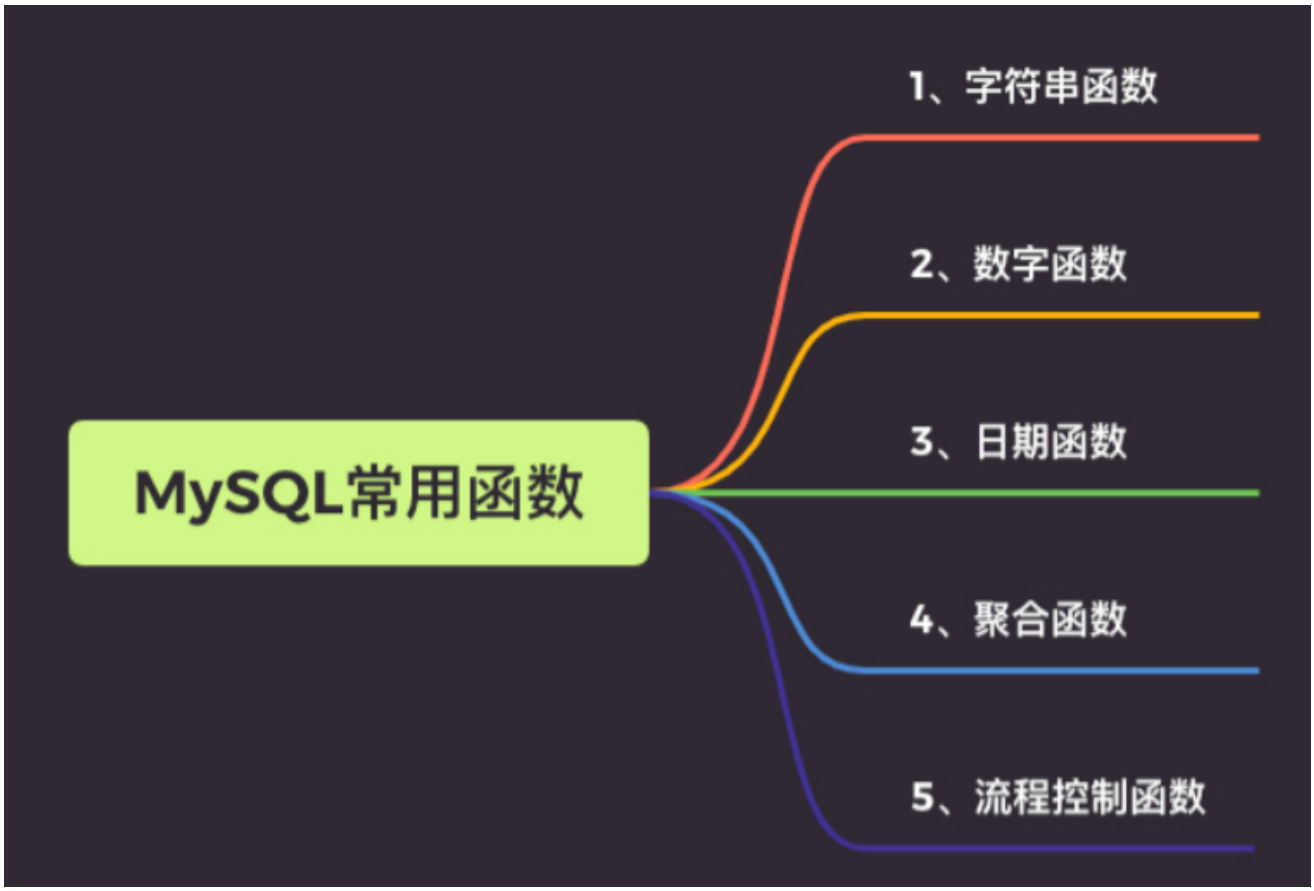
MySQL 常用函数



常见函数

- 字符串函数
- 数字函数
- 日期函数
- 聚合函数
- 流程控制函数

字符串函数

```
concat(s1,s2...,sn)
-- 将s1,s2...,sn连接成字符串，如果该函数中的任何参数为 null，返回结果为 null
concat_ws(sep,s1,s2...,sn)
-- 将s1,s2...,sn连接成字符串,并用sep字符间隔
substr(str,start,len)
-- 从start位置开始截取字符串，len表示要截取的长度;
lcase(str)或lower(str)
-- 将字符串str转换为小写形式
upper(str)和ucase(str)
-- 将字符串转换为大写形式
length(s)          -- 返回字符串str中的字符数

trim(str)          -- 去除字符串首部和尾部的所有空格



left(str,x)        -- 返回字符串str中最左边的x个字符
right(str,x)       -- 返回字符串str中最右边的x个字符
strcmp(s1,s2)      -- 比较字符串s1和s2是否相同 相同返回0，否则-1
```

下面对个别举例

1、concat函数

```
select concat('my  ', 'mongodb', 'es') as
str;
+----------------+
```

```
| str              |
+-----------------+
| my  mongodbes |
+-----------------+
```

如果该函数中的任何参数为 null，返回结果为 null

```
select concat('my  ', null, 'es') as str;
+------+
| str  |
+------+
| null |
+------+
```

2、concat_ws函数

```
select concat_ws('-', 'my  ', 'mongodb',
'es') as str1;
+------------------+
| str1             |
+------------------+
| my  -mongodb-es |
+------------------+
```

如果该函数中的存在参数为 null，则会过滤null

```
select concat_ws('-', 'my  ', null, 'es') as
str1;
+----------+
| str1     |
+----------+
```

```
| my  -es |
+----------+
```

3、substr函数

没有指定len长度：表示从start开始起，截取到字符串末尾。

```
select substr('今天天气真不错',3) as str;
+---------+
| str     |
+---------+
|天气真不错 |
+---------+
```

指定了len长度：表示从start开始起，截取len个长度。

```
select substr('今天天气真不错',3,2) as str;
+------+
| str  |
+------+
| 天气  |
+------+
```

4、trim函数

```
select trim('  今天哈尔滨天气30度  ') as str1;
+--------------+
| str1         |
+--------------+
|今天杭州天气40度|
```

```
+--------------+
```

默认是空格，我们可以指定子串

```
select trim('-' from '---今天哈尔滨天气30度---')
as str1;
+--------------+
| str1         |
+--------------+
|今天哈尔滨天气30度|
+--------------+
```

数字函数

```
round(x,y)          -- 四舍五入,y表示保留小数位(y非必填)
ceil(x)             -- 向上取整，返回>=该参数的最小整数。
floor(x)            -- 向下取整，返回<=该参数的最大整数。
mod(x,y)            -- 返回x/y的模（余数）
greatest(x1,x2,...,xn)
                    -- 返回集合中最大的值
least(x1,x2,...,xn)
                    -- 返回集合中最小的值
rand()              -- 返回0到1内的随机值,可以通过提供一个参数(种子)使rand()随机数生成器生成一个指定的值。
truncate(x,y)       -- 返回数字x截短为y位小数的结果
```

示例：

ROUND(x, y) – 四舍五入，保留y位小数（y非必填）。

```
SELECT ROUND(3.14159, 2); -- 结果为3.14
SELECT ROUND(3.6789);     -- 结果为4（未指定保留小数位，默认为0）
```

CEIL(x) – 向上取整，返回不小于x的最小整数。

```
SELECT CEIL(5.3); -- 结果为6
SELECT CEIL(7.8); -- 结果为8
```

FLOOR(x) – 向下取整，返回不大于x的最大整数。

```
SELECT FLOOR(5.3); -- 结果为5
SELECT FLOOR(7.8); -- 结果为7
```

MOD(x, y) – 返回x除以y的模（余数）。

```
SELECT MOD(10, 3); -- 结果为1
SELECT MOD(15, 6); -- 结果为3
```

GREATEST(x1, x2, ..., xn) – 返回集合中最大的值。

```
SELECT GREATEST(10, 5, 8, 12); -- 结果为12
SELECT GREATEST(1.5, 2.7, 0.8); -- 结果为2.7
```

LEAST(x1, x2, ..., xn) – 返回集合中最小的值。

```
SELECT LEAST(10, 5, 8, 12); -- 结果为5
SELECT LEAST(1.5, 2.7, 0.8); -- 结果为0.8
```

RAND() – 返回0到1内的随机值，可以通过提供一个参数（种子）使rand()随机数生成器生成一个指定的值。

```
SELECT RAND(); -- 返回0到1内的随机值
SELECT RAND(123); -- 返回一个基于种子123的随机值
```

TRUNCATE(x, y) – 返回数字x截短为y位小数的结果。

```
SELECT TRUNCATE(3.14159, 2); -- 结果为3.14
SELECT TRUNCATE(5.6789, 1);  -- 结果为5.6
```

日期函数

序号	格式	功能
1	%Y	年，4 位
2	%y	年，2 位
3	%m	月份(01,02...11,12)
4	%c	月份(1,2...11,12)
5	%d	日 (01,02,31)
6	%e	日(1,2,31)
7	%H	小时(24小时制)
8	%h	小时(12小时制)
9	%i	分钟 (00,01,-59)
10	%s	秒 (00,01,-59)
11	%T	时间,24-小时 (hh:mm:ss)

```
now()                    -- 返回当前的日期和时间
curdate()或current_date()
                         -- 返回当前的日期
curtime()或current_time()
                         -- 返回当前的时间
year(date)      -- 获取年份
```

```
month(date)    -- 获取月份
day(date)      -- 获取日
hour(date)     -- 获取小时
minute(date)   -- 获取分钟
second(date)   -- 获取秒数
str_to_date(str,format)    -- 将日期格式的字符串,转换成指定格式的日期
date_format(date,format)
                  -- 将日期转换为对应的字符串格式

date_add(date,interval expr unit)
                    -- 增加日期时间
date_sub(date,interval expr unit)
                    -- 减少日期时间
dayofweek(date)    -- 返回date所代表的一星期中的第几天(1~7)
dayofmonth(date)   -- 返回date是一个月的第几天(1~31)
dayofyear(date)    -- 返回date是一年的第几天(1~366)
quarter(date)      -- 返回date在一年中的季度(1~4)
unix_timestamp(date)
                    -- 将当前时间转为时间戳
如:select unix_timestamp(now())
from_unixtime(tr) -- 将时间戳转为时间 如:select from_unixtime(unix_timestamp());
select period_diff(200302,199802);
```

```
                                    -- 返回两个日期值之间的差值(月数)
```

示例

1、str_to_date(str,format)

srt:要格式化为日期的字符串(输入字符串)

format:要使用的格式字符串

如果不能按照format解析str，str_to_date函数将返回null

```
-- 2017-01-06 10:20:30
select str_to_date('2017-01-06 10:20:30','%Y-%m-%d %H:%i:%s');
-- 2022-07-06
select str_to_date('2022-07-06 10:20:30','%Y-%m-%d');
-- 2022-05-25
select str_to_date('25,5,2022','%d,%m,%Y');
-- 2022-05-25 11:30:00
select str_to_date('20220525 1130','%Y%m%d %h%i');
-- 2022-05-27 10:40:10
select str_to_date('2022,5,27 10,40,10', '%Y,%m,%d %h,%i,%s');
```

```
-- 2022-05-25
select str_to_date('25,5,2022 extra
characters','%d,%m,%Y');
```

2、date_format(date,format)

使用场景：开始结束时间的查询条件，根据年月日星期分组的查询

```
SELECT
  id,
  DATE_FORMAT(create_time, '%Y-%c-%d')
FROM order
WHERE DATE_FORMAT(create_time, '%Y-%m-%d') >=
'2022-07-15';
```

使用场景：计算年龄

```
-- 计算年龄方式一
SELECT DATE_FORMAT(NOW(), '%Y') -
DATE_FORMAT(Birthday, '%Y') as age from
employee
--计算年龄方式二
SELECT DATE_FORMAT(FROM_DAYS(TO_DAYS(NOW())-
TO_DAYS(create_time)),'%Y')+0 AS age FROM
employee;
```

3、date_add(date,interval expr unit)

作用：增加日期时间

应用场景：当前时间的前一天，前几分钟。 常用于数据统计。
例子：

```
select date_add(now(),interval 1 day) as day;
+---------------------+
| day                 |
+---------------------+
| 2022-07-16 02:23:15 |
+---------------------+
```

其中 Date 表示日期格式，其中就包括：如 2017-12-27，now() 等格式。

expr：表示数量。

unit：表示单位，支持毫秒(microsecond)，秒(second)，小时(hour)，天(day)，周(week)，月(month),年(year)等。

聚合函数

```
avg(x)              -- 返回指定列的平均值
count(x)            -- 返回指定列中非null值的个数
min(x)              -- 返回指定列的最小值
max(x)              -- 返回指定列的最大值
sum(x)              -- 返回指定列的所有值之和
group_concat(x)     -- 返回由属于一组的列值连接组合而成的结果，非常有用
```

group_concat函数在开发中确实很有用，group_concat需要和group by联合使用，用于将某一列的值按指定的分隔符进行拼接，mysql默认的分隔符是逗号。

例如：

```
-- 1、有个部门表，我想看下每个部门一共有多少人
select dept, count(*) from employee group by dept;
-- 2、每个部门的具体人名，我们就可以用group_concat函数
select dept,group_concat(name) from employee group by dept;
-- 3、可以对人名排序
select dept,group_concat(name order by name desc) from employee group by dept;
```

流程控制函数

1、if函数
语法

```
if(test,t,f)   -- 如果test是真，返回t；否则返回f
```

示例

```
select name,age,if(age>60,'老年','不老')  as remark from user
```

```
+------+------+--------+
| name | age  | remark |
+------+------+--------+
| 张三  |   10 | 不老   |
| 李四  |   69 | 老年   |
| 王五  |   72 | 老年   |
| 赵六  |   30 | 不老   |
+------+------+--------+
```

2、ifnull函数
判断值是否为null，是null用指定值填充;

语法

```
ifnull(arg1,arg2)    -- 如果arg1不是空，返回arg1，否则返回arg2
nullif(arg1,arg2)    -- 如果arg1=arg2返回null;否则返回arg1
```

示例

```
select ifnull(1,2) r1, ifnull(null,5) r2, ifnull(2*null,'false') r3;
+----+----+-------+
| r1 | r2 | r3    |
+----+----+-------+
|  1 | 5  | false |
+----+----+-------+
```

```
select nullif(1,1) as r1,nullif('a','b') as
r2,nullif(2+3,4+1) as r2;
+------+------+------+
| r1   | r2   | r2   |
+------+------+------+
| null | a    | null |
+------+------+------+
```

3、case...when函数

case when 有两种写法:

搜索case when，好处是每一次假设都可以指定不同的列
其实只会这一种方式就可以，下面是不同写法而已，功能都能实现

```
case
when <求值表达式> then <表达式1>
when <求值表达式> then <表达式2>
else <表达式>
end
```

<求值表达式> ：一般为字段 【=、>、<、in、等】如 字段 = "1"

<表达式1> ： 一般为字段或者字符串或者数值等。

2）简单case表达式

```
case <表达式>
```

```
when <表达式> then <表达式>
when <表达式> then <表达式>
else <表达式>
end
```

注意

else 可以不写，默认返回null

end 不可以忘记

当一个case子句中有多个判断逻辑时、字段类型需要一致

当一个case子句中有多个判断逻辑时、第一个为真的结果会被输出

这里举个完整的示例，我们先看原始数据

id	age	sex	name	score
1	22	0	周蓉	91
2	18	0	周秉昆	45
3	25	1	周秉义	92
4	22	1	蔡晓光	82
5	24	0	郝冬梅	82
6	17	0	乔春燕	56
7	21	0	郑娟	38
8	18	1	韩德宝	61

需求一:统计下每个人的分数在什么阶段。

```
select id,name,score,
case
 when  score>=90 then '优秀'
 when  score>=80 and score <90 then '良好'
 when  score>=60 and score<80 then '合格'
 when  score<60 then '不合格'
end as '阶段' from t_score;
```

运行结果

id	name	score	阶段
1	周蓉	91	优秀
2	周秉昆	45	不合格
3	周秉义	92	优秀
4	蔡晓光	82	良好
5	郝冬梅	82	良好
6	乔春燕	56	不合格
7	郑娟	38	不合格
8	韩德宝	61	合格

需求2:统计优秀、良好、合格、不合格人数。

```
select
case
 when  score>=90 then '优秀'
 when  score>=80 and score <90 then '良好'
 when  score>=60 and score<80 then '合格'
 when  score<60 then '不合格'
end as '阶段',count(*) as '人数' from t_score
group by case
```

```
 when  score>=90 then '优秀'
 when  score>=80 and score <90 then '良好'
 when  score>=60 and score<80 then '合格'
 when  score<60 then '不合格'
end;

-- 上面sql我们也可以使用别名
select
case
 when  score>=90 then '优秀'
 when  score>=80 and score <90 then '良好'
 when  score>=60 and score<80 then '合格'
 when  score<60 then '不合格'
end as type,count(*) as '人数' from t_score
group by type;
```

运行结果

阶段	人数
优秀	2
不合格	3
良好	2
合格	1

需求3：不使用group by 的聚合来统计优秀、良好、合格、不合格人数。

需求2返回的是4条数据，如果我们想变成一条数据，这样是不是更加方便。

```
select
sum(case when  score>=90 then 1 else 0 end )
as '优秀' ,
sum(case when  score>=80 and score <90  then
1 else 0 end ) as '良好' ,
sum(case when  score>=60 and score<80 then 1
else 0 end ) as '合格' ,
sum(case when  score<60 then 1 else 0 end )
as '不合格'
from t_score;
```

运行结果

优秀	良好	合格	不合格
2	2	1	3