

分析型数据库 AnalyticDB

SQL函数使用指南

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

引擎说明：

- AnalyticDB 目前支持2种计算引擎
- 不同版本默认引擎有差异
- 默认计算引擎尚未切换到MPP引擎情况下，2.6.*版本开始默认引擎可以切换至MPP
--部分运算符需要在SQL语句前增加 `/*+engine=MPP*/` hint提示信息。
--`/*+engine=MPP*/ select` 语句

为更好的理解和使用本教程包含的函数，建议先创建如下测试表及生成简单测试数据

```
CREATE TABLE t_fact_customers (  
  customer_id bigint COMMENT "",  
  customer_name varchar COMMENT "",  
  sex int COMMENT "",  
  age double COMMENT "",  
  birth_day date COMMENT "",  
  phone_number varchar COMMENT "",  
  home_addr varchar COMMENT "",  
  first_login timestamp COMMENT "",  
  PRIMARY KEY (customer_id)  
)  
PARTITION BY HASH KEY (customer_id) PARTITION  
NUM 8  
TABLEGROUP ads_demo  
OPTIONS (UPDATETYPE='realtime')  
COMMENT ""
```

```
INSERT INTO t_fact_customers (customer_id, customer_name, sex, age,  
  birth_day, phone_number, home_addr, first_login)  
VALUES (1, '王小二', 0, 20.1, '1997-10-01', '11126411777', '北京市朝阳区  
望京xxxx', '2017-01-10 10:00:11'), (2, '张大山', 0, 21.4, '1998-02-22',  
'15126411778', '北京市海淀区中关村xxxx', '2016-02-13 12:08:13'), (3, '李  
春梅', 1, 32.6, '1985-11-06', '16126411548', '北京市西城区复兴门xxxx',  
'2014-11-22 23:04:45'), (4, '孙大帅', 0, 43.2, '1974-08-04', '17126415478',  
'北京市东城区王府井xxxx', '2015-09-28 02:03:26'), (5, '韩美美', 1, 23.8,  
'1994-12-18', '18126414478', '北京市海淀区中关村xxxx', '2012-08-14  
09:18:23'), (6, '胡衣衣', 1, 54.9, '1963-05-29', '16126412378', '北京市海淀  
区定慧寺xxxx', '2014-02-09 16:28:42'), (7, '知文君', 0, 25.9, '1992-10-17',  
'12126414578', '北京市东城区东郊胡同xxxx', '2017-06-05 12:38:533'), (8, '  
段宗宝', 0, 46.1, '1971-07-04', '13126423778', '北京市海淀区杏石口xxxx',  
'2014-04-28 15:45:19'), (9, '石小英', 1, 22.3, '1995-04-22', '14126423778',  
'北京市西城区白云路xxxx', '2015-06-23 22:09:25'), (10, '林瑜', 0, 28.6,  
'1989-02-09', '15126423778', '北京市东城区SOHUxxxx', '2016-12-18  
15:23:24');
```

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

函数名	函数表达式-返回值	函数说明
GREATEST	<code>greatest(value1, value2) → [same as input]</code>	返回参数中的最大值。
LEAST	<code>least(value1, value2) → [same as input]</code>	返回参数中的最小值。
ALL	<code>ALL() → true/false</code>	ALL, ANY and SOME 量词可以和比较运算符结合使用
ANY	<code>ANY() →true/false</code>	
SOME	<code>SOME() →true/false</code>	


```
/*+engine=mpp*/  
select customer_id,age, greatest(age,100),least(age,30)  
from t_fact_customers
```

```
/*+engine=mpp*/ SELECT 'hello' = ANY (VALUES 'hello', 'world');
```

```
/*+engine=mpp*/ SELECT 21 < ALL (VALUES 19, 20, 21);
```

```
/*+engine=mpp*/ SELECT 42 >= SOME (SELECT 41 UNION ALL SELECT 42);
```

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

函数名	函数表达式-返回值	函数说明
CASE when	多个条件then返回的数据类型必须一致	支持2种形式的case when语句
DECODE	DECODE(expr, search, result, default)	default必须提供， 否则提示函数不存在
IF	if(condition, true_value) if(condition, true_value, false_value)	计算返回 true_value 如果 condition 为 TRUE， 否则返回 NULL 计算返回 true_value 如果 condition 为 TRUE， 否则计算返回 false_value
IFNULL	nullif(value1, value2)	等同 CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END
COALESCE	coalesce(value[, ...])	返回第一个非 NULL 的 value。 和 CASE 表达式类似, 参数仅在需要的时候计算
NVL2	NVL2(expr1, expr2, expr3)	
TRY	try(expression)	计算返回表达式 expression 如果表达式计算遇到错误则返回 NULL

```
select sex, CASE WHEN sex=0 then 'Man' when sex=1 then 'women' else 'unkown' end as result_name,  
CASE sex WHEN 0 then 'Man' when 1 then 'women' else 'unkown' end as result_name1  
from t_fact_customers  
where customer_id<10
```

```
/*+engine=mpp*/ select sex, IF(sex=0,'man') from t_fact_customers where customer_id<10
```

```
/*+engine=mpp*/ select sex, coalesce(sex,age) from t_fact_customers where customer_id<10
```

```
SELECT NVL2(NULL, 2, 3);  
SELECT NVL2(1, 2, 3);  
select nvl2(age,sex,1) from t_fact_customers order by customer_id;
```

```
SELECT DECODE(1, 1, '1A', 2, '2A', '3A');  
SELECT DECODE(2, 1, '1A', 2, '2A', '3A');  
select DECODE(sex,1,'Man',0,'Women','other') from t_fact_customers order by customer_id;
```

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

函数名	函数表达式-返回值	函数说明
CAST	cast(value AS type) → new type value	显式把value转换到type类型。可用于把字符类型值转变为数值类型，反之亦然。
try_cast	try_cast(value AS type) → new type value	类似 cast(), 但是会在转型失败的时候返回 NULL
typeof(expr)	typeof(expr) → type	返回 expr 表达式的结果类型

```
/*+engine=mpp*/ SELECT cast(123 as double) , try_cast('abc' as bigint) ,typeof('abc');
```

```
/*+engine=mpp*/ select cast(phone_number as bigint) varchar2bigint , cast(sex as VARCHAR ) int2varchar,  
cast(age as VARCHAR) double2varchar  
from t_fact_customers  
order by customer_id
```

```
/*+engine=mpp*/  
select typeof(customer_id),typeof(customer_name), typeof(phone_number),typeof(age),  
typeof(first_login)  
from t_fact_customers limit 10;
```

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

数学函数列表（1）：

函数名	表达式及返回值	函数说明
abs	abs(x) → [same as input]	返回 x 的绝对值
cbrt	cbrt(x) → double	返回 x 的立方根.
ceiling	ceiling(x) → [same as input]	返回 x 的向上取整的数值.
ceil	ceil(x) → [same as input]	ceiling() 的同名方法.
degrees	degrees(x) → double	将角度 x 以弧度转换为度.
e	e() → double	返回欧拉常量.
exp	exp(x) → double	返回 x 的欧拉常量次幂.
floor	floor(x) → [same as input]	返回 x 向下取整的最近整数值
from_base	from_base(string, radix) → bigint	返回 radix 进制的字符串 string 代表的数值
ln	ln(x) → double	返回 x 的自然对数.
log2	log2(x) → double	返回 x 以2为底的对数.
log10	log10(x) → double	返回 x 以10为底的对数.
log	log(x, b) → double	返回 x 以 b 为底的对数.
mod	mod(n, m) → [same as input]	返回 n 除 m 的模数(余数).
pi	pi() → double	返回常量Pi.
power	power(x, p) → double	返回 x 的 p 次幂.
crc32	crc32(varchar) → bigint	返回crc32()值

数学函数列表（2）：

函数名	表达式及返回值	函数说明
pow	pow(x, p) → double	power() 的同名方法.
radians	radians(x) → double	将角度 x 以度为单位转换为弧度.
random	random() → double	返回 0.0 <= x < 1.0 范围内的伪随机数.
rand	rand() → double	random() 的同名方法.
round	round(x) → [same as input]	返回 x 四舍五入后的最近的整数值.
round	round(x, d) → [same as input]	返回 x 四舍五入到 d 位小数位的值.
sign	sign(x) → [same as input]	x 的正负号函数, 即: x 为 0, 返回 0, x 为正, 返回 1, x 为负, 返回 -1. 对于 double 类型参数, 则: x 为 NaN, 返回 NaN, x 为正无穷, 返回 1, x 为负无穷, 返回 -1.
sqrt	sqrt(x) → double	返回 x 的平方根.
truncate	truncate(x) → double	舍弃 x 的小数位, 返回整数值.
to_base	to_base(x, radix) → varchar	返回 x 的 radix 进制表示的字符串.

数学函数列表（3）：

函数名	返回类型	函数说明
acos(x)	double	返回 x 的反余弦
asin(x)	double	返回 x 的正弦
atan(x)	double	返回 x 的正切
atan2(y, x)	double	返回 y / x 的正切
cos(x)	double	返回 x 的余弦值
cosh(x)	double	返回 x 的双曲余弦值
cot(x)	double	返回 x 的余切值
sin(x)	double	返回 x 的正弦值
tan(x)	double	返回 x 的正切值
tanh(x)	double	返回 x 的双曲正切
infinity()	double	返回表示正无穷大的常量
is_finite(x)	boolean	判定 x 是否有限
is_infinite(x)	boolean	判定 x 是否无限
is_nan(x)	boolean	判定 x 是非法数值
nan()	double	返回代表非数值的常量值
remainder(n1,n2)	double	返回n1除以n2的余数
bitand(expr1, expr2)	bigint	返回两个数值型数值在按位进行 AND 运算后的结果

```
SELECT ABS(-32), ACOS(1.0001), ASIN(0.2), ATAN2(PI(),0) ,CEILING(-1.23), CRC32('AnalyticDB'), MOD(34.5,3),  
TRUNCATE(-1.999,1);
```

```
select abs(-1),cbirt(9),ceiling(3.4),ceil(3.6),degrees(34.7),e(),exp(3),floor(3.5)  
from t_fact_customers where customer_id=1 limit 1;
```

```
select from_base('0110', 2),from_base('0110', 8),from_base('00a0', 16), to_base(2,2),to_base(72,8),to_base(160,  
16)  
from t_fact_customers where customer_id=1 limit 1;
```

```
select ln(100),log2(100),log10(100),log(100), mod(10,3),pi(),power(2,3),pow(2,3)  
from t_fact_customers where customer_id=1 limit 1;
```

```
select radians(30), random(), rand(), round(10.339), round(10.339,2) ,truncate(10.339)  
from t_fact_customers where customer_id=1 limit 1;
```

```
/*+engine=mpp*/ select sign(1),sign(0),sign(-1), sqrt(9)  
from t_fact_customers where customer_id=1 limit 1;
```


比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数









JSON函数

地理空间函数

全文检索函数

函数名	表达式及返回值	函数说明
bit_count	bit_count(x, bits) → bigint	返回 x (视为 bits 位的有符号整形) 的 bits 位补码表示中, 为1的位的个数:
bitwise_and	bitwise_and(x, y) → bigint	返回 x 和 y 按位与的补码表示.
bitwise_not	bitwise_not(x) → bigint	返回 x 取反的补码表示.
bitwise_or	bitwise_or(x, y) → bigint	返回 x 和 y 按位或的补码表示


```
SELECT bit_count(9, 64),bitwise_and(1,2),bitwise_not(2),bitwise_or(1,2);
```

 bit_count(9, 64) 	 bitwise_and(1, 2) 	 bitwise_not(2) 	 bitwise_or(1, 2) 
2	0	-3	3

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

数名	函数表达式-返回值	函数说明
length	length(binary) → bigint	返回 binary 字节长度。
to_base64	to_base64(binary) → varchar	将 binary 编码为base64字符串。
from_base64	from_base64(string) → varbinary	将base64编码的 string 解码为二进制数据。
to_base64url	to_base64url(binary) → varchar	使用URL安全字符将 binary 编码为base64字符串。
from_base64url	from_base64url(string) → varbinary	使用URL安全字符，将base64编码的 string 解码为二进制数据。
to_hex	to_hex(binary) → varchar	将 binary 编码为16进制字符串。
from_hex	from_hex(string) → varbinary	将16进制编码的字符串 string 解码为二进制数据
to_big_endian_64	to_big_endian_64(bigint) → varbinary	将bigint编码为大字节序的二进制格式。
from_big_endian_64	from_big_endian_64(binary) → bigint	将一个大字节序的 binary 解码为bigint。
md5	md5(binary) → varbinary	返回 binary md5哈希值。
sha1	sha1(binary) → varbinary	返回 binary sha1哈希值。
sha256	sha256(binary) → varbinar	返回 binary sha256哈希值。
sha512	sha512(binary) → varbinary	返回 binary sha512哈希值
xxhash64	xxhash64(binary) → varbinary	返回 binary 的xxhash64哈希值

```
select  
xxhash64(to_big_endian_64(1990)),  
sha512(to_big_endian_64(1990)),  
sha256(to_big_endian_64(1990)),  
md5(to_big_endian_64(1990)),  
from_big_endian_64(to_big_endian_64(1990)),  
to_big_endian_64(1990),  
to_hex(from_hex('09')),  
from_hex('09')  
from t_fact_customers  
where customer_id=1 limit 1;
```

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

函数名/运算符	函数表达式-返回值	函数说明
	String string	两个字符串连接
concat	concat(string1, ..., stringN) → varchar	字符串连接操作,与标准SQL的连接运算符 ()功能相同。

使用示例

```
select 'AAA' || 'BBB', concat('AAA','BBB','CCC');
```


函数名	函数表达式-返回值	函数说明
chr	chr(n) → varchar	返回下标为 n 位置的字符。如chr(65)返回：'A'
ASCII	ASCII(string)->bigint	返回字符或者字符串最左边字符对应的ASCII值
char	char(N,N,...)->varchar	返回每个数字代表的字符形成的字符串
concat	concat(string1, ..., stringN) → varchar	字符串连接操作,与标准SQL的连接运算符 ()功能相同。
length	length(string) → bigint	返回字符串 string 长度。
lower	lower(string) → varchar	转换字符串 string 为小写。
lcase	lcase(string)->varchar	同lower
upper	upper(string) → varchar	转换字符串为大写
ucase	ucase(string)->varchar	同upper
lpad	lpad(string, size, padstring) → varchar	将字符串 string 左边拼接 padstring 直到长度达到 size 并返回填充后的字符串。如果 size 比 string 长度小，则截断。 size 不能为负数， padstring 必须非空。
rpadd	rpadd(string, size, padstring) → varchar	将字符串 string 右边拼接 padstring 直到长度达到 size ,返回填充后的字符串。如果 size 比 string 长度小，则截断。 size 不能为负数， padstring 必须非空。
trim	trim(string) → varchar	删除字符串 string 前后的空格。
ltrim	ltrim(string) → varchar	删除字符串所有前导空格。
rtrim	rtrim(string) → varchar	删除字符串 string 右边所有空格。

函数名	函数表达式-返回值	函数说明
replace	replace(string, search) → varchar	删除字符串 string 中的所有子串 search。
replace	replace(string, search, replace) → varchar	将字符串 string 中所有子串 search 替换为 replace。
reverse	reverse(string) → varchar	将字符串 string 逆序后返回。
split	split(string, delimiter) → array<varchar>	将字符串 string 按分隔符 delimiter 进行分隔，并返回数组。
split	split(string, delimiter, limit) → array<varchar>	将字符串 string 按分隔符 delimiter 分隔，并返回按 limit 大小限制的数组。数组中的最后一个元素包含字符串中的所有剩余内容。limit 必须是正数。
split_part	split_part(string, delimiter, index) → varchar	将字符串 string 按分隔符 delimiter 分隔，并返回分隔后数组下标为 index 的子串 index 以 1 开头，如果大于字段数则返回null。
strpos	strpos(string, substring) → bigint	返回字符串中子字符串的第一次出现的起始位置。以 1 开始 未找到则返回 0
substr	substr(string, start) → varchar	返回 start 位置开始到字符串结束。位置从 1 开始。如果 start 为负数，则起始位置代表从字符串的末尾开始倒数。
substr	substr(string, start, length) → varchar	返回 start 位置开始长度为 length 的子串，位置从 1 开始。如果 start 为负数，则起始位置代表从字符串的末尾开始倒数。
substring	substring	同substr
mid	MID(string, start, length)	同substr(string, start, length)
repeat	REPEAT(string,int)->string	返回字符串重复多次的字符串
initcap	INITCAP(string)->string	返回输入字符串首字符转大写
translate	TRANSLATE(expr, from_string, to_string)	将expr字符串中，符合from_string的字符，并替换为to_string
instr	INSTR(string, substring)	返回string中匹配substring的第一个位置信息
instr	INSTR(string, substring, position)	返回string中匹配substring的位置信息，从string字符的第position位置开始搜索
instr	INSTR(string, substring, position, occurence)	返回string中匹配substring的位置信息，从string字符的第position位置开始搜索，第occurence次出现的位置
hex	hex(bigint)->hexadecimal string	返回16进制字符串

```
/*+engine=mpp*/ select chr(65),chr(66),concat('a','b','c','123'),length('abc')  
, lower('ABC'),upper('abc'),lpad('ab',5,'1'),rpad('ab',5,'1') ;
```

```
/*+engine=mpp*/ select trim(' abc '),ltrim(' abc'),rtrim('abc '), replace('abcccc','c'),replace('abcccc','c','1'),revers  
e('cba') ;
```

```
select strpos('ads','s'), strpos('ads','p'), substr('ads server',1),substr('ads server',5),substr('ads server',-6), substr  
( 'ads server',1,3),substr('ads server',5,6),substr('ads server',-6,6) ;
```

```
/*+engine=mpp*/ select ASCII('a'),ASCII('ab'), hex(10),hex(1010), REPEAT('abc',3), INITCAP('the soap'), TRANS  
LATE('acbd','ab','12'), TRANSLATE('acbd','ac','1'), CHAR(65,68,83);
```

```
/*+engine=mpp*/ SELECT INSTR('CORPORATE FLOOR','OR'), INSTR('CORPORATE FLOOR','OR', 3), INSTR('C  
ORPORATE FLOOR','OR', 3, 2), mid('Quadratically',5,6), LCASE('ADS'), ucase('ads'), INITCAP('ads')
```

```
/*+engine=mpp*/ SELECT split('a,b,cd,d',''), split('a,b,cd,d',' ',2), split_part('a,b,cd,d',' ',3)
```

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

- 日期时间运算符可以进行时间运算，时间类型包括：date,time,timestamp,interval。
- interval为时间单位，包含：Year,Month,Day,Hour,Minute,Second
- 运算格式：interval 'N' [Year,Month,Day,Hour,Minute,Second]，其中'N'必须单引号一个数字。

运算符	示例	结果
+	date '2012-08-08' + interval '2' day	2012-08-10
+	time '01:00' + interval '3' hour	04:00:00.000
+	timestamp '2012-08-08 01:00' + interval '29' hour	2012-08-09 06:00:00.000
+	timestamp '2012-10-31 01:00' + interval '1' month	2012-11-30 01:00:00.000
+	interval '2' day + interval '3' hour	2 03:00:00.000
+	interval '3' year + interval '5' month	3-5
-	date '2012-08-08' - interval '2' day	2012-08-06
-	time '01:00' - interval '3' hour	22:00:00.000
-	timestamp '2012-08-08 01:00' - interval '29' hour	2012-08-06 20:00:00.000
-	interval '2' day - interval '3' hour	1 21:00:00.000
-	interval '3' year - interval '5' month	2-7

函数名	描述	示例	返回类型	结果
current_date/curdate	返回当前日期	select current_date()	DATE	2018-03-29
current_time/curtime	返回当前时间	select current_time()	TIME	10:54:16
current_timestamp	返回当前时间戳	select current_timestamp()	TIMESTAMP	2018-03-29 10:55:13
current_timezone	返回当前时区	select current_timezone()	VARCHAR	Asia/Shanghai
from_unixtime	返回unixtime时间戳	select from_unixtime(1522292553);	TIMESTAMP	2018-03-29 03:02:33.0
localtime	返回本地时间	select localtime()	TIME	11:17:50
localtimestamp()	返回当前本地时间戳	select localtimestamp();	TIMESTAMP	2018-03-29 11:18:44.3
now()	返回当前时间戳	select now()	TIMESTAMP	2018-03-29 11:19:35.0
to_unixtime(timestamp)	转换为unix时间戳	select to_unixtime(now())	DOUBLE	1.522322425646E9
utc_date()	返回utc日期	SELECT UTC_DATE();	VARCHAR	2018-03-29
utc_time()	返回utc时间	SELECT UTC_TIME();	VARCHAR	13:03:04
utc_timestamp()	返回utc时间戳	SELECT utc_timestamp();	VARCHAR	2018-03-29 1


```
SELECT UNIX_TIMESTAMP('2015-11-13 10:20:19.1'), UNIX_TIMESTAMP(now()), UNIX_TIMESTAMP(curdate());
```

```
select current_date(), curdate();
```

```
select current_time();
```

```
select current_timestamp(), now ();
```

- DATE_TRUNC 函数根据您指定的日期部分（如小时、周或月）截断时间戳表达式或文本
- 语法：DATE_TRUNC('unit', timestamp)
- 参数unit支持的单位有：

单位	示例	示例结果	说明
SECOND	select date_trunc('second', now());	2018-03-29 11:39:45.0	
MINUTE	select date_trunc('minute', now());	2018-03-29 11:39:00.0	
HOURL	select date_trunc('hour', now());	2018-03-29 11:00:00.0	
DAY	select date_trunc('day', now());	2018-03-29 00:00:00.0	
WEEK	select date_trunc('week', now());	2018-03-26 00:00:00.0	返回周一零点
MONTH	select date_trunc('month', now());	2018-03-01 00:00:00.0	返回当月第一天零点
QUARTER	select date_trunc('quarter', now());	2018-01-01 00:00:00.0	返回本季度第一天零点
YEAR	select date_trunc('year', now());	2018-01-01 00:00:00.0	返回本年度第一天零点

函数名称	参数	返回类型	说明
date()	varchar/int/timestamp	date	返回日期，等价于cast(x as date)
date_format(timestamp,format)	timestamp,format	varchar	将timestamp按照指定格式转为varchar
date_parse(vvarchar, format)	varchar,format	timestamp	将varchar按照指定格式转为timestamp
to_char(datetime,format)	datetime,format	varchar	datetime：支持timestamp、date、varchar类型
Str_to_date(str,format)	Vvarchar,format	date	与date_format相反
cast()	x as dataType	dataType	将任意类型转换为目标类型
unix_timestamp()	无	bigint	返回Unix Timestamp
unix_timestamp(date)	date	bigint	返回Unix Timestamp，输入日期即为UTC日期
unix_timestamp(timestamp)	timestamp	bigint	返回Unix Timestamp，输入时间戳即为UTC时间戳
from_unixtime(unixtime)	unix_time	timestamp	返回unixtime对应的时间戳
from_unixtime(unixtime,format)	unix_time,format	timestamp	返回unixtime对应的时间戳或日期
to_days(date)	date varchar,timesatamp	bigint	返回从0年开始以来的天数
to_seconds	date varchar,timesatamp	bigint	返回从0年开始以来的秒数

格式符	说明
%a	Abbreviated weekday name (Sun .. Sat)
%b	Abbreviated month name (Jan .. Dec)
%c	Month, numeric (0 .. 12)
%d	Day of the month, numeric (00 .. 31)
%e	Day of the month, numeric (0 .. 31)
%f	Fraction of second (6 digits for printing: 000000 .. 999000; 1 - 9 digits for parsing: 0 .. 999999999)
%H	Hour (00 .. 23)
%h	Hour (01 .. 12)
%l	Hour (01 .. 12)
%i	Minutes, numeric (00 .. 59)
%j	Day of year (001 .. 366)
%k	Hour (0 .. 23)
%l	Hour (1 .. 12)

格式符	说明
%M	Month name (January .. December)
%m	Month, numeric (00 .. 12)
%p	AM or PM
%r	Time, 12-hour (hh:mm:ss followed by AM or PM)
%S	Seconds (00 .. 59)
%s	Seconds (00 .. 59)
%T	Time, 24-hour (hh:mm:ss)
%v	Week (01 .. 53), where Monday is the first day of the week; used with %x
%W	Weekday name (Sunday .. Saturday)
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)
%%	A literal % character
%x	x, for any x not listed above

时间格式及转换函数使用示例

```
select date(sysdate()), date('2018-03-04 01:01:01'), date(20180304), date('20180304');
```

```
/*+engine=mpp*/ SELECT date_parse('20180516144851','%Y%m%d%k%i%S')
```

```
SELECT DATE_FORMAT('2017-11-11 11:11:11', '%Y-%m-%d') ,  
DATE_FORMAT('2017-11-11 11:11:11', '%Y-%m-%d %H:%i:%s') ,  
DATE_FORMAT('2017-11-11 11:11:11', '%Y-%m-%d %T') ,  
DATE_FORMAT('2017-11-11 11:11:11', '%W %M %Y')  
from t_fact_customers where customer_id =1;
```

```
SELECT to_char(ADDDATE(curdate(),-6),'%Y-%m-%d'), to_char(curdate(),'%Y%m%d') f  
rom t_fact_customers WHERE customer_id=1
```

```
SELECT STR_TO_DATE('01,5,2013','%d,%m,%Y');
```

```
select customer_id,customer_name,first_login  
from t_fact_customers WHERE 1=1 and first_login> str_to_date('2017-01-01 10:00:00', '%Y-%m-%d %H:%i:%s')
```

```
SELECT TO_DAYS('2007-10-07');
```

```
SELECT TO_SECONDS('2009-11-29');
```


时间格式及转换函数使用示例

```
select from_unixtime(unix_timestamp(sysdate()));
```

```
select unix_timestamp('1970-01-01'),  
       unix_timestamp('1970-01-01 00:00:00'),  
       unix_timestamp(sysdate()),  
       unix_timestamp();
```

- 语法：to_char(datetime, format)
- 参数说明：
- datetime：支持timestamp、date、varchar类型
- format：使用了和Teradata SQL 的datetime函数兼容的字符串格式
- format格式说明见下表：

格式	说明
- / , . ; :	标点符号被忽略
dd	天 (1-31)
hh	12小时制 (1-12)
hh24	24小时制 (0-23)
mi	分钟 (0-59)
mm	月 (01-12)
ss	秒 (0-59)
yyyy	4位年
yy	2位年

```
SELECT to_char(now(), '%Y-%m-%d %H%i:%s');
SELECT to_char(current_date(), '%Y-%m-%d %H%i:%s');
SELECT to_char('2018-03-29 15:09:19', '%Y-%m-%d %H%i:%s');
```

- 使用抽取函数来抽取日期时间字段如下域：
- 支持TIMESTAMP、DATE、VARCHAR类型

域	描述	示例/结果
YEAR	year()	SELECT year(now());->2018. SELECT year(current_date());->2018
QUARTER	quarter()	SELECT quarter(now());->1 SELECT quarter(current_date());->1
MONTH	month()	SELECT month(now());->3 SELECT month(current_date());->3;
DAY/DAY_OF_MONTH	day()	SELECT day(current_date());->29 SELECT day_of_month(now());->29
DAY_OF_WEEK/DOW	day_of_week()	SELECT day_of_week(now());->4 SELECT dow(current_date());->4
DAY_OF_YEAR/DOY	day_of_year()	SELECT day_of_year(current_date());->88 SELECT doy(now());->88
YEAR_OF_WEEK/YOW	year_of_week()	SELECT year_of_week(now());->2018 SELECT yow(current_date());->2018
HOUR	hour()	SELECT hour(now());->14 SELECT hour(current_time());->14
MINUTE	minute()	SELECT minute(current_time());->59 SELECT minute(now());->59
SECOND	second()	SELECT second(current_time());->14 SELECT second(now());->20

函数名称	返回类型	说明
date_add(unit, value, timestamp)	timestamp	对时间进行增减运算
date_diff(unit, timestamp1, timestamp2)	bigint	求两个时间戳之间的差
ADDTIME	datetime	返回expr1 + expr2结果，参数仅支持VARCHAR类型
PERIOD_ADD(YYYYMM, monthNum)	bigint	YYYYMM： bigint类型YYYYMM格式，如： 201803
PERIOD_DIFF(YYYYMM, YYYYMM)	bigint	返回参数1 - 参数2 的月份
SUBTIME(expr1,expr2)	datetime	返回expr1 - expr2后的时间，要求expr1与expr2相同格式
SUBDATE/DATE_SUB(date,INTERVAL expr unit) SUBDATE(expr,days)	datetime	返回参数1减去单位天数后的日期
TIMESTAMPADD(unit,interval,datetime_expr)	Date／datetime	返回增加指定单位interval后的日期时间
TIMESTAMPDIFF(unit,datetime_expr1,datetime_expr2)	bigint	返回datetime_expr2 - datetime_expr1 结果，单位为unit

单位	描述
millisecond	Milliseconds
second	Seconds
minute	Minutes
hour	Hours
day	Days
week	Weeks
month	Months
quarter	Quarters of a year
year	Years


```
SELECT DATE_ADD('DAY', 31, current_date()), current_date();
```

```
SELECT DATE_ADD('second', 31, current_time()), current_time();
```

```
SELECT DATE_ADD('second', 31, current_timestamp()), current_timestamp();
```

```
SELECT addDate(now(), interval '2' day);
```

```
SELECT first_login, ADDDATE(first_login,INTERVAL 2 HOUR), ADDDATE(first_login,INTERVAL 2 SECOND),  
ADDDATE('2017-01-12 10:00:11',INTERVAL 11222 SECOND)  
from t_fact_customers where customer_id <=3;
```

```
SELECT addDate(now(), interval '2' minute);
```

```
SELECT ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002');
```

```
SELECT ADDTIME('23:59:59.999999', '1:1:1.000002');
```

```
SELECT PERIOD_ADD(200801,20);
```

```
SELECT PERIOD_DIFF(200802,200703);
```

```
SELECT date_diff('day', current_date(), date_add('day', 2, current_date()));
```

```
SELECT date_diff('minute', now(), date_add('day', 2, now()));
```

```
SELECT SUBTIME('2007-12-31 23:59:59.999999','1 1:1:1.000002');
```

```
SELECT DATE_SUB(now(), INTERVAL 31 DAY);
```

```
SELECT SUBDATE(current_date(), INTERVAL 31 DAY);
```

```
SELECT TIMESTAMPADD(WEEK,1,'2003-01-02');
```

```
SELECT TIMESTAMPADD(MONTH,1,'2003-01-02');
```

```
SELECT TIMESTAMPDIFF(SECOND,curtime(),'20:39:39');
```

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

数名	表达式及返回值类型	说明描述
arbitrary	arbitrary(x) → [类型与输入参数相同]	返回 x 的任意非null值
array_agg	array_agg(x) → array<[类型和输入参数相同]>	返回以输入参数 x 为元素的数组
avg	avg(x) → double	返回所有输入值的平均数（算数平均数）
bool_and	bool_and(boolean) → boolean	只有所有参数均为 TRUE 则返回 TRUE，否则返回 FALSE。
bool_or	bool_or(boolean) → boolean	任何一个参数为 TRUE，则返回 TRUE，否则返回 FALSE。
checksum	checksum(x) → varbinary	返回不受给定参数值顺序影响的校验值。
count	count(*) → bigint	返回输入数据行的统计个数。
	count(x) → bigint	返回非null值的输入参数个数。
count_if	count_if(x) → bigint	返回输入参数中 TRUE 的个数。这个函数和 count(CASE WHEN x THEN 1 END) 等同
every	every(boolean) → boolean	这个函数是 bool_and() 的别名。
geometric_mean	geometric_mean(x) → double	返回所有输入参数的几何平均值
max	max(x) → [与输入类型相同]	返回输入参数中最大的值
	max(x, n) → array<[与x类型相同]>	返回所有参数 x 中前 n 大的值.
min	min(x) → [与输入类型相同]	返回所有输入参数中最小的值。
	min(x, n) → array<[与x类型相同]>	返回所有输入参数 x 中，前 n 小的值。
sum	sum(x) → [和输入类型相同]	返回所有输入参数的和

```
select sex, array_agg(customer_name) from t_fact_customers group by sex;
```

```
select avg(customer_id) from t_fact_customers;  
select bool_and(sex = 0) from t_fact_customers;  
select bool_or(sex = 0) from t_fact_customers;  
select checksum(customer_id) from t_fact_customers;
```

```
select count(*) from t_fact_customers;  
select count(customer_id) from t_fact_customers;  
select count_if(sex = 0) from t_fact_customers;  
select every(sex = 1) from t_fact_customers;  
select geometric_mean(customer_id) from t_fact_customers;  
select sum(customer_id) from t_fact_customers;
```


比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

窗口函数基于查询结果的行数据进行计算。

窗口函数运行在 HAVING 子句之后，但是在 ORDER BY 子句之前。

触发一个窗口函数需要特殊的关键字 OVER 子句来指定窗口。

一个窗口包含三个组成部分：

分区规范，用于将输入行分裂到不同的分区中。这个过程和 GROUP BY 子句的分裂过程相似。

排序规范，用于决定输入数据行在窗口函数中执行的顺序。

窗口框架，用于指定一个滑动窗口的数据给窗口函数处理给定的行数据。

函数 OVER (PARTITION BY column1 ORDER BY column2)

函数：聚集函数、排序函数、值函数

```
SELECT customer_id,customer_name,sex,birth_day,  
cume_dist() OVER (PARTITION BY sex ORDER BY birth_day) AS rolling_cume_dist  
FROM t_fact_customers;
```

```
SELECT customer_id,customer_name,sex,birth_day,  
dense_rank() OVER (PARTITION BY sex ORDER BY birth_day) AS rolling_dense_rank FROM t_fact_customers;
```

```
SELECT customer_id,customer_name,sex,birth_day, ntile(2)  
OVER (PARTITION BY sex ORDER BY birth_day) AS rolling_ntile FROM t_fact_customers;
```

```
SELECT customer_id,customer_name,sex,birth_day,  
percent_rank() OVER (PARTITION BY sex ORDER BY birth_day) AS rolling_percent_rank  
FROM t_fact_customers;
```

```
SELECT customer_id,customer_name,sex,birth_day,  
rank() OVER (PARTITION BY sex ORDER BY birth_day) AS rolling_rank FROM t_fact_customers;
```

```
SELECT customer_id,customer_name,sex,birth_day, row_number()  
OVER (PARTITION BY sex ORDER BY birth_day) FROM t_fact_customers;
```

```
SELECT customer_id, customer_name, sex, birth_day,  
sum(age) OVER (PARTITION BY sex ORDER BY birth_day) AS rolling_sum  
FROM t_fact_customers;
```

```
SELECT customer_id, customer_name, sex, birth_day,  
first_value(customer_id) OVER (PARTITION BY sex ORDER BY birth_day)  
FROM t_fact_customers;
```

```
SELECT customer_id, customer_name, sex, birth_day,  
last_value(customer_id) OVER (PARTITION BY sex ORDER BY birth_day)  
FROM t_fact_customers;
```

```
SELECT customer_id, customer_name, sex, birth_day, ntile(2)  
OVER (PARTITION BY sex ORDER BY birth_day) AS rolling_ntile  
FROM t_fact_customers;
```

```
SELECT customer_id, customer_name, sex, birth_day,  
nth_value(customer_id, 2) OVER (PARTITION BY sex ORDER BY birth_day)  
FROM t_fact_customers;
```

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

函数名	表达式及返回值类型	描述说明
json_array_contains	json_array_contains(json, value) → boolean	确定json中是否存在值（包含JSON数组的字符串）
json_array_get	json_array_get(json_array, index) → varchar	将指定index处的元素返回到json数组中,index从0开始计数
json_array_length	json_array_length(json) → bigint	返回json的数组长度（包含JSON数组的字符串）
json_extract	json_extract(json, json_path) → json	评估json上的JSONPath表达式json_path（包含JSON的字符串），并将结果作为JSON字符串返回
json_extract_scalar	json_extract_scalar(json, json_path) → varchar	和json_extract（）类似，但返回结果值作为一个字符串（而不是编码为JSON）。 json_path引用的值必须是scalar（boolean, number or string）
json_format	json_format(json) → varchar	将json作为字符串返回
json_parse	json_parse(string) → json	解析字符串作为json
json_size	json_size(json, json_path) → bigint	跟json_extract（）一样，但返回size的大小。对于对象或数组，大小是成员数，scalar的大小为零

JSON函数使用示例(1)

```
CREATE TABLE json_tbl ( id bigint COMMENT 'ID数字', sid bigint COMMENT '',  
json_test json comment '这一列是json类型，默认会建立json索引',  
PRIMARY KEY (id) )  
PARTITION BY HASH KEY (id)  
PARTITION NUM 8 TABLEGROUP ads_demo  
OPTIONS (UPDATETYPE='realtime') COMMENT ''
```

```
insert into json_tbl (id, sid, json_test) values(0, 0, '{"id":0, "name":"tjy", "age":0}');  
insert into json_tbl (id, sid, json_test) values(1, 1, '{"id":1, "name":"tjy", "age":10, "company":"alibaba"}');  
insert into json_tbl (id, sid, json_test) values(2, 2, '{"id":2, "name":"yjt", "age":20}');  
insert into json_tbl (id, sid, json_test) values(5, 5, '{"id":5, "name":"tjy", "age":50, "company":{"name":"microsoft",  
"place":"america"}}');
```

```
select * from json_tbl where json_extract(json_test_col, '$.company.company_name') = 'alibaba';  
select * from json_tbl where json_extract(json_test_col, '$.id') between 0 and 10;  
select * from json_tbl where json_extract(json_test, '$.company.name') = 'alibaba';  
select id, json_test, json_extract(json_test, '$.name') from json_tbl where json_extract(json_test, '$.id') = 0;
```

JSON函数使用示例(2)

```
SELECT json_array_contains('[1, 2, 3]', 2);
SELECT json_array_contains('[1, 2, 3]', 2);
select json_array_get('["a","b","c"]', 0);
select json_array_get('["a","b","c"]', 0);
select json_array_get('["a","b","c"]', -1);
select json_array_get('["a","b","c"]', -1);
select json_array_get('["a","b","c"]', 3);
select json_array_get('["a","b","c"]', 3);
SELECT json_array_length('[1, 2, 3]');
SELECT json_array_length('[1, 2, 3]');
select json_extract('{"key":{"book":[1,2,3]}}', '$.key.book')
select json_extract('{"key":{"book":[1,2,3]}}', '$.key.book');

select json_extract_scalar('{"key":{"book":[1,2,3]}}', '$.key.book[0]');
SELECT json_format(JSON '[1, 2, 3]');
SELECT json_format(JSON '"a"');
SELECT json_parse('[1, 2, 3]');
SELECT json_size('{"x": {"a": 1, "b": 2}}', '$.x');
SELECT json_size('{"x": [1, 2, 3]}', '$.x');
```

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

udf_sys_geo_in_cycle

作用：用于做基于地理位置的经纬度画圈

格式：UDF_SYS_GEO_IN_CYCLE(longitude, latitude, point, radius)

第一个参数为经度列名称， 类型float

第二个参数为纬度列名称， 类型float

第三个参数为圆圈中心点的位置， 格式=>"经度， 维度", =>"120.85979,30.011984"

第四个参数为圆圈的半径， 单位米

返回：返回一个boolean值

udf_sys_geo_in_rectangle

作用：用于做基于地理位置的经纬度画矩形

格式：UDF_SYS_GEO_IN_RECTANGLE(longitude, latitude, pointA, pointB)

第一个参数为经度列名称, 类型float

第二个参数为纬度列名称, 类型float

第三个参数为矩形的左下角坐标， 格式=>"经度， 维度", =>"120.85979,30.011984"

第四个参数为矩形的右上角坐标， 格式=>"经度， 维度", =>"120.88450,31.21011"

返回：返回一个boolean值


```
select longitude,latitude  
from test_table_ly  
where udf_sys_geo_in_cycle(longitude,latitude, "120.85979,30.011984", 50)=true  
order by longitude;
```

```
select *  
from test_table_ly  
where udf_sys_geo_in_rectangle(longitude,latitude, "110.85979, 20.011986", "130.85930, 40.011987")=true;
```

比较函数

条件运算函数

类型转换函数

数学函数

位函数

二进制函数

字符串函数

时间日期函数

聚合函数

窗口函数

JSON函数

地理空间函数

全文检索函数

需要建立全文索引的列，类型设置为clob或blob或varchar;
通过子句FULLTEXT INDEX <idx_name> (<col_name>) 指定为这一列建立索引

查询语法

match(<已经创建全文索引的列名>) against('<待查询关键词>') 多种查询方式

普通查询

match(<多个已经创建全文索引的列名>) against('<待查询关键词>') match...against... 子句的
AND/OR/NOT
ORDER BY, GROUP BY, JOIN, UNION

精准匹配，通配符匹配

```
select id, title, author, body from articles where match(title) against ('紫砂');  
select id, title, author, body, match(title) against ('紫砂') as score from articles where match(title) against ('紫砂');  
select id, title, author, body, match(body) against ('北京') as score, match(title) against ('美术馆') as score2 from  
articles where match(body) against ('北京') and match(title) against ('紫砂');
```

```
select id, title, author, body, match(body, title) against ('北京 紫砂') as score from articles where match(body,  
title) against ('北京 紫砂') order by score desc;
```

```
select articles.id, articles.title, articles.author, articles.body, dim_rt7.id, dim_rt7.name, dim_rt7.gender,  
match(articles.title) against ('紫砂'), match(dim_rt7.name) against ('张三')  
from articles join dim_rt7 on articles.id = dim_rt7.id  
where match(articles.title) against ('悬念') and match(dim_rt7.name) against ('张三');
```

```
select id, title, author, body, match(body, title) against ('北京 紫砂') as score from articles where match(body,  
title) against ('阿里巴巴') order by score desc;
```

```
select id, title, author, body, match(body, title) against ('北京 OR 紫砂') as score  
from articles  
where match(body, title) against ('北京 OR 紫砂') order by score desc;
```



Thanks!

咨询邮箱：ADB_SUPPORT@service.alibaba.com