大多数SQL语句都可以包含*标量表达式*，用于计算数据中的新值。 例如，在查询SELECT ceil（price）FROM items中，表达式ceil（price）计算来自price列的值的向上舍入值。

标量表达式生成适合存储在单个表的单元格中（一行中的一列）。 它们可以和table表达式和选择查询形成对比，生成表的结构化内容。

下文提供了有关每个选项的详细信息。

常量

常量表达式表示不会更改原来的值。 它们在SQL常量部分中有进一步描述。

列参考

查询中的表达式可以通过两种方式引用当前数据源中的列：

* 使用列的名称，例如SELECT price FROM items中的price。
* 如果列的名称也是SQL关键字，则必须适当引用该名称。 例如： SELECT "Default" FROM configuration.
* 如果名称不明确（例如，当跨多个表连接时），则可以通过使用表名称为列名添加前缀来消除歧义。 例如SELECT items.price FROM items
* 使用列的序号位置。 例如，SELECT @1 FROM items选择items中的第一列。

这是一个CockroachDB SQL扩展。

注意：在生产代码中应谨慎使用序数引用！ 在schema更新期间，列序号位置可以更改并使基于先前版本的schema使用序号位置的现有查询无效。

一元和二元操作

以一元运算符为前缀的表达式或由二元运算符分隔的两个表达式形成一个新表达式。

有关CockroachDB运算符的完整列表，有关其优先顺序的详细信息以及哪些数据类型是每个运算符的有效操作数，查看[Functions and Operators](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/#operators)。

值比较

标准运算符<（小于），>（大于），<=（低于或等于），> =（大于或等于），=（等于）， <>和！=（不等于），IS（相同）和IS NOT（不相同）可以应用于单个数据类型的任何一对值，以及来自不同数据类型的一些值。

更多请查看： [this section over which data types are valid operands for each operator](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/#operators).

以下特殊规则适用：

* NULL总是比其他每个值都小，甚至自身。
* NULL永远不会达到=，甚至是NULL等于任何东西。 去检查 值是否为NULL，使用IS运算符或条件表达式IFNULL（..）。

See also [NULLs and Ternary Logic](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/null-handling/#nulls-and-ternary-logic).

类型

所有的比较都接受任何参数类型的组合，并产生类型BOOL。

与NaN比较

CockroachDB识别FLOAT或DECIMAL类型的标量的特殊值NaN（非数字）。

根据IEEE 754标准，NaN被认为与比较中的每个其他数值不同。

但为了与PostgreSQL兼容，有两个例外，：

* 在比较中，NaN被认为与自身相同。 IEEE 754规定NaN与其自身不同。
* NaN被认为小于其他所有值，包括-INFINITY。 IEEE 754规定NaN不与任何其他值排序，即x <= NaN和x> = NaN对于包括无穷大的x的每个值都是假的。

存在这些特例，以便可以在WHERE子句和索引中使用值NaN。

举个例子：

> **SELECT** FLOAT 'NaN < 1, 1 < FLOAT 'NaN', FLOAT 'NaN' < FLOAT 'NaN';

+-----------------+-----------------+---------------------------+

| FLOAT 'NaN' < 1 | 1 < FLOAT 'NaN' | FLOAT 'NaN' < FLOAT 'NaN' |

+-----------------+-----------------+---------------------------+

| true | false | false |

+-----------------+-----------------+---------------------------+

> SELECT FLOAT 'NaN' = FLOAT 'NaN' AS result;

+--------+

| result |

+--------+

| true |

+--------+

> SELECT FLOAT 'NaN' < FLOAT '-INFINITY' AS result;

+--------+

| result |

+--------+

| true |

+--------+

多个值比较

<expr> <comparison> ANY <expr>

<expr> <comparison> SOME <expr>

<expr> <comparison> ALL <expr>

值比较运算符<，>，=，<=，> =，<>和！=，以及模式匹配运算符[NOT] LIKE和 [NOT] ILIKE，可用于将左侧的单个值与右侧的多个值进行比较。

这是通过使用关键字ANY/SOME或ALL组合运算符来完成的。

右操作数可以是数组，元组或子查询。

当且仅当以下情况时，比较结果为真：

* 对于ANY/SOME，左侧值的比较对于右侧的任何元素都是正确的。
* 对于ALL，左侧值的比较对于右侧的每个元素都是真的。

举个例子：

> **SELECT** 12 = **ANY** (10, 12, 13); *-- returns true*

> **SELECT** 12 = **ALL** (10, 12, 13); *-- returns false*

> **SELECT** 1 = **ANY** ARRAY[2, 3, 1]; *-- using an array*

> **SELECT** 1 = **ALL** (**SELECT** \* **FROM** **rows**); *-- using a tuple generated by a subquery*

类型规则

左侧类型与右侧操作数的元素类型之间必须是可比较的。

设置成员

语法：

<expr> **IN** <expr>

<expr> **IN** ( ... subquery ... )

<expr> NOT **IN** <expr>

<expr> NOT **IN** ( ... subquery ... )

当且仅当左操作数的值是评估右操作数的结果的一部分时，返回TRUE。 在子查询表单中，可以使用任何选择查询。

举个例子：

> **SELECT** a **IN** (1, 2, 3) **FROM** sometable;

> **SELECT** a **IN** (**SELECT** \* **FROM** allowedvalues) **FROM** sometable;

> **SELECT** ('x', 123) **IN** (**SELECT** \* **FROM** **rows**);

有关更多详细信息和性能最佳实践，另请参阅[子查询](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/subqueries/)。

类型规则

IN要求它的右操作数是一个同类元组类型，它的左操作数要匹配元组元素类型。结果类型为BOOL。

字符串模式匹配

语法：

<expr> LIKE <expr>

<expr> ILIKE <expr>

<expr> NOT LIKE <expr>

<expr> NOT ILIKE <expr>

将两个表达式计算为字符串，然后测试左侧的字符串是否与右侧给出的模式匹配。 如果找到匹配则返回TRUE，否则返回FALSE，或者返回NOT变量的倒数。

模式可以包含\_以匹配任何单个字符，或％来匹配任何零个或多个字符的序列。 ILIKE导致匹配会对大小写地进行测试。

举个例子：

> **SELECT** 'monday' **LIKE** '%day' **AS** a, 'tuesday' **LIKE** 'tue\_day' **AS** b, 'wednesday' **ILIKE** 'W%' **AS** c;

+------+------+------+

| a | b | c |

+------+------+------+

| true | true | true |

+------+------+------+

类型规则

操作数必须是STRING或两者都是BYTES。 结果类型为BOOL。

使用POSIX正则表达式进行字符串匹配

语法：

<expr> ~ <expr>

<expr> ~**\*** <expr>

<expr> !~ <expr>

<expr> !~**\*** <expr>

将两个表达式计算为字符串，然后测试左侧的字符串是否与右侧给出的模式匹配。 如果找到匹配则返回TRUE，否则返回FALSE，或者返回！变量的倒数。

星号“\*”使用不区分大小写的匹配; 否则匹配区分大小写。

该模式使用POSIX正则表达式语法表示。 与LIKE模式不同，正则表达式允许匹配字符串内的任何位置，而不仅仅是在开头。

举个例子：

> **SELECT** 'monday' ~ 'onday' **AS** a, 'tuEsday' ~ 't[uU][eE]sday' **AS** b, 'wednesday' ~\* 'W.\*y' **AS** c;

+------+------+------+

| a | b | c |

+------+------+------+

| true | true | true |

+------+------+------+

类型规则

操作数必须是STRING或两者都是BYTES。 结果类型为BOOL。

使用SQL正则表达式进行字符串匹配

语法：

<expr> SIMILAR TO <expr>

<expr> NOT SIMILAR TO <expr>

将两个表达式计算为字符串，然后测试左侧的字符串是否与右侧给出的模式匹配。 如果找到匹配则返回TRUE，否则返回FALSE，或者返回NOT变量的倒数。

使用SQL标准的正则表达式定义表示模式。

这是SQLLIKE模式和POSIX正则表达式的混合：

* \_和%分别表示任何字符或任何字符串。
* .特别匹配句点字符，不像POSIX，它是一个通配符。
* 大多数其他POSIX语法照常应用。
* 模式匹配整个字符串（如LIKE，与POSIX正则表达式不同）。

举个例子：

> **SELECT** 'monday' SIMILAR **TO** '\_onday' **AS** a, 'tuEsday' SIMILAR **TO** 't[uU][eE]sday' **AS** b, 'wednesday' SIMILAR **TO** 'w%y' **AS** c;

+------+------+------+

| a | b | c |

+------+------+------+

| true | true | true |

+------+------+------+

类型规则

操作数必须是STRING或两者都是BYTES。 结果类型为BOOL。

函数调用和SQL特殊表单

普通语法：

<name> ( <arguments...> )

内置函数名后跟一个左括号，后跟逗号分隔的表达式列表，后跟一个右括号。

这将命名函数应用于括号之间的参数。 当函数的命名空间没有前缀时，名称解析规则决定调用哪个函数。

另请参阅[有关支持的内置函数部分](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/)。

此外，还支持以下SQL特殊表单：

| **Special form** | **Equivalent to** |
| --- | --- |
| CURRENT\_CATALOG | current\_catalog() |
| CURRENT\_DATE | current\_date() |
| CURRENT\_ROLE | current\_user() |
| CURRENT\_SCHEMA | current\_schema() |
| CURRENT\_TIMESTAMP | current\_timestamp() |
| CURRENT\_TIMESTAMP | current\_timestamp() |
| CURRENT\_TIME | current\_time() |
| CURRENT\_USER | current\_user() |
| EXTRACT( FROM ) | extract("", ) |
| EXTRACT\_DURATION( FROM ) | extract\_duration("", ) |
| OVERLAY( PLACING FROM FOR ) | overlay(, , , ) |
| OVERLAY( PLACING FROM ) | overlay(, , ) |
| POSITION( IN ) | strpos(, ) |
| SESSION\_USER | current\_user() |
| SUBSTRING( FOR FROM ) | substring(, , ) |
| SUBSTRING( FOR ) | substring(, 1, ) |
| SUBSTRING( FROM FOR ) | substring(, , ) |
| SUBSTRING( FROM ) | substring(, ) |
| TRIM( FROM ) | btrim(, ) |
| TRIM(, ) | btrim(, ) |
| TRIM(FROM ) | btrim() |
| TRIM(LEADING FROM ) | ltrim(, ) |
| TRIM(LEADING FROM ) | ltrim() |
| TRIM(TRAILING FROM ) | rtrim(, ) |
| TRIM(TRAILING FROM ) | rtrim() |
| USER | current\_user() |

类型规则

通常，函数调用要求参数属于函数接受的类型，并返回由函数确定类型的值。

但是，由于SQL支持函数重载，函数调用的类型很复杂。更多细节[查看我们的博客](https://www.cockroachlabs.com/blog/revisiting-sql-typing-in-cockroachdb/)。

下标表达式

可以使用[...]运算符访问数组值中的一个项。

例如，如果名称“a”指的是10个值的数组，a [3]将检索第3个值。 第一个值的索引为1。

如果索引小于或等于0，或者大于数组的大小，则下标表达式的结果为NULL。

类型规则

下标表达式必须具有数组类型; 索引表达式必须具有类型INT。 结果具有下标表达式的元素类型。

条件表达式

表达式可以测试条件表达式，并根据是否满足或满足哪个条件，判断出一个或多个其他操作。

这些表达式格式共享以下属性：仅在条件为真时才评估其某些操作数。 特别是当操作数无效时，这点很重要。 例如，如果a为0，则IF（a = 0,0，x/a）返回0，否则返回x/a。

IF 表达式

语法：

IF ( <cond>, <expr1>, <expr2> )

计算<cond>，然后在条件为真时计算<expr1>，否则计算<expr2>。

不评估对应于条件为假时的表达式。

类型规则

条件必须具有类型BOOL，其余两个表达式必须具有相同的类型。 结果与已计算的表达式具有相同的类型。

简单的CASE表达式

语法：

**CASE** <cond>

**WHEN** <condval1> **THEN** <expr1>

[ **WHEN** <condvalx> **THEN** <exprx> ] ...

[ **ELSE** <expr2> ]

**END**

计算<cond>，然后选择WHEN分支，其中<condval>等于<cond>，然后计算并返回相应的THEN表达式。 如果没有WHEN匹配分支，则评估并返回ELSE表达式（如果有的话）。 否则，返回NULL。

在第一次匹配之后的条件和结果表达式不评估。

类型规则

条件和WHEN表达式必须具有相同的类型。 THEN表达式和ELSE表达式（如果有的话）必须具有相同的类型。 结果与THEN /ELSE表达式的类型相同。

搜索CASE表达式

语法：

**CASE** **WHEN** <cond1> **THEN** <expr1>

[ **WHEN** <cond2> **THEN** <expr2> ] ...

[ **ELSE** <expr> ]

**END**

按顺序计算每个<cond>表达式; 在第一个计算为TRUE的<cond>表达式中，返回计算相应的THEN表达式的结果。 如果<cond>表达式都没有计算为true，则计算并返回ELSE表达式的值（如果有），否则返回NULL。

条件和结果表达式在第一次匹配之后不再计算比较。

类型规则

所有WHEN表达式都必须具有类型BOOL。 THEN表达式和ELSE表达式（如果有的话）必须具有相同的类型。 结果与THEN /ELSE表达式的类型相同。

NULLIF表达式

语法：

NULLIF ( <expr1>, <expr2> )

相当于: IF ( <expr1> = <expr2>, NULL, <expr1> )

类型规则

两个操作数必须具有相同的类型，这也是结果的类型。

COALESCE和IFNULL表达式

语法：

IFNULL ( <expr1>, <expr2> )

COALESCE ( <expr1> [, <expr2> [, <expr3> ] ...] )

COALESCE evaluates the first expression first. If its value is not NULL, its value is returned directly. Otherwise, it returns the result of applying COALESCE on the remaining expressions. If all the expressions are NULL, NULL is returned. COALESCE首先对第一个表达式比较。 如果其值不是NULL，则直接返回其值。 否则，它返回在剩余表达式上应用COALESCE的结果。 如果所有表达式都是NULL，则返回NULL。

不计算第一个非null参数右侧的参数。

IFNULL(a, b) 等效于 COALESCE(a, b).

类型规则

操作数必须具有相同的类型，这也是结果的类型。

逻辑运算符

可以使用布尔运算符AND，OR和NOT。

语法：

NOT <expr>

<expr1> AND <expr2>

<expr1> OR <expr2>

AND和OR是可交换的。 此外，AND和OR的输入不按任何特定顺序进行评估。 如果仅使用另一个操作数可以完全确定结果，则甚至可能根本不计算某些操作。

这与其他编程语言中的从左到右的逻辑不同。 若要强制执行计算顺序，请使用条件表达式。

See also [NULLs and Ternary Logic](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/null-handling/#nulls-and-ternary-logic).

类型规则

操作数必须具有类型BOOL。 结果类型为BOOL。

聚合表达式

聚合表达式与函数调用具有相同的语法，具有COUNT的特殊情况：

<name> ( <arguments...> )

COUNT ( \* )

聚合表达式和函数调用之间的区别在于前者使用聚合函数，并且只能出现在SELECT子句中的rendered表达式列表中。

聚合表达式计算组合值，具体取决于当前所选的所有行中使用的聚合函数。

类型规则

[操作数和返回类型的确定与常规函数调用类似](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#function-calls-and-sql-special-forms).

窗口函数调用

窗口函数调用具有函数调用的语法，后跟OVER子句：

<name> ( <arguments...> ) OVER <window>

<name> ( **\*** ) OVER <window>

它表示在查询选择的行的子集（“窗口”）上的窗口或聚合函数。

类型规则

[The operand and return types are determined like for regular function calls](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#function-calls-and-sql-special-forms).

明确的类型强制

语法：

<expr> :: <type>

CAST (<expr> AS <type>)

计算表达式并将结果值转换为指定的类型。 如果转换无效，则会报错。

举个例子： CAST(now() AS DATE)

请注意，在许多情况下，类型注释优于类型强制。 有关详细信息，请参阅下面的类型注释部分。

类型规则

操作数可以是任何类型。 结果具有在CAST表达式中指定的类型。

作为一种特殊情况，如果操作数是文字、常量表达式或占位符，则CAST类型用于指导操作数的输入。[有关详细信息，请参阅我们的博文](https://www.cockroachlabs.com/blog/revisiting-sql-typing-in-cockroachdb/).

Collation表达式

语法：

<expr> COLLATE <collation>

计算表达式并将其结果转换为具有指定排序规则的Collation字符串。

举个例子： 'a' COLLATE de

类型规则

操作数必须是STRING类型。 结果是类型COLLATEDSTRING。

数组构造函数

语法：

ARRAY[ <expr>, <expr>, ... ]

计算包含指定值的数组。

举个例子：

> **SELECT** ARRAY[1,2,3] **AS** a;

+---------+

| a |

+---------+

| {1,2,3} |

+---------+

数组的数据类型是从提供的表达式的值推断出来的。 数组中的所有位置必须具有相同的数据类型。

如果没有指定表达式（空数组），或者所有值都是NULL，则必须使用类型注释显式指定数组的类型。 举个例子：

> **SELECT** ARRAY[]:::int[];

要将子查询的结果转换为数组，请改用ARRAY(...)。

类型规则

操作数必须都具有相同的类型。 结果具有操作数为元素类型的数组类型。

元组构造函数

语法：

(<expr>, <expr>, ...)

ROW (<expr>, <expr>, ...)

计算包含所提供表达式值的元组。

举个例子：

> **SELECT** ('x', 123, 12.3) **AS** a;

+----------------+

| a |

+----------------+

| ('x',123,12.3) |

+----------------+

从值中推断出结果元组的数据类型。 元组中的每个位置都可以具有不同的数据类型。

类型规则

操作数可以是任何类型。 结果有一个元组类型，其项的类型是操作数的类型。

显式类型表达式

语法：

<expr>:::<type>

ANNOTATE\_TYPE(<expr>, <type>)

计算给定的表达式，要求表达式具有给定的类型。 如果表达式没有给定的类型，则返回错误。

类型注释对于指导数值的算术特别有用，举个例子：

> **SELECT** (1 / 0):::FLOAT; *--> +Inf*

> **SELECT** (1 / 0); *--> error "division by zero"*

> **SELECT** (1 / 0)::FLOAT; *--> error "division by zero"*

类型注释也与强制转换表达式（见上文）不同，因为它们不会导致转换值。 例如，now():: DATE将当前时间戳转换为日期值（并丢弃当前时间），而now()::: DATE则触发错误消息（即now()不会 有类型DATE）。

更多请查看博客：[more information about context-dependent typing](https://www.cockroachlabs.com/blog/revisiting-sql-typing-in-cockroachdb/).

类型规则

操作数必须隐式得强制给定类型。 结果具有给定的类型。

子查询表达式

Scalar 子查询

语法：

( **...** subquery **...** )

计算子查询，声明它返回单行和单列，然后计算该单个单元格的值。 任何选择查询都可以用作子查询。

举个例子：

> **SELECT** (**SELECT** **COUNT**(\*) **FROM** users) > (**SELECT** **COUNT**(\*) **FROM** admins);

如果表users中的行多于表admins中的行，则返回TRUE。

有关更多详细信息和性能最佳实践，另请参阅子查询。

类型规则

操作数必须具有只有一列的表类型。 结果具有该单列的类型。

子查询结果的存在性检验

语法：

EXISTS ( ... subquery ... )

NOT EXISTS ( ... subquery ... )

计算子查询，然后返回TRUE或FALSE，具体取决于子查询是返回任何行（对于EXISTS）还是没有返回任何行（对于NOT EXISTS）。 任何选择查询都可以用作子查询。

类型规则

操作数可以具有任何表类型。 结果类型为BOOL。

Conversion of Subquery Results to an Array

语法：

**ARRAY**( ... subquery ... )

计算子查询并将其结果转换为数组。 任何选择查询都可以用作子查询。

有关更多详细信息和性能最佳实践，另请参阅子查询。 要将标量表达式列表转换为数组，请改用ARRAY[...]。

See Also

* [Constants](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/constants/)
* [Selection Queries](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/selection-queries/)
* [Table Expressions](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/table-expressions/)
* [Data Types](http://doc.cockroachchina.baidu.com/#develop/data-types/overview/)
* [Functions and Operators](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/)
* [Subqueries](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/subqueries/)