南京审计大学

国产数据库

主讲：唐伟

4.4.2 函数

(教材149页扩展阅读)

函数和操作符的功能是类似的，都接受参数并返回结果。神通数据库支持的函数包括：数学函数、字符函数、时间和日期函数、聚集函数、大对象函数等。

函数可以出现在查询语句中，也可以用于表达式。调用函数的格式如下：

function_name(parameters)

其中：参数表中的参数可以0个到多个。当有多个参数时，各参数之间用逗号隔开。

1 数学函数

（1） ABS

说明：返回给定数字表达式的绝对值

语法：ABS (num_expr)

示例：SELECT ABS(1-2);

（2） CEIL

说明：取顶函数。返回大于或等于给定数字表达式的最小整数。

另，CEILING和CEIL功能相同。

语法：CEIL (num_expr)

示例：SELECT CEIL (1.2);

（3）FLOOR

说明：取底函数。返回小于或等于给定数字表达式的最大整数；对 interval day to second 类型，返回值为整数天。

语法：FLOOR (num_expr)

示例：

```
SELECT FLOOR (1.2);

SELECT FLOOR (interval '1 12:34:56' day to second);
```

（4）MOD

说明：求余数（模）。

语法：MOD（num_expr1, num_expr2)

示例：

```
SELECT MOD(22, 10);
```

（5）POWER

说明：返回给定表达式乘指定次方的值。

语法：POWER（num_expr1, num_expr2)

示例：

```
SELECT POWER(2,4);
```

（6）ROUND

说明：返回数字表达式并四舍五入为指定的长度或精度。

语法：ROUND(num_expr [,length])

示例：

```
SELECT ROUND(192.362,1);
SELECT ROUND(192.362, 0), ROUND(192.362);
SELECT ROUND(192.362, -2), ROUND(192.362, -4);
```

（7）TRUNC

说明：截断为指定小数位置的数字；按指定的日期格式截断日期值。

语法：TRUNC(num_expr [,length])

TRUNC(date [, fmt])

示例：

```
SELECT TRUNC(192.362, 1);

SELECT TRUNC(192.362);

SELECT TRUNC(timestamp '2020-10-1 9:8:7.01', 'YY');

SELECT TRUNC(timestamp '2020-10-1 9:8:7.01');
```

2 字符函数

（1）CHARINDEX

说明：返回字符或者字符串在另一个字符串中的起始位置。

语法：CHARINDEX(char_expr1, char_expr2 [,start_location])

示例：

```
SELECT CHARINDEX('数据库','神通数据库');
SELECT CHARINDEX('数据库','神通数据库',3);
SELECT CHARINDEX('数据库','神通数据库',4);
```

（2）CONCAT

说明：字符串连接函数，把字符串连接起来组成一个新的字符串并返回。

语法：CONCAT (char_expr1, char_expr2 [,char_expr3])

示例：

```
SELECT CONCAT('1','2'), CONCAT('1','2','3');
```

（3）LEFT

说明：返回从字符串左边开始指定字符个数的子串。

语法：LEFT(char_expr,num_expr)

示例：

```
SELECT LEFT('12',1);
SELECT LEFT ('神通数据库',2);
```

（4）RIGHT

说明：返回从字符串右边开始指定字符个数的子串，语法与LEFT相同。

（5）LENGTH

说明：求字符串长度（字符数）。

语法：LENGTH(char_expr)

示例：

```
SELECT LENGTH('神通OSCAR');
```

（6）LTRIM

说明：字符串左截断函数。指定一个字符串（被截断的字符串）要删除左边出现的char_expr2中任意字符，并返回删除后的字符串。如果不指定要删除的字符串，则系统将删除起始空格后返回。

语法：LTRIM(char_expr1 [, char_expr2])

示例：

```
SELECT LTRIM('abchelloabc', 'abc');

SELECT LTRIM('abchelloabc', 'he');

SELECT LTRIM('  hello');
```

使用RTRIM和LTRIM函数时的注意事项：参数char_expr2不表示按整个char_expr2字符串进行匹配，而是发现char_expr2中任意的字符均做删除操作。

（7）RTRIM

说明：字符串右截断函数。指定一个字符串（被截断的字符串）要删除右边出现的char_expr2中任意字符，并返回删除后的字符串。如果不指定要删除的字符串，则系统将删除尾随空格后返回。

语法：RTRIM(char_expr1 [, char_expr2])

（8）REPLACE

说明：用一个字符串替换另一个字符串的指定子串，返回被替换后的字符串。

语法：REPLACE(char_expr1 ,char_expr2, char_expr3)

示例：

```
SELECT REPLACE('abcdefcde', 'cde', 'x');

SELECT REPLACE('abcdefcde', 'cde', '');
```

（9）SUBSTR

说明：返回字符表达式从指定位置开始，指定长度的字符。

语法：SUBSTR(char_expr , start_location,num_expr)

示例：

```
SELECT SUBSTR('HELLO',2,3), SUBSTR('HELLO',-2,3);
```

3 日期和时间函数

（1）SYSDATE

说明：返回系统当前时间。

示例：

```
SELECT SYSDATE,  SYSDATE -1;
```

（2）ADD_MONTHS

说明：返回给定日期之前或之后几个月的时间。

语法：ADD_MONTHS(date_expr,num_expr)

示例：

```
SELECT ADD_MONTHS('2020-10-1',-2);
SELECT ADD_MONTHS('2020-10-1',2);
```

（3）MONTHS_BETWEEN

说明：返回给定的两个日期之间的月份差。

语法：MONTHS_BETWEEN(date_expr1, date_expr2）

示例：

```
SELECT MONTHS_BETWEEN(
        '2008-8-8 20:00:00 ', '2022-2-4 20:00:00');
SELECT MONTHS_BETWEEN(sysdate,'1949-10-1');
```

（4）DATE_TRUNC

说明：给定一个日期，截断成指定的精度。

语法：DATE_TRUNC(field, date_expr）

示例：

```
SELECT DATE_TRUNC('month',date '2022-2-4');
SELECT DATE_TRUNC('day',timestamp '2022-2-4 7:8:9');
SELECT DATE_TRUNC('year',interval '2022-2' year(4) to month);
SELECT DATE_TRUNC('day',interval '2 7:8:9' day TO second);
SELECT DATE_TRUNC('minute',interval '2 7:8:9' day TO second);
```

（5）DATENAME

说明：给定一个日期，返回指定的域（整数）。

语法：DATENAME(field, date_expr）

示例：

```
SELECT DATENAME('month',date '2022-2-4');
SELECT DATENAME('day',timestamp '2022-2-4 7:8:9');
SELECT DATENAME('minute',timestamp '2022-2-4 7:8:9');
```

（6）DATEADD

说明：返回指定日期加上一个时间间隔后的新日期值，field域指明了需要加到哪个域上。

语法：DATEADD(field, date_expr,num_expr）

DATEADD(field, num_expr,date_expr）

示例：

```
SELECT DATEADD('day',2,timestamp '2022-2-4 7:8:9');
SELECT DATEADD('day',timestamp '2022-2-4 7:8:9',-2);
SELECT DATEADD('month',2,timestamp '2022-2-4 7:8:9');
SELECT DATEADD('year',2,'2022-2-4');
```

（6）DATEDIFF

说明：计算两个日期之间的间隔。

语法：DATEDIFF(field, date_expr1, date_expr2）

示例：

```
SELECT DATEDIFF('year','2022-12-31', '2023-1-1');
SELECT DATEDIFF('month','2022-12-31', '2023-1-1');
SELECT DATEDIFF('day','2022-12-31', '2023-1-1');
SELECT DATEDIFF('month','2022-12-31 23:59:59', '2023-1-1 00:00:01');
SELECT DATEDIFF('second','2022-12-31 23:59:59', '2023-1-1 00:00:01');
```

另外，YEAR、MONTH、HOUR、MINUTE、SECOND函数分别返回指定日期/时间中的年、月、时、分和秒域，返回值都为INT类型。DAYOFWEEK、DAYOFMONTH、DAYOFYEAR函数分别返回指定日期为一周中的第几天、一个月中的第几天、一年中的第几天，返回值也都为INT类型。如：

```
SELECT YEAR(SYSDATE), MONTH(SYSDATE);

SELECT HOUR(SYSDATE), MINUTE(SYSDATE), SECOND(SYSDATE);

SELECT DAYOFWEEK(SYSDATE);

SELECT DAYOFMONTH(SYSDATE);

SELECT DAYOFYEAR(SYSDATE);
```

其中，DAYOFWEEK返回1 ~7之间的一个整数，代表一周中的第几天，1代表星期日，2代表星期一，依此类推。

4 聚集函数

（1）AVG

说明：计算一组值的平均值，空值将被忽略。

语法：AVG ([ALL | DISTINCT] num_expr)

参数：

- ALL：对所有的值进行聚合函数运算。ALL是默认值
- DISTINCT：只在每个值的唯一实例上进行聚合函数运算，不论该值出现了多少次
- num_expr：必须是数值类型的表达式，不允许使用聚合函数或子查询。

示例：

```
--查询所有客户的平均工资。
SELECT AVG(Cincome)  FROM Client
```

（2）COUNT

说明：计算输入参数中元素的个数，一般用于计算查询结果中的行数。

语法：COUNT({ [[ALL | DISTINCT] expr] | * })

参数：

- ALL：对组中的每一行都计算表达式的值，并返回非空值的数量。ALL是默认值
- DISTINCT：返回组中表达式唯一非空值的数量。
- expr：一个表达式，不允许使用聚合函数或子查询。
- *：返回表或组中的所有行的数量，包括含有空值的行，COUNT(*)不需要表达式参数。

示例：

```
--查询所有的客户数。
SELECT COUNT(*)  FROM Client
```

（3）SUM

说明：返回表达式中所有值的和，忽略空值。

语法：SUM([ALL | DISTINCT] num_expr)

参数：

- ALL：对所有的值进行聚合函数运算。ALL是默认值
- DISTINCT：只在每个值的唯一实例上进行聚合函数运算，不论该值出现了多少次
- num_expr：必须是数字类型的表达式，不允许使用聚合函数或子查询。

示例：

```
--查询所有客户的工资总和。
SELECT SUM(Cincome)  FROM Client
```

（4）MAX

说明：返回表达式中的最大值，忽略空值。

语法：MAX([ALL | DISTINCT] expr)

参数说明：

- ✧ ALL：对所有的值进行聚合函数运算。ALL是默认值
- ✧ DISTINCT：只在每个值的唯一实例上进行聚合函数运算，不论该值出现了多少次
- ✧ expr：一个表达式，不允许使用聚合函数或子查询。

示例：

```
--查询所有客户的最高工资。
SELECT MAX(Cincome)  FROM Client
```

（5）MIN

说明：返回表达式中的最小值，忽略空值。

语法：MIN([ALL | DISTINCT] expr)

参数：

- ✧ ALL：对所有的值进行聚合函数运算。ALL是默认值
- ✧ DISTINCT：只在每个值的唯一实例上进行聚合函数运算，不论该值出现了多少次
- ✧ expr：一个表达式，不允许使用聚合函数或子查询。

示例：

```
--查询所有客户的最低工资。
SELECT MIN(Cincome) FROM Client
```

5 其他函数

（1）TO_CHAR

说明：转换成CHAR类型。

语法：TO_CHAR(expr [,text_expr])

参数：

- ✧ text_expr 指定的格式表达式。对于数值型表达式的转换，每一位数字位都要对应一个0；对于TIMESTAMP类型的转换，其格式应与TIMESTAMP类型数据的书写形式一样。若年月日时分秒之间的间隔用其它的字符来代替，则返回值也一样发生了相应的改变。若间隔符之外的其它格式发生了变化，则会出错。

示例：

```
SELECT TO_CHAR(123.45,'0000.0');
SELECT TO_CHAR(SYSDATE,'yyyy-mm-dd'),TO_CHAR(SYSDATE,'dd hh:mi:ss');
SELECT TO_CHAR(DATE '2020-10-1','mm/dd/yy');
SELECT TO_CHAR(TIMESTAMP '2020-10-1 9:8:7');
```

（2）TO_DATE

说明：将文本型数据转换成DATE类型。

语法：TO_DATE(char_expr [,text_expr])

参数：

- ✧ char_expr：文本型数据，必须是 TIMESTAMP 类型的格式，只是间隔符可以不是“-”。
- ✧ text_expr：指定的格式表达式，应为date类型的格式。

示例：

```
SELECT TO_DATE('2022.02.04', 'YYYY-MM-DD');
SELECT TO_DATE('2022-02%04 11-12-01', 'YY-MM-DD');
SELECT TO_DATE('20-06-01','DD-YY-MM');
SELECT TO_DATE('2020-01-01');
```

（3）TO_NUMBER

说明：将整型或者文本数据转换成 NUMERIC 类型。

语法：TO_ NUMBER (expr ,text_expr)

参数：

- ✧ expr：字符表达式（可以包含数字、符号）或者整型数据。
- ✧ text_expr：指定的格式表达式。当expr为字符型表达式，对于要得到的数字位，在转换格式中的对应位处填入0，反之以任一数字（除了0）或符号代替。

示例：

```
SELECT TO_NUMBER(5/3,'00.00'), TO_NUMBER('5/3','00');
SELECT TO_NUMBER('5/3','000'), TO_NUMBER('5/3','+00');
```

（4）TO_TIMESTAMP

说明：将一个文本类型的数据转换成 TIMESTAMP 类型

语法：TO_ TIMESTAMP (char_expr [, text_expr])

参数：

- ✧ char_expr：文本型数据，必须是 TIMESTAMP 数据类型的格式，只是间隔符可以不是“-”。
- ✧ text_expr：指定的格式表达式。如果不指定格式，则类型转换需要参照 TIMESTAMPFORMAT 参数的格式。

示例：

```
SELECT TO_TIMESTAMP('2020*10-1 10*10a10', 'YYYY-MM-DD HH:MI:SS');

SET TIMESTAMPFORMAT='MM-DD-YYYY HH24:MI:SS.FF';

SELECT TO_TIMESTAMP('10-1*2020 10*10a10.1234 ');
```

补充：CAST

说明：将一种类型的数据转换成另一种类型数据

语法：CAST(expr as data_type)

参数：

- ✧ expr：要被进行类型转换的表达式。
- ✧ data_type：要转换成的数据类型。

示例：

```
SELECT cast('123' as INT);
SELECT CAST('1234.567' as decimal(7,2))+89
```

（5）ISNULL

说明：如果第一个参数不为空，则返回第一个参数值，否则返回第二个参数值，若两个参数都为空，则返回空。当参数为一个时，如果参数值为空，返回真，否则返回假。

语法：ISNULL (expr1 [，expr2])

示例：

```
SELECT ISNULL(2,3);

SELECT ISNULL(NULL,'1');

SELECT ISNULL('1');
```

（6） DECODE

说明：函数相当于一个条件语句 (IF)。它将输入值与函数中的参数列表相比较，根据输入值返回一个对应值。函数的参数列表是由若干输入值及其对应的结果值组成的多组序偶形式。如果未能与任何一个输入轴匹配成功，则返回默认值。

语法：DECODE(expr,serach_expr1,result_expr1

[, serach_expr2,result_expr2, …… ,default])

示例：

```
SELECT DECODE(2, 1, 'a', 2, 'b', 'z');


SELECT*, DECODE(x,101, 'FIRST',102,'SECOND',103, 'THIRD', 'OTHER')
FROM a;
```

（7） ROW_NUMBER

说明：针对SELECT语句返回的每一行，从1开始编号，赋予其连续的编号。

语法：ROW_NUMBER() OVER([PARTITION BY expr1] ORDER BY expr2)

参数：PARTITION BY expr1将记录分区，缺省时，所有记录为一个分区

ORDER BY expr2确定将ROW_NUMBER值分配给分区中的行的顺序

示例：

```
SELECT *, ROW_NUMBER() OVER(ORDER BY ldate) 贷款时间次序
FROM loan;

SELECT *, ROW_NUMBER() OVER(PARTITION BY cno ORDER BY ldate) 贷款时间次序
FROM loan;
```

（8） RANK、DENSE_RANK

说明：RANK和DENSE_RANK函数与ROW_NUMBER函数的功能类似。

ROW_NUMBER函数：排序时，不考虑并列的情况，排序编号不会重复，总数不变

RANK函数：排序时，考虑并列的情况，排序编号会重复，但总数不变

DENSE_RANK：排序时，考虑并列的情况，排序编号会重复，总数会减少（当排序编号有重复时）

示例：

```
SELECT *, ROW_NUMBER() OVER(ORDER BY lamount) FROM loan;

SELECT *, RANK() OVER(ORDER BY lamount) FROM loan;

SELECT *, DENSE_RANK() OVER(ORDER BY lamount) FROM loan;
```

南京审计大学

THANKS