聚合函数一起使用。

如果有ORDER BY字句，就按照”属性名2”指定的字段进行排序。排序方式由“ASC”和“DESC”两个参数指出。默认的情况下是“ASC”。

准备环境

创建一个表，命令如下：

CREATE table employee( num INT NOT NULL PRIMARY KEY AUTO\_INCREMENT, d\_id INT NOT NULL, name VARCHAR(20), age INT, sex VARCHAR(4), homeaddr VARCHAR(50));

插入数据：

INSERT INTO employee values(null,1001,'zhangsan',26,'M','haidian');

INSERT INTO employee values(null,1002,'lisi',24,'F','changping');

INSERT INTO employee values(null,1003,'wangwu',25,'F','daxing');

INSERT INTO employee values(null,1004,'anic',15,'M','England');

INSERT INTO employee values(null,1005,'zhaoliu',30,'M','fangshan');

案列1：

下面是一个简单的SELECT语句来查询employee表。SELECT语句的代码如下：

SELECT num,name,age,sex,homeaddr FROM employee;

测试结果：

mysql> SELECT num,name,age,sex,homeaddr FROM employee;

+-----+----------+------+------+-----------+

| num | name | age | sex | homeaddr |

+-----+----------+------+------+-----------+

| 1 | zhangsan | 26 | M | haidian |

| 2 | lisi | 24 | F | changping |

| 3 | wangwu | 24 | F | daxing |

| 4 | anic | 15 | M | England |

| 5 | zhaoliu | 30 | M | fangshan |

+-----+----------+------+------+-----------+

5 rows in set (0.00 sec)

案列2：

下面是一个包含WHERE和ORDER BY子句的SELECT语句。SELECT语句的代码如下：

SELECT num,d\_id,name,age,sex,homeaddr FROM employee WHERE age<26 ORDER BY d\_id DESC;

测试结果：

mysql> SELECT num,d\_id,name,age,sex,homeaddr FROM employee WHERE age<26 ORDER BY d\_id DESC;

+-----+------+--------+------+------+-----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+--------+------+------+-----------+

| 4 | 1004 | anic | 15 | M | England |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 2 | 1002 | lisi | 24 | F | changping |

+-----+------+--------+------+------+-----------+

3 rows in set (0.00 sec)

案列3：

下面用SELECT语句查询employee表中的所有字段的数据。

SELECT num,d\_id,name,age,sex,homeaddr FROM employee;

测试结果：

mysql> SELECT num,d\_id,name,age,sex,homeaddr FROM employee;

+-----+------+----------+------+------+-----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+-----------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 4 | 1004 | anic | 15 | M | England |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

+-----+------+----------+------+------+-----------+

5 rows in set (0.00 sec)

案列4：

下面使用SELECT语句来查询employee表中的所有字段的数据，此处用“\*”来替代“属性列表”。SELECT语句的代码如下

SELECT \* FROM employee；

测试结果：

mysql> SELECT \* FROM employee;

+-----+------+----------+------+------+-----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+-----------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 4 | 1004 | anic | 15 | M | England |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

+-----+------+----------+------+------+-----------+

5 rows in set (0.00 sec)

注：“\*”代表表中的所有字段。

## 查询指定字段

查询数据时，可以再SELECT语句的“属性列表”中列出所要查询的字段。这种方式可以指定需要查询的字段，而不需要查询出所有的字段

案列5：（查询指定字段的记录）

下面查询employee表中的num,name,age和homeaddr这四个字段的数据。SELECT语句的代码 如下：

SELECT num,name,age,homeaddr FROM employee;

测试结果：

mysql> SELECT num,name,age,homeaddr FROM employee;

+-----+----------+------+-----------+

| num | name | age | homeaddr |

+-----+----------+------+-----------+

| 1 | zhangsan | 26 | haidian |

| 2 | lisi | 24 | changping |

| 3 | wangwu | 24 | daxing |

| 4 | anic | 15 | England |

| 5 | zhaoliu | 30 | fangshan |

+-----+----------+------+-----------+

5 rows in set (0.00 sec)

## 查询指定记录WHERE

SELECT语句中可以设置查询条件。用户可以根据自己的需要来设置查询条件，按条件进行查询。查询的结果必须满足查询条件。例如。用户需要查找d\_id为1001的记录，那么可以设置“d\_id=1001”为查询条件。这样查询结果中的记录就都会满足“d\_id=1001”这个条件。WHERE子句可以用来指定查询条件。其语法规则为：WHERE 条件表达式

案列6：（查询指定记录）

下面查询employee表中d\_id为1001的记录。SELECT语句的代码如下 ：

SELECT \*FROM employee WHERE d\_id=1001;

测试结果：

mysql> SELECT \* FROM employee WHERE d\_id=1001;

+-----+------+----------+------+------+----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

+-----+------+----------+------+------+----------+

1 row in set (0.00 sec)

## 带IN关键字的查询

IN关键字可以判断某个字段的值是否在指定的集合内中。如果字段的值在集合中，测满足查询条件，该记录将被查询出来。如何不在集合中，则不满足查询条件。其语法规则如下：[NOT] IN （元素1,元素2,……,元素n）

案列7：

下面使用IN关键字进行查询。SELECT语句的代码如下：

SELECT \* FROM employee WHERE d\_id in(1001,1004);

测试结果：

mysql> SELECT \* FROM employee WHERE d\_id in(1001,1004);

+-----+------+----------+------+------+----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 4 | 1004 | anic | 15 | M | England |

+-----+------+----------+------+------+----------+

2 rows in set (0.00 sec)

注：表示，查询的结果必须满足括号里的条件，即d\_id为1001或1004的记录。

案列8：

下面使用NOT IN关键字进行查询，而且集合的元素为字符型数据。SELECT语句的代码如下：

SELECT \* FROM employee WHERE name not in(‘zhangsan’,‘lisi’);

测试结果：

mysql> SELECT \* FROM employee WHERE name not in(“zhangsan”,”lisi”);

ERROR 1054 (42S22): Unknown column '“zhangsan”' in 'WHERE clause'

mysql> SELECT \* FROM employee WHERE name not in('zhangsan','lisi');

+-----+------+---------+------+------+----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+---------+------+------+----------+

| 3 | 1003 | wangwu | 24 | F | daxing |

| 4 | 1004 | anic | 15 | M | England |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

+-----+------+---------+------+------+----------+

3 rows in set (0.00 sec)

注：名字不为zhangsan或者lisi记录被查询出来了。

## 带between and的范围查询

Between and关键可以判读某个字段的值是否存在指定的范围内。如果字段的值在指定的范围内，则满足查询条件，该记录将被查询出来。如果不在指定的范围内，则不满足查询条件。其语法规则如下：[not] between 取值1 and 取值2;

案列9：

下面使用between and关键字进行查询，查询 条件是age字段的取值为15到25。SELECT语句的代码如下：

SELECT \* FROM employee WHERE age between 15 and 25;

测试结果：

mysql> SELECT \* FROM employee WHERE age between 15 and 25;

+-----+------+--------+------+------+-----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+--------+------+------+-----------+

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 4 | 1004 | anic | 15 | M | England |

+-----+------+--------+------+------+-----------+

3 rows in set (0.00 sec)

案列10：

下面使用not between and关键字查询employee表。查询条件是age字段的取值不在15到25之间。SELECT语句的代码如下：

SELECT \* FROM employee WHERE age not between 15 and 25;

测试结果：

mysql> SELECT \* FROM employee WHERE age not between 15 and 25;

+-----+------+----------+------+------+----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

+-----+------+----------+------+------+----------+

2 rows in set (0.00 sec)

## 带LIKE的字符匹配查询

LIKE关键字可以匹配字符串是否相等。如果字段的值与指定的字符串相匹配，则满足查询条件，该记录将被查询出来。如果与指定的字符串不匹配，则不满足查询条件。其语法规则如下：[NOT] LIKE ‘字符串’

案列11：

下面使用LIKE关键字来匹配一个完整的字符串‘Aric’。SELECT语句的代码如下：

SELECT \* FROM employee WHERE name like ‘Aric’;

测试结果：

mysql> SELECT \* FROM employee WHERE name like 'anic';

+-----+------+------+------+------+----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+------+------+------+----------+

| 4 | 1004 | anic | 15 | M | England |

+-----+------+------+------+------+----------+

1 row in set (0.00 sec)

案列12：

下面使用LIKE关键字来匹配带有通配符”%“的字符串”da“。SELECT语句的代码如下：

SELECT \* FROM employee WHERE homeaddr like 'da%';

测试结果：

mysql> SELECT \* FROM employee WHERE homeaddr like 'da%';

+-----+------+--------+------+------+----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+--------+------+------+----------+

| 3 | 1003 | wangwu | 24 | F | daxing |

+-----+------+--------+------+------+----------+

1 row in set (0.00 sec)

案列13：

下面使用like关键来匹配带有通配符”\_“的字符串”an\_c“。SELECT语句的代码如下：

SELECT \* FROM employee WHERE name like "an\_c";

测试结果：

mysql> SELECT \* FROM employee WHERE name like "an\_c";

+-----+------+------+------+------+----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+------+------+------+----------+

| 4 | 1004 | anic | 15 | M | England |

+-----+------+------+------+------+----------+

1 row in set (0.00 sec)

案列14：

下面使用not like关键字来查询不是姓张的所有人的记录。SELECT语句的代码如下：

SELECT \* FROM employee WHERE name not like "zhang%";

测试结果：

mysql> SELECT \* FROM employee WHERE name not like "zhang%";

+-----+------+---------+------+------+-----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+---------+------+------+-----------+

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 4 | 1004 | anic | 15 | M | England |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

+-----+------+---------+------+------+-----------+

4 rows in set (0.00 sec)

## 查询空值IS NULL

IS NULL关键字可以用来判断字段的值是否为空值（null）。如果字段的值是空值，则满足查询条件，该记录将被查询出来。如果字段的值不是空值，则不满足查询条件。其语法规则为：is [not] null。其中，”not“是可选参数，加上not表示字段不是空值时满足条件。

创建一个work表用来测试：

CREATE table work( id int, name varchar(20), sex varchar(4), info varchar(50));

插入数据：

INSERT INTO work values(1001,'hjh',null,null);

INSERT INTO work values(1002,'cch',null,null);

INSERT INTO work values(1003,'zk',null,'student');

查看结果：

mysql> SELECT \* FROM work;

+------+------+------+---------+

| id | name | sex | info |

+------+------+------+---------+

| 1001 | hjh | NULL | NULL |

| 1002 | cch | NULL | NULL |

| 1003 | zk | NULL | student |

+------+------+------+---------+

3 rows in set (0.00 sec)

案列15：

下面使用IS NULL关键字来查询work表中的inifo字段为空值的记录。SELECT语句的代码如下：

SELECT \* FROM work WHERE info IS NULL;

测试结果：

mysql> SELECT \* FROM work WHERE info IS NULL;

+------+------+------+------+

| id | name | sex | info |

+------+------+------+------+

| 1001 | hjh | NULL | NULL |

| 1002 | cch | NULL | NULL |

+------+------+------+------+

2 rows in set (0.00 sec)

案列16：

下面使用is not null关键字来查询work表中的inifo字段不为空值的记录。SELECT语句的代码如下：

SELECT \* FROM work WHERE info is not null;

测试结果：

mysql> SELECT \* FROM work WHERE info is not null;

+------+------+------+---------+

| id | name | sex | info |

+------+------+------+---------+

| 1003 | zk | NULL | student |

+------+------+------+---------+

1 row in set (0.00 sec)

## 带AND的多条件查询

AND关键字可以用来联合多个条件进行查询。使用AND关键字时，只有同时满足所有查询条件的记录会被查询出来。如果不满足这些查询条件的其中一个，这样的记录将被排除掉。AND关键字的语法规则为：条件表达式1 AND 条件表达式2 […… AND 条件表达式n]

其中，AND可以连接两个条件表达式。而且，可以同时使用多个AND关键字，这样可以连接更多的条件表达式。

案列17：

下面使用AND关键字来查询employee表中的d\_id为1001，而且sex为“男”的记录。SELECT语句的代码如下：

SELECT \* FROM employee WHERE d\_id=1001 and sex like 'M';

测试结果：

mysql> SELECT \* FROM employee WHERE d\_id=1001 and sex like 'M';

+-----+------+----------+------+------+----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

+-----+------+----------+------+------+----------+

1 row in set (0.00 sec)

案列18：

下面在employee表中查询d\_id小于1004，age小于27，而且sex为“男”的记录。SELECT语句的代码如下：

SELECT \* FROM employee WHERE d\_id<1004 and age<27 and sex='M';

测试结果：

mysql> SELECT \* FROM employee WHERE d\_id<1004 and age<27 and sex='M';

+-----+------+----------+------+------+----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

+-----+------+----------+------+------+----------+

1 row in set (0.00 sec)

案列19：

下面使用AND关键字查询employee表中的记录。查询条件为num取值在{1,2,3}这个集合中，age从15到25这个范围，而且，homeaddr的取值中包含“beijing”。SELECT语句的代码如下：

SELECT \* FROM employee WHERE num in (1,2,7）and age between 15 and 26 and homeaddr like ‘beijing%’;

测试结果：

mysql> SELECT \* FROM employee WHERE num in(1,2,7) and age between 15 and 26 and homeaddr like '%beijing%';

+-----+------+--------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+--------+------+------+----------------+

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

+-----+------+--------+------+------+----------------+

1 row in set (0.02 sec)

## 带OR的条件查询

OR关键字也可以用来联合多个条件进行查询，但是与AND关键字不同。使用OR关键字时，只要满足这几个查询条件的其中一个，这样的记录就会被查询出来。如果不满足这些查询条件中的任何一个，这样的记录将被排除掉。OR关键字的语法规则为：条件表达式1 OR 条件表达式2 […OR 条件表达式n]

其中，OR可以用来连接两个条件表达式。而且，可以同时使用多个OR关键字，这样可以连接更多的条件表达式。

案列20：

下面使用OR关键字来查询employee表中d\_id为1001，或者sex为“M”的记录。SELECT语句的代码如下：

SELECT \* FROM employee WHERE d\_id=1 or sex='M';

测试结果：

mysql> SELECT \* FROM employee WHERE d\_id=1 or sex='M';

+-----+------+----------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 4 | 1004 | anic | 15 | M | England |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

| 6 | 1006 | tianqi | 23 | M | NULL |

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

+-----+------+----------+------+------+----------------+

5 rows in set (0.00 sec)

案例21：

下面使用OR关键字来查询employee表中的记录，查询条件为num取值在（1,2,3）这个集合中，或者homeaddr的取值中包含“beijing”。SELECT语句的代码如下：

SELECT \* FROM employee WHERE num in (1,2,3) or age between 24 and 26 or homeaddr like '%beijing%';

测试结果：

mysql> SELECT \* FROM employee WHERE num in (1,2,3) or age between 24 and 26 or homeaddr like '%beijing%';

+-----+------+----------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

+-----+------+----------+------+------+----------------+

4 rows in set (0.00 sec)

案列22：

下面同时使用or和and关键字查询employee表中的记录。SELECT语句的代码如下：

SELECT \* FROM employee WHERE num in(1,3,4) and age=25 or sex='F';

测试结果：

mysql> SELECT \* FROM employee WHERE num in(1,3,4) and age=25 or sex='F';

+-----+------+--------+------+------+-----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+--------+------+------+-----------+

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

+-----+------+--------+------+------+-----------+

2 rows in set (0.00 sec)

## 查询结果不重复DISTINCT

如果表中的某些字段没有唯一性约束，这些字段可能存在着重复的值。例如，employee表中的d\_id字段就存在着重复的情况。

+------+

| d\_id |

+------+

| 1006 |

| 1007 |

| 1007 |

+------+

案列23：

下面使用DISTINCT关键字来查询d\_id字段中的重复记录。带DISTINCT关键字的SELECT语句如下：

SELECT DISTINCT d\_id FROM employee;

测试结果：

mysql> SELECT DISTINCT d\_id FROM employee;

+------+

| d\_id |

+------+

| 1001 |

| 1002 |

| 1003 |

| 1004 |

| 1005 |

| 1006 |

| 1007 |

+------+

7 rows in set (0.00 sec)

## 对查询结果排序ASC、DESC

从表中查询出来的数据可能是无序的，或者其排列顺序不是用户所期望的顺序。为了使查询结果的顺序满足用户的要求，可以使用ORDER BY关键字对记录进行排序。其语法规则为：ORDER BY 属性名 [ASC|DESC]

案列24：

下面查询employee表中所有记录，按照age字段进行排序。带ORDER BY关键字的SELECT语句如下：

SELECT \* FROM employee ORDER BY age;

测试结果：

mysql> SELECT \* FROM employee ORDER BY age;

+-----+------+----------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------------+

| 4 | 1004 | anic | 15 | M | England |

| 6 | 1006 | tianqi | 23 | M | NULL |

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 8 | 1007 | baidu | 25 | M | beingpinggu |

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

+-----+------+----------+------+------+----------------+

8 rows in set (0.00 sec)

案列25：

下面查询employee表中所有记录，按照age字段的升序方式进行排序。SELECT语句如下：

SELECT \* FROM employee ORDER BY age ASC;

测试结果：

mysql> SELECT \* FROM employee ORDER BY age;

+-----+------+----------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------------+

| 4 | 1004 | anic | 15 | M | England |

| 6 | 1006 | tianqi | 23 | M | NULL |

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 8 | 1007 | baidu | 25 | M | beingpinggu |

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

+-----+------+----------+------+------+----------------+

8 rows in set (0.00 sec)

注：在按升序排序的时候，省略ASC即可，因为它是默认参数。

案列26：

下面查询employee表中所有记录，按照age字段的降序方式进行排序。SELECT语句的代码如下:

SELECT \* FROM employee ORDER BY age DESC;

测试结果：

mysql> SELECT \* FROM employee ORDER BY age DESC;

+-----+------+----------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------------+

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 8 | 1007 | baidu | 25 | M | beingpinggu |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 6 | 1006 | tianqi | 23 | M | NULL |

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

| 4 | 1004 | anic | 15 | M | England |

+-----+------+----------+------+------+----------------+

8 rows in set (0.00 sec)

案列27：

下面查询employ表中的所有记录，按照d\_id字段的升序和age字段的降序方式进行排序。SELECT语句如下：

SELECT \* FROM employee ORDER BY d\_id ASC,age DESC;

测试结果：

mysql> SELECT \* FROM employee ORDER BY d\_id ASC,age DESC;

+-----+------+----------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 4 | 1004 | anic | 15 | M | England |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

| 6 | 1006 | tianqi | 23 | M | NULL |

| 8 | 1007 | baidu | 25 | M | beingpinggu |

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

+-----+------+----------+------+------+----------------+

8 rows in set (0.00 sec)

注：在有多个排序条件是，首先根据现在的条件进行排序，前面的记录有重复时，在根据后面的条件进行排序。

## 分组查询GROUP BY

GROUP BY 关键字可以将查询结果按某个字段或多个字段进行分组。字段中值相等的为一组。其语法规则如下：

GROUP BY 属性名[having 条件表达式][with rollup]

1. 单独使用GROUP BY关键字来分组
2. GROUP BY关键字与GROUP\_concat（）函数一起使用
3. GROUP BY关键字与集合函数一起使用
4. GROUP BY关键字与having一起使用
5. 按多个字段进行分组
6. GROUP BY关键与with rollup一起使用

案列28：

下面按employee表的sex字段进行分组查询，查询结果与分组前结果进行对比。先执行不带GROUP BY关键字的SELECT语句。语句执行如下：

SELECT \* FROM employee GROUP BY sex;

测试结果：

mysql> select \* from employee group by sex;

+-----+------+----------+------+------+-----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+-----------+

| 2 | 1002 | lisi | 24 | F | changping |

| 1 | 1001 | zhangsan | 26 | M | haidian |

+-----+------+----------+------+------+-----------+

2 rows in set (0.00 sec)

案列29：

下面按employee表的sex字段进行成分组查询，使用GROUP CONCAT()函数将每个分子的name字段的值显示出来、SELECT语句的代码如下：

SELECT sex,GROUP\_CONCAT(name) FROM employee GROUP BY sex;

测试结果:

mysql> SELECT sex,GROUP\_CONCAT(name) FROM employee GROUP BY sex;

+------+-------------------------------------------+

| sex | GROUP\_CONCAT(name) |

+------+-------------------------------------------+

| F | lisi,wangwu |

| M | zhangsan,anic,zhaoliu,tianqi,tianqi,baidu |

+------+-------------------------------------------+

2 rows in set (0.03 sec)

案列30：

下面按employee表的sex字段进行分组查询。sex字段取值相同的为一组。然后对每一组使用集合函数COUNT()进行计算，求出每一组的记录数。SELECT语句的代码如下：

SELECT sex,COUNT(sex) FROM employee GROUP BY sex ;

测试结果：

mysql> SELECT sex,COUNT(sex) FROM employee GROUP BY sex ;

+------+------------+

| sex | COUNT(sex) |

+------+------------+

| F | 2 |

| M | 6 |

+------+------------+

2 rows in set (0.02 sec)

案列31：

下面按employee表的sex字段进行分组查询。然后显示记录数大于等于3的分组。SELECT语句的代码如下：

SELECT sex,COUNT(sex) FROM employee GROUP BY sex HAVING COUNT(sex)>=3;

测试结果：

mysql> SELECT sex,COUNT(sex) FROM employee GROUP BY sex HAVING COUNT(sex)>=3;

+------+------------+

| sex | COUNT(sex) |

+------+------------+

| M | 6 |

+------+------------+

1 row in set (0.00 sec)

案列32

下面employee表按照d\_id字段和sex字段进行查询。SELECT语句如下：

SELECT \* FROM employee GROUP BY d\_id,sex;

测试结果：

mysql> SELECT \* FROM employee GROUP BY d\_id,sex;

+-----+------+----------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 4 | 1004 | anic | 15 | M | England |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

| 6 | 1006 | tianqi | 23 | M | NULL |

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

+-----+------+----------+------+------+----------------+

7 rows in set (0.00 sec)

案列33：

下面按employee表的sex字段进行分组查询。使用COUNT()函数来计算魅族的记录数。并且加上WITH ROLL UP。SELECT语句如下：

SELECT sex,COUNT(sex) FROM employee GROUP BY sex WITH ROLLUP;

测试结果

mysql> SELECT sex,COUNT(sex) FROM employee GROUP BY sex WITH ROLLUP;

+------+------------+

| sex | COUNT(sex) |

+------+------------+

| F | 2 |

| M | 6 |

| NULL | 8 |

+------+------------+

3 rows in set (0.00 sec)

注：使用ROLLUP关键字后，会对分组的结果进行求和。

案列34：

下面按employee表的sex字段进行查询。使用GROUP\_CONCAT()函数查询每组的name字段的值。并且加上WITH ROLLUP。SELECT语句如下：

SELECT sex,GROUP\_CONCAT(name) FROM employee GROUP BY sex WITH ROLLUP;

测试结果：

mysql> SELECT sex,GROUP\_CONCAT(name) FROM employee GROUP BY sex WITH ROLLUP;

+------+-------------------------------------------------------+

| sex | GROUP\_CONCAT(name) |

+------+-------------------------------------------------------+

| F | lisi,wangwu |

| M | zhangsan,anic,zhaoliu,tianqi,tianqi,baidu |

| NULL | lisi,wangwu,zhangsan,anic,zhaoliu,tianqi,tianqi,baidu |

+------+-------------------------------------------------------+

3 rows in set (0.00 sec)

## 用LIMIT闲置查询结果的数量

查询数据时，可能会查询出很多的记录。而用户需要的记录可能只是很少的一部分。这样就需要来限制查询结果的数量。LIMIT是MySQL中的一个特殊关键字。其可以用来指定查询结果从哪条记录开始显示。还可以指定一共显示多少条记录。LIMIT关键字有两种使用的方式。这两种方式分别是不指定初始位置和指定初始位置。

1. 不指定初始位置
2. 指定初始位置

案列35：

下面查询employee表的所有记录。但只显示前两条。SELECT语句如下：

SELECT \* FROM employee LIMIT 2;

测试结果：

mysql> SELECT \* FROM employee LIMIT 2;

+-----+------+----------+------+------+-----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+-----------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 2 | 1002 | lisi | 24 | F | changping |

+-----+------+----------+------+------+-----------+

2 rows in set (0.00 sec)

案列36：

下面查询employee表的所有记录，但只显示前十条。SELECT语句如下：

mysql> SELECT \* FROM employee LIMIT 10;

测试结果：

mysql> SELECT \* FROM employee LIMIT 10;

+-----+------+----------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 4 | 1004 | anic | 15 | M | England |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

| 6 | 1006 | tianqi | 23 | M | NULL |

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

| 8 | 1007 | baidu | 25 | M | beingpinggu |

+-----+------+----------+------+------+----------------+

8 rows in set (0.00 sec)

注：如果指定查询的记录条数大于表的最大记录数，数据库会显示表的全部内容

案列37：

下面查询employee表的所有记录，显示前两条记录。SELECT语句如下：

SELECT \* FROM employee LIMIT 0,2;

测试结果：

mysql> SELECT \* FROM employee LIMIT 0,2;

+-----+------+----------+------+------+-----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+-----------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 2 | 1002 | lisi | 24 | F | changping |

+-----+------+----------+------+------+-----------+

2 rows in set (0.00 sec)

注：0表示该表的第一条记录。是默认值。

案列38：

下面使用employee表的所有记录，从第二条记录开始显示，共显示两条记录。SELECT语句如下：

SELECT \* FROM employee LIMIT 1,2;

测试结果：

mysql> SELECT \* FROM employee LIMIT 1,2;

+-----+------+--------+------+------+-----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+--------+------+------+-----------+

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

+-----+------+--------+------+------+-----------+

2 rows in set (0.00 sec)

## 使用集合函数查询COUNT()、SUM()、AVG()、MAX()、MIN()

集合函数包括COUNT()、SUM()、AVG()、MAX()和MIN()。其中，COUNT()用来统计记录的条数；SUM()用来计算字段的值的总和；AVG()用来计算字段的值的平均值；MAX()用来查询字段的最大值；MIN()用来查询字段的最小值。当需要对表中的记录求和、求平均值、查询最大值、查询最小值等操作时，可以使用集合函数。例如，需要计算学生成绩表中的平均成绩，可以使用AVG()函数。GROUP BY关键字通常需要与集合函数一起使用。本节中将详细讲解各种集合函数。

### COUNT()函数使用

案列39：

下面使用COUNT()函数 统计employee表的记录数。SELECT语句如下：

SELECT COUNT(\*) FROM employee;

测试结果：

mysql> SELECT COUNT(\*) FROM employee;

+----------+

| COUNT(\*) |

+----------+

| 8 |

+----------+

1 row in set (0.00 sec)

案列40：

下面使用COUNT()函数统计employee表中不同d\_id值的记录数。COUNT()函数与GROUP BY关键字一起使用。SELECT语句如下：

SELECT d\_id,COUNT(\*) FROM employee GROUP BY d\_id;

测试结果:

mysql> SELECT d\_id,COUNT(\*) FROM employee GROUP BY d\_id;

+------+----------+

| d\_id | COUNT(\*) |

+------+----------+

| 1001 | 1 |

| 1002 | 1 |

| 1003 | 1 |

| 1004 | 1 |

| 1005 | 1 |

| 1006 | 1 |

| 1007 | 2 |

+------+----------+

7 rows in set (0.00 sec)

### SUM()函数使用

SUM()函数式求和函数。使用SUM()函数可以求出表中某个字段取值的总和。例如，可以使用SUM()函数来求学生的总成绩。

=================================================================================

创建一个测试表。语句如下：

CREATE TABLE grade（num INT(10) NOT NULL,course VARCHAR(10) NOT NULL,score FLOAT）;

insert into grade values(1001,'shuxue',80);

insert into grade values(1001,'yuwen',90);

insert into grade values(1001,'yingyu',85);

insert into grade values(1001,'jisuanji',95);

insert into grade values(1002,'shuxue',88);

insert into grade values(1002,'yuwen',90);

insert into grade values(1002,'yingyu',89);

insert into grade values(1002,'jisuanji',90);

insert into grade values(1003,'shuxue',80);

insert into grade values(1003,'yuwen',98);

insert into grade values(1003,'yingyu',85);

insert into grade values(1003,'jisuanji',85);

建表后表内容：

mysql> select \* from grade;

+------+----------+-------+

| num | course | score |

+------+----------+-------+

| 1001 | yuwen | 90 |

| 1001 | yingyu | 85 |

| 1001 | jisuanji | 95 |

| 1001 | shuxue | 80 |

| 1002 | shuxue | 88 |

| 1002 | yuwen | 90 |

| 1002 | yingyu | 89 |

| 1002 | jisuanji | 90 |

| 1003 | shuxue | 80 |

| 1003 | yuwen | 98 |

| 1003 | yingyu | 85 |

| 1003 | jisuanji | 95 |

+------+----------+-------+

12 rows in set (0.00 sec)

案列41：

下面使用SUM()函数统计grade表中学号为1001的同学的总成绩。SELECT语句如下：

SELECT num,SUM(score) FROM grade WHERE num=1001;

测试结果：

mysql> SELECT num,SUM(score) FROM grade WHERE num=1001;

+------+------------+

| num | SUM(score) |

+------+------------+

| 1001 | 350 |

+------+------------+

1 row in set (0.00 sec)

案列42：

下面将grade表按照num字段进行分组。然后，使用SUM()函数统计各分组的总成绩。SELECT语句如下：

SELECT num,SUM(score) FROM grade GROUP BY num;

测试结果：

mysql> SELECT num,SUM(score) FROM grade GROUP BY num;

+------+------------+

| num | SUM(score) |

+------+------------+

| 1001 | 350 |

| 1002 | 357 |

| 1003 | 358 |

+------+------------+

3 rows in set (0.00 sec)

### AVG()函数

AVG()函数式求平均值的函数。使用AVG()函数可以求出表中某个字段取值的平均值。例如，可以用AVG()函数来求平均年龄，也可以使用AVG()函数来求学生的平均成绩。

案列43：

下面使用AVG()函数计算employee表中平均年龄（age）。SELECT语句如下：

SELECT AVG(age) FROM employee;

测试结果：

mysql> SELECT AVG(age) FROM employee;

+----------+

| AVG(age) |

+----------+

| 23.7500 |

+----------+

1 row in set (0.02 sec)

案列44：

下面使用AVG()函数计算grade表中不同科目的平均成绩。SELECT语句如下：

SELECT course,AVG(score) FROM grade GROUP BY course;

测试结果：

mysql> SELECT course,AVG(score) FROM grade GROUP BY course;

+----------+-------------------+

| course | AVG(score) |

+----------+-------------------+

| jisuanji | 93.33333333333333 |

| shuxue | 82.66666666666667 |

| yingyu | 86.33333333333333 |

| yuwen | 92.66666666666667 |

+----------+-------------------+

4 rows in set (0.00 sec)

### MAX()函数

MAX()函数式求最大值的函数。使用MAX()函数可以求出表中某个字段取值的最大值。开入，可以用MAX()函数来查询最大年龄，也可以使用MAX()函数来求各科的最高成绩

案列45：

下面使用MAX()函数查询employee表中的最大年龄（age）。SELECT语句如下：

SELECT MAX(age) FROM employee;

测试结果

mysql> SELECT MAX(age) FROM employee;

+----------+

| MAX(age) |

+----------+

| 30 |

+----------+

1 row in set (0.00 sec)

案列46：

下面使用MAX()函数来查询grade表中不同科目的最高成绩。SELECT语句如下：

SELECT num,MAX(course) FROM grade GROUP BY course;

测试结果：

mysql> SELECT num,MAX(course) FROM grade GROUP BY course;

+------+-------------+

| num | MAX(course) |

+------+-------------+

| 1001 | jisuanji |

| 1001 | shuxue |

| 1001 | yingyu |

| 1001 | yuwen |

+------+-------------+

4 rows in set (0.00 sec)

案列47：

下面使用MAX()函数查询work表中name字段的最大值。SELECT语句如下：

SELECT MAX(name) FROM work;

测试结果：

mysql> SELECT MAX(name) FROM work;

+-----------+

| MAX(name) |

+-----------+

| zk |

+-----------+

1 row in set (0.00 sec)

注：字符串大小是根据字母顺序进行排列，z最大。

### MIN()函数

MIN()函数式求最小值的函数。使用MIN()函数可以求出表中某个字段取值的最小值。例如，可以用MIN()函数来查询最小年龄，也可以使用MIN()函数来求各科的最低成绩。

案列48：

下面使用MIN()函数查询employee表中的最小年龄。SELECT语句如下：

SELECT MIN(age) FROM employee;

测试结果：

mysql> SELECT MIN(age) FROM employee;

+----------+

| MIN(age) |

+----------+

| 15 |

+----------+

1 row in set (0.00 sec)

案列49：

下面使用MIN()函数查询grade表中不同科目的最低成绩。SELECT语句如下：

SELECT course,MIN(score) FROM grade GROUP BY course;

测试结果：

mysql> SELECT course,MIN(score) FROM grade GROUP BY course;

+----------+------------+

| course | MIN(score) |

+----------+------------+

| jisuanji | 90 |

| shuxue | 80 |

| yingyu | 85 |

| yuwen | 90 |

+----------+------------+

4 rows in set (0.00 sec)

## 连接查询

连接查询时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使用的。当不同的表中存在表示相同意义的字段时，可以通过该字段来连接着几个表。例如学生表中有crouse\_id字段来表示所学课程的课程号，课程表中有num字段来表示课程号。那么，可以通过学生表中的course\_id字段与课程中的num字段来进行连接查询。连接查询包括内连接查询和外连接查询。本小节将详细讲解内连接查询和外连接查询。同时，还会讲解多个条件结合在一起进行复核连接查询。

### 内连接查询

内连接查询时一种最常用的连接查询。内连接查询可以查询两个或两个以上的表。为了更好的理解，暂时只讲解两个表的连接查询。当两个表中存在相同意义的字段时，可以通过该字段俩连接这两个表。当该字段的值相等时，就查询出该记录。

创建一个表用于测试，代码如下：

CREATE table department(d\_id INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,d\_name VARCHAR(10),function VARCHAR(10) ,address VARCHAR(10));

INSERT INTO department values(1001,'keyanbu','yanfachanpin','3haolou');

INSERT INTO department values(1002,'shengchanbu','shengchanchanpin','5haolou');

INSERT INTO department values(1003,'xiaoshoubu','cehuaxiaoshou','1haolou');

查询结果：

mysql> select \* from department;

+------+------------+------------+---------+

| d\_id | d\_name | function | address |

+------+------------+------------+---------+

| 1001 | keyanbu | yanfachanp | 3haolou |

| 1002 | shengchanb | shengchanc | 5haolou |

| 1003 | xiaoshoubu | cehuaxiaos | 1haolou |

+------+------------+------------+---------+

3 rows in set (0.00 sec)

案列50：

下面使用内连接查询的方式查询employee表和department。在执行内连接查询之前，先分别查看employee表和department表中的记录，以便进行比较。查询结果如下：

employee表内容

mysql> SELECT \* FROM employee;

+-----+------+----------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

| 4 | 1004 | anic | 15 | M | England |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

| 6 | 1006 | tianqi | 23 | M | NULL |

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

| 8 | 1007 | baidu | 25 | M | beingpinggu |

+-----+------+----------+------+------+----------------+

8 rows in set (0.00 sec)

department表内容

mysql> SELECT \* FROM department;

+------+------------+------------+---------+

| d\_id | d\_name | function | address |

+------+------------+------------+---------+

| 1001 | keyanbu | yanfachanp | 3haolou |

| 1002 | shengchanb | shengchanc | 5haolou |

| 1003 | xiaoshoubu | cehuaxiaos | 1haolou |

+------+------------+------------+---------+

3 rows in set (0.00 sec)

SELECT语句为：

SELECT num,name,employee.d\_id,age,sex,d\_name,function FROM employee,department WHERE employee.d\_id=department.d\_id;

测试结果：

mysql> SELECT num,name,employee.d\_id,age,sex,d\_name,function FROM employee,department WHERE employee.d\_id=department.d\_id;

+-----+----------+------+------+------+------------+------------+

| num | name | d\_id | age | sex | d\_name | function |

+-----+----------+------+------+------+------------+------------+

| 1 | zhangsan | 1001 | 26 | M | keyanbu | yanfachanp |

| 2 | lisi | 1002 | 24 | F | shengchanb | shengchanc |

| 3 | wangwu | 1003 | 24 | F | xiaoshoubu | cehuaxiaos |

+-----+----------+------+------+------+------------+------------+

3 rows in set (0.00 sec)

### 外连接查询

外连接查询可以查询两个或两个以上的表。外连接查询也需要通过指定字段来进行连接。当该字段取值相等时，可以查询出该记录。而且，该字段取值不相等的记录也可以查询出来。外连接查询包括做连接查询和右连接查询。其基本语法如下：

SELECT属性名列表：

RFOM 表名1 LEFT|RIGHT JOIN 表名2

ON 表名1.属性名1=表名2.属性名2；

1. 左链接查询
2. 右连接查询

案列51：

下面使用左连接查询的方式查询employee表和department表。两表通过d\_id字段进行连接。做链接的SELECT语句如下：

SELECT num,name,employee.d\_id,age,sex,d\_name,function FROM employee LEFT JOIN department ON employee.d\_id=department.d\_id;

测试结果：

mysql> SELECT num,name,employee.d\_id,age,sex,d\_name,function FROM employee LEFT JOIN department ON employee.d\_id=department.d\_id;

+-----+----------+------+------+------+------------+------------+

| num | name | d\_id | age | sex | d\_name | function |

+-----+----------+------+------+------+------------+------------+

| 1 | zhangsan | 1001 | 26 | M | keyanbu | yanfachanp |

| 2 | lisi | 1002 | 24 | F | shengchanb | shengchanc |

| 3 | wangwu | 1003 | 24 | F | xiaoshoubu | cehuaxiaos |

| 4 | anic | 1004 | 15 | M | NULL | NULL |

| 5 | zhaoliu | 1005 | 30 | M | NULL | NULL |

| 6 | tianqi | 1006 | 23 | M | NULL | NULL |

| 7 | tianqi | 1007 | 23 | M | NULL | NULL |

| 8 | baidu | 1007 | 25 | M | NULL | NULL |

+-----+----------+------+------+------+------------+------------+

8 rows in set (0.02 sec)

案列52：

下面使用右连接查询的方法查询employee表和department表。两表通过d\_id字段进行连接。右连接的SELECT语句如下：

SELECT num,name,age,sex,department.d\_id,d\_name,function FROM employee RIGHT JOIN department ON employee.d\_id=department.d\_id;

测试结果：

mysql> SELECT num,name,age,sex,department.d\_id,d\_name,function FROM employee RIGHT JOIN department ON employee.d\_id=department.d\_id;

+------+----------+------+------+------+------------+------------+

| num | name | age | sex | d\_id | d\_name | function |

+------+----------+------+------+------+------------+------------+

| 1 | zhangsan | 26 | M | 1001 | keyanbu | yanfachanp |

| 2 | lisi | 24 | F | 1002 | shengchanb | shengchanc |

| 3 | wangwu | 24 | F | 1003 | xiaoshoubu | cehuaxiaos |

+------+----------+------+------+------+------------+------------+

3 rows in set (0.00 sec)

### 复合条件连接查询

在连接查询时，也可以增加其他的闲置条件。通过多个条件的复合查询，可以使查询结果更加准确。例如，employee表和department表进行连接查询是，可以闲置age字段的取值必须大于24.这样，可以更加准确的查询出年龄大于24岁的员工的信息。

案列53：

下面使用内连接查询的方式查询employee表和department表。并且employee表总的age字段的值必须大于24。右连接的SELECT语句如下：

SELECT num,name,employee.d\_id,age,sex,d\_name,function FROM employee,department WHERE employee.d\_id=department.d\_id AND age>24;

测试结果：

mysql> SELECT num,name,employee.d\_id,age,sex,d\_name,function FROM employee,department WHERE employee.d\_id=department.d\_id AND age>24;

+-----+----------+------+------+------+---------+------------+

| num | name | d\_id | age | sex | d\_name | function |

+-----+----------+------+------+------+---------+------------+

| 1 | zhangsan | 1001 | 26 | M | keyanbu | yanfachanp |

+-----+----------+------+------+------+---------+------------+

1 row in set (0.04 sec)

案列54：

下面使用内连接查询的方式查询employee表和department表。并且以age字段的升序显示查询结果。SELECT语句如下：

SELECT num,name,employee.d\_id,age,sex,d\_name,function FROM employee,department WHERE employee.d\_id=department.d\_id ORDER BY age ASC;

测试结果：

mysql> SELECT num,name,employee.d\_id,age,sex,d\_name,function FROM employee,department WHERE employee.d\_id=department.d\_id ORDER BY age ASC;

+-----+----------+------+------+------+------------+------------+

| num | name | d\_id | age | sex | d\_name | function |

+-----+----------+------+------+------+------------+------------+

| 2 | lisi | 1002 | 24 | F | shengchanb | shengchanc |

| 3 | wangwu | 1003 | 24 | F | xiaoshoubu | cehuaxiaos |

| 1 | zhangsan | 1001 | 26 | M | keyanbu | yanfachanp |

+-----+----------+------+------+------+------------+------------+

3 rows in set (0.00 sec)

## 子查询

子查询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，可以为外层查询语句提供查询条件。因为在特定情况下，一个查询语句的条件需要另一个查询语句来获取。例如，现在需要从学生成绩表中查询计算机系学生的各科成绩。那么就必须知道哪些课程是计算机系学生选修的。因为，必须先查询计算机系学生选修的课程，然后根据这些课程来查询计算机的各科成绩。通过子查询，可以实现多表指尖的查询。子查询可能包括IN、NOT IN、ANY、ALL、EXISTS、NOT EXISTS等关键字。子查询中还可能包含比较运算符，如“=”、“!=”、“>”和“<”等。本小节将详细讲解子查询的只是。

### 带IN关键字的子查询

一个查询语句中的条件可能落在另一个SELECT语句的查询结果中。这可以通过IN关键字来判断。例如，要查询哪些同学选择了计算机系开设的课程。然后在从学生表中进行查询。如果学生选修的课程在前面查询出来的课程中，则查询出该同学的信息。这可以用带IN关键字的子查询来实现。

案列55：

下面查询employee表中的记录。这些记录的d\_id字段值必须在department表中出现过。SELECT语句如下：

SELECT \* FROM employee WHERE d\_id IN (SELECT d\_id FROM department);

测试结果：

mysql> SELECT \* FROM employee WHERE d\_id IN (SELECT d\_id FROM department);

+-----+------+----------+------+------+-----------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+-----------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 24 | F | daxing |

+-----+------+----------+------+------+-----------+

3 rows in set (0.03 sec)

案列56：

下面查询employee表中的记录。这些记录的d\_id字段的值必须没有在department表中出现过。SELECT语句如下：

SELECT \* FROM employee WHERE d\_id NOT IN (SELECT d\_id FROM department);

测试结果：

mysql> SELECT \* FROM employee WHERE d\_id NOT IN (SELECT d\_id FROM department);

+-----+------+---------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+---------+------+------+----------------+

| 4 | 1004 | anic | 15 | M | England |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

| 6 | 1006 | tianqi | 23 | M | NULL |

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

| 8 | 1007 | baidu | 25 | M | beingpinggu |

+-----+------+---------+------+------+----------------+

5 rows in set (0.00 sec)

首先创建两个表用于下面案列的试验。

表一：

CREATE TABLE computer\_stu(id INT PRIMARY KEY,name VARCHAR(20),score FLOAT);

INSERT INTO computer\_stu VALUES(1001,'Lily',85);

INSERT INTO computer\_stu VALUES(1002,'Tom',91);

INSERT INTO computer\_stu VALUES(1003,Jim',87);

INSERT INTO computer\_stu VALUES(1004,'Aric',77);

INSERT INTO computer\_stu VALUES(1005,'Lucy',65);

INSERT INTO computer\_stu VALUES(1006,'Andy',99);

INSERT INTO computer\_stu VALUES(1008,'Jeck',70);

表二：

CREATE TABLE scholarship(level INT PRIMARY KEY,score INT);

INSERT INTO scholarship VALUES(1,90);

INSERT INTO scholarship VALUES(2,80);

INSERT INTO scholarship VALUES(3,70);

### 带比较运算符的子查询

子查询可以使用比较运算符。这些比较运算符包括=、!>、=<、<=、<>等。其中，<>与!=是等价的。比较运算符在子查询时使用的非常广泛。如查询分数、年龄、价格、收入等。

案列57：

下面从computer\_stu表中查询获得一等奖学金的学生的学号、姓名和分数。各个等级的奖学金最低分存储在scholarship表中。SELECT语句如下：

SELECT id,name,score FROM computer\_stu WHERE score>=(SELECT score FROM scholarship WHERE level=1);

测试结果：

mysql> SELECT id,name,score FROM computer\_stu WHERE score>=(SELECT score FROM scholarship WHERE level=1);

+------+------+-------+

| id | name | score |

+------+------+-------+

| 1002 | Tom | 91 |

| 1006 | Andy | 99 |

+------+------+-------+

2 rows in set (0.03 sec)

案列58：

下面在department表中查询哪些部门没有年龄为24岁的员工。员工的年龄存储在employee表中。先查询一下employee表和department表，以便进行对比。查询结果如下：

SELECT d\_id,d\_name FROM department WHERE d\_id!=(SELECT d\_id FROM employee WHERE age=24);

测试结果：

mysql> SELECT d\_id,d\_name FROM department WHERE d\_id!=(SELECT d\_id FROM employee WHERE age=24);

+------+------------+

| d\_id | d\_name |

+------+------------+

| 1001 | keyanbu |

| 1003 | xiaoshoubu |

+------+------------+

2 rows in set (0.00 sec)

案列59：

下面用<>替代!=来完成一个例子。SELECT语句如下：

SELECT d\_id,d\_name FROM department WHERE d\_id<>(SELECT d\_id FROM employee WHERE age=24);

测试结果：

mysql> SELECT d\_id,d\_name FROM department WHERE d\_id<>(SELECT d\_id FROM employee WHERE age=24);

+------+------------+

| d\_id | d\_name |

+------+------------+

| 1001 | keyanbu |

| 1003 | xiaoshoubu |

+------+------------+

2 rows in set (0.00 sec)

### 带EXISTS关键字的子查询

EXISTS关键字表示存在。使用EXISTS关键字时，内层查询语句不返回查询的记录。而是返回一个真假值。如果内层查询语句查询到满足条件的记录，就返回一个真值（True）。否则，将返回一个假值（False）。当返回的值是真值时，外层查询语句将进行查询。当返回值是假值时，外层查询语句不进行查询或者查询不出任何记录。

案列60：

下面如果department表中存在d\_id取值为1003的记录，则查询employee表的记录。SELECT语句如下：

SELECT \* FROM employee WHERE EXISTS (SELECT d\_name FROM department WHERE d\_id=1003);

测试结果：

mysql> SELECT \* FROM employee WHERE EXISTS (SELECT d\_name FROM department WHERE d\_id=1003);

+-----+------+----------+------+------+----------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+----------------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 2 | 1002 | lisi | 24 | F | changping |

| 3 | 1003 | wangwu | 31 | F | daxing |

| 4 | 1004 | anic | 15 | M | England |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

| 6 | 1006 | tianqi | 23 | M | NULL |

| 7 | 1007 | tianqi | 23 | M | beijingxicheng |

| 8 | 1007 | baidu | 25 | M | beingpinggu |

+-----+------+----------+------+------+----------------+

8 rows in set (0.03 sec)

案列61：

下面如果department表中存在d\_id取值为1004的记录，则查询employee表中age大于24的记录。SELECT语句如下：

SELECT \* FROM employee WHERE EXISTS (SELECT d\_name FROM department WHERE d\_id=1004);

测试结果：

mysql> SELECT \* FROM employee WHERE EXISTS (SELECT d\_name FROM department WHERE d\_id=1004);

Empty set (0.00 sec)

案列62：

下面如果department表中存在d\_id取值为1003的记录，则查询employee表中age大于24的记录。SELECT语句如下：

SELECT \* FROM employee WHERE age>24 AND EXISTS (SELECT d\_name FROM department WHERE d\_id=1003);

测试结果：

mysql> SELECT \* FROM employee WHERE age>24 AND EXISTS (SELECT d\_name FROM department WHERE d\_id=1003);

+-----+------+----------+------+------+-------------+

| num | d\_id | name | age | sex | homeaddr |

+-----+------+----------+------+------+-------------+

| 1 | 1001 | zhangsan | 26 | M | haidian |

| 3 | 1003 | wangwu | 31 | F | daxing |

| 5 | 1005 | zhaoliu | 30 | M | fangshan |

| 8 | 1007 | baidu | 25 | M | beingpinggu |

+-----+------+----------+------+------+-------------+

4 rows in set (0.00 sec)

案列63：

下面如果department表中不存在d\_id取值为1003的记录，则查询employee表的记录。SELECT语句如下：

SELECT \* FROM employee WHERE NOT EXISTS (SELECT d\_name FROM department WHERE d\_id=1003);

测试结果：

mysql> SELECT \* FROM employee WHERE NOT EXISTS (SELECT d\_name FROM department WHERE d\_id=1003);

Empty set (0.00 sec)

### 带ANY关键字的子查询

ANY关键字表示满足其中任一条件。使用ANY关键字时，只要满足该条件来执行外层查询语句。例如，需要查询哪些同学能够获得奖学金。那么，首先必须从奖学金表中查询出各种奖金要求的最低分。只要一个同学的成绩高于不同奖学金最低分的任何一个，这个同学就可以获得奖学金。ANY关键字通常与比较运算符一起使用。

案列64：

下面从computer\_stu表中查询出那些同学可以获得奖学金。奖学金的信息存储在scholarship表中。先查看一下computer\_stu表和scholarship表。SELECT语句如下：

SELECT \* FROM computer\_stu WHERE score>=ANY(SELECT score FROM scholarship);

测试结果：

mysql> SELECT \* FROM computer\_stu WHERE score>=ANY(SELECT score FROM scholarship);

+------+------+-------+

| id | name | score |

+------+------+-------+

| 1001 | Lily | 85 |

| 1002 | Tom | 91 |

| 1003 | Jim | 87 |

| 1004 | Aric | 77 |

| 1006 | Andy | 99 |

| 1007 | Ada | 85 |

| 1008 | Jeck | 70 |

+------+------+-------+

7 rows in set (0.00 sec)

### 带ALL关键字的子查询

ALL关键字表示满足所有条件。使用ALL关键字时，只有满足内查询语句返回的所有结果，才可以执行外层查询语句。例如，需要查询哪些同学能够获得一等奖学金。首先必须从奖学金表中查询出各种奖学金要求的最低分。因为一等奖学金要求的分数最高。只有当同学的成绩高于所有奖学金最低分时，这个同学才可以获得一奖学金。ALL关键字也经常与比较运算符一起使用。

案列65：

下面从computer\_stu表中查询出哪些同学可以获得一等奖学金。奖学金的信息存储在scholarship表中。先需要从scholarship表中查询出各种奖学金的最低分。然后，从computer\_stu表中查询那些人的分数高于所有奖学金的最低分。SELECT语句代码如下：

SELECT \* FROM computer\_stu WHERE score>=ALL(SELECT score FROM scholarship);

测试结果：

mysql> SELECT \* FROM computer\_stu WHERE score>=ALL(SELECT score FROM scholarship);

+------+------+-------+

| id | name | score |

+------+------+-------+

| 1002 | Tom | 91 |

| 1006 | Andy | 99 |

+------+------+-------+

2 rows in set (0.00 sec)

### 合并查询结果

合并查询结果是将多个SELECT语句的查询结果合并到一起。因为某种情况下，需要将几个SELECT语句查询出来的结果合并起来显示。例如，现在需要查询公司甲和乙这两个公司的所有员工的信息。这就需要从公司甲中查询出所有的员工信息，然后再从公司乙中查询出所有员工的信息。然后将两次的查询结果合并到一起，进行合并操作使用UNION和UNION ALL关键字。以下将详细介绍讲解合并查询结果的方法。

使用UNION关键字时，数据库系统会将所有的查询结果合并到一起，然后去除相同的记录。而UNION ALL关键字则只是简单的合并到一起。其语法规则如下：

SELECT语句1

UNION|UNION ALL

SELECT语句2:

UNION|UNION ALL….

SELECT语句n…

案列66：

下面从department表和employee表中查询d\_id字段的取值。然后通过UNION关键字将结果合并到一起。首先，先查看department表和employee表中d\_id字段的取值。查询结果如下：

SELECT d\_id FROM department UNION SELECT d\_id FROM employee;

测试结果：

mysql> SELECT d\_id FROM department UNION SELECT d\_id FROM employee;

+------+

| d\_id |

+------+

| 1001 |

| 1002 |

| 1003 |

| 1004 |

| 1005 |

| 1006 |

| 1007 |

+------+

7 rows in set (0.02 sec)

案列67：

下面从department表和employee表中查询d\_id字段的取值。然后通过UNION ALL关键字将结果合并到一起。首先，先查看department表和employee表中d\_id字段的取值。查询结果如下：

SELECT d\_id FROM department UNION ALL SELECT d\_id FROM employee;

测试结果：

mysql> SELECT d\_id FROM department UNION ALL SELECT d\_id FROM employee;

+------+

| d\_id |

+------+

| 1001 |

| 1002 |

| 1003 |

| 1001 |

| 1002 |

| 1003 |

| 1004 |

| 1005 |

| 1006 |

| 1007 |

| 1007 |

+------+

11 rows in set (0.00 sec)

## 为表和字段取别名

在查询时，可以为表和字段取一个别名。这个别名可以代替其指定的表和字段。以下将详细的讲解怎样为表和字段取别名、

### 为表取别名

当表的名称特别长时，在查询中直接使用表名很不方便。这时可以为表取一个别名。用这个别名来代替表的名称。例如，电力软件中的变压器的表的名称为：power\_system\_transform。如果要使用该表下面的字段id，但同时查询的其他表中也有id字段。这样就必须指明是哪个表下的id字段，如power\_system\_transform.id。因为变压器表取名太长，使用起来很不方便。为了解决这个问题，可以将变压器表取一个别名。如将power\_system\_transform去个别名为t，那么t就代表了变压器表。t.id与power\_system\_transform表示的意思相同了。本小节中讲解怎么样为表取一个别名，以及查询时如何使用别名。

MySQL中为表取名的基本形式如下：

表名 表的别名

案列68：

下面为department表去个别名d。然后查询表中d\_id字段取值为1001的记录。SQL代码如下:

SELECT \* FROM department d WHERE d.d\_id=1001;

测试结果：

mysql> SELECT \* FROM department d WHERE d.d\_id=1001;

+------+---------+------------+---------+

| d\_id | d\_name | function | address |

+------+---------+------------+---------+

| 1001 | keyanbu | yanfachanp | 3haolou |

+------+---------+------------+---------+

1 row in set (0.00 sec)

### 为字段取别名

当查询数据时，MySQL会显示每个输出列的名词。默认的情况下，显示的列名是创建表时定义的列名。例如，department表的列表名分别是d\_id、d\_name、function和address。当查询department表时，就会相应的显示这几个列名。有时为了显示结果更加直观，需要一个更加直观的名字来表示这一列。如department\_name可以很直接的知道这是部门名称。这时就需要将d\_name字段取别名为department\_name。下面将详细讲解如何为字段取别名。

MySQL中为字段取别名的基本形式如下：

属性名[AS] bieming

案列69:

下面为department表中的d\_id字段取名为department\_id，d\_name字段取名为department\_name。改名前，先查看department表的d\_id字段和d\_name字段的数据。以便与改名后进行对比。查询结果如下：

SELECT d\_id AS department\_id,d\_name AS department\_name FROM department;

测试结果：

mysql> SELECT d\_id AS department\_id,d\_name AS department\_name FROM department;

+---------------+-----------------+

| department\_id | department\_name |

+---------------+-----------------+

| 1001 | keyanbu |

| 1002 | shengchanb |

| 1003 | xiaoshoubu |

+---------------+-----------------+

3 rows in set (0.01 sec)

案列70：

下面为department表中的d\_id字段取名为department\_id，d\_name字段取名为department\_name。而且department表取名为d。然后查询表中d\_id字段取值为1001的记录。SQL代码如下：

SELECT d\_id AS department\_id,d\_name AS department\_name,d.function,d.address FROM department d WHERE d.d\_id=1001;

测试结果：

mysql> SELECT d\_id AS department\_id,d\_name AS department\_name,d.function,d.address FROM department d WHERE d.d\_id=1001;

+---------------+-----------------+------------+---------+

| department\_id | department\_name | function | address |

+---------------+-----------------+------------+---------+

| 1001 | keyanbu | yanfachanp | 3haolou |

+---------------+-----------------+------------+---------+

1 row in set (0.00 sec)

## 使用正则表达式查询

正则表达式是用某种模式去匹配一类字符串的一个方式。例如，使用正则表达式可以查询出包含A、B、C其中任一字母字符串。正则表达式的查询能力比通配符的查询能力更强大，而且更加的灵活。正则表达式可以应用与非常复杂的查询。本节将详细讲解如何使用正则表达式来查询。

MySQL中，使用REGEXP关键字来查询正则表达式。其基本形式如下：

属性名 REGEXP ‘匹配方式’

### 查询以特定字符或字符开头的的记录

使用字符“^”可以匹配以特定字符或字符串开头的记录。

创建一个info表用于下面的实例

CREATE TABLE info(id INT,name VARCHAR(20));

INSERT INTO info VALUES(1,'Aric');

INSERT INTO info VALUES(2,'Eric');

INSERT INTO info VALUES(4,'Jack');

INSERT INTO info VALUES(5,'Lucy');

INSERT INTO info VALUES(6,'Lily');

INSERT INTO info VALUES(7,'Tom');

INSERT INTO info VALUES(8,'aaa');

INSERT INTO info VALUES(9,'dadaaa');

INSERT INTO info VALUES(10,'aaabd');

INSERT INTO info VALUES(11,'abc12');

INSERT INTO info VALUES(12,'ad321');

INSERT INTO info VALUES(17,'ababab');

案列71：

下面从info表name字段总查询以字母“L”开头的记录.SQL代码如下：

SELECT \* FROM info WHERE name REGEXP "^L";

测试结果

mysql> SELECT \* FROM info WHERE name REGEXP "^L";

+------+------+

| id | name |

+------+------+

| 5 | Lucy |

| 6 | Lily |

+------+------+

2 rows in set (0.01 sec)

案列72：

下面从info表name字段中查询以字符串“aaa”开头的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP '^aaa';

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP '^aaa';

+------+-------+

| id | name |

+------+-------+

| 8 | aaa |

| 10 | aaabd |

+------+-------+

2 rows in set (0.00 sec)

### 查询以特定字符或字符串结尾的记录

使用字符”$“可以匹配以特定字符或字符串结尾的记录

案列73：

下面从info表name字段中查询以字母”c“结尾的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP 'c$'

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP 'c$';

+------+------+

| id | name |

+------+------+

| 1 | Aric |

| 2 | Eric |

+------+------+

2 rows in set (0.00 sec)

案列74：

下面从info表name字段中查询以字母”aaa“结尾的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP 'aaa$';

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP 'aaa$';

+------+--------+

| id | name |

+------+--------+

| 8 | aaa |

| 9 | dadaaa |

+------+--------+

2 rows in set (0.00 sec)

### 用符号”.“来替代字符串中的任意一个字符

用正则表达式来查询，可以用”.“来替代字符串中的任意一个字符。

案列75：

下面从info表name字段中查询字母”L“开头，以字母”y“结尾，中间有两个任意字符的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP '^L..y$'；

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP '^L..y$';

+------+------+

| id | name |

+------+------+

| 5 | Lucy |

| 6 | Lily |

+------+------+

2 rows in set (0.00 sec)

### 匹配指定字符中的任意一个

使用方括号（[]）可以将需要查询字符组成一个字符集。只要记录中包含方括号中的任意字符，该记录将被查询出来。例如，通过”[abc]“可以查询包含a、b、c这三个字母中任何一个记录。

案列76：

下面从info表name字段中查询包含”c“，”e“，”o“这三个字母中的任意一个的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP '[ceo]';

测试结果

mysql> SELECT \* FROM info WHERE name REGEXP '[ceo]';

+------+-------+

| id | name |

+------+-------+

| 1 | Aric |

| 2 | Eric |

| 4 | Jack |

| 5 | Lucy |

| 7 | Tom |

| 11 | abc12 |

+------+-------+

6 rows in set (0.00 sec)

案列77：

下面从info表name字段中查询包含数据的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP '[0-9]';

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP '[0-9]';

+------+-------+

| id | name |

+------+-------+

| 11 | abc12 |

| 12 | ad321 |

+------+-------+

2 rows in set (0.00 sec)

案列78：

下面从info表name字段中查询包含束河或者字母a、b、c的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP '[0-9a-c]';

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP '[0-9a-c]';

+------+--------+

| id | name |

+------+--------+

| 1 | Aric |

| 2 | Eric |

| 4 | Jack |

| 5 | Lucy |

| 8 | aaa |

| 9 | dadaaa |

| 10 | aaabd |

| 11 | abc12 |

| 12 | ad321 |

| 17 | ababab |

+------+--------+

10 rows in set (0.00 sec)

### 匹配指定字符以外的字符

使用”[^字符集合]“可以匹配指定字符以外的字符。

案列79：

下面从info表name字段中查询包含a到w字母和数据以外的字符的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP '[^a-w0-9]';

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP '[^a-w0-9]';

+------+------+

| id | name |

+------+------+

| 5 | Lucy |

| 6 | Lily |

+------+------+

2 rows in set (0.00 sec)

### 匹配指定字符串

正则表达式可以匹配字符串。当表中的记录包含这个字符串时，就可以将该记录查询出来。如果指定多个字符串时，需要用符号”|“。只要匹配这些字符串中的任意一个即可。

案列80：

下面从info表name字段中查询包含”ic”。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP 'ic';

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP 'ic';

+------+------+

| id | name |

+------+------+

| 1 | Aric |

| 2 | Eric |

+------+------+

2 rows in set (0.00 sec)

案列81：

下面从info表name字段中查询包含”ic“，”uc“，”ab“的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP 'ic|uc|ab';

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP 'ic|uc|ab';

+------+--------+

| id | name |

+------+--------+

| 1 | Aric |

| 2 | Eric |

| 5 | Lucy |

| 10 | aaabd |

| 11 | abc12 |

| 17 | ababab |

+------+--------+

6 rows in set (0.00 sec)

### 使用”\*“和”+“来匹配多个字符

正则表达式中，”\*“和”+“都可以匹配多个该符号之前的字符。但是，”+“表示至少一个或多个字符，而”\*“可以表示零个或多个字符。

案列82：

下面从info表name字段中查询字母”c“之前出现过”a“的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP 'a\*c';

SELECT \* FROM info WHERE name REGEXP 'a+c';

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP 'a\*c';

+------+-------+

| id | name |

+------+-------+

| 1 | Aric |

| 2 | Eric |

| 4 | Jack |

| 5 | Lucy |

| 11 | abc12 |

+------+-------+

5 rows in set (0.00 sec)

mysql> SELECT \* FROM info WHERE name REGEXP 'a+c';

+------+------+

| id | name |

+------+------+

| 4 | Jack |

+------+------+

1 row in set (0.00 sec)

### 使用{M}或者{M、N}来指定字符串连续出现的次数

正则表达式中，”字符串{M}“表示字符串至少出现M此；”字符串{M,N}“表示字符串连续出现至少M此，最多N次。例如，”ab{2}“表示字符串”ab“连续出现两次。”ab{2,4}“表示字符串”ab“连续出现至少两次，最多四次。

案列83：

下面从info表name字段中查询出现过的“a“三次的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP 'a{3}';

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP 'a{3}';

+------+--------+

| id | name |

+------+--------+

| 8 | aaa |

| 9 | dadaaa |

| 10 | aaabd |

+------+--------+

3 rows in set (0.00 sec)

案列84：

下面从info表name字段中查询出现过的“ab“最少一次，最多三次的记录。SQL代码如下：

SELECT \* FROM info WHERE name REGEXP 'ab{1,3}';

测试结果：

mysql> SELECT \* FROM info WHERE name REGEXP 'ab{1,3}';

+------+--------+

| id | name |

+------+--------+

| 10 | aaabd |

| 11 | abc12 |

| 17 | ababab |

+------+--------+

3 rows in set (0.00 sec)