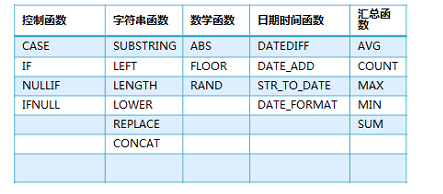
**系统函数和联合查询**

1. **系统函数**

函数是SQL里的关键字，用于对字段里的数据进行操作。函数是一个命令，通常与字段名称或者是表达式联合使用，处理输入的数据并产生结果。

**1、常用函数：**



**1.1控制函数**

**CASE：**

**语法格式：**

CASE *value* WHEN [*compare-value*] THEN *result* [WHEN [*compare-value*] THEN *result* ...] [ELSE *result*] END CASE WHEN [*condition*] THEN *result* [WHEN [*condition*] THEN *result* ...] [ELSE *result*] END

在第一个方案的返回结果中，*value*=*compare-value*。而第二个方案的返回结果是第一种情况的真实结果。如果没有匹配的结果值，则返回结果为ELSE后的结果，如果没有ELSE 部分，则返回值为 NULL。

例如：select s\_name,case s\_sex when 1 then '男' when 2 then '女' else '不详' end from students;

select s\_name,case sex when '男' then '1' when '女' then '2' else '0' end as s\_sex,case age when '18' then '成年人' else '未成年' end as age from students;

注意：也可以如上第二个查询语句进行多个字段转换。

**IF:**

**语法格式：**

IF(*expr1*,*expr2*,*expr3*) 如果*expr1* 是TRUE (*expr1* <> 0 and *expr1* <> NULL)，则 IF()的返回值为*expr2*; 否则返回值则为 *expr3*。IF()的返回值为数字值或字符串值，具体情况视其所在语境而定。

例如：

select s\_name,if(s\_age < 18,'未成年人','成年人') from students;

**IFNULL:**

**语法格式：**

IFNULL(*expr1*,*expr2*)

假如*expr1*不为NULL，则IFNULL()的返回值为*expr1*; 否则其返回值为*expr2*。IFNULL()的返回值是数字或是字符串，具体情况取决于其所使用的语境。

例如：

select s\_name,ifnull(s\_no,0) s\_no from students;

**NULLIF:**

**语法格式：**

NULLIF(*expr1*,*expr2*)

如果*expr1* = *expr2* 成立，那么返回值为NULL，否则返回值为*expr1*。这和CASE

WHEN *expr1* = *expr2* THEN NULL ELSE *expr1* END相同。

select s\_name,nullif(s\_height,0) s\_height from students;

select s\_name,case when s\_height=0 then null else '不详' end from students;

**1.2字符串函数**

**SUBSTRING：**

**语法格式：**

SUBSTRING(*str*,*pos*)==SUBSTRING(*str* FROM *pos*)，

SUBSTRING(*str*,*pos*,*len*) == SUBSTRING(*str* FROM *pos* FOR *len*)

不带有*len* 参数的格式从字符串*str*返回一个子字符串，起始于位置 *pos*。带有*len*参数的格式从字符串*str*返回一个长度同*len*字符相同的子字符串，起始于位置 *pos*。 使用 FROM的格式为标准SQL语法。也可能对*pos*使用一个负值。假若这样，则子字符串的位置起始于字符串结尾的*pos* 字符，而不是字符串的开头位置。

例如：

select substring(s\_name,1) s\_name from students;

select substring('habby new year 2017',-4);>>2017

注意：MySQL中字符串的起始位置是从1开始截取，包括起始位置的字符。

**LEFT、RIGHT：**

LEFT(*str*,*len*) 返回从字符串*str* 开始的*len* 最左字符。

RIGHT(*str*,*len*) 从字符串*str* 开始，返回最右*len* 字符。

例如：

select left(s\_name,1) from students;

select right('hello',2);

**LENGTH、CHAR\_LENGTH、CHARACTER\_LENGTH**

LENGTH(str)，返回字符串str的字节长度，一个中文字符占3个字节（gbk编码）

CHAR\_LENGTH(*str*)，返回值为字符串*str* 的长度，长度的单位为字符。一个多字节字符算作一个单字符。*对于一个*包含五个二字节字符集, LENGTH()返回值为10, 而CHAR\_LENGTH()的返回值为5。

CHARACTER\_LENGTH(*str*) ，CHARACTER\_LENGTH()是CHAR\_LENGTH()的同义词。

例如：

select length(s\_name) s\_name from students;

select length('黑黝黝') from students;>>3

select character\_length('黑黝黝');>>3

select char\_length('黑黝黝');>>3

**LOWER、UPPER：**

**LOWER(*str*)**返回字符串 *str* 以及所有根据最新的字符集映射表变为小写字母的字符(默认为 cp1252 Latin1)。

**UPPER(*str*)** 返回字符串*str，* 以及根据最新字符集映射转化为大写字母的字符 (默认为cp1252 Latin1).

例如：

SELECT LOWER('QUADRATICALLY'); ->> 'quadratically'

SELECT LOWER('quadratically'); ->> 'QUADRATICALLY'

**REPLACE：**

**REPLACE(*str*,*from\_str*,*to\_str*)**返回字符串*str* 以及所有被字符串*to\_str*替代的字符串*from\_str* 。

例如：

mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');

        -> 'WwWwWw.mysql.com'

**CONCAT(*str1*,*str2*,...)：**

返回结果为连接参数产生的字符串。如有任何一个参数为NULL，则返回值为 NULL。或许有一个或多个参数。 如果所有参数均为非二进制字符串，则结果为非二进制字符串。 如果自变量中含有任一二进制字符串，则结果为一个二进制字符串。一个数字参数被转化为与之相等的二进制字符串格式；若要避免这种情况，可使用显式类型cast, 例如： SELECT CONCAT(CAST(int\_col AS CHAR), char\_col);

例如：

mysql> SELECT CONCAT('My', 'S', 'QL');

-> 'MySQL'

mysql> SELECT CONCAT('My', NULL, 'QL');

-> NULL

mysql> SELECT CONCAT(14.3);

-> '14.3'

**1.3数学函数**

若发生错误，所有数学函数会返回NULL。

**ABS(*X*)**，返回*X* 的绝对值。

例如：

mysql> SELECT ABS(2); -> 2

mysql> SELECT ABS(-32); -> 32

FLOOR(*X*)，返回不大于*X*的最大整数值 。

例如：

mysql> SELECT FLOOR(1.23); -> 1

mysql> SELECT FLOOR(-1.23); -> -2

注意，返回值会被转化为一个 BIGINT。

**CEILING(*X*)、CEIL(*X*)**

返回不小于*X* 的最小整数值。

例如：

mysql> SELECT CEILING(1.23); -> 2

mysql> SELECT CEIL(-1.23); -> -1

这两个函数的意义相同。注意返回值会被转化为一个BIGINT。

**RAND()、 RAND(*N*)**

返回一个随机浮点值*v* ，范围在0到1 之间(即, 其范围为 0 ≤ *v* ≤ 1.0)。若已指定一个整数参数*N* ，则它被用作种子值，用来产生重复序列。

例如：

mysql> SELECT RAND();

        -> 0.9233482386203

mysql> SELECT RAND(20);

        -> 0.15888261251047

mysql> SELECT RAND(20);

        -> 0.15888261251047

mysql> SELECT RAND();

        -> 0.63553050033332

若要在*i* ≤ *R* ≤ *j* 这个范围得到一个随机整数*R* 需要用到表达式 FLOOR(*i* + RAND() \* (*j* – *i* + 1))。例如，若要在7到12的范围（包括7和12）内得到一个随机整数,可使用以下语句： select floor(7+rand(12-7+1));

**1.4日期时间函数**

**DATEDIFF(*expr*,*expr2*)**

DATEDIFF()返回起始时间*expr*和结束时间*expr2*之间的天数。*Expr*和*expr2*为日期或date-and-time 表达式。计算中只用到这些值的日期部分。

例如：

mysql> SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-30');

        -> 1

mysql> SELECT DATEDIFF('1997-11-30 23:59:59','1997-12-31');

        -> -31

mysql> select datediff('2016-12-31 11:39:43','2015-11-30 11:39:43');>> 397

**DATE\_ADD(*date*,INTERVAL *expr* *type*)，用于date日期时间往后推**

**DATE\_SUB(*date*,INTERVAL *expr* *type*) ，用于****date日期时间前推**

这些函数执行日期运算。*date* 是一个DATETIME或DATE值，用来指定起始时间。 *expr* 是一个表达式，用来指定从起始日期添加或减去的时间间隔值。*Expr*是一个字符串;对于负值的时间间隔，它可以以一个 ‘-’开头。 *type*为关键词，它指示了表达式被解释的方式。 关键词INTERVA及 *type* 分类符均不区分大小写。

例如：

select adddate('2016-12-31',interval 1 day);

>> 2017-01-01

等同于：

select '2016-12-31' - interval 1 day; >> 2017-01-01

select adddate(now(),interval 1 month);

>> 2017-01-31 11:52:00

select date\_add('2016-12-31 22:14:57',interval -1 minute);

>> 2016-12-31 22:13:57

等同于：

select '2016-12-31 22:14:57' - interval 1 minute; >> 2016-12-31 22:13:57

select date\_sub('2016-12-31 22:08:24',interval 1 minute);

>> 2016-12-31 22:07:24

select date\_sub('2016-12-31',interval 1 minute);

>> 2016-12-30 23:59:00

**STR\_TO\_DATE(*****str*,*format*)**

这是DATE\_FORMAT()函数的倒转。它获取一个字符串*str* 和一个格式字符串*format*。若格式字符串包含日期和时间部分，则STR\_TO\_DATE()返回一个DATETIME 值，若该字符串只包含日期部分或时间部分，则返回一个DATE或TIME值。

例如：

select str\_to\_date('2016/12/31','%Y/%m/%d');>>2016-12-31

select str\_to\_date('2016/12/31','%Y-%m/%d');>>null

select str\_to\_date('2016/12/31 13:03:53','%Y/%m/%d %H:%i:%s');

>> 2016-12-31 13:03:53

select str\_to\_date('2016/12/31 13:03:53','%y/%m/%d %H:%i:%s');

>>null

select str\_to\_date('2016/02)31','%Y/%m)%d');

>> 2016-02-31

注意：要转换的字符串日期或者时间必须格式和字符串的日期时间格式一样，否则返回的是null值。

**DATE\_FORMAT(*****date*,*format*)**

格式化字符串日期时间***date***成指定格式***format***的日期时间类型。

以下说明符可用在 *format* 字符串中：

|  |  |
| --- | --- |
| **说明符** | **说明** |
| %a | 工作日的缩写名称  (Sun..Sat) |
| %b | 月份的缩写名称  (Jan..Dec) |
| %c | 月份，数字形式(0..12) |
| %D | 带有英语后缀的该月日期  (0th, 1st, 2nd, 3rd, ...) |
| %d | 该月日期, 数字形式 (00..31) |
| %e | 该月日期, 数字形式(0..31) |
| %f | 微秒 (000000..999999) |
| %H | 小时(00..23) |
| %h | 小时(01..12) |
| %I | 小时 (01..12) |
| %i | 分钟,数字形式 (00..59) |
| %j | 一年中的天数 (001..366) |
| %k | 小时 (0..23) |
| %l | 小时 (1..12) |
| %M | 月份名称 (January..December) |
| %m | 月份, 数字形式 (00..12) |
| %p | 上午（AM）或下午（ PM） |
| %r | 时间 , 12小时制 (小时hh:分钟mm:秒数ss 后加 AM或PM) |
| %S | 秒 (00..59) |
| %s | 秒 (00..59) |
| %T | 时间 , 24小时制 (小时hh:分钟mm:秒数ss) |
| %U | 周 (00..53), 其中周日为每周的第一天 |
| %u | 周 (00..53), 其中周一为每周的第一天 |
| %V | 周 (01..53), 其中周日为每周的第一天 ; 和 %X同时使用 |
| %v | 周 (01..53), 其中周一为每周的第一天 ; 和 %x同时使用 |
| %W | 工作日名称 (周日..周六) |
| %w | 一周中的每日 (0=周日..6=周六) |
| %X | 该周的年份，其中周日为每周的第一天, 数字形式,4位数;和%V同时使用 |
| %x | 该周的年份，其中周一为每周的第一天, 数字形式,4位数;和%v同时使用 |
| %Y | 年份, 数字形式,4位数 |
| %y | 年份, 数字形式 (2位数) |
| %% | ‘%’文字字符 |

注意，‘%’字符要求在格式指定符之前。月份和日期说明符的范围从零开始，原因是MySQL允许存储诸如 '2004-00-00'的不完全日期。

例如：

select Date\_format(‘2016-12-31’,'%Y-%m-%d');>>2016-12-31

select Date\_format('2016/12/31 13:12:23','%Y-%m-%d %H:%i:%s');

>> 2016-12-31 13:12:23

select Date\_format('2016-12-31 22:23:00','%W %M %Y');

>> Saturday December 2016

select date\_format('2017/01/01 11:24:34','%Y-%m-%d %H-%i-%s');

>>2017-01-01 11-24-34

select date\_format('2017/00/00 11:24:34','%Y-%m-%d');

>>2017-00-00

1. **联合查询**

**1.左连接：**以left 左边表为主查询数据

**2.右连接：**以right 右边表为主查询数据

**3.内连接：**典型的连接运算，使用像 = 或 <> 之类的比较运算符）。包括相等连接和自然连接。内连接使用比较运算符根据每个表共有的列的值匹配两个表中的行。

**左连接**

左连接的结果集包括LEFT子句中指定的左表的所有行，而不仅仅是连接列所匹配的行。如果左表的某行在右表中没有匹配行，则在相关联的结果集行中右表的所有选择列表列均为空值。

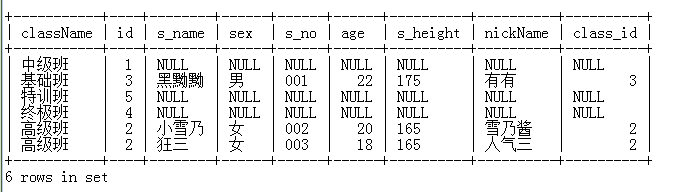
**LEFT JOIN**

**语法：**

select A.\* ,B.\* from A left join B on A.colName = B.colName

例如：

mysql>select \* from class c left join students s on c.id = s.class\_id;

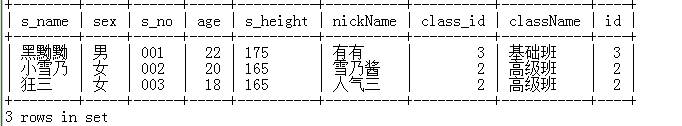


mysql> select \* from class c left join students s on s.classs\_id = c.id;

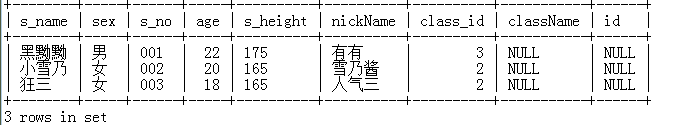
1054 - Unknown column 's.classs\_id' in 'on clause'

注意：以上左连接的on关键字后面的字段，必须是左表的字段在等号左边，右边的字段在等号右边，否则报错。

mysql> select \* from students s left join class c on s.class\_id=c.id;



mysql> select \* from students s left join class c on s.s\_height=c.id;



注意：以上左连接查询必须关联的是两个表中有关系的字段，否则查询不正确。

**右连接**

右连接是左连接的反向连接。将返回右表的所有行。如果右表的某行在左表中没有匹配行，则将为左表返回空值。

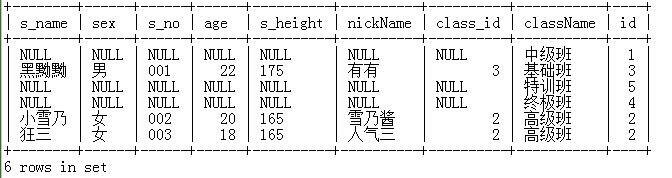
**RIGHT JOIN**

**语法：**

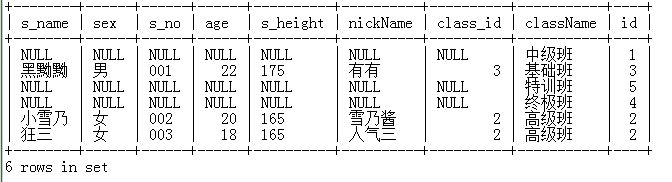
select A.\* ,B.\* from A right join B on A.colName = B.colName;

例如：

mysql> select \* from students s right join class c on c.id=s.class\_id;

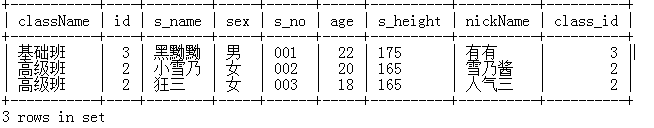


mysql> select \* from students s right join class c on s.class\_id=c.id;



注意：右连接查询中的on关键字后面的字段顺序没有先后之分。

mysql> select \* from class c right join students s on s.class\_id = c.id;



**内连接**

把两个表相关联的数据展示出来，相当于求两个集合的交集。

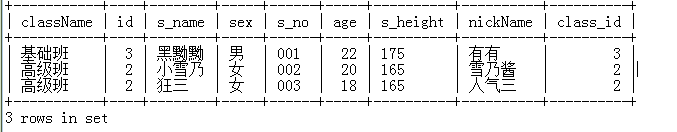
**INNER JOIN**

**语法**：

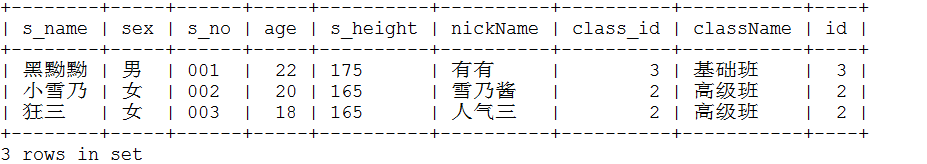
select A.\* ,B.\* from A inner join B on A.colName = B.colName。

例如：

mysql> select \* from class c inner join students s on s.class\_id = c.id;



select students.\*,class.\* from class inner join students on students.class\_id = class.id;



1. **组合查询**

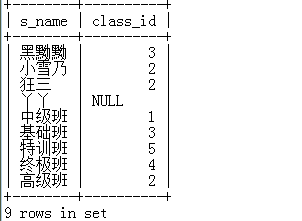
**Union [all]**

Union操作符可以组合两个或多个的select语句的结果，**不包含重复项**。换句话说，如果某行的输出存在于一个查询结果里，那么其他查询结果同一行的记录就不会再输出了。在使用union操作符时，每个select语句里必须选择**相同数量的字段**、**同样的数量的字段表达式**、**同样的数据类型**、**同样的次序**（但长度不必一样）。

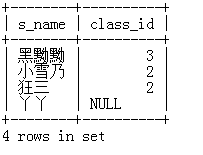
**如果需要不去除重复项，则加上all关键字。**

例如：

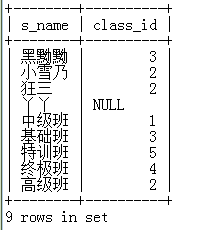
mysql> select s\_name,class\_id from students union select \* from class;



mysql> select s\_name,class\_id from students union select s\_name,class\_id from students;



mysql> select s\_name,class\_id from students union select s\_name,class\_id from students union select \* from class;



mysql> select s\_name,class\_id from students union all select s\_name,class\_id from students;

