SQL 集合运算及常用函数

# SQL 函数

## 一、字符转换函数

### ASCII ()

返回字符表达式最左端字符的ASCII 码值。在ASCII（）函数中，纯数字的字符串可不用‘’括起来，但含其它字符的字符串必须用‘’括起来使用，否则会出错。

|  |
| --- |
| SELECT ASCII('iTalkbb')  ----------  105 |

### CHAR ()

将ASCII 码转换为字符。如果没有输入0 ~ 255 之间的ASCII 码值，CHAR（） 返回NULL 。

|  |
| --- |
| SELECT char(105)  ----------  i |

### LOWER ()

LOWER()将字符串全部转为小写；

|  |
| --- |
| SELECT **lower**('iTalkbb'),**upper**('iTalkbb')  ---------- ----------  italkbb ITALKBB |

### UPPER ()

UPPER()将字符串全部转为大写。

|  |
| --- |
| SELECT **lower**('iTalkbb'),**upper**('iTalkbb')  ---------- ----------  italkbb ITALKBB |

### STR ()

把数值型数据转换为字符型数据。

STR (<float\_expression>[，length[， <decimal>]])

length 指定返回的字符串的长度，decimal 指定返回的小数位数。如果没有指定长度，缺省的length 值为10， decimal 缺省值为0。

当length 或者decimal 为负值时，返回NULL；

当length 小于小数点左边（包括符号位）的位数时，返回length 个\*；

先服从length ，再取decimal ；

当返回的字符串位数小于length ，左边补足空格。

|  |
| --- |
| SELECT **STR**(123.45, 6, 1);  ----------  123.5 |

## 二、去空格函数

### LTRIM ()

把字符串头部的空格去掉。

|  |
| --- |
| SELECT **ltrim**(' iTalkBB')  ----------  iTalkBB |

### RTRIM ()

把字符串尾部的空格去掉。

|  |
| --- |
| SELECT **rtrim**('iTalkBB ')  ----------  iTalkBB |

## 三、取子串函数

### LEFT ()

LEFT (<character\_expression>， <integer\_expression>)

返回character\_expression 左起 integer\_expression 个字符。

|  |
| --- |
| SELECT **left**('iTalkbb',2)  ----------  iT |

### RIGHT ()

RIGHT (<character\_expression>， <integer\_expression>)

返回character\_expression 右起 integer\_expression 个字符。

|  |
| --- |
| SELECT **RIGHT**('iTalkbb',2)  ----------  bb |

### SUBSTRING ()

SUBSTRING (<expression>， <starting\_ position>， length)

返回从字符串左边第starting\_ position 个字符起length个字符的部分。

|  |
| --- |
| SELECT **SUBSTRING**('iTalkbb',2,4)  ----------  Talk |

## 四、字符串比较函数

### CHARINDEX ()

返回字符串中某个指定的子串出现的开始位置。

CHARINDEX (<’substring\_expression’>， <expression>)

其中substring \_expression 是所要查找的字符表达式，expression 可为字符串也可为列名表达式。如果没有发现子串，则返回0 值。

此函数不能用于TEXT 和IMAGE 数据类型。

|  |
| --- |
| SELECT **CHARINDEX**('a','italkbb')  ----------  3  SELECT **CHARINDEX**('Hooyes’,’ ItalkBB')  ----------  0 |

### PATINDEX ()

返回字符串中某个指定的子串出现的开始位置。

PATINDEX (<’%substring \_expression%’>， <column\_ name>)其中子串表达式前后必须有百分号“%”否则返回值为0。

与CHARINDEX 函数不同的是，PATINDEX函数的子串中可以使用通配符，且此函数可用于CHAR、 VARCHAR 和TEXT 数据类型。

|  |
| --- |
| SELECT **PATINDEX**('AB%','1ABCD')  ----------  0  SELECT **PATINDEX**('%AB','1ABCD')  ----------  0  SELECT **CHARINDEX**('AB','1ABCD')  ----------  2  SELECT **PATINDEX**('%[B,C,0-9]%','XYZABC123')  ----------  5 |

## 五、字符串操作函数

### QUOTENAME ()

返回被特定字符括起来的字符串。

QUOTENAME (<’character\_expression’>[， quote\_ character]) 其中quote\_ character 标明括字符串所用的字符，缺省值为“[]”。

|  |
| --- |
| SELECT **QUOTENAME**('iTalkbb')  ----------  [iTalkbb]  SELECT **QUOTENAME**('iTalkbb','<>')  ----------  <iTalkbb> |

### REPLICATE ()

返回一个重复character\_expression 指定次数的字符串。

REPLICATE (character\_expression integer\_expression) 如果integer\_expression 值为负值，则返回NULL 。

|  |
| --- |
| SELECT **REPLICATE**('iTalkBB',2)  --------------  iTalkBBiTalkBB |

### REVERSE ()

将指定的字符串的字符排列顺序颠倒。

REVERSE (<character\_expression>) 其中character\_expression 可以是字符串、常数或一个列的值。

|  |
| --- |
| SELECT **REVERSE**('iTalkBB')  ----------  BBklaTi |

### REPLACE ()

返回被替换了指定子串的字符串。

REPLACE (<string\_expression1>， <string\_expression2>， <string\_expression3>) 用string\_expression3 替换在string\_expression1 中的子串string\_expression2。

|  |
| --- |
| SELECT **REPLACE**('iTalkBB','BB','MOBO')  ----------  iTalkMOBO |

### SPACE ()

返回一个有指定长度的空白字符串。

SPACE (<integer\_expression>) 如果integer\_expression 值为负值，则返回NULL 。

|  |
| --- |
| SELECT 'i'+**SPACE**(2)+'am'  ----------  i am |

### STUFF ()

用另一子串替换字符串指定位置、长度的子串。

STUFF (<character\_expression1>， <start\_ position>， <length>，<character\_expression2>)

如果起始位置为负或长度值为负，或者起始位置大于character\_expression1 的长度，则返回NULL 值。

如果length 长度大于character\_expression1 中 start\_ position 以右的长度，则character\_expression1 只保留首字符。

|  |
| --- |
| SELECT **STUFF**('www.italkbb.com',5,7,'mobo')  ------------  www.mobo.com  例1：查询信用卡的表，要求只显示前两位和后4位，中间用星号显示。  SELECT **STUFF**(CREDIT\_CARD,3,**LEN**(CREDIT\_CARD)-6 ,**REPLICATE**('\*',**LEN**(CREDIT\_CARD)-6)) FROM ACCOUNT\_CC  ------------------  40\*\*\*\*\*\*\*\*\*\*9900  40\*\*\*\*\*\*\*\*\*\*0130  40\*\*\*\*\*\*\*\*\*\*2623  例2：查询用户Email 要示Email只显示前1位和@后面的内容。  SELECT top 100 **STUFF**(EMAIL,2,charindex('@',EMAIL)-2,**REPLICATE**('\*',charindex('@',EMAIL)-2)) from MEMBER  order by acctid  --------------------------  t\*\*\*\*\*\*\*\*\*\*\*@yahoo.com  c\*\*\*\*\*\*\*@hotmail.com  w\*\*\*\*\*@comcast.net  t\*\*\*@yahoo.com  a\*\*\*\*\*\*\*\*\*\*@hotmail.com  l\*\*\*@gmu.edu  y\*\*\*\*\*\*@gmail.com  b\*\*\*\*\*\*\*@gmail.com |

### LEN ()

LEN ( string\_expression ) 计算字符串长度

### DATALENGTH ()

DataLength() 函数返回的是字符串字节的长度，包含后缀空格。而Len()函数返回的是字符串的字符长度，不包含后缀的空格。

|  |
| --- |
| SELECT **LEN**('iTalkBB '),**DATALENGTH**('iTalkBB ')  column1 column2  ---------- ----------  7 11 |

## 六、数据类型转换函数

### CAST ()

CAST (expression AS data\_type [ (length ) ] )

### CONVERT ()

CONVERT (<data\_ type>[ length ]， <expression> [， style])

1）data\_type为SQL Server系统定义的数据类型，用户自定义的数据类型不能在此使用。

2）length用于指定数据的长度，缺省值为30。

3）把CHAR或VARCHAR类型转换为诸如INT或SAMLLINT这样的INTEGER类型、结果必须是带正号或负号的数值。

4）TEXT类型到CHAR或VARCHAR类型转换最多为8000个字符，即CHAR或VARCHAR数据类型是最大长度。

5）IMAGE类型存储的数据转换到BINARY或VARBINARY类型，最多为8000个字符。

6）把整数值转换为MONEY或SMALLMONEY类型，按定义的国家的货币单位来处理，如人民币、美元、英镑等。

7）BIT类型的转换把非零值转换为1，并仍以BIT类型存储。

8）试图转换到不同长度的数据类型，会截短转换值并在转换值后显示“+”，以标识发生了这种截断。

9）用CONVERT()函数的style 选项能以不同的格式显示日期和时间。style 是将DATATIME 和SMALLDATETIME 数据转换为字符串时所选用的由SQL Server 系统提供的转换样式编号，不同的样式编号有不同的输出格式。

## 七、日期函数

### DAY ()

Day (date\_expression)

返回date\_expression中的日期值

|  |
| --- |
| SELECT **DAY**(**getdate**()) |

### MONTH ()

Month (date\_expression)

返回date\_expression中的月份值

|  |
| --- |
| SELECT **MONTH**(**getdate**()) |

### YEAR ()

Year (date\_expression)

返回date\_expression中的年份值

|  |
| --- |
| SELECT **YEAR**(**getdate**()) |

### DATEADD ()

DATEADD (<datepart>， <number>， <date>)

返回指定日期date 加上指定的额外日期间隔number 产生的新日期。

**Datepart** 参数是与 integernumber 相加的 date 部分。下表列出了所有有效的 datepart 参数。用户定义的变量等效项是无效的。

|  |  |  |
| --- | --- | --- |
| Datepart | 缩写 | 备注 |
| Year | yy,yyyy | 年 |
| Quarter | qq,q | 季度 |
| Month | mm,m | 月 |
| Dayofyear | dy,y | 年中的日 |
| Day | dd,d | 日 |
| Week | wk,ww | 周 |
| Weekday | dw,w | 星期 |
| Hour | hh | 小时 |
| Minute | mi,n | 分钟 |
| Second | ss,s | 秒 |
| Millisecond | ms | 豪秒 |
| Microsecond | mcs | 微秒 |
| Nanosecond | ns | 纳秒 |

|  |
| --- |
| SELECT top 10 **dateadd**(day,1,mod\_date) from account\_extension order by mod\_date desc |

### DATEDIFF ()

DATEDIFF (<datepart>， <date1>， <date2>)

返回两个指定日期在datepart 方面的不同之处，即date2 超过date1的差距值，其结果值是一个带有正负号的整数值。

|  |
| --- |
| SELECT **DATEDIFF**(**day**, **getdate**(),**dateadd**(**day**,2,**getdate**()));  ----------  2 |

### DATENAME ()

DATENAME (<datepart>， <date>)

以字符串的形式返回日期的指定部分此部分。由datepart 来指定。

|  |
| --- |
| SELECT **DATENAME**(**year**, **getdate**())  ----------  2011 |

### DATEPART ()

DATEPART (<datepart>， <date>)

以整数值的形式返回日期的指定部分。此部分由datepart 来指定。

DATEPART (dd， date) 等同于DAY (date)

DATEPART (mm， date) 等同于MONTH (date)

DATEPART (yy， date) 等同于YEAR (date)

### GETDATE ()

以DATETIME 的缺省格式返回系统当前的日期和时间。

### CONVERT ()

CONVERT (<data\_ type>[ length ]， <expression> [， style])

SQL Server 中提供多种样式的日期格式化函数

常用：

Select CONVERT (varchar(100), GETDATE(), 8): 10:57:46

Select CONVERT (varchar(100), GETDATE(), 24): 10:57:47

Select CONVERT (varchar(100), GETDATE(), 108): 10:57:49

Select CONVERT (varchar(100), GETDATE(), 12): 060516

Select CONVERT (varchar(100), GETDATE(), 23): 2006-05-16

其它:

Select CONVERT (varchar(100), GETDATE(), 0): 05 16 2006 10:57AM

Select CONVERT (varchar(100), GETDATE(), 1): 05/16/06

Select CONVERT (varchar(100), GETDATE(), 2): 06.05.16

Select CONVERT (varchar(100), GETDATE(), 3): 16/05/06

Select CONVERT (varchar(100), GETDATE(), 4): 16.05.06

Select CONVERT (varchar(100), GETDATE(), 5): 16-05-06

Select CONVERT(varchar(100), GETDATE(), 6): 16 05 06

Select CONVERT(varchar(100), GETDATE(), 7): 05 16, 06

Select CONVERT(varchar(100), GETDATE(), 8): 10:57:46

Select CONVERT(varchar(100), GETDATE(), 9): 05 16 2006 10:57:46:827AM

Select CONVERT(varchar(100), GETDATE(), 10): 05-16-06

Select CONVERT(varchar(100), GETDATE(), 11): 06/05/16

Select CONVERT(varchar(100), GETDATE(), 12): 060516

Select CONVERT(varchar(100), GETDATE(), 13): 16 05 2006 10:57:46:937

Select CONVERT(varchar(100), GETDATE(), 14): 10:57:46:967

Select CONVERT(varchar(100), GETDATE(), 20): 2006-05-16 10:57:47

Select CONVERT(varchar(100), GETDATE(), 21): 2006-05-16 10:57:47.157

Select CONVERT(varchar(100), GETDATE(), 22): 05/16/06 10:57:47 AM

Select CONVERT(varchar(100), GETDATE(), 23): 2006-05-16

Select CONVERT(varchar(100), GETDATE(), 24): 10:57:47

Select CONVERT(varchar(100), GETDATE(), 25): 2006-05-16 10:57:47.250

Select CONVERT(varchar(100), GETDATE(), 100): 05 16 2006 10:57AM

Select CONVERT(varchar(100), GETDATE(), 101): 05/16/2006

Select CONVERT(varchar(100), GETDATE(), 102): 2006.05.16

Select CONVERT(varchar(100), GETDATE(), 103): 16/05/2006

Select CONVERT(varchar(100), GETDATE(), 104): 16.05.2006

Select CONVERT(varchar(100), GETDATE(), 105): 16-05-2006

Select CONVERT(varchar(100), GETDATE(), 106): 16 05 2006

Select CONVERT(varchar(100), GETDATE(), 107): 05 16, 2006

Select CONVERT(varchar(100), GETDATE(), 108): 10:57:49

Select CONVERT(varchar(100), GETDATE(), 109): 05 16 2006 10:57:49:437AM

Select CONVERT(varchar(100), GETDATE(), 110): 05-16-2006

Select CONVERT(varchar(100), GETDATE(), 111): 2006/05/16

Select CONVERT(varchar(100), GETDATE(), 112): 20060516

Select CONVERT(varchar(100), GETDATE(), 113): 16 05 2006 10:57:49:513

Select CONVERT(varchar(100), GETDATE(), 114): 10:57:49:547

Select CONVERT(varchar(100), GETDATE(), 120): 2006-05-16 10:57:49

Select CONVERT(varchar(100), GETDATE(), 121): 2006-05-16 10:57:49.700

Select CONVERT(varchar(100), GETDATE(), 126): 2006-05-16T10:57:49.827

Select CONVERT(varchar(100), GETDATE(), 130): 18 ???? ?????? 1427 10:57:49:907AM

Select CONVERT(varchar(100), GETDATE(), 131): 18/04/1427 10:57:49:920AM

## 八、统计函数

### AVG ()

返回的平均值

|  |
| --- |
| 例1：求出当前系统中所有计划月费的平均值  SELECT **AVG**(MRC) from calling\_plan  ----------  13.6222 |

### COUNT ()

返回的行数

### FIRST ()

返回第一个值

### LAST ()

返回最后一个值

### MAX ()

返回的最大价值

|  |
| --- |
| SELECT **MAX**(plan\_id) FROM calling\_plan  ----------  9700 |

### MIN ()

返回最小的价值

|  |
| --- |
| SELECT **MIN**(plan\_id) FROM calling\_plan  ----------  -1 |

### TOTAL ()

返回的总和

## 九、数学函数

### ABS ()

abs(numeric\_expr) 求绝对值

|  |
| --- |
| SELECT **ABS**(-100)  ----------  100 |

### CEILING ()

ceiling(numeric\_expr) 取大于等于指定值的最小整数

|  |
| --- |
| SELECT **CEILING**(12.35),**CEILING**(-12.35)  column1 column2  ---------- ----------  13 -12 |

### FLOOR ()

floor(numeric\_expr) 取小于等于指定值得最大整数

|  |
| --- |
| SELECT **FLOOR**(12.35),**FLOOR**(-12.35)  column1 column2  ---------- ----------  12 -13 |

### ROUND ()

round(numeric\_expr,int\_expr) 按int\_expr规定的精度四舍五入

|  |
| --- |
| SELECT **ROUND**(12.545,1),**ROUND**(12.545,2),**ROUND**(12.545,0)  column1 column2 column3  ---------- ---------- ----------  12.5 12.55 13 |

### EXP ()

exp(float\_expr) 取指数，该函数是以e为底的指数函数，即对e的自乘（e是一个参数表达式中常用的数学常数,约等于2.718281828459045）

|  |
| --- |
| SELECT **EXP**(2)  ----------------  7.38905609893065 |

### PI ()

Pi () 数学中的圆周率

|  |
| --- |
| SELECT **PI**()  -----------------  3.141592653589793 |

### POWER ()

power(numeric\_expr, power) 返回numeric\_expr 的power次方

|  |
| --- |
| SELECT **POWER**(4,2)  ----------  16 |

### RAND ()

Rand ([int\_expr]) 返回小于0的正数随机数

### SIGN ()

sign(int\_expr) 根据正数,0,负数,对应返回+1,0,-1

|  |
| --- |
| SELECT **SIGN**(-100),**SIGN**(0),**SIGN**(121)  column1 column2 column3  ---------- ---------- ----------  -1 0 1 |

### SQRT ()

sqrt(float\_expr) 平方根

|  |
| --- |
| SELECT **SQRT**(121)  ----------  11 |

## 十、系统函数

### suser\_name()

用户登录名

### user\_name()

用户在数据库中的名字

### user

用户在数据库中的名字

### show\_role()

对当前用户起作用的规则

### db\_name()

数据库名

### object\_name(obj\_id)

数据库对象名

### col\_name(obj\_id,col\_id)

列名

### col\_length(objname,colname)

列长度

valid\_name(char\_expr)

是否是有效标识符

## 十一、其它函数

### ROW\_NUMBER()

SQL2005

|  |
| --- |
| 例2:  SELECT row\_number() over(order by MRC )  as row\_number,MRC,plan\_id  from calling\_plan  where MRC>0  例2:  SELECT row\_number() over(PARTITION BY AREA order by MRC )  as row\_number,MRC,plan\_id,AREA  from calling\_plan where MRC>0 |

### RANK ()

SQL2005

|  |
| --- |
| 例1：  SELECT rank() over(PARTITION BY AREA order by MRC )  as rank,MRC,plan\_id,AREA  from calling\_plan where MRC>0 |

### DENSE\_RANK()

SQL2005

|  |
| --- |
| SELECT dense\_rank() over(PARTITION BY AREA order by MRC )  as rank,MRC,plan\_id,AREA  from calling\_plan where MRC>0 |

### NTILE ()

这个函数按照ntile(n)中的N,把记录强制分成多少段。

### SOUNDEX ()

SOUNDEX (character\_expression)

SOUNDEX 将字母数字字符串转换成由四个字符组成的代码，用于查找发音相似的词或名称。代码的第一个字符是 character\_expression 的第一个字符，代码的第二个字符到第四个字符是数字。除非元音字母是字符串的第一个字母，否则 character\_expression 中的元音字母将被忽略。可以嵌套字符串函数。

### DIFFERENCE ()

DIFFERENCE (character\_expression, character\_expression)

返回的整数是 SOUNDEX 值中相同字符的个数。返回的值从 0 到 4 不等：0 表示几乎不同或完全不同，4 表示几乎相同或完全相同。

|  |
| --- |
| SELECT **SOUNDEX**('Green'), **SOUNDEX**('Greene'), **DIFFERENCE**('Green’,’ Greene')  column1 column2 column3  ---------- ---------- ----------  G650 G650 4 |

# SQL 集合运算

SQL-3标准中提供了三种对检索结果进行集合运算的命令：并集**UNION**；交集**INTERSECT**；差集**EXCEPT**（在Oracle中叫做 MINUS）。在有些数据库中对此的支持不够充分，如MySql中只有**UNION**，没有其他两种

## UNION 运算符

UNION 运算符通过组合其他两个结果表（例如 TABLE1 和 TABLE2）并消去表中任何重复行而派生出一个结果表。当 ALL 随 UNION 一起使用时（即 UNION ALL），不消除重复行。两种情况下，派生表的每一行不是来自 TABLE1 就是来自 TABLE2。

合并查询结果，如：

SELECT \* FROM students WHERE name like ‘张%’ UNION [ALL] SELECT \* FROM students WHERE name like ‘李%’

## EXCEPT 运算符

EXCEPT 运算符通过包括所有在 TABLE1 中但不在 TABLE2 中的行并消除所有重复行而派生出一个结果表。当 ALL 随 EXCEPT 一起使用时 (EXCEPT ALL)，不消除重复行。

## INTERSECT 运算符

INTERSECT 运算符通过只包括 TABLE1 和 TABLE2 中都有的行并消除所有重复行而派生出一个结果表。当 ALL 随 INTERSECT 一起使用时 (INTERSECT ALL)，不消除重复行。

注：使用运算词的几个查询结果行必须是一致的