CockroachDB支持以下用于[scalar expressions](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/)的SQL 函数和运算符。

提示：在内建SQL shell中，使用\hf [function]可以获取指定函数的help信息，

特殊语法形式

为了兼容SQL标准和PostgreSQL，可识别以下语法形式，相当于常规内置函数。

| **特殊形式** | **等效于** |
| --- | --- |
| 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(<part> FROM <value>) | extract("<part>", <value>) |
| EXTRACT\_DURATION(<part> FROM <value>) | extract\_duration("<part>", <value>) |
| OVERLAY(<text1> PLACING <text2> FROM <int1> FOR <int2>) | overlay(<text1>, <text2>, <int1>, <int2>) |
| OVERLAY(<text1> PLACING <text2> FROM <int>) | overlay(<text1>, <text2>, <int>) |
| POSITION(<text1> IN <text2>) | strpos(<text2>, <text1>) |
| SESSION\_USER | current\_user() |
| SUBSTRING(<text> FOR <int1> FROM <int2>) | substring(<text>, <int2>, <int1>) |
| SUBSTRING(<text> FOR <int>) | substring(<text>, 1, <int>) |
| SUBSTRING(<text> FROM <int1> FOR <int2>) | substring(<text>, <int1>, <int2>) |
| SUBSTRING(<text> FROM <int>) | substring(<text>, <int>) |
| TRIM(<text1> FROM <text2>) | btrim(<text2>, <text1>) |
| TRIM(<text1>, <text2>) | btrim(<text1>, <text2>) |
| TRIM(FROM <text>) | btrim(<text>) |
| TRIM(LEADING <text1> FROM <text2>) | ltrim(<text2>, <text1>) |
| TRIM(LEADING FROM <text>) | ltrim(<text>) |
| TRIM(TRAILING <text1> FROM <text2>) | rtrim(<text2>, <text1>) |
| TRIM(TRAILING FROM <text>) | rtrim(<text>) |
| USER | current\_user() |

条件和类函数运算符

下表列出了看起来像内置函数但具有特殊评估规则的运算符：

| **Operator** | **Description** |
| --- | --- |
| ANNOTATE\_TYPE(...) | [显式类型表达式](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#explicitly-typed-expressions) |
| ARRAY(...) | [子查询结果转换为数组](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#conversion-of-subquery-results-to-an-array) |
| ARRAY[...] | [scalar表达式转换为数组](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#array-constructors) |
| CAST(...) | [Type Cast](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#explicit-type-coercions) |
| COALESCE(...) | [First non-NULL expression with Short Circuit](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#coalesce-and-ifnull-expressions) |
| EXISTS(...) | [Existence Test on the Result of Subqueries](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#existence-test-on-the-result-of-subqueries) |
| IF(...) | [Conditional Evaluation](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#if-expressions) |
| IFNULL(...) | COALESCE的别名，仅限于两个操作数 |
| NULLIF(...) | [Return NULL conditionally](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#nullif-expressions) |
| ROW(...) | [Tuple Constructor](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#tuple-constructor) |

内置函数

ANYELEMENT函数

| **Function → Returns** | **Description** |
| --- | --- |
| to\_json(val: anyelement) → jsonb | 返回值为JSON或JSONB。 |
| to\_jsonb(val: anyelement) → jsonb | 返回值为JSON或JSONB。 |

数组函数

| **Function -> Returns** | **Description** |
| --- | --- |
| array\_append(array: bool[], elem: bool) -> bool[] | 将元素追加到数组，返回结果 |
| array\_append(array: bytes[], elem: bytes) -> bytes[] | 将元素追加到数组，返回结果 |
| array\_append(array: date[], elem: date) -> date[] | 将元素追加到数组，返回结果 |
| array\_append(array: decimal[], elem: decimal) -> decimal[] | 将元素追加到数组，返回结果 |
| array\_append(array: float[], elem: float) -> float[] | 将元素追加到数组，返回结果 |
| array\_append(array: inet[], elem: inet) -> inet[] | 将元素追加到数组，返回结果 |
| array\_append(array: int[], elem: int) -> int[] | 将元素追加到数组，返回结果 |
| array\_append(array: interval[], elem: interval) -> interval[] | 将元素追加到数组，返回结果 |
| array\_append(array: string[], elem: string) -> string[] | 将元素追加到数组，返回结果 |
| array\_append(array: time[], elem: time) -> time[] | 将元素追加到数组，返回结果 |
| array\_append(array: timestamp[], elem: timestamp) -> timestamp[] | 将元素追加到数组，返回结果 |
| array\_append(array: timestamptz[], elem: timestamptz) -> timestamptz[] | 将元素追加到数组，返回结果 |
| array\_append(array: uuid[], elem: uuid) -> uuid[] | 将元素追加到数组，返回结果 |
| array\_append(array: oid[], elem: oid) -> oid[] | 将元素追加到数组，返回结果 |
| array\_cat(left: bool[], right: bool[]) -> bool[] | 附加两个数组。 |
| array\_cat(left: bytes[], right: bytes[]) -> bytes[] | 附加两个数组。 |
| array\_cat(left: date[], right: date[]) -> date[] | 附加两个数组。 |
| array\_cat(left: decimal[], right: decimal[]) -> decimal[] | 附加两个数组。 |
| array\_cat(left: float[], right: float[]) -> float[] | 附加两个数组。 |
| array\_cat(left: inet[], right: inet[]) -> inet[] | 附加两个数组。 |
| array\_cat(left: int[], right: int[]) -> int[] | 附加两个数组。 |
| array\_cat(left: interval[], right: interval[]) -> interval[] | 附加两个数组。 |
| array\_cat(left: string[], right: string[]) -> string[] | 附加两个数组。 |
| array\_cat(left: time[], right: time[]) -> time[] | 附加两个数组。 |
| array\_cat(left: timestamp[], right: timestamp[]) -> timestamp[] | 附加两个数组。 |
| array\_cat(left: timestamptz[], right: timestamptz[]) -> timestamptz[] | 附加两个数组。 |
| array\_cat(left: uuid[], right: uuid[]) -> uuid[] | 附加两个数组。 |
| array\_cat(left: oid[], right: oid[]) -> oid[] | 附加两个数组。 |
| array\_length(input: anyelement[], array\_dimension: int) -> int | 计算给定的array\_dimension维度上的input长度。 但是，由于CockroachDB尚不支持多维数组，因此唯一支持的array\_dimension为1。 |
| array\_lower(input: anyelement[], array\_dimension: int) -> int | 计算给定的array\_dimension维度上的input长度。 但是，由于CockroachDB尚不支持多维数组，因此唯一支持的array\_dimension为1。 |
| array\_position(array: bool[], elem: bool) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: bytes[], elem: bytes) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: date[], elem: date) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: decimal[], elem: decimal) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: float[], elem: float) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: inet[], elem: inet) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: int[], elem: int) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: interval[], elem: interval) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: string[], elem: string) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: time[], elem: time) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: timestamp[], elem: timestamp) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: timestamptz[], elem: timestamptz) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: uuid[], elem: uuid) -> int | 返回数组中第一次出现元素的索引。 |
| array\_position(array: oid[], elem: oid) -> int | 返回数组中第一次出现元素的索引。 |
| array\_positions(array: bool[], elem: bool) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: bytes[], elem: bytes) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: date[], elem: date) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: decimal[], elem: decimal) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: float[], elem: float) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: inet[], elem: inet) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: int[], elem: int) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: interval[], elem: interval) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: string[], elem: string) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: time[], elem: time) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: timestamp[], elem: timestamp) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: timestamptz[], elem: timestamptz) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: uuid[], elem: uuid) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_positions(array: oid[], elem: oid) -> int[] | 返回一个elem在array中所有出现位置的索引的数组 |
| array\_prepend(elem: bool, array: bool[]) -> bool[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: bytes, array: bytes[]) -> bytes[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: date, array: date[]) -> date[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: decimal, array: decimal[]) -> decimal[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: float, array: float[]) -> float[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: inet, array: inet[]) -> inet[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: int, array: int[]) -> int[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: interval, array: interval[]) -> interval[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: string, array: string[]) -> string[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: time, array: time[]) -> time[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: timestamp, array: timestamp[]) -> timestamp[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: timestamptz, array: timestamptz[]) -> timestamptz[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: uuid, array: uuid[]) -> uuid[] | 将元素放到数组前，返回结果。 |
| array\_prepend(elem: oid, array: oid[]) -> oid[] | 将元素放到数组前，返回结果。 |
| array\_remove(array: bool[], elem: bool) -> bool[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: bytes[], elem: bytes) -> bytes[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: date[], elem: date) -> date[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: decimal[], elem: decimal) -> decimal[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: float[], elem: float) -> float[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: inet[], elem: inet) -> inet[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: int[], elem: int) -> int[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: interval[], elem: interval) -> interval[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: string[], elem: string) -> string[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: time[], elem: time) -> time[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: timestamp[], elem: timestamp) -> timestamp[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: timestamptz[], elem: timestamptz) -> timestamptz[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: uuid[], elem: uuid) -> uuid[] | 从数组中删除所有等于elem的元素。 |
| array\_remove(array: oid[], elem: oid) -> oid[] | 从数组中删除所有等于elem的元素。 |
| array\_replace(array: bool[], toreplace: bool, replacewith: bool) -> bool[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: bytes[], toreplace: bytes, replacewith: bytes) -> bytes[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: date[], toreplace: date, replacewith: date) -> date[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: decimal[], toreplace: decimal, replacewith: decimal) -> decimal[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: float[], toreplace: float, replacewith: float) -> float[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: inet[], toreplace: inet, replacewith: inet) -> inet[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: int[], toreplace: int, replacewith: int) -> int[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: interval[], toreplace: interval, replacewith: interval) -> interval[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: string[], toreplace: string, replacewith: string) -> string[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: time[], toreplace: time, replacewith: time) -> time[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: timestamp[], toreplace: timestamp, replacewith: timestamp) -> timestamp[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: timestamptz[], toreplace: timestamptz, replacewith: timestamptz) -> timestamptz[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: uuid[], toreplace: uuid, replacewith: uuid) -> uuid[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_replace(array: oid[], toreplace: oid, replacewith: oid) -> oid[] | 用replacewith替换数组中所有出现的toreplace。 |
| array\_upper(input: anyelement[], array\_dimension: int) -> int | 计算给定的array\_dimension维度上的输入的最大值。 但是，由于CockroachDB尚不支持多维数组，因此唯一支持的array\_dimension为1。 |
| string\_to\_array(str: string, delimiter: string) -> string[] | 依据delimiter将字符串进行拆分。 |
| string\_to\_array(str: string, delimiter: string, null: string) -> string[] | 依据delimiter将字符串进行拆分，并使用指定的字符串来作为NULL。 |

布尔函数

| **Function → Returns** | **Description** |
| --- | --- |
| inet\_contained\_by\_or\_equals(val: inet, container: inet) → bool | 仅使用地址的网络部分测试子网是否包含或相等。 地址的主机部分被忽略。 |
| inet\_contains\_or\_contained\_by(val: inet, val: inet) → bool | 仅使用地址的网络部分测试子网是否包含。 地址的主机部分被忽略。 |
| inet\_contains\_or\_equals(container: inet, val: inet) → bool | 仅使用地址的网络部分测试子网是否包含或相等。 地址的主机部分被忽略。 |
| inet\_same\_family(val: inet, val: inet) → bool | 检查两个IP是否属于同一个IP family. |

比较函数

| **Function → Returns** | **Description** |
| --- | --- |
| greatest(anyelement...) → anyelement | 返回值最大的元素。 |
| least(anyelement...) → anyelement | 返回值最小的元素。 |

日期和时间函数

| **Function → Returns** | **Description** |
| --- | --- |
| age(begin: timestamptz, end: timestamptz) → interval | 计算begin和end之间的间隔。 |
| age(val: timestamptz) → interval | 计算当前时间和val之间的间隔。 |
| clock\_timestamp() → timestamp | 返回其中一个群集节点上的当前系统时间。 |
| clock\_timestamp() → timestamptz | 返回其中一个群集节点上的当前系统时间。 |
| current\_date() → date | 返回当前事务的日期。  该值基于事务开始时选择的时间戳，并且在整个事务中保持不变。 此时间戳与并发事务的提交顺序无关。 |
| current\_timestamp() → timestamp | 返回当前事务的时间。  该值基于事务开始时选择的时间戳，并且在整个事务中保持不变。 此时间戳与并发事务的提交顺序无关。 |
| current\_timestamp() → timestamptz | 返回当前事务的时间。  该值基于事务开始时选择的时间戳，并且在整个事务中保持不变。 此时间戳与并发事务的提交顺序无关。 |
| date\_trunc(element: string, input: date) → timestamptz | 将input截断成精度为element，将所有无意义的字段设为0(或对于日期和月份则设为1)  兼容元素： year, quarter, month, week, hour, minute, second, millisecond, microsecond. |
| date\_trunc(element: string, input: time) → interval | 将input截断成精度为element，将所有无意义的字段设为0(或对于日期和月份则设为1)  兼容元素： year, quarter, month, week, hour, minute, second, millisecond, microsecond. |
| date\_trunc(element: string, input: timestamp) → timestamp | 将input截断成精度为element，将所有无意义的字段设为0(或对于日期和月份则设为1)  兼容元素： year, quarter, month, week, hour, minute, second, millisecond, microsecond. |
| date\_trunc(element: string, input: timestamptz) → timestamptz | 将input截断成精度为element，将所有无意义的字段设为0(或对于日期和月份则设为1)  兼容元素： year, quarter, month, week, hour, minute, second, millisecond, microsecond. |
| experimental\_strftime(input: date, extract\_format: string) → string | 从input中提取并格式化成extract\_format标识的时间，extract\_format使用标准的strftime表示法（尽管不支持所有格式）。 |
| experimental\_strftime(input: timestamp, extract\_format: string) → string | 从input中提取并格式化成extract\_format标识的时间，extract\_format使用标准的strftime表示法（尽管不支持所有格式）。 |
| experimental\_strftime(input: timestamptz, extract\_format: string) → string | 从input中提取并格式化成extract\_format标识的时间，extract\_format使用标准的strftime表示法（尽管不支持所有格式）。 |
| experimental\_strptime(input: string, format: string) → timestamptz | 使用format（使用标准strptime格式）将input作为timestamptz返回。 |
| extract(element: string, input: date) → int | 从input中提取element。  兼容元素：year, quarter, month, week, dayofweek, dayofyear, hour, minute, second, millisecond, microsecond, epoch |
| extract(element: string, input: time) → int | 从input中提取element。  兼容元素：hour, minute, second, millisecond, microsecond, epoch |
| extract(element: string, input: timestamp) → int | 从input中提取element。  兼容元素： year, quarter, month, week, dayofweek, dayofyear, hour, minute, second, millisecond, microsecond, epoch |
| extract(element: string, input: timestamptz) → int | 从input中提取element。  兼容元素：year, quarter, month, week, dayofweek, dayofyear, hour, minute, second, millisecond, microsecond, epoch |
| extract\_duration(element: string, input: interval) → int | 从input中提取element。 兼容元素: hour, minute, second, millisecond, microsecond. |
| now() → timestamp | 返回当前事务的时间。  该值基于事务开始时选择的时间戳，并且在整个事务中保持不变。 此时间戳与并发事务的提交顺序无关。 |
| now() → timestamptz | 返回当前事务的时间。  该值基于事务开始时选择的时间戳，并且在整个事务中保持不变。 此时间戳与并发事务的提交顺序无关。 |
| statement\_timestamp() → timestamp | 返回当前语句的开始时间。 |
| statement\_timestamp() → timestamptz | 返回当前语句的开始时间。 |
| transaction\_timestamp() → timestamp | 返回当前事务的时间。  该值基于事务开始时选择的时间戳，并且在整个事务中保持不变。 此时间戳与并发事务的提交顺序无关。 |
| transaction\_timestamp() → timestamptz | 返回当前事务的时间。  该值基于事务开始时选择的时间戳，并且在整个事务中保持不变。 此时间戳与并发事务的提交顺序无关。 |

ID生成函数

| **Function → Returns** | **Description** |
| --- | --- |
| experimental\_uuid\_v4() → bytes | 返回一个 UUID. |
| gen\_random\_uuid() → uuid | 生成随机 UUID 并将其作为 UUID 类型的值返回。 |
| unique\_rowid() → int | 如果表未定义主键，则返回一个唯一 ID，该 ID 被 CockroachDB 用于生成唯一的行 ID。该值是插入时间戳和执行语句的节点的ID的组合，这保证了此组合是全局唯一的。 |
| uuid\_v4() → bytes | 返回一个 UUID. |

INET函数

| **Function → Returns** | **Description** |
| --- | --- |
| abbrev(val: inet) → string | 将IP地址和前缀长度转换为文本类型的缩写显示格式。 对于INET类型，如果它不是默认值(32或IPv4，IPv6为128)，将省略前缀长度  例如，abbrev('192.168.1.2/24')返回'192.168.1.2/24' |
| broadcast(val: inet) → inet | 获取val表示的网络地址的广播地址。  例如，广播('192.168.1.2/24')返回'192.168.1.255/24' |
| family(val: inet) → int | 提取val的IP family; IPv4为4，IPv6为6。  例如，family('::1')返回6 |
| host(val: inet) → string | 将组合地址/前缀值的地址部分提取为文本。  例如，host('192.168.1.2/16')返回'192.168.1.2' |
| hostmask(val: inet) → inet | 创建与val中的前缀长度对应的IP主机掩码。  例如，hostmask('192.168.1.2/16')返回'0.0.255.255' |
| masklen(val: inet) → int | 检索存储在val中的前缀长度。  例如，masklen('192.168.1.2/16')返回16 |
| netmask(val: inet) → inet | 创建与val中的前缀长度对应的IP网络掩码。  例如，网络掩码('192.168.1.2/16')返回'255.255.0.0' |
| set\_masklen(val: inet, prefixlen: int) → inet | 将val的前缀长度设置为prefixlen。  例如，set\_masklen('192.168.1.2'，16)返回'192.168.1.2/16'。 |
| text(val: inet) → string | 将IP地址和前缀长度转换为文本格式。 |

JSONB函数

| **Function → Returns** | **Description** |
| --- | --- |
| json\_array\_length(json: jsonb) → int | 返回JSON或JSONB数组中最外层元素的数量。 |
| json\_build\_array(anyelement...) → jsonb | 从可变参数列表构建可能异构类型(possibly-heterogeneously-typed)的JSON或JSONB数组。 |
| json\_build\_object(anyelement...) → jsonb | 从可变参数列表构建JSON对象。 |
| json\_extract\_path(jsonb, string...) → jsonb | 返回可变参数指向的JSON值。 |
| json\_object(keys: string[], values: string[]) → jsonb | 这种形式的json\_object从两个独立的数组中成对获取键和值。 在所有其他方面，它与单参数形式相同。 |
| json\_remove\_path(val: jsonb, path: string[]) → jsonb | 从JSON对象中删除指定的路径。 |
| json\_set(val: jsonb, path: string[], to: jsonb) → jsonb | 返回可变参数指向的JSON值。 |
| json\_set(val: jsonb, path: string[], to: jsonb, create\_missing: bool) → jsonb | 返回可变参数指向的JSON值。 如果create\_missing为false，则不会将新key插入到对象中，并且值也不会被添加到数组。 |
| json\_strip\_nulls(from\_json: jsonb) → jsonb | 返回from\_json，其中包含省略空值的所有对象字段。 其他空值不受影响。 |
| json\_typeof(val: jsonb) → string | 以文本字符串形式返回最外层JSON值的类型。 |
| jsonb\_array\_length(json: jsonb) → int | 返回最外层JSON或JSONB数组中的元素的数量。 |
| jsonb\_build\_array(anyelement...) → jsonb | 从可变参数列表构建可能异构类型(possibly-heterogeneously-typed)的JSON或JSONB数组。 |
| jsonb\_build\_object(anyelement...) → jsonb | 从可变参数列表构建JSON对象。 |
| jsonb\_extract\_path(jsonb, string...) → jsonb | 返回可变参数指向的JSON值。 |
| jsonb\_object(keys: string[], values: string[]) → jsonb | 这种形式的json\_object从两个独立的数组中成对获取键和值。 在所有其他方面，它与单参数形式相同。 |
| jsonb\_pretty(val: jsonb) → string | 返回给定的JSON值，作为缩进的STRING和换行符。 |
| jsonb\_set(val: jsonb, path: string[], to: jsonb) → jsonb | 返回可变参数指向的JSON值。 |
| jsonb\_set(val: jsonb, path: string[], to: jsonb, create\_missing: bool) → jsonb | 返回可变参数指向的JSON值。 如果create\_missing为false，则不会将新key插入到对象中，并且值也不会被添加到数组。 |
| jsonb\_strip\_nulls(from\_json: jsonb) → jsonb | 返回from\_json，其中包含省略空值的所有对象字段。 其他空值不受影响。 |
| jsonb\_typeof(val: jsonb) → string | 以文本字符串形式返回最外层JSON值的类型。 |

数学和数字(Numeric)函数

| **Function → Returns** | **Description** |
| --- | --- |
| abs(val: decimal) → decimal | 计算val的绝对值。 |
| abs(val: float) → float | 计算val的绝对值。 |
| abs(val: int) → int | 计算val的绝对值。 |
| acos(val: float) → float | 计算val的反余弦值。 |
| asin(val: float) → float | 计算val的反正弦值。 |
| atan(val: float) → float | 计算val的反正切值。 |
| atan2(x: float, y: float) → float | 计算x / y的反正切。 |
| cbrt(val: decimal) → decimal | 计算val的立方根(∛)。 |
| cbrt(val: float) → float | 计算val的立方根(∛)。 |
| ceil(val: decimal) → decimal | 计算大于val的最小整数。 |
| ceil(val: float) → float | 计算大于val的最小整数。 |
| ceiling(val: decimal) → decimal | 计算大于val的最小整数。 |
| ceiling(val: float) → float | 计算大于val的最小整数。 |
| cos(val: float) → float | 计算val的余弦值。 |
| cot(val: float) → float | 计算val的余切值。 |
| crc32c(bytes...) → int | 使用Castagnoli多项式计算CRC-32哈希值。 |
| crc32c(string...) → int | 使用Castagnoli多项式计算CRC-32哈希值。 |
| crc32ieee(bytes...) → int | 使用IEEE多项式计算CRC-32哈希值。 |
| crc32ieee(string...) → int | 使用IEEE多项式计算CRC-32哈希值。 |
| degrees(val: float) → float | 将val作为弧度值转换为度数值。(a degree value) |
| div(x: decimal, y: decimal) → decimal | 计算x/y的整数商。 |
| div(x: float, y: float) → float | 计算x / y的整数商。 |
| div(x: int, y: int) → int | 计算x / y的整数商。 |
| exp(val: decimal) → decimal | 计算e ^ val。 |
| exp(val: float) → float | 计算e ^ val。 |
| floor(val: decimal) → decimal | 计算不大于val的最大整数。 |
| floor(val: float) → float | 计算不大于val的最大整数。 |
| fnv32(bytes...) → int | 计算一组值的32位FNV-1哈希值。 |
| fnv32(string...) → int | 计算一组值的32位FNV-1哈希值。 |
| fnv32a(bytes...) → int | 计算不大于val的最大整数。 |
| fnv32a(string...) → int | 计算不大于val的最大整数。 |
| fnv64(bytes...) → int | 计算一组值的64位FNV-1哈希值。 |
| fnv64(string...) → int | 计算一组值的64位FNV-1哈希值。 |
| fnv64a(bytes...) → int | 计算一组值的64位FNV-1a哈希值。 |
| fnv64a(string...) → int | 计算一组值的64位FNV-1a哈希值。 |
| isnan(val: decimal) → bool | 如果val是NaN，则返回true，否则返回false。 |
| isnan(val: float) → bool | 如果val是NaN，则返回true，否则返回false。 |
| ln(val: decimal) → decimal | 计算val的自然对数。 |
| ln(val: float) → float | 计算val的自然对数。 |
| log(val: decimal) → decimal | 计算val的基数为10的log。 |
| log(val: float) → float | 计算val的基数为10的log。 |
| mod(x: decimal, y: decimal) → decimal | 计算x％y。 |
| mod(x: float, y: float) → float | 计算x％y。 |
| mod(x: int, y: int) → int | 计算x％y。 |
| pi() → float | 返回pi的值(3.141592653589793)。 |
| pow(x: decimal, y: decimal) → decimal | 计算x^y。 |
| pow(x: float, y: float) → float | 计算x^y。 |
| pow(x: int, y: int) → int | 计算x^y。 |
| power(x: decimal, y: decimal) → decimal | 计算x^y。 |
| power(x: float, y: float) → float | 计算x^y。 |
| power(x: int, y: int) → int | 计算x^y。 |
| radians(val: float) → float | 将val作为度数值(a degree value)转换为弧度值(a radians value)。 |
| random() → float | 返回0到1之间的随机浮点数。 |
| round(input: decimal, decimal\_accuracy: int) → decimal | 使用四舍五入(using half away from zero rounding.)保留input中零右侧decimal\_accuracy位数字。 如果decimal\_accuracy不在-2^31…(2^31-1)范围内，则结果是未定义(undefined)的。 |
| round(input: float, decimal\_accuracy: int) → float | 使用一半到偶数(银行家)舍入(using half to even (banker’s) rounding.)，在input的零的右侧保留decimal\_accuracy位数字。 |
| round(val: decimal) → decimal | 将val舍入到最接近的整数，距离零的一半：ROUND(+/- 2.4)= +/- 2，ROUND(+/- 2.5)= +/- 3。 |
| round(val: float) → float | 使用一半到偶数(银行家)(using half to even (banker’s) rounding)舍入将val舍入到最接近的整数。 |
| sign(val: decimal) → decimal | 确定val的符号：1为正; 0表示值0; -1表示负 |
| sign(val: float) → float | 确定val的符号：1为正; 0表示值0; -1表示负 |
| sign(val: int) → int | 确定val的符号：1为正; 0表示值0; -1表示负 |
| sin(val: float) → float | 计算val的正弦值。 |
| sqrt(val: decimal) → decimal | 计算val的平方根。 |
| sqrt(val: float) → float | 计算val的平方根。 |
| tan(val: float) → float | 计算val的正切值。 |
| to\_hex(val: int) → string | 将val转换为十六进制表示形式。 |
| trunc(val: decimal) → decimal | 截断val的十进制值。 |
| trunc(val: float) → float | 截断val的十进制值。 |

STRING[] 函数

| **Function → Returns** | **Description** |
| --- | --- |
| json\_object(texts: string[]) → jsonb | 从文本数组中构建JSON或JSONB对象。 数组必须是一个具有偶数个成员的一维数组，在这种情况下，它们被视为交替的键/值对。 |
| jsonb\_object(texts: string[]) → jsonb | 从文本数组中构建JSON或JSONB对象。 数组必须是一个具有偶数个成员的一维数组，这种情况下，它们被视为交替的键/值对。 |

序列函数

| **Function → Returns** | **Description** |
| --- | --- |
| currval(sequence\_name: string) → int | 返回此会话中此序列的nextval获取的最新值。 |
| lastval() → int | 在此会话中使用nextval最近获得的返回值。 |
| nextval(sequence\_name: string) → int | 推进给定的序列并返回其新值。 |
| setval(sequence\_name: string, value: int) → int | 设置给定序列的当前值。 nextval的下一次调用将返回value + Increment |
| setval(sequence\_name: string, value: int, is\_called: bool) → int | 设置给定序列的当前值。 如果is\_called为false，则对nextval的下一次调用将返回value; 否则返回value + Increment。 |

字符串和字节函数

| **Function → Returns** | **Description** |
| --- | --- |
| ascii(val: string) → int | 计算val中第一个字符的ASCII值。 |
| btrim(input: string, trim\_chars: string) → string | 从input的开头或结尾删除trim\_chars中包含的任何字符(递归应用)。  例如，btrim('doggie'，'eod')返回ggi。 |
| btrim(val: string) → string | 删除val的开头和结尾的所有空格。 |
| concat(string...) → string | 连接以逗号分隔的字符串列表。 |
| concat\_ws(string...) → string | 使用第一个参数作为后续参数的串联分隔符。  例如，concat\_ws('！'，'wow'，'great')返回wow!great。 |
| decode(text: string, format: string) → bytes | 以format指定的格式解码数据（仅支持“hex”和“escape”）。 |
| encode(data: bytes, format: string) → string | 以format指定的文本格式对数据进行编码（仅支持“hex”和“escape”）。 |
| from\_ip(val: bytes) → string | 将IP的字节字符串表示形式转换为其字符串表示形式。 |
| from\_uuid(val: bytes) → string | 将UUID的字节字符串表示形式转换为其字符串表示形式。 |
| initcap(val: string) → string | 将val的第一个字母大写。 |
| left(input: bytes, return\_set: int) → bytes | 从input返回前return\_set个字节。 |
| left(input: string, return\_set: int) → string | 从input返回前return\_set个字符。 |
| length(val: bytes) → int | 计算val中的字节数。 |
| length(val: string) → int | 计算val中的字符数。 |
| lower(val: string) → string | 将val中的所有字符转换为小写。 |
| ltrim(input: string, trim\_chars: string) → string | 从input的开头（左侧）删除trim\_chars中包含的任何字符（递归应用）。  例如，ltrim('doggie'，'od')返回ggie。 |
| ltrim(val: string) → string | 删除val开头（左侧）的所有空格。 |
| md5(bytes...) → string | 计算一组值的MD5哈希值。 |
| md5(string...) → string | 计算一组值的MD5哈希值。 |
| octet\_length(val: bytes) → int | 计算val中的字节数。 |
| octet\_length(val: string) → int | 计算用于表示val的字节数。 |
| overlay(input: string, overlay\_val: string, start\_pos: int) → string | 使用overlay\_val替换input中的字符，从start\_pos开始（从1开始）。  例如，overlay（'doggie'，'CAT'，2返回dCATie。 |
| overlay(input: string, overlay\_val: string, start\_pos: int, end\_pos: int) → string | 删除start\_pos和end\_pos之间input中的字符（count从1开始），然后在start\_pos中插入overlay\_val。 |
| regexp\_extract(input: string, regex: string) → string | 返回input中正则表达式regex的第一个匹配项。 |
| regexp\_replace(input: string, regex: string, replace: string) → string | 使用replace来替换input中正则表达式的匹配项 |
| repeat(input: string, repeat\_counter: int) → string | 将input进行repeat\_counter次连接。  例如， repeat（'dog'，2）返回 dogdog 。 |
| replace(input: string, find: string, replace: string) → string | 用replace替换input中的所有出现的find |
| reverse(val: string) → string | 反转字符串字符的顺序。 |
| right(input: bytes, return\_set: int) → bytes | 从input返回最后return\_set个字节。 |
| right(input: string, return\_set: int) → string | 返回input中最后return\_set个字符。 |
| rtrim(input: string, trim\_chars: string) → string | 从input的末尾（右侧）删除trim\_chars中包含的任何字符（递归应用）。  例如，rtrim（'doggie'，'ei'）返回dogg。 |
| rtrim(val: string) → string | 从val的末端（右侧）移除所有空格。 |
| sha1(bytes...) → string | 计算一组值的SHA1哈希值。 |
| sha1(string...) → string | 计算一组值的SHA1哈希值。 |
| sha256(bytes...) → string | 计算一组值的SHA256哈希值。 |
| sha256(string...) → string | 计算一组值的SHA256哈希值。 |
| sha512(bytes...) → string | 计算一组值的SHA512哈希值。 |
| sha512(string...) → string | 计算一组值的SHA512哈希值。 |
| split\_part(input: string, delimiter: string, return\_index\_pos: int) → string | 使用delimiter分隔符拆分input并返回return\_index\_pos位置的值（从1开始）。  例如，split\_part（'123.456.789.0'，'.'，3）返回789。 |
| strpos(input: string, find: string) → int | 计算字符串find在输入中开始的位置。  例如，strpos（'doggie'，'gie'）返回4。 |
| substr(input: string, regex: string) → string | 返回与正则表达式regex匹配的input子字符串。 |
| substr(input: string, regex: string, escape\_char: string) → string | 返回与正则表达式regex匹配的input子字符串，使用escape\_char作为转义字符而不是</code> |
| substr(input: string, start\_pos: int, end\_pos: int) → string | 返回start\_pos和end\_pos之间的input子字符串（count从1开始）。 |
| substr(input: string, substr\_pos: int) → string | 返回从substr\_pos开始的input子字符串（count从1开始）。 |
| substring(input: string, regex: string) → string | 返回与正则表达式匹配regex的input子字符串。 |
| substring(input: string, regex: string, escape\_char: string) → string | 返回与正则表达式匹配regex的input子字符串，使用escape\_char作为转义字符而不是</code>. |
| substring(input: string, start\_pos: int, end\_pos: int) → string | 返回start\_pos和end\_pos之间的input子字符串（count从1开始）。 |
| substring(input: string, substr\_pos: int) → string | 返回从substr\_pos开始的input子字符串（count从1开始）。 |
| to\_english(val: int) → string | 这个函数用英语基数说明它的参数的值。 |
| to\_hex(val: bytes) → string | 将val转换为十六进制表示形式。 |
| to\_ip(val: string) → bytes | 将IP的字符串表示形式转换为其字节字符串表示形式。 |
| to\_uuid(val: string) → bytes | 将UUID的字符串表示形式转换为其字节字符串表示形式。 |
| translate(input: string, find: string, replace: string) → string | 在输入中，将find中的第一个字符替换为replace的第一个字符; 对查找中的每个字符重复进行。  例如，translate（'doggie'，'dog'，'123'）; 返回1233ie。 |
| upper(val: string) → string | 将val中的所有字符转换为大写字母。 |

系统信息函数

| **Function → Returns** | **Description** |
| --- | --- |
| cluster\_logical\_timestamp() → decimal | 返回当前事务的逻辑时间。  此函数由CockroachDB开发人员保留用于测试目的，其定义可能会更改，恕不另行通知。  请注意，使用此功能会禁用服务器端优化，并可能增加争用或重试错误，或两者兼而有之。 |
| crdb\_internal.cluster\_id() → uuid | 返回集群ID。 |
| crdb\_internal.force\_error(errorCode: string, msg: string) → int | 此函数仅供CockroachDB的开发人员用于测试目的。 |
| crdb\_internal.force\_log\_fatal(msg: string) → int | 此函数仅供CockroachDB的开发人员用于测试目的。 |
| crdb\_internal.force\_panic(msg: string) → int | 此函数仅供CockroachDB的开发人员用于测试目的。 |
| crdb\_internal.force\_retry(val: interval) → int | 此函数仅供CockroachDB的开发人员用于测试目的。 |
| crdb\_internal.no\_constant\_folding(input: anyelement) → anyelement | 此函数仅供CockroachDB的开发人员用于测试目的。 |
| crdb\_internal.node\_executable\_version() → string | 返回此节点正在运行的CockroachDB的版本。 |
| crdb\_internal.set\_vmodule(vmodule\_string: string) → int | 此功能用于内部调试目的。 使用不当会严重影响性能。 |
| current\_database() → string | 返回当前数据库 |
| current\_schema() → string | 返回当前schema。 |
| current\_schemas(include\_pg\_catalog: bool) → string[] | 返回搜索路径中的有效schema。 |
| current\_user() → string | 返回当前用户。 提供此功能是为了与PostgreSQL兼容。 |
| version() → string | 返回节点的CockroachDB版本。 |

兼容性函数

| **Function → Returns** | **Description** |
| --- | --- |
| crdb\_internal.unary\_table() → setof tuple{} | 生成不包含任何值的单行的虚拟表。  此函数仅供CockroachDB的开发人员用于测试目的。 |
| format\_type(type\_oid: oid, typemod: int) → string | 返回由其类型OID和可能的类型修饰符标识的数据类型的SQL名称。 目前，类型修饰符会被忽略。 |
| generate\_series(start: int, end: int) → setof tuple{int} | 生成一个虚拟表，其中包含从start到end的整数值。 |
| generate\_series(start: int, end: int, step: int) → setof tuple{int} | 生成一个虚拟表，其中包含从start到end，并以step为步长的整型值。 |
| generate\_series(start: timestamp, end: timestamp, step: interval) → setof tuple{timestamp} | 生成一个虚拟表，其中包含从start到end，并以step为间隔的时间戳值。 |
| has\_any\_column\_privilege(table: string, privilege: string) → bool | 当前用户是否对表的所有行具有权限 |
| has\_any\_column\_privilege(table: oid, privilege: string) → bool | 当前用户是否对表的所有行具有权限 |
| has\_any\_column\_privilege(user: string, table: string, privilege: string) → bool | 用户是否对表的所有行具有权限 |
| has\_any\_column\_privilege(user: string, table: oid, privilege: string) → bool | 用户是否对表的所有行具有权限 |
| has\_any\_column\_privilege(user: oid, table: string, privilege: string) → bool | 用户是否对表的所有行具有权限 |
| has\_any\_column\_privilege(user: oid, table: oid, privilege: string) → bool | 用户是否对表的所有行具有权限 |
| has\_column\_privilege(table: string, column: int, privilege: string) → bool | 当前用户是否具有列权限 |
| has\_column\_privilege(table: string, column: string, privilege: string) → bool | 当前用户是否具有列权限 |
| has\_column\_privilege(table: oid, column: int, privilege: string) → bool | 当前用户是否具有列权限 |
| has\_column\_privilege(table: oid, column: string, privilege: string) → bool | 当前用户是否具有列权限 |
| has\_column\_privilege(user: string, table: string, column: int, privilege: string) → bool | 用户是否拥有列的权限 |
| has\_column\_privilege(user: string, table: string, column: string, privilege: string) → bool | 用户是否拥有列的权限 |
| has\_column\_privilege(user: string, table: oid, column: int, privilege: string) → bool | 用户是否拥有列的权限 |
| has\_column\_privilege(user: string, table: oid, column: string, privilege: string) → bool | 用户是否拥有列的权限 |
| has\_column\_privilege(user: oid, table: string, column: int, privilege: string) → bool | 用户是否拥有列的权限 |
| has\_column\_privilege(user: oid, table: string, column: string, privilege: string) → bool | 用户是否拥有列的权限 |
| has\_column\_privilege(user: oid, table: oid, column: int, privilege: string) → bool | 用户是否拥有列的权限 |
| has\_column\_privilege(user: oid, table: oid, column: string, privilege: string) → bool | 用户是否拥有列的权限 |
| has\_database\_privilege(database: string, privilege: string) → bool | 当前用户是否拥有数据库权限 |
| has\_database\_privilege(database: oid, privilege: string) → bool | 当前用户是否拥有数据库权限 |
| has\_database\_privilege(user: string, database: string, privilege: string) → bool | 用户是否拥有数据库权限 |
| has\_database\_privilege(user: string, database: oid, privilege: string) → bool | 用户是否拥有数据库权限 |
| has\_database\_privilege(user: oid, database: string, privilege: string) → bool | 用户是否拥有数据库权限 |
| has\_database\_privilege(user: oid, database: oid, privilege: string) → bool | 用户是否拥有数据库权限 |
| has\_foreign\_data\_wrapper\_privilege(fdw: string, privilege: string) → bool | 当前用户是否拥有 foreign-data wrapper 的权限 |
| has\_foreign\_data\_wrapper\_privilege(fdw: oid, privilege: string) → bool | 当前用户是否拥有 foreign-data wrapper 的权限 |
| has\_foreign\_data\_wrapper\_privilege(user: string, fdw: string, privilege: string) → bool | 当前用户是否拥有 foreign-data wrapper 的权限 |
| has\_foreign\_data\_wrapper\_privilege(user: string, fdw: oid, privilege: string) → bool | 当前用户是否拥有 foreign-data wrapper 的权限 |
| has\_foreign\_data\_wrapper\_privilege(user: oid, fdw: string, privilege: string) → bool | 当前用户是否拥有 foreign-data wrapper 的权限 |
| has\_foreign\_data\_wrapper\_privilege(user: oid, fdw: oid, privilege: string) → bool | 当前用户是否拥有 foreign-data wrapper 的权限 |
| has\_function\_privilege(function: string, privilege: string) → bool | 当前用户是否具有函数权限 |
| has\_function\_privilege(function: oid, privilege: string) → bool | 当前用户是否具有函数权限 |
| has\_function\_privilege(user: string, function: string, privilege: string) → bool | 用户是否具有函数权限 |
| has\_function\_privilege(user: string, function: oid, privilege: string) → bool | 用户是否具有函数权限 |
| has\_function\_privilege(user: oid, function: string, privilege: string) → bool | 用户是否具有函数权限 |
| has\_function\_privilege(user: oid, function: oid, privilege: string) → bool | 用户是否具有函数权限 |
| has\_language\_privilege(language: string, privilege: string) → bool | 当前用户是否拥有language权限 |
| has\_language\_privilege(language: oid, privilege: string) → bool | 当前用户是否拥有language权限 |
| has\_language\_privilege(user: string, language: string, privilege: string) → bool | 用户是否拥有language权限 |
| has\_language\_privilege(user: string, language: oid, privilege: string) → bool | 用户是否拥有language权限 |
| has\_language\_privilege(user: oid, language: string, privilege: string) → bool | 用户是否拥有language权限 |
| has\_language\_privilege(user: oid, language: oid, privilege: string) → bool | 用户是否拥有language权限 |
| has\_schema\_privilege(schema: string, privilege: string) → bool | 当前用户是否具有schema权限 |
| has\_schema\_privilege(schema: oid, privilege: string) → bool | 当前用户是否具有schema权限 |
| has\_schema\_privilege(user: string, schema: string, privilege: string) → bool | 用户是否拥有schema权限 |
| has\_schema\_privilege(user: string, schema: oid, privilege: string) → bool | 用户是否拥有schema权限 |
| has\_schema\_privilege(user: oid, schema: string, privilege: string) → bool | 用户是否拥有schema权限 |
| has\_schema\_privilege(user: oid, schema: oid, privilege: string) → bool | 用户是否拥有schema权限 |
| has\_sequence\_privilege(sequence: string, privilege: string) → bool | 当前用户是否具有序列权限 |
| has\_sequence\_privilege(sequence: oid, privilege: string) → bool | 当前用户是否具有序列权限 |
| has\_sequence\_privilege(user: string, sequence: string, privilege: string) → bool | 用户是否拥有序列权限 |
| has\_sequence\_privilege(user: string, sequence: oid, privilege: string) → bool | 用户是否拥有序列权限 |
| has\_sequence\_privilege(user: oid, sequence: string, privilege: string) → bool | 用户是否拥有序列权限 |
| has\_sequence\_privilege(user: oid, sequence: oid, privilege: string) → bool | 用户是否拥有序列权限 |
| has\_server\_privilege(server: string, privilege: string) → bool | 当前用户是否拥有外部服务器的权限 |
| has\_server\_privilege(server: oid, privilege: string) → bool | 当前用户是否拥有外部服务器的权限 |
| has\_server\_privilege(user: string, server: string, privilege: string) → bool | 用户是否拥有外部服务器的权限 |
| has\_server\_privilege(user: string, server: oid, privilege: string) → bool | 用户是否拥有外部服务器的权限 |
| has\_server\_privilege(user: oid, server: string, privilege: string) → bool | 用户是否拥有外部服务器的权限 |
| has\_server\_privilege(user: oid, server: oid, privilege: string) → bool | 用户是否拥有外部服务器的权限 |
| has\_table\_privilege(table: string, privilege: string) → bool | 当前用户是否拥有表的权限 |
| has\_table\_privilege(table: oid, privilege: string) → bool | 当前用户是否拥有表的权限 |
| has\_table\_privilege(user: string, table: string, privilege: string) → bool | 用户是否拥有表的权限 |
| has\_table\_privilege(user: string, table: oid, privilege: string) → bool | 用户是否拥有表的权限 |
| has\_table\_privilege(user: oid, table: string, privilege: string) → bool | 用户是否拥有表的权限 |
| has\_table\_privilege(user: oid, table: oid, privilege: string) → bool | 用户是否拥有表的权限 |
| has\_tablespace\_privilege(tablespace: string, privilege: string) → bool | 用户是否拥有表空间权限 |
| has\_tablespace\_privilege(tablespace: oid, privilege: string) → bool | 用户是否拥有表空间权限 |
| has\_tablespace\_privilege(user: string, tablespace: string, privilege: string) → bool | 用户是否拥有表空间权限 |
| has\_tablespace\_privilege(user: string, tablespace: oid, privilege: string) → bool | 用户是否拥有表空间权限 |
| has\_tablespace\_privilege(user: oid, tablespace: string, privilege: string) → bool | 用户是否拥有表空间权限 |
| has\_tablespace\_privilege(user: oid, tablespace: oid, privilege: string) → bool | 用户是否拥有表空间权限 |
| has\_type\_privilege(type: string, privilege: string) → bool | 当前用户是否具有type的权限 |
| has\_type\_privilege(type: oid, privilege: string) → bool | 当前用户是否具有type的权限 |
| has\_type\_privilege(user: string, type: string, privilege: string) → bool | 用户是否具有type的权限 |
| has\_type\_privilege(user: string, type: oid, privilege: string) → bool | 用户是否具有type的权限 |
| has\_type\_privilege(user: oid, type: string, privilege: string) → bool | 用户是否具有type的权限 |
| has\_type\_privilege(user: oid, type: oid, privilege: string) → bool | 用户是否具有type的权限 |
| json\_array\_elements(input: jsonb) → setof tuple{jsonb} | 将JSON数组扩展为JSON值集合。 |
| json\_array\_elements\_text(input: jsonb) → setof tuple{string} | 将JSON数组扩展为文本值集合。 |
| json\_each(input: jsonb) → setof tuple{string, jsonb} | 将最外层的JSON或JSONB对象扩展为键/值对集合。 |
| json\_each\_text(input: jsonb) → setof tuple{string, string} | 将最外层的JSON或JSONB对象扩展为键/值对集合。 返回的值是text类型。 |
| json\_object\_keys(input: jsonb) → setof tuple{string} | 返回最外层JSON对象中的有序键集。 |
| jsonb\_array\_elements(input: jsonb) → setof tuple{jsonb} | 将JSON数组扩展为一组JSON值。 |
| jsonb\_array\_elements\_text(input: jsonb) → setof tuple{string} | 将JSON数组扩展为一组文本值。 |
| jsonb\_each(input: jsonb) → setof tuple{string, jsonb} | 将最外层的JSON或JSONB对象扩展为一组键/值对。 |
| jsonb\_each\_text(input: jsonb) → setof tuple{string, string} | 将最外层的JSON或JSONB对象扩展为一组键/值对。 返回的值是text类型。 |
| jsonb\_object\_keys(input: jsonb) → setof tuple{string} | 返回最外层JSON对象中的有序键集。 |
| oid(int: int) → oid | 将整数转换为OID。 |
| pg\_get\_keywords() → setof tuple{string, string, string} | 生成包含SQL解析器能识别的关键字的虚拟表。 |
| unnest(input: anyelement[]) → anyelement | 以一组行（a set of rows）的形式返回输入数组 |

聚合函数

| **Function → Returns** | **Description** |
| --- | --- |
| array\_agg(arg1: bool) → bool[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: bytes) → bytes[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: date) → date[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: decimal) → decimal[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: float) → float[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: inet) → inet[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: int) → int[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: interval) → interval[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: string) → string[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: time) → time[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: timestamp) → timestamp[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: timestamptz) → timestamptz[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: uuid) → uuid[] | 将所选值聚合到一个数组中。 |
| array\_agg(arg1: oid) → oid[] | 将所选值聚合到一个数组中。 |
| avg(arg1: decimal) → decimal | 计算所选值的平均值。 |
| avg(arg1: float) → float | 计算所选值的平均值。 |
| avg(arg1: int) → decimal | 计算所选值的平均值。 |
| bool\_and(arg1: bool) → bool | 计算将所有选定值进行AND运算的布尔值。 |
| bool\_or(arg1: bool) → bool | 计算将所有选定值进行OR运算的布尔值。 |
| concat\_agg(arg1: bytes) → bytes | 连接所有选定的值。 |
| concat\_agg(arg1: string) → string | 连接所有选定的值。 |
| count(arg1: anyelement) → int | 计算所选元素的数量。 |
| count\_rows() → int | 计算行数。 |
| final\_stddev(arg1: decimal, arg2: decimal, arg3: int) → decimal | 计算所选的本地计算的平方差值的标准偏差。 |
| final\_stddev(arg1: float, arg2: float, arg3: int) → float | 计算所选的本地计算的平方差值的标准偏差。 |
| final\_variance(arg1: decimal, arg2: decimal, arg3: int) → decimal | 计算所选本地计算的平方差值的方差。 |
| final\_variance(arg1: float, arg2: float, arg3: int) → float | 计算所选本地计算的平方差值的方差。 |
| json\_agg(arg1: anyelement) → jsonb | 将值聚合为JSON或JSONB数组 |
| jsonb\_agg(arg1: anyelement) → jsonb | 将值聚合为JSON或JSONB数组 |
| max(arg1: bool) → bool | 返回所选值的最大值。 |
| max(arg1: bytes) → bytes | 返回所选值的最大值。 |
| max(arg1: date) → date | 返回所选值的最大值。 |
| max(arg1: decimal) → decimal | 返回所选值的最大值。 |
| max(arg1: float) → float | 返回所选值的最大值。 |
| max(arg1: inet) → inet | 返回所选值的最大值。 |
| max(arg1: int) → int | 返回所选值的最大值。 |
| max(arg1: interval) → interval | 返回所选值的最大值。 |
| max(arg1: string) → string | 返回所选值的最大值。 |
| max(arg1: time) → time | 返回所选值的最大值。 |
| max(arg1: timestamp) → timestamp | 返回所选值的最大值。 |
| max(arg1: timestamptz) → timestamptz | 返回所选值的最大值。 |
| max(arg1: uuid) → uuid | 返回所选值的最大值。 |
| max(arg1: jsonb) → jsonb | 返回所选值的最大值。 |
| max(arg1: oid) → oid | 返回所选值的最大值。 |
| min(arg1: bool) → bool | 返回选定值的最小值。 |
| min(arg1: bytes) → bytes | 返回选定值的最小值。 |
| min(arg1: date) → date | 返回选定值的最小值。 |
| min(arg1: decimal) → decimal | 返回选定值的最小值。 |
| min(arg1: float) → float | 返回选定值的最小值。 |
| min(arg1: inet) → inet | 返回选定值的最小值。 |
| min(arg1: int) → int | 返回选定值的最小值。 |
| min(arg1: interval) → interval | 返回选定值的最小值。 |
| min(arg1: string) → string | 返回选定值的最小值。 |
| min(arg1: time) → time | 返回选定值的最小值。 |
| min(arg1: timestamp) → timestamp | 返回选定值的最小值。 |
| min(arg1: timestamptz) → timestamptz | 返回选定值的最小值。 |
| min(arg1: uuid) → uuid | 返回选定值的最小值。 |
| min(arg1: jsonb) → jsonb | 返回选定值的最小值。 |
| min(arg1: oid) → oid | 返回选定值的最小值。 |
| sqrdiff(arg1: decimal) → decimal | 计算所选值的平均值的平方差的总和。 |
| sqrdiff(arg1: float) → float | 计算所选值的平均值的平方差的总和。 |
| sqrdiff(arg1: int) → decimal | 计算所选值的平均值的平方差的总和。 |
| stddev(arg1: decimal) → decimal | 计算所选值的标准偏差。 |
| stddev(arg1: float) → float | 计算所选值的标准偏差。 |
| stddev(arg1: int) → decimal | 计算所选值的标准偏差。 |
| sum(arg1: decimal) → decimal | 计算所选值的总和。 |
| sum(arg1: float) → float | 计算所选值的总和。 |
| sum(arg1: int) → decimal | 计算所选值的总和。 |
| sum(arg1: interval) → interval | 计算所选值的总和。 |
| sum\_int(arg1: int) → int | 计算所选值的总和。 |
| variance(arg1: decimal) → decimal | 计算所选值的方差。 |
| variance(arg1: float) → float | 计算所选值的方差。 |
| variance(arg1: int) → decimal | 计算所选值的方差。 |
| xor\_agg(arg1: bytes) → bytes | 计算所选值的按位XOR的值 |
| xor\_agg(arg1: int) → int | 计算所选值的按位XOR的值 |

操作符

下表按照从最高优先级到最低优先级的顺序列出了所有CockroachDB运算符，即它们在语句中的计算顺序。具有相同优先级的运算符是左关联的。 这意味着这些运算符从左侧开始向右移动。

| **Order of Precedence** | **Operator** | **Name** | **Operator Arity** |
| --- | --- | --- | --- |
| 1 | . | Member field access operator | binary |
| 2 | :: | [Type cast](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#explicit-type-coercions) | binary |
| 3 | - | Unary minus | unary (prefix) |
|  | ~ | Bitwise not | unary (prefix) |
| 4 | ^ | Exponentiation | binary |
| 5 | \* | Multiplication | binary |
|  | / | Division | binary |
|  | // | Floor division | binary |
|  | % | Modulo | binary |
| 6 | + | Addition | binary |
|  | - | Subtraction | binary |
| 7 | << | Bitwise left-shift | binary |
|  | >> | Bitwise right-shift | binary |
| 8 | & | Bitwise AND | binary |
| 9 | # | Bitwise XOR | binary |
| 10 | | | Bitwise OR | binary |
| 11 | || | Concatenation | binary |
|  | < ANY, SOME, ALL | [Multi-valued](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions.html#multi-valued-comparisons) "less than" comparison | binary |
|  | > ANY, SOME, ALL | [Multi-valued](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions.html#multi-valued-comparisons) "greater than" comparison | binary |
|  | = ANY, SOME, ALL | [Multi-valued](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions.html#multi-valued-comparisons) "equal" comparison | binary |
|  | <= ANY, SOME, ALL | [Multi-valued](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions.html#multi-valued-comparisons) "less than or equal" comparison | binary |
|  | >= ANY, SOME, ALL | [Multi-valued](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions.html#multi-valued-comparisons) "greater than or equal" comparison | binary |
|  | <> ANY / != ANY, <> SOME / != SOME, <> ALL / != ALL | [Multi-valued](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions.html#multi-valued-comparisons) "not equal" comparison | binary |
|  | [NOT] LIKE ANY, [NOT] LIKE SOME, [NOT] LIKE ALL | [Multi-valued](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions.html#multi-valued-comparisons) LIKE comparison | binary |
|  | [NOT] ILIKE ANY, [NOT] ILIKE SOME, [NOT] ILIKE ALL | [Multi-valued](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions.html#multi-valued-comparisons) ILIKE comparison | binary |
| 12 | [NOT] BETWEEN | Value is [not] within the range specified | binary |
|  | [NOT] BETWEEN SYMMETRIC | Like [NOT] BETWEEN, but in non-sorted order. For example, whereas a BETWEEN b AND c means b <= a <= c, a BETWEEN SYMMETRIC b AND c means (b <= a <= c) OR (c <= a <= b). | binary |
|  | [NOT] IN | Value is [not] in the set of values specified | binary |
|  | [NOT] LIKE | Matches [or not] LIKE expression, case sensitive | binary |
|  | [NOT] ILIKE | Matches [or not] LIKE expression, case insensitive | binary |
|  | [NOT] SIMILAR | Matches [or not] SIMILAR TO regular expression | binary |
|  | ~ | Matches regular expression, case sensitive | binary |
|  | !~ | Does not match regular expression, case sensitive | binary |
|  | ~\* | Matches regular expression, case insensitive | binary |
|  | !~\* | Does not match regular expression, case insensitive | binary |
| 13 | = | Equal | binary |
|  | < | Less than | binary |
|  | > | Greater than | binary |
|  | <= | Less than or equal to | binary |
|  | >= | Greater than or equal to | binary |
|  | !=, <> | Not equal | binary |
| 14 | IS [DISTINCT FROM] | Equal, considering NULL as value | binary |
|  | IS NOT [DISTINCT FROM] | a IS NOT b equivalent to NOT (a IS b) | binary |
|  | ISNULL, IS UNKNOWN , NOTNULL, IS NOT UNKNOWN | Equivalent to IS NULL / IS NOT NULL | unary (postfix) |
|  | IS NAN, IS NOT NAN | [Comparison with the floating-point NaN value](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#comparison-with-nan) | unary (postfix) |
|  | IS OF(...) | Type predicate | unary (postfix) |
| 15 | NOT | [Logical NOT](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#logical-operators) | unary |
| 16 | AND | [Logical AND](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#logical-operators) | binary |
| 17 | OR | [Logical OR](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/#logical-operators) | binary |

Supported Operations

| **#** | **Return** |
| --- | --- |
| int # int | int |
| #> | Return |
| jsonb #> string[] | jsonb |
| #>> | Return |
| jsonb #>> string[] | string |
| % | Return |
| decimal % decimal | decimal |
| decimal % int | decimal |
| float % float | float |
| int % decimal | decimal |
| int % int | int |
| & | Return |
| inet & inet | inet |
| int & int | int |
| \* | Return |
| decimal \* decimal | decimal |
| decimal \* int | decimal |
| decimal \* interval | interval |
| float \* float | float |
| float \* interval | interval |
| int \* decimal | decimal |
| int \* int | int |
| int \* interval | interval |
| interval \* decimal | interval |
| interval \* float | interval |
| interval \* int | interval |
| + | Return |
| +decimal | decimal |
| +float | float |
| +int | int |
| date + int | date |
| date + interval | timestamptz |
| date + time | timestamp |
| decimal + decimal | decimal |
| decimal + int | decimal |
| float + float | float |
| inet + int | inet |
| int + date | date |
| int + decimal | decimal |
| int + inet | inet |
| int + int | int |
| interval + date | timestamptz |
| interval + interval | interval |
| interval + time | time |
| interval + timestamp | timestamp |
| interval + timestamptz | timestamptz |
| time + date | timestamp |
| time + interval | time |
| timestamp + interval | timestamp |
| timestamptz + interval | timestamptz |
| - | Return |
| -decimal | decimal |
| -float | float |
| -int | int |
| -interval | interval |
| date - date | int |
| date - int | date |
| date - interval | timestamptz |
| date - time | timestamp |
| decimal - decimal | decimal |
| decimal - int | decimal |
| float - float | float |
| inet - inet | int |
| inet - int | inet |
| int - decimal | decimal |
| int - int | int |
| interval - interval | interval |
| jsonb - int | jsonb |
| jsonb - string | jsonb |
| time - interval | time |
| time - time | interval |
| timestamp - interval | timestamp |
| timestamp - timestamp | interval |
| timestamp - timestamptz | interval |
| timestamptz - interval | timestamptz |
| timestamptz - timestamp | interval |
| timestamptz - timestamptz | interval |
| -> | Return |
| jsonb -> int | jsonb |
| jsonb -> string | jsonb |
| ->> | Return |
| jsonb ->> int | string |
| jsonb ->> string | string |
| / | Return |
| decimal / decimal | decimal |
| decimal / int | decimal |
| float / float | float |
| int / decimal | decimal |
| int / int | decimal |
| interval / float | interval |
| interval / int | interval |
| // | Return |
| decimal // decimal | decimal |
| decimal // int | decimal |
| float // float | float |
| int // decimal | decimal |
| int // int | int |
| < | Return |
| bool < bool | bool |
| bytes < bytes | bool |
| collatedstring < collatedstring | bool |
| date < date | bool |
| date < timestamp | bool |
| date < timestamptz | bool |
| decimal < decimal | bool |
| decimal < float | bool |
| decimal < int | bool |
| float < decimal | bool |
| float < float | bool |
| float < int | bool |
| inet < inet | bool |
| int < decimal | bool |
| int < float | bool |
| int < int | bool |
| interval < interval | bool |
| oid < oid | bool |
| string < string | bool |
| time < time | bool |
| timestamp < date | bool |
| timestamp < timestamp | bool |
| timestamp < timestamptz | bool |
| timestamptz < date | bool |
| timestamptz < timestamp | bool |
| timestamptz < timestamptz | bool |
| tuple < tuple | bool |
| uuid < uuid | bool |
| << | Return |
| inet << inet | bool |
| int << int | int |
| <= | Return |
| bool <= bool | bool |
| bytes <= bytes | bool |
| collatedstring <= collatedstring | bool |
| date <= date | bool |
| date <= timestamp | bool |
| date <= timestamptz | bool |
| decimal <= decimal | bool |
| decimal <= float | bool |
| decimal <= int | bool |
| float <= decimal | bool |
| float <= float | bool |
| float <= int | bool |
| inet <= inet | bool |
| int <= decimal | bool |
| int <= float | bool |
| int <= int | bool |
| interval <= interval | bool |
| oid <= oid | bool |
| string <= string | bool |
| time <= time | bool |
| timestamp <= date | bool |
| timestamp <= timestamp | bool |
| timestamp <= timestamptz | bool |
| timestamptz <= date | bool |
| timestamptz <= timestamp | bool |
| timestamptz <= timestamptz | bool |
| tuple <= tuple | bool |
| uuid <= uuid | bool |
| <@ | Return |
| jsonb <@ jsonb | bool |
| = | Return |
| bool = bool | bool |
| bool[] = bool[] | bool |
| bytes = bytes | bool |
| bytes[] = bytes[] | bool |
| collatedstring = collatedstring | bool |
| date = date | bool |
| date = timestamp | bool |
| date = timestamptz | bool |
| date[] = date[] | bool |
| decimal = decimal | bool |
| decimal = float | bool |
| decimal = int | bool |
| decimal[] = decimal[] | bool |
| float = decimal | bool |
| float = float | bool |
| float = int | bool |
| float[] = float[] | bool |
| inet = inet | bool |
| inet[] = inet[] | bool |
| int = decimal | bool |
| int = float | bool |
| int = int | bool |
| int[] = int[] | bool |
| interval = interval | bool |
| interval[] = interval[] | bool |
| jsonb = jsonb | bool |
| oid = oid | bool |
| string = string | bool |
| string[] = string[] | bool |
| time = time | bool |
| time[] = time[] | bool |
| timestamp = date | bool |
| timestamp = timestamp | bool |
| timestamp = timestamptz | bool |
| timestamp[] = timestamp[] | bool |
| timestamptz = date | bool |
| timestamptz = timestamp | bool |
| timestamptz = timestamptz | bool |
| timestamptz = timestamptz | bool |
| tuple = tuple | bool |
| uuid = uuid | bool |
| uuid[] = uuid[] | bool |
| >> | Return |
| inet >> inet | bool |
| int >> int | int |
| ? | Return |
| jsonb ? string | bool |
| ?& | Return |
| jsonb ?& string[] | bool |
| ? | Return |
| jsonb ? string[] | bool |
| @> | Return |
| jsonb @> jsonb | bool |
| ILIKE | Return |
| string ILIKE string | bool |
| IN | Return |
| bool IN tuple | bool |
| bytes IN tuple | bool |
| collatedstring IN tuple | bool |
| date IN tuple | bool |
| decimal IN tuple | bool |
| float IN tuple | bool |
| inet IN tuple | bool |
| int IN tuple | bool |
| interval IN tuple | bool |
| jsonb IN tuple | bool |
| oid IN tuple | bool |
| string IN tuple | bool |
| time IN tuple | bool |
| timestamp IN tuple | bool |
| timestamptz IN tuple | bool |
| tuple IN tuple | bool |
| uuid IN tuple | bool |
| IS NOT DISTINCT FROM | Return |
| bool IS NOT DISTINCT FROM bool | bool |
| bool[] IS NOT DISTINCT FROM bool[] | bool |
| bytes IS NOT DISTINCT FROM bytes | bool |
| bytes[] IS NOT DISTINCT FROM bytes[] | bool |
| collatedstring IS NOT DISTINCT FROM collatedstring | bool |
| date IS NOT DISTINCT FROM date | bool |
| date IS NOT DISTINCT FROM timestamp | bool |
| date IS NOT DISTINCT FROM timestamptz | bool |
| date[] IS NOT DISTINCT FROM date[] | bool |
| decimal IS NOT DISTINCT FROM decimal | bool |
| decimal IS NOT DISTINCT FROM float | bool |
| decimal IS NOT DISTINCT FROM int | bool |
| decimal[] IS NOT DISTINCT FROM decimal[] | bool |
| float IS NOT DISTINCT FROM decimal | bool |
| float IS NOT DISTINCT FROM float | bool |
| float IS NOT DISTINCT FROM int | bool |
| float[] IS NOT DISTINCT FROM float[] | bool |
| inet IS NOT DISTINCT FROM inet | bool |
| inet[] IS NOT DISTINCT FROM inet[] | bool |
| int IS NOT DISTINCT FROM decimal | bool |
| int IS NOT DISTINCT FROM float | bool |
| int IS NOT DISTINCT FROM int | bool |
| int[] IS NOT DISTINCT FROM int[] | bool |
| interval IS NOT DISTINCT FROM interval | bool |
| interval[] IS NOT DISTINCT FROM interval[] | bool |
| jsonb IS NOT DISTINCT FROM jsonb | bool |
| oid IS NOT DISTINCT FROM oid | bool |
| string IS NOT DISTINCT FROM string | bool |
| string[] IS NOT DISTINCT FROM string[] | bool |
| time IS NOT DISTINCT FROM time | bool |
| time[] IS NOT DISTINCT FROM time[] | bool |
| timestamp IS NOT DISTINCT FROM date | bool |
| timestamp IS NOT DISTINCT FROM timestamp | bool |
| timestamp IS NOT DISTINCT FROM timestamptz | bool |
| timestamp[] IS NOT DISTINCT FROM timestamp[] | bool |
| timestamptz IS NOT DISTINCT FROM date | bool |
| timestamptz IS NOT DISTINCT FROM timestamp | bool |
| timestamptz IS NOT DISTINCT FROM timestamptz | bool |
| timestamptz IS NOT DISTINCT FROM timestamptz | bool |
| tuple IS NOT DISTINCT FROM tuple | bool |
| unknown IS NOT DISTINCT FROM unknown | bool |
| uuid IS NOT DISTINCT FROM uuid | bool |
| uuid[] IS NOT DISTINCT FROM uuid[] | bool |
| LIKE | Return |
| string LIKE string | bool |
| SIMILAR TO | Return |
| string SIMILAR TO string | bool |
| ^ | Return |
| decimal ^ decimal | decimal |
| decimal ^ int | decimal |
| float ^ float | float |
| int ^ decimal | decimal |
| int ^ int | int |
|  | Return |
| inet inet | inet |
| int int | int |
|  | Return |
| bool bool[] | bool[] |
| bool[] bool | bool[] |
| bool[] bool[] | bool[] |
| bytes bytes | bytes |
| bytes bytes[] | bytes[] |
| bytes[] bytes | bytes[] |
| bytes[] bytes[] | bytes[] |
| date date[] | date[] |
| date[] date | date[] |
| date[] date[] | date[] |
| decimal decimal[] | decimal[] |
| decimal[] decimal | decimal[] |
| decimal[] decimal[] | decimal[] |
| float float[] | float[] |
| float[] float | float[] |
| float[] float[] | float[] |
| inet inet[] | inet[] |
| inet[] inet | inet[] |
| inet[] inet[] | inet[] |
| int int[] | int[] |
| int[] int | int[] |
| int[] int[] | int[] |
| interval interval[] | interval[] |
| interval[] interval | interval[] |
| interval[] interval[] | interval[] |
| jsonb jsonb | jsonb |
| oid oid | oid |
| string string | string |
| string string[] | string[] |
| string[] string | string[] |
| string[] string[] | string[] |
| time time[] | time[] |
| time[] time | time[] |
| time[] time[] | time[] |
| timestamp timestamp[] | timestamp[] |
| timestamp[] timestamp | timestamp[] |
| timestamp[] timestamp[] | timestamp[] |
| timestamptz timestamptz | timestamptz |
| timestamptz timestamptz | timestamptz |
| timestamptz timestamptz | timestamptz |
| uuid uuid[] | uuid[] |
| uuid[] uuid | uuid[] |
| uuid[] uuid[] | uuid[] |
| ~ | Return |
| ~inet | inet |
| ~int | int |
| string ~ string | bool |
| ~\* | Return |
| string ~\* string | bool |