第04章_运算符

1. 算术运算符

算术运算符主要用于数学运算，其可以连接运算符前后的两个数值或表达式

对数值或表达式进行加（+）、减（-）、乘（*）、除（/）和取模（%）运算

运　算　符	名　　称	作　　用	示　　例
+	加法运算符	计算两个值或表达式的和	SELECT A + B
-	减法运算符	计算两个值或表达式的差	SELECT A - B
*	乘法运算符	计算两个值或表达式的乘积	SELECT A * B
/或DIV	除法运算符	计算两个值或表达式的商	SELECT A / B 或者 SELECT A DIV B
%或MOD	求模（求余）运算符	计算两个值或表达式的余数	SELECT A % B 或者 SELECT A MOD B

1．加法与减法运算符

```
mysql> SELECT 100, 100 + 0, 100 - 0, 100 + 50, 100 + 50 -30, 100 + 35.5, 100 -
35.5
FROM dual;

+-----+---------+---------+----------+--------------+------------+------------
+
| 100 | 100 + 0 | 100 - 0 | 100 + 50 | 100 + 50 -30 | 100 + 35.5 | 100 - 35.5
|
+-----+---------+---------+----------+--------------+------------+------------
+
| 100 |     100 |     100 |      150 |          120 |      135.5 |       64.5
|
+-----+---------+---------+----------+--------------+------------+------------
+
1 row in set (0.00 sec)
```

由运算结果可以得出如下结论：

> - 一个整数类型的值对整数进行加法和减法操作，结果还是一个整数；
> - 一个整数类型的值对浮点数进行加法和减法操作，结果是一个浮点数；
> - 加法和减法的优先级相同，进行先加后减操作与进行先减后加操作的结果是一样的；
> - 在Java中，+的左右两边如果有字符串，那么表示字符串的拼接。但是在MySQL中+只表示数值相加。如果遇到非数值类型，先尝试转成数值，如果转失败，就按0计算。（补充：

> MySQL中字符串拼接要使用字符串函数CONCAT()实现)

字符串于数字的转换规则

- **隐式转换**

注意点：

1、在SQL中，+没有连接的作用，就表示加法运算。此时，会将字符串转换为数值（隐式转换）【数字字符串转为对应数字，字符字符串转为0】

2、即字符与数字参与运算，先将字符转为数字，能转数字就转数字，不能就转0

```
SELECT 100 + '1'  # 在Java语言中，结果是：1001。
FROM DUAL;#伪表
+-----------+
| 100 + '1' |
+-----------+
|       101 |
+-----------+
```

```
SELECT 100 + 'a' #此时将'a'看做0处理
FROM DUAL;
+-----------+
| 100 + 'a' |
+-----------+
|       100 |
+-----------+
```

2. 乘法与除法运算符

```
-- 分母如果为0，则结果为null
-- SQL中只要涉及除法运算，结果都是浮点数

SELECT 100, 100 * 1, 100 * 1.0, 100 / 1.0, 100 / 2,
100 + 2 * 5 / 2,100 / 3, 100 DIV 0  # 分母如果为0，则结果为null 【SQL中只要涉及除法运算，结果都是浮点数】
FROM DUAL;
+-----+---------+-----------+-----------+---------+-----------------+---------
+-----------+
| 100 | 100 * 1 | 100 * 1.0 | 100 / 1.0 | 100 / 2 | 100 + 2 * 5 / 2 | 100 / 3
| 100 DIV 0 |
+-----+---------+-----------+-----------+---------+-----------------+---------
+-----------+
| 100 |     100 |     100.0 |  100.0000 | 50.0000 |        105.0000 | 33.3333
|      NULL |
+-----+---------+-----------+-----------+---------+-----------------+---------
+-----------+
```

```
-- null值参与运算，结果为null
SELECT 100 + NULL  # null值参与运算，结果为null
FROM DUAL;
+------------+
| 100 + NULL |
+------------+
|       NULL |
+------------+
```

```
#计算出员工的年基本工资
SELECT employee_id,salary,salary * 12 annual_sal
FROM employees;

+-------------+----------+------------+
| employee_id | salary   | annual_sal |
+-------------+----------+------------+
|         100 | 24000.00 |  288000.00 |
|         101 | 17000.00 |  204000.00 |
|         102 | 17000.00 |  204000.00 |
|         202 |  6000.00 |   72000.00 |
|         203 |  6500.00 |   78000.00 |
|         204 | 10000.00 |  120000.00 |
|         205 | 12000.00 |  144000.00 |
|         206 |  8300.00 |   99600.00 |
+-------------+----------+------------+
```

由运算结果可以得出如下结论：

- 一个数乘以整数1和除以整数1后仍得原数
- 一个数乘以浮点数1和除以浮点数1后变成浮点数，数值与原数相等
- 一个数除以整数后，不管是否能除尽，结果都为一个浮点数
- 一个数除以另一个数，除不尽时，结果为一个浮点数，并保留到小数点后4位
- 乘法和除法的优先级相同，进行先乘后除操作与先除后乘操作，得出的结果相同
- 在数学运算中，0不能用作除数，在MySQL中，一个数除以0为NULL

3. 求模（求余）运算符

```
# 取模运算： % mod
SELECT 12 % 3,12 % 5, 12 MOD -5,-12 % 5,-12 % -5
FROM DUAL;#取模结果，与被模数有关，与模数无关
+--------+--------+-----------+---------+----------+
| 12 % 3 | 12 % 5 | 12 MOD -5 | -12 % 5 | -12 % -5 |
+--------+--------+-----------+---------+----------+
|      0 |      2 |         2 |      -2 |       -2 |
+--------+--------+-----------+---------+----------+
```

```
#筛选出employee_id是偶数的员工
SELECT employee_id,salary,last_name,phone_number FROM employees
WHERE employee_id MOD 2 = 0;
```

```
+-------------+----------+-------------+--------------------+
| employee_id | salary   | last_name   | phone_number       |
+-------------+----------+-------------+--------------------+
|         100 | 24000.00 | King        | 515.123.4567       |
|         102 | 17000.00 | De Haan     | 515.123.4569       |
|         104 |  6000.00 | Ernst       | 590.423.4568       |
|         106 |  4800.00 | Pataballa   | 590.423.4560       |
|         202 |  6000.00 | Fay         | 603.123.6666       |
|         204 | 10000.00 | Baer        | 515.123.8888       |
|         206 |  8300.00 | Gietz       | 515.123.8181       |
+-------------+----------+-------------+--------------------+
```

2. 比较运算符

比较运算符用来对**表达式左边的操作数和右边的操作数进行比较**，比较的结果为真则返回1，比较的结果为假则返回0，其他情况则返回NULL

比较运算符经常被用来作为SELECT查询语句的条件来使用，返回符合条件的结果记录

运　算　符	名　　称	作　　用	示　　例
=	等于运算符	判断两个值、字符串或表达式是否相等	SELECT C FROM TABLE WHERE A = B
<=>	安全等于运算符	安全地判断两个值、字符串或表达式是否相等	SELECT C FROM TABLE WHERE A <=> B
<>(!=)	不等于运算符	判断两个值、字符串或表达式是否不相等	SELECT C FROM TABLE WHERE A <> B SELECT C FROM TABLE WHERE A != B
<	小于运算符	判断前面的值、字符串或表达式是否小于后面的值、字符串或表达式	SELECT C FROM TABLE WHERE A < B
<=	小于等于运算符	判断前面的值、字符串或表达式是否小于等于后面的值、字符串或表达式	SELECT C FROM TABLE WHERE A <= B
>	大于运算符	判断前面的值、字符串或表达式是否大于后面的值、字符串或表达式	SELECT C FROM TABLE WHERE A > B
>=	大于等于运算符	判断前面的值、字符串或表达式是否大于等于后面的值、字符串或表达式	SELECT C FROM TABLE WHERE A >= B

1. 等号运算符

- 等号运算符（=）判断等号两边的值、字符串或表达式是否相等，如果相等则返回1，不相等则返回0。
- 在使用等号运算符时，遵循如下规则：
  - 如果等号两边的值、字符串或表达式都为字符串，则MySQL会按照字符串进行比较，其比较的是每个字符串中字符的ANSI编码是否相等。
  - **如果等号两边的值都是整数，则MySQL会按照整数来**比较两个值的大小。
  - 如果等号两边的值一个是整数，另一个是字符串，则MySQL会将字符串转化为数字进行比较。
  - 如果等号两边的值、字符串或表达式中有一个为NULL，则比较结果为NULL。
- 对比：SQL中赋值符号使用　　=
- **字符串的隐式转换**

```
-- 字符串存在隐式转换。如果转换数值不成功，则看做0
SELECT 1 = 2,1 != 2,1 = '1',1 = 'a',0 = 'a' #字符串存在隐式转换。如果转换数值不成功，则看做0
FROM DUAL;
+-------+--------+---------+---------+---------+
| 1 = 2 | 1 != 2 | 1 = '1' | 1 = 'a' | 0 = 'a' |
+-------+--------+---------+---------+---------+
|     0 |      1 |       1 |       0 |       1 |
+-------+--------+---------+---------+---------+
```

```
mysql> SELECT 1 = 1, 1 = '1', 1 = 0, 'a' = 'a', (5 + 3) = (2 + 6), '' = NULL , NULL = NULL;
+-------+---------+-------+-----------+-------------------+-----------+-------------+
| 1 = 1 | 1 = '1' | 1 = 0 | 'a' = 'a' | (5 + 3) = (2 + 6) | '' = NULL | NULL = NULL |
+-------+---------+-------+-----------+-------------------+-----------+-------------+
|     1 |       1 |     0 |         1 |                 1 |      NULL |        NULL |
+-------+---------+-------+-----------+-------------------+-----------+-------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT 1 = 2, 0 = 'abc', 1 = 'abc' FROM dual;
+-------+-----------+-----------+
| 1 = 2 | 0 = 'abc' | 1 = 'abc' |
+-------+-----------+-----------+
|     0 |         1 |         0 |
+-------+-----------+-----------+
1 row in set, 2 warnings (0.00 sec)
```

- **字符串的比较**

```
SELECT 'a' = 'a','ab' = 'ab','a' = 'b' #两边都是字符串的话，则按照ANSI的比较规则进行比较。
FROM DUAL;
+-----------+-------------+-----------+
| 'a' = 'a' | 'ab' = 'ab' | 'a' = 'b' |
+-----------+-------------+-----------+
|         1 |           1 |         0 |
+-----------+-------------+-----------+
```

- **有null参与的判断结果为null**

```
SELECT 1 = NULL,NULL = NULL # 只要有null参与判断，结果就为null
FROM DUAL;
+----------+-------------+
| 1 = NULL | NULL = NULL |
+----------+-------------+
|     NULL |        NULL |
+----------+-------------+
```

```
#查询salary=10000，注意在Java中比较是==
SELECT employee_id,salary FROM employees WHERE salary = 10000;
mysql> SELECT employee_id,salary FROM employees WHERE salary = 10000;
+-------------+----------+
| employee_id | salary   |
+-------------+----------+
|         150 | 10000.00 |
|         156 | 10000.00 |
|         169 | 10000.00 |
|         204 | 10000.00 |
+-------------+----------+
```

- **null作为过滤条件**

```
-- null作为过滤条件，相当于没有过滤
SELECT last_name,salary,commission_pct
FROM employees
#where salary = 6000;
WHERE commission_pct = NULL;  #此时执行，不会有任何的结果【只有当过滤条件为真1时，才会取出这条记录；左右两边有一边为null则结果为null】
```

2. 安全等于运算符

安全等于运算符（<=>）与等于运算符（=）的作用是相似的，`唯一的区别`是'<=>'可以用来**对NULL进行判断**

在两个操作数均为NULL时，其**返回值为1**，而不为NULL

当一个操作数为NULL时，其**返回值为0**，而不为NULL

```
-- <=> ：安全等于。 记忆技巧：为NULL而生，与null进行比较结果不再只是null


SELECT 1 <=> 2,1 <=> '1',1 <=> 'a',0 <=> 'a'
FROM DUAL;
+---------+----------+----------+----------+
| 1 <=> 2 | 1 <=> '1' | 1 <=> 'a' | 0 <=> 'a' |
+---------+----------+----------+----------+
|       0 |         1 |         0 |         1 |
+---------+----------+----------+----------+

SELECT 1 <=> NULL, NULL <=> NULL #安全等于 为NULL而生 假、真
FROM DUAL;
+------------+---------------+
| 1 <=> NULL | NULL <=> NULL |
+------------+---------------+
|          0 |             1 |
+------------+---------------+
```

```
#查询commission_pct等于0.40
SELECT employee_id,commission_pct FROM employees WHERE commission_pct = 0.40;

SELECT employee_id,commission_pct FROM employees WHERE commission_pct <=>
0.40;

#如果把0.40改成 NULL 呢？

#练习：查询表中commission_pct为null的数据有哪些
SELECT last_name,salary,commission_pct
FROM employees
WHERE commission_pct <=> NULL;
+-------------+----------+----------------+
| last_name   | salary   | commission_pct |
+-------------+----------+----------------+
| King        | 24000.00 |           NULL |
| Kochhar     | 17000.00 |           NULL |
| Gietz       |  8300.00 |           NULL |
+-------------+----------+----------------+
```

可以看到，使用安全等于运算符时，两边的操作数的值都为NULL时，返回的结果为1而不是NULL，其他返回结果与等于运算符相同

3. 不等于运算符

不等于运算符（<>和!=）用于判断两边的数字、字符串或者表达式的值是否不相等，如果不相等则返回1，相等则返回0。不等于运算符不能判断NULL值。如果两边的值有任意一个为NULL，或两边都为NULL，则结果为NULL

```
<>  !=
```

SQL语句示例如下：

```
mysql> SELECT 1 <> 1, 1 != 2, 'a' != 'b', (3+4) <> (2+6), 'a' != NULL, NULL <>
NULL;
+--------+--------+------------+----------------+-------------+--------------+
| 1 <> 1 | 1 != 2 | 'a' != 'b' | (3+4) <> (2+6) | 'a' != NULL | NULL <> NULL |
+--------+--------+------------+----------------+-------------+--------------+
|      0 |      1 |          1 |              1 |        NULL |         NULL |
+--------+--------+------------+----------------+-------------+--------------+
1 row in set (0.00 sec)
```

此外，还有非符号类型的运算符：

运 算 符	名 称	作 用	示 例
IS NULL	为空运算符	判断值、字符串或表达式是否为空	SELECT B FROM TABLE WHERE A IS NULL
		判断值、字符串或表达式是否	SELECT B FROM TABLE WHERE A

LIKE	模糊匹配运算符	判断一个值是否符合模糊匹配规则	SELECT C FROM TABLE WHERE A LIKE B
REGEXP	正则表达式运算符	判断一个值是否符合正则表达式的规则	SELECT C FROM TABLE WHERE A REGEXP B
RLIKE	正则表达式运算符	判断一个值是否符合正则表达式的规则	SELECT C FROM TABLE WHERE A RLIKE B

4. 空运算符

空运算符（IS NULL或者ISNULL）判断一个值是否为NULL，如果为NULL则返回1，否则返回0。
SQL语句示例如下：

空值三种表现形式：

> 字段 IS NULL
>
> ISNULL(字段)
>
> 字段<=>NULL

```
mysql> SELECT NULL IS NULL, ISNULL(NULL), ISNULL('a'), 1 IS NULL;
+--------------+--------------+-------------+-----------+
| NULL IS NULL | ISNULL(NULL) | ISNULL('a') | 1 IS NULL |
+--------------+--------------+-------------+-----------+
|            1 |            1 |           0 |         0 |
+--------------+--------------+-------------+-----------+
1 row in set (0.00 sec)
```

```
#查询commission_pct等于NULL。比较如下的四种写法
SELECT employee_id,commission_pct FROM employees WHERE commission_pct IS
NULL;#结果为0 1
SELECT employee_id,commission_pct FROM employees WHERE commission_pct <=>
NULL;#结果为0 1
SELECT employee_id,commission_pct FROM employees WHERE ISNULL(commission_pct);
#结果为0 1
SELECT employee_id,commission_pct FROM employees WHERE commission_pct = NULL;
#结果仅为null 没有看到1就不会出结果
```

```
#练习：查询表中commission_pct为null的数据有哪些

-- 1 使用关键字判断某个字段是否为空
SELECT last_name,salary,commission_pct
FROM employees
WHERE commission_pct IS NULL;
+-------------+----------+----------------+
| last_name   | salary   | commission_pct |
+-------------+----------+----------------+
| King        | 24000.00 |           NULL |
| Kochhar     | 17000.00 |           NULL |
| Gietz       |  8300.00 |           NULL |
+-------------+----------+----------------+

#或
-- 2 使用函数判断某个字段是否为空
SELECT last_name,salary,commission_pct
FROM employees
WHERE ISNULL(commission_pct);

+-------------+----------+----------------+
| last_name   | salary   | commission_pct |
+-------------+----------+----------------+
| King        | 24000.00 |           NULL |
```

```
| Kochhar     | 17000.00 |           NULL |
| Gietz       |  8300.00 |           NULL |
+-------------+----------+----------------+
```

5. 非空运算符

非空运算符（IS NOT NULL）判断一个值是否不为NULL，如果不为NULL则返回1，否则返回0。
SQL语句示例如下：

非空三种表现形式：

> 字段 IS NOT NULL
>
> NOT 字段 <=>NULL
>
> NOT ISNULL(字段)

```
mysql> SELECT NULL IS NOT NULL, 'a' IS NOT NULL,  1 IS NOT NULL;
+------------------+-----------------+---------------+
| NULL IS NOT NULL | 'a' IS NOT NULL | 1 IS NOT NULL |
+------------------+-----------------+---------------+
|                0 |               1 |             1 |
+------------------+-----------------+---------------+
1 row in set (0.01 sec)
```

```
#查询commission_pct不等于NULL
SELECT employee_id,commission_pct FROM employees WHERE commission_pct IS NOT
NULL;
SELECT employee_id,commission_pct FROM employees WHERE NOT commission_pct <=>
NULL;
SELECT employee_id,commission_pct FROM employees WHERE NOT
ISNULL(commission_pct);
```

```
#练习：查询表中commission_pct不为null的数据有哪些
-- 1 使用关键字判断某个字段是否不为空
SELECT last_name,salary,commission_pct
FROM employees
WHERE commission_pct IS NOT NULL;

#或
-- 2 使用安全等与某个字段是否为null，然后取反 NOT
SELECT last_name,salary,commission_pct
FROM employees
WHERE NOT commission_pct <=> NULL;
```

6. 最小值运算符

语法格式为：LEAST(值1，值2，...，值n)。其中，“值n”表示参数列表中有n个值。在有两个或多个参数的情况下，返回最小值。

```
mysql> SELECT LEAST (1,0,2), LEAST('b','a','c'), LEAST(1,NULL,2);
+--------------+--------------------+-----------------+
| LEAST (1,0,2) | LEAST('b','a','c') | LEAST(1,NULL,2) |
+--------------+--------------------+-----------------+
|       0       |         a          |      NULL       |
+---------------+--------------------+-----------------+
1 row in set (0.00 sec)
```

```
-- LEAST（）取集合中的最小
-- GREATEST（）取集合中的最大


SELECT LEAST('g','b','t','m'),GREATEST('g','b','t','m')
FROM DUAL;
+-----------------------+--------------------------+
| LEAST('g','b','t','m') | GREATEST('g','b','t','m') |
+-----------------------+--------------------------+
| b                     | t                        |
+-----------------------+--------------------------+


SELECT LEAST(first_name,last_name)
a,LEAST(LENGTH(first_name),LENGTH(last_name))
b,first_name,last_name,length(first_name) from employees;
+-------------+------+-------------+-------------+--------------------+
| a           | b    | first_name  | last_name   | length(first_name) |
+-------------+------+-------------+-------------+--------------------+
| King        |    4 | Steven      | King        |                  6 |
| Kochhar     |    5 | Neena       | Kochhar     |                  5 |
| Gietz       |    5 | William     | Gietz       |                  7 |
+-------------+------+-------------+-------------+--------------------+
```

由结果可以看到，当参数是整数或者浮点数时，LEAST将返回其中最小的值；当参数为字符串时，返回字母表中顺序最靠前的字符；当比较值列表中有NULL时，不能判断大小，返回值为NULL。

7. 最大值运算符

语法格式为：GREATEST(值1，值2，...，值n)。其中，n表示参数列表中有n个值。当有两个或多个参数时，返回值为最大值。假如任意一个自变量为NULL，则GREATEST()的返回值为NULL。

```
mysql> SELECT GREATEST(1,0,2), GREATEST('b','a','c'), GREATEST(1,NULL,2);
+----------------+---------------------+-------------------+
| GREATEST(1,0,2) | GREATEST('b','a','c') | GREATEST(1,NULL,2) |
+----------------+---------------------+-------------------+
|              2 | c                   |              NULL |
+----------------+---------------------+-------------------+
1 row in set (0.00 sec)
```

由结果可以看到，当参数中是整数或者浮点数时，GREATEST将返回其中最大的值；当参数为字符串时，返回字母表中顺序最靠后的字符；当比较值列表中有NULL时，不能判断大小，返回值为NULL。

8. BETWEEN AND运算符

BETWEEN运算符使用的格式通常为SELECT D FROM TABLE WHERE C BETWEEN A AND B，此时，当C大于或等于A，并且C小于或等于B时，结果为1，否则结果为0。

```
mysql> SELECT 1 BETWEEN 0 AND 1, 10 BETWEEN 11 AND 12, 'b' BETWEEN 'a' AND
'c';
+------------------+---------------------+-------------------------+
| 1 BETWEEN 0 AND 1 | 10 BETWEEN 11 AND 12 | 'b' BETWEEN 'a' AND 'c' |
+------------------+---------------------+-------------------------+
|                 1 |                   0 |                       1 |
+------------------+---------------------+-------------------------+
1 row in set (0.00 sec)
```

```
-- BETWEEN 条件下界1 AND 条件上界2  （查询条件1和条件2范围内的数据，包含边界）

#查询工资在6000 到 8000的员工信息
SELECT employee_id,last_name,salary
FROM employees
#where salary between 6000 and 7000;
WHERE salary >= 6000 && salary <= 7000; #等同
+-------------+-----------+---------+
| employee_id | last_name | salary  |
+-------------+-----------+---------+
|         104 | Ernst     | 6000.00 |
|         113 | Popp      | 6900.00 |
|         179 | Johnson   | 6200.00 |
|         202 | Fay       | 6000.00 |
|         203 | Mavris    | 6500.00 |
+-------------+-----------+---------+

-- between ... and ... 是有上界值和下界值的
#交换6000 和 8000之后，查询不到数据
SELECT employee_id,last_name,salary
FROM employees
WHERE salary BETWEEN 7000 AND 6000;


-- 确定区间范围，再取反
#查询工资不在6000 到 8000的员工信息
SELECT employee_id,last_name,salary
FROM employees
WHERE salary NOT BETWEEN 2500 AND 15000;
#where salary < 2500 or salary > 15000;
+-------------+------------+----------+
| employee_id | last_name  | salary   |
+-------------+------------+----------+
|         100 | King       | 24000.00 |
```

```
|         101 | Kochhar     | 17000.00 |
|         102 | De Haan     | 17000.00 |
|         127 | Landry      |  2400.00 |
|         128 | Markle      |  2200.00 |
|         132 | Olson       |  2100.00 |
|         135 | Gee         |  2400.00 |
|         136 | Philtanker  |  2200.00 |
+-------------+-------------+----------+
```

9. IN运算符

IN运算符用于判断给定的值是否是IN列表中的一个值，如果是则返回1，否则返回0。如果给定的值为NULL，或者IN列表中存在NULL，则结果为NULL。

```
mysql> SELECT 'a' IN ('a','b','c'), 1 IN (2,3), NULL IN ('a','b'), 'a' IN
('a', NULL);
+----------------------+------------+-------------------+--------------------+
| 'a' IN ('a','b','c') | 1 IN (2,3) | NULL IN ('a','b') | 'a' IN ('a', NULL) |
+----------------------+------------+-------------------+--------------------+
|            1         |        0   |         NULL      |         1          |
+----------------------+------------+-------------------+--------------------+
1 row in set (0.00 sec)
```

```
-- in((set) 相当于使用过滤条件的or

#练习：查询部门为10,20,30部门的员工信息
SELECT last_name,salary,department_id
FROM employees
#where department_id = 10 or 20 or 30; #在MySQL中逻辑运算符只要有一边是1；则会过滤出该条件
#where department_id = 10 or department_id = 20 or department_id = 30;
WHERE department_id IN (10,20,30);
+------------+----------+---------------+
| last_name  | salary   | department_id |
+------------+----------+---------------+
| Whalen     |  4400.00 |            10 |
| Hartstein  | 13000.00 |            20 |
| Fay        |  6000.00 |            20 |
| Raphaely   | 11000.00 |            30 |
| Khoo       |  3100.00 |            30 |
| Baida      |  2900.00 |            30 |
| Tobias     |  2800.00 |            30 |
| Himuro     |  2600.00 |            30 |
| Colmenares |  2500.00 |            30 |
+------------+----------+---------------+
9 rows in set (0.00 sec)
```

10. NOT IN运算符

NOT IN运算符用于判断给定的值是否不是IN列表中的一个值，如果不是IN列表中的一个值，则返回1，否则返回0。

```
mysql> SELECT 'a' NOT IN ('a','b','c'), 1 NOT IN (2,3);
+-------------------------+---------------+
| 'a' NOT IN ('a','b','c') | 1 NOT IN (2,3) |
+-------------------------+---------------+
|                  0      |             1  |
+-------------------------+---------------+
1 row in set (0.00 sec)
```

```
SELECT last_name,salary,department_id
FROM employees
WHERE department_id NOT IN (10,20,30);
mysql> SELECT last_name,salary,department_id
    -> FROM employees
    -> WHERE department_id NOT IN (10,20,30);
+-------------+----------+---------------+
| last_name   | salary   | department_id |
+-------------+----------+---------------+
| King        | 24000.00 |            90 |
| Higgins     | 12000.00 |           110 |
| Gietz       |  8300.00 |           110 |
+-------------+----------+---------------+
97 rows in set (0.00 sec)
```

11. LIKE运算符

LIKE运算符主要用来匹配字符串，通常用于模糊匹配，如果满足条件则返回1，否则返回0

如果给定的值或者匹配条件为NULL，则返回结果为NULL。

LIKE运算符通常使用如下通配符：

```
“%”：匹配0个或多个字符。
“_”：只能匹配一个字符。
```

SQL语句示例如下：

```
mysql> SELECT NULL LIKE 'abc', 'abc' LIKE NULL;
+-----------------+-----------------+
| NULL LIKE 'abc' | 'abc' LIKE NULL |
+-----------------+-----------------+
|          NULL   |          NULL   |
+-----------------+-----------------+
1 row in set (0.00 sec)
```

```
#练习：查询last_name中包含字符'a'的员工信息
```

```
SELECT last_name
FROM employees
WHERE last_name LIKE '%a%';
+------------+
| last_name  |
+------------+
| Kochhar    |
| De Haan    |
| Baer       |
+------------+

#练习：查询last_name中以字符'a'开头的员工信息
SELECT last_name
FROM employees
WHERE last_name LIKE 'a%';
+-----------+
| last_name |
+-----------+
| Austin    |
| Atkinson  |
| Ande      |
| Abel      |
+-----------+

#练习：查询last_name中包含字符'a'且包含字符'e'的员工信息
#写法1：
SELECT last_name
FROM employees
WHERE last_name LIKE '%a%' AND last_name LIKE '%e%';
#写法2：
SELECT last_name
FROM employees
WHERE last_name LIKE '%a%e%' OR last_name LIKE '%e%a%';
+------------+
| last_name  |
+------------+
| De Haan    |
| Faviet     |
| Baer       |
+------------+

#练习：查询第3个字符是'a'的员工信息
SELECT last_name
FROM employees
WHERE last_name LIKE '__a%';
+-----------+
| last_name |
+-----------+
| Grant     |
| Grant     |
| Whalen    |
+-----------+
```

ESCAPE

- 回避特殊符号的：**使用转义符**
- 如果使用\表示转义，要省略ESCAPE。如果不是\，则要加上ESCAPE。

```
#练习：查询第2个字符是_且第3个字符是'a'的员工信息
#需要使用转义字符: \
SELECT last_name
FROM employees
WHERE last_name LIKE '_\_a%';

#或者  (了解)【定义$作为转义标识】
SELECT last_name
FROM employees
WHERE last_name LIKE '_$_a%' ESCAPE '$';
-- Empty set (0.00 sec)
```

12. REGEXP运算符

REGEXP运算符用来匹配字符串，语法格式为：`expr REGEXP 匹配条件`。如果expr满足匹配条件，返回1；如果不满足，则返回0。若expr或匹配条件任意一个为NULL，则结果为NULL。

REGEXP运算符在进行匹配时，常用的有下面几种通配符：

（1）‘^’匹配以该字符后面的字符开头的字符串。
（2）‘$’匹配以该字符前面的字符结尾的字符串。
（3）‘.’匹配任何一个单字符。
（4）“[...]”匹配在方括号内的任何字符。例如，“[abc]”匹配“a”或“b”或“c”。为了命名字符的范围，使用一个‘-’。“[a-z]”匹配任何字母，而“[0-9]”匹配任何数字。
（5）“*匹配零个或多个在它前面的字符。例如，“x”*匹配任何数量的‘x’字符，“[0-9]”*匹配任何数量的数字，而“*”匹配任何数量的任何字符。

SQL语句示例如下：

```
-- 以shk开头，以t结尾，包含hk
SELECT 'shkstart' REGEXP '^shk', 'shkstart' REGEXP 't$', 'shkstart' REGEXP
'hk'
FROM DUAL;
+--------------------------+-----------------------+-----------------------+
| 'shkstart' REGEXP '^shk' | 'shkstart' REGEXP 't$' | 'shkstart' REGEXP 'hk' |
+--------------------------+-----------------------+-----------------------+
|                        1 |                     1 |                     1 |
+--------------------------+-----------------------+-----------------------+

-- 通配符 . 代表除换行符之外的全部字符
SELECT 'atguigu' REGEXP 'gu.gu','atguigu' REGEXP '[ab]'
FROM DUAL;
+--------------------------+------------------------+
```

```
| 'atguigu' REGEXP 'gu.gu' | 'atguigu' REGEXP '[ab]' |
+--------------------------+-------------------------+
|                        1 |                       1 |
+--------------------------+-------------------------+
```

3. 逻辑运算符

逻辑运算符主要用来判断表达式的真假，在MySQL中，逻辑运算符的返回结果为1、0或者NULL。

MySQL中支持4种逻辑运算符如下：

运　算　符	作　　用	示　　例
NOT 或 ！	逻辑非	SELECT NOT A
AND 或 &&	逻辑与	SELECT A AND B SELECT A && B
OR 或 ||	逻辑或	SELECT A OR B SELECT A || B
XOR	逻辑异或	SELECT A XOR B

1．逻辑非运算符

逻辑非（NOT或!）运算符表示当给定的值为0时返回1；当给定的值为非0值时返回0；当给定的值为NULL时，返回NULL。

```
mysql> SELECT NOT 1, NOT 0, NOT(1+1), NOT !1, NOT NULL;
+-------+-------+----------+--------+----------+
| NOT 1 | NOT 0 | NOT(1+1) | NOT !1 | NOT NULL |
+-------+-------+----------+--------+----------+
|     0 |     1 |        0 |      1 |     NULL |
+-------+-------+----------+--------+----------+
1 row in set, 1 warning (0.00 sec)
```

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');
```

2．逻辑与运算符

逻辑与（AND或&&）运算符是当给定的所有值均为非0值，并且都不为NULL时，返回1；

当给定的一个值或者多个值为0时则返回0；

否则返回NULL。

```
mysql> SELECT 1 AND -1, 0 AND 1, 0 AND NULL, 1 AND NULL;
+----------+---------+------------+------------+
| 1 AND -1 | 0 AND 1 | 0 AND NULL | 1 AND NULL |
+----------+---------+------------+------------+
|        1 |       0 |          0 |       NULL |
+----------+---------+------------+------------+
1 row in set (0.00 sec)
```

```
-- or  并集
-- and 交集
SELECT last_name,salary,department_id
FROM employees
#where department_id = 10 or department_id = 20;
#where department_id = 10 and department_id = 20;
WHERE department_id = 50 AND salary > 6000;
+-----------+---------+---------------+
| last_name | salary  | department_id |
+-----------+---------+---------------+
| Weiss     | 8000.00 |            50 |
| Fripp     | 8200.00 |            50 |
| Kaufling  | 7900.00 |            50 |
| Vollman   | 6500.00 |            50 |
+-----------+---------+---------------+

#注意：AND的优先级高于OR
```

**

3. 逻辑或运算符

逻辑或（OR或||）运算符是当给定的值都不为NULL，并且任何一个值为非0值时，则返回1，否则返回0；当一个值为NULL，并且另一个值为非0值时，返回1，否则返回NULL；当两个值都为NULL时，返回NULL。

```
mysql> SELECT 1 OR -1, 1 OR 0, 1 OR NULL, 0 || NULL, NULL || NULL;
+---------+--------+-----------+-----------+--------------+
| 1 OR -1 | 1 OR 0 | 1 OR NULL | 0 || NULL | NULL || NULL |
+---------+--------+-----------+-----------+--------------+
|       1 |      1 |         1 |    NULL   |       NULL   |
+---------+--------+-----------+-----------+--------------+
1 row in set, 2 warnings (0.00 sec)
```

```
#查询基本薪资不在9000-12000之间的员工编号和基本薪资
SELECT employee_id,salary FROM employees
WHERE NOT (salary >= 9000 AND salary <= 12000);

SELECT employee_id,salary FROM employees
WHERE salary <9000 OR salary > 12000;

SELECT employee_id,salary FROM employees
WHERE salary NOT BETWEEN 9000 AND 12000;
```

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%';
```

> 注意：
>
> OR可以和AND一起使用，但是在使用时要注意两者的优先级，由于AND的优先级高于OR，因此先对AND两边的操作数进行操作，再与OR中的操作数结合。

4. 逻辑异或运算符

逻辑异或（XOR）运算符是当给定的值中任意一个值为NULL时，则返回NULL；如果两个非NULL的值都是0或者都不等于0时，则返回0；如果一个值为0，另一个值不为0时，则返回1。

```
mysql> SELECT 1 XOR -1, 1 XOR 0, 0 XOR 0, 1 XOR NULL, 1 XOR 1 XOR 1, 0 XOR 0
XOR 0;
+----------+---------+---------+------------+---------------+---------------+
| 1 XOR -1 | 1 XOR 0 | 0 XOR 0 | 1 XOR NULL | 1 XOR 1 XOR 1 | 0 XOR 0 XOR 0 |
+----------+---------+---------+------------+---------------+---------------+
|        0 |       1 |       0 |       NULL |             1 |             0 |
+----------+---------+---------+------------+---------------+---------------+
1 row in set (0.00 sec)
```

```
# XOR :追求的"异"
SELECT last_name,salary,department_id
FROM employees
WHERE department_id = 50 XOR salary > 6000;#是50号部门的工资一定不大于6000 工资大于
6000的一定不是50号部门
+-------------+----------+---------------+
| last_name   | salary   | department_id |
+-------------+----------+---------------+
| King        | 24000.00 |            90 |
| Kochhar     | 17000.00 |            90 |
| De Haan     | 17000.00 |            90 |
| Nayer       |  3200.00 |            50 |
| Mikkilineni |  2700.00 |            50 |
| Landry      |  2400.00 |            50 |
| Markle      |  2200.00 |            50 |
| Bissot      |  3300.00 |            50 |
+-------------+----------+---------------+
```

4. 位运算符

位运算符是在二进制数上进行计算的运算符

位运算符会先将操作数变成二进制数，然后进行位运算，最后将计算结果从二进制变回十进制数。

MySQL支持的位运算符如下：

运　算　符	作　　用	示　　例
&	按位与（位AND）	SELECT A & B
|	按位或（位OR）	SELECT A | B
^	按位异或（位XOR）	SELECT A ^ B
~	按位取反	SELECT ~ A
>>	按位右移	SELECT A >> 2
<<	按位左移	SELECT B << 2

1. 按位与运算符

按位与（&）运算符将给定值对应的二进制数逐位进行逻辑与运算。当给定值对应的二进制位的数值都为1时，则该位返回1，否则返回0。

```
mysql> SELECT 1 & 10, 20 & 30;
+--------+---------+
| 1 & 10 | 20 & 30 |
+--------+---------+
|      0 |      20 |
+--------+---------+
1 row in set (0.00 sec)
```

1的二进制数为0001，10的二进制数为1010，所以1 & 10的结果为0000，对应的十进制数为0。

20的二进制数为10100，30的二进制数为11110，所以20 & 30的结果为10100，对应的十进制数为20。

2. 按位或运算符

按位或（|）运算符将给定的值对应的二进制数逐位进行逻辑或运算。当给定值对应的二进制位的数值有一个或两个为1时，则该位返回1，否则返回0。

```
mysql> SELECT 1 | 10, 20 | 30;
+--------+---------+
| 1 | 10 | 20 | 30 |
+--------+---------+
|     11 |      30 |
+--------+---------+
1 row in set (0.00 sec)
```

1的二进制数为0001，10的二进制数为1010，所以1 | 10的结果为1011，对应的十进制数为11。

20的二进制数为10100，30的二进制数为11110，所以20 | 30的结果为11110，对应的十进制数为30。

3. 按位异或运算符

按位异或（^）运算符将给定的值对应的二进制数逐位进行逻辑异或运算。当给定值对应的二进制位的数值不同时，则该位返回1，否则返回0。

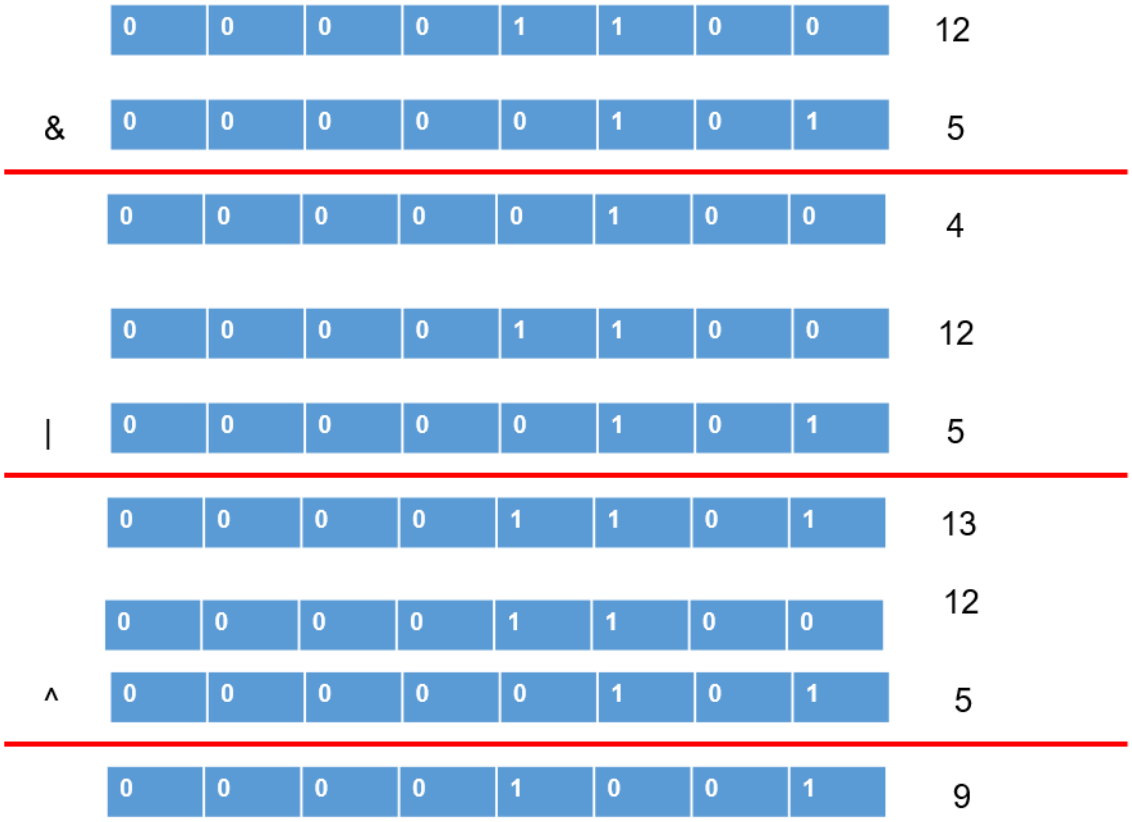
```
mysql> SELECT 1 ^ 10, 20 ^ 30;
+--------+---------+
| 1 ^ 10 | 20 ^ 30 |
+--------+---------+
|     11 |      10 |
+--------+---------+
1 row in set (0.00 sec)
```

1的二进制数为0001，10的二进制数为1010，所以1 ^ 10的结果为1011，对应的十进制数为11。

20的二进制数为10100，30的二进制数为11110，所以20 ^ 30的结果为01010，对应的十进制数为10。

再举例：

```
mysql> SELECT 12 & 5, 12 | 5,12 ^ 5 FROM DUAL;
+--------+--------+--------+
| 12 & 5 | 12 | 5 | 12 ^ 5 |
+--------+--------+--------+
|      4 |     13 |      9 |
+--------+--------+--------+
1 row in set (0.00 sec)
```



4. 按位取反运算符

按位取反（~）运算符将给定的值的二进制数逐位进行取反操作，即将1变为0，将0变为1。

```
mysql> SELECT 10 & ~1;
+---------+
| 10 & ~1 |
+---------+
|      10 |
+---------+
1 row in set (0.00 sec)
```

由于按位取反（~）运算符的优先级高于按位与（&）运算符的优先级，所以10 & ~1

首先，对数字1进行按位取反操作，结果除了最低位为0，其他位都为1，然后与10进行按位与操作，结果为10。

5. 按位右移运算符

按位右移（>>）运算符将给定的值的二进制数的所有位右移指定的位数。

右移指定的位数后，右边低位的数值被移出并丢弃，左边高位空出的位置用0补齐。

```
mysql> SELECT 1 >> 2, 4 >> 2;
+--------+--------+
| 1 >> 2 | 4 >> 2 |
+--------+--------+
|      0 |      1 |
+--------+--------+
1 row in set (0.00 sec)
```

1的二进制数为0000 0001，右移2位为0000 0000，对应的十进制数为0。4的二进制数为0000 0100，右移2位为0000 0001，对应的十进制数为1。

6. 按位左移运算符

按位左移（<<）运算符将给定的值的二进制数的所有位左移指定的位数。左移指定的位数后，左边高位的数值被移出并丢弃，右边低位空出的位置用0补齐。

```
mysql> SELECT 1 << 2, 4 << 2;
+--------+--------+
| 1 << 2 | 4 << 2 |
+--------+--------+
|      4 |     16 |
+--------+--------+
1 row in set (0.00 sec)
```

1的二进制数为0000 0001，左移两位为0000 0100，对应的十进制数为4。4的二进制数为0000 0100，左移两位为0001 0000，对应的十进制数为16。

5. 运算符的优先级

优 先 级	运 算 符
1	:=，=（赋值）
2	||，OR，XOR
3	&&，AND
4	NOT
5	BETWEEN，CASE，WHEN，THEN 和 ELSE
6	=（比较运算符），<=>，>=，>，<=，<，<>，!=，IS，LIKE，REGEXP和IN
7	|
8	&
9	<<与>>
10	-和+
11	*，/，DIV，%和MOD
12	^
13	-（负号）和~（按位取反）
14	!
15	()

数字编号越大，优先级越高，优先级高的运算符先进行计算。可以看到，赋值运算符的优先级最低，使用“()”括起来的表达式的优先级最高

拓展：使用正则表达式查询

正则表达式通常被用来检索或替换那些符合某个模式的文本内容，根据指定的匹配模式匹配文本中符合要求的特殊字符串。例如，从一个文本文件中提取电话号码，查找一篇文章中重复的单词或者替换用户输入的某些敏感词语等，这些地方都可以使用正则表达式。正则表达式强大而且灵活，可以应用于非常复杂的查询。

MySQL中使用REGEXP关键字指定正则表达式的字符匹配模式。下表列出了REGEXP操作符中常用字符匹配列表。

选项	说明	例子	匹配值示例

1. 查询以特定字符或字符串开头的记录

字符'^'匹配以特定字符或者字符串开头的文本。

在fruits表中，查询f_name字段以字母'b'开头的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_name REGEXP '^b';
```

2. 查询以特定字符或字符串结尾的记录

字符'$'匹配以特定字符或者字符串结尾的文本。

在fruits表中，查询f_name字段以字母'y'结尾的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_name REGEXP 'y$';
```

3. 用符号"."来替代字符串中的任意一个字符

字符'.'匹配任意一个字符。

在fruits表中，查询f_name字段值包含字母'a'与'g'且两个字母之间只有一个字母的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_name REGEXP 'a.g';
```

4. 使用"*"和"+"来匹配多个字符

星号'*'匹配前面的字符任意多次，包括0次。加号'+'匹配前面的字符至少一次。

在fruits表中，查询f_name字段值以字母'b'开头且'b'后面出现字母'a'的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_name REGEXP '^ba*';
```

在fruits表中，查询f_name字段值以字母'b'开头且'b'后面出现字母'a'至少一次的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_name REGEXP '^ba+';
```

5. 匹配指定字符串

正则表达式可以匹配指定字符串，只要这个字符串在查询文本中即可，如要匹配多个字符串，多个字符串之间使用分隔符'|'隔开。

在fruits表中，查询f_name字段值包含字符串"on"的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_name REGEXP 'on';
```

在fruits表中，查询f_name字段值包含字符串"on"或者"ap"的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_name REGEXP 'on|ap';
```

之前介绍过，LIKE运算符也可以匹配指定的字符串，但与REGEXP不同，LIKE匹配的字符串如果在文本中间出现，则找不到它，相应的行也不会返回。REGEXP在文本内进行匹配，如果被匹配的字符串在文本中出现，REGEXP将会找到它，相应的行也会被返回。对比结果如下所示。

在fruits表中，使用LIKE运算符查询f_name字段值为"on"的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_name like 'on';
Empty set(0.00 sec)
```

6. 匹配指定字符中的任意一个

方括号"[]"指定一个字符集合，只匹配其中任何一个字符，即为所查找的文本。

在fruits表中，查找f_name字段中包含字母'o'或者't'的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_name REGEXP '[ot]';
```

在fruits表中，查询s_id字段中包含4、5或者6的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE s_id REGEXP '[456]';
```

7. 匹配指定字符以外的字符

"[^字符集合]"匹配不在指定集合中的任何字符。

在fruits表中，查询f_id字段中包含字母a_{e和数字1}2以外字符的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_id REGEXP '[^a-e1-2]';
```

8. 使用{n,}或者{n,m}来指定字符串连续出现的次数

"字符串{n,}"表示至少匹配n次前面的字符；"字符串{n,m}"表示匹配前面的字符串不少于n次，不多于m次。例如，a{2,}表示字母a连续出现至少2次，也可以大于2次；a{2,4}表示字母a连续出现最少2次，最多不能超过4次。

在fruits表中，查询f_name字段值出现字母'x'至少2次的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_name REGEXP 'x{2,}';
```

在fruits表中，查询f_name字段值出现字符串“ba”最少1次、最多3次的记录，SQL语句如下：

```
mysql> SELECT * FROM fruits WHERE f_name REGEXP 'ba{1,3}';
```

课后练习

- **选择工资不在5000到12000的员工的姓名和工资**

```
SELECT last_name,salary
FROM employees
#where salary not between 5000 and 12000;
WHERE salary < 5000 OR salary > 12000;
+-------------+----------+
| last_name   | salary   |
+-------------+----------+
| King        | 24000.00 |
| Kochhar     | 17000.00 |
+-------------+----------+
```

- **选择在20或50号部门工作的员工姓名和部门号**

```
SELECT last_name,department_id
FROM employees
# where department_id in (20,50);
WHERE department_id = 20 OR department_id = 50;
+-------------+---------------+
| last_name   | department_id |
+-------------+---------------+
| Weiss       |            50 |
| Fripp       |            50 |
| Hartstein   |            20 |
| Fay         |            20 |
+-------------+---------------+
```

- **选择公司中没有管理者的 员工姓名及job_id**

```
SELECT last_name,job_id,manager_id
FROM employees
WHERE manager_id IS NULL;

SELECT last_name,job_id,manager_id
FROM employees
WHERE manager_id <=> NULL;
+-----------+---------+------------+
| last_name | job_id  | manager_id |
+-----------+---------+------------+
| King      | AD_PRES |       NULL |
+-----------+---------+------------+
```

- **选择公司中有奖金的员工姓名，工资和奖金级别**

```
SELECT last_name,salary,commission_pct
FROM employees
WHERE commission_pct IS NOT NULL;


SELECT last_name,salary,commission_pct
FROM employees
WHERE NOT commission_pct <=> NULL;
+------------+----------+----------------+
| last_name  | salary   | commission_pct |
+------------+----------+----------------+
| Russell    | 14000.00 |           0.40 |
| Partners   | 13500.00 |           0.30 |
| Johnson    |  6200.00 |           0.10 |
+------------+----------+----------------+
```

- **选择员工姓名的第三个字母是a的员工姓名**

```
SELECT last_name
FROM employees
WHERE last_name LIKE '__a%';
+-----------+
| last_name |
+-----------+
| Grant     |
| Grant     |
| Whalen    |
+-----------+
3 rows in set (0.00 sec)
```

- **选择姓名中有字母a和k的员工姓名**

```
SELECT last_name
FROM employees
WHERE last_name LIKE '%a%k%' OR last_name LIKE '%k%a%';
#where last_name like '%a%' and last_name LIKE '%k%';
+------------+
| last_name  |
+------------+
| Kochhar    |
| Kaufling   |
| Markle     |
| Atkinson   |
| Philtanker |
| Kumar      |
+------------+
6 rows in set (0.00 sec)
```

- **显示出表 employees 表中 first_name 以 'e'结尾的员工信息**

```
SELECT first_name,last_name
FROM employees
WHERE first_name LIKE '%e';
```

```
+------------+-------------+
| first_name | last_name   |
+------------+-------------+
| Bruce      | Ernst       |
| Irene      | Mikkilineni |
| Vance      | Jones       |
+------------+-------------+
9 rows in set (0.00 sec)

SELECT first_name,last_name
FROM employees
WHERE first_name REGEXP 'e$'; # 以e开头的写法：'^e'
+------------+-------------+
| first_name | last_name   |
+------------+-------------+
| Bruce      | Ernst       |
| Irene      | Mikkilineni |
| Vance      | Jones       |
+------------+-------------+
9 rows in set (0.01 sec)
```

- **显示出表 employees 部门编号在 80-100 之间的姓名、工种**

```
SELECT last_name,job_id
FROM employees
#方式1：推荐
WHERE department_id BETWEEN 80 AND 100;
#方式2：推荐，与方式1相同
#where department_id >= 80 and department_id <= 100;
#方式3：仅适用于本题的方式。
#where department_id in (80,90,100);
+------------+------------+
| last_name  | job_id     |
+------------+------------+
| King       | AD_PRES    |
| Kochhar    | AD_VP      |
| Johnson    | SA_REP     |
+------------+------------+
43 rows in set (0.00 sec)
```

- **显示出表 employees 的 manager_id 是 100,101,110 的员工姓名、工资、管理者id**

```
SELECT last_name,salary,manager_id
FROM employees
WHERE manager_id IN (100,101,110);
+-----------+----------+------------+
| last_name | salary   | manager_id |
+-----------+----------+------------+
| Kochhar   | 17000.00 |        100 |
| De Haan   | 17000.00 |        100 |
| Raphaely  | 11000.00 |        100 |
| Greenberg | 12000.00 |        101 |
```

```
| Whalen    |  4400.00 |        101 |
| Mavris    |  6500.00 |        101 |
| Baer      | 10000.00 |        101 |
| Higgins   | 12000.00 |        101 |
+-----------+----------+------------+
19 rows in set (0.00 sec)
```