查询（SELECT）

基本语法：

```
SELECT [DISTINCT] expr_list
[FROM [db.]table | (subquery) | table_function] [FINAL]
[SAMPLE sample_coeff]
[ARRAY JOIN ...]
[GLOBAL] ANY|ALL INNER|LEFT JOIN (subquery)|table USING columns_list
[PREWHERE expr]
[WHERE expr]
[GROUP BY expr_list] [WITH TOTALS]
[HAVING expr]
[ORDER BY expr_list]
[LIMIT [n, ]m]
[UNION ALL ...]
[INTO OUTFILE filename]
[FORMAT format]
[LIMIT n BY columns]
```

所有的子句都是可选的，除了SELECT之后的表达式列表（expr_list）。下面将选择部分子句进行说明。ClickHouse官网中文文档有更详细说明，请参考查询语法。

简单查询语句示例：

```
SELECT SUM(LO_REVENUE) AS REVENUE
FROM lineorder
WHERE ((LO_DISCOUNT >= 1) AND (LO_DISCOUNT <= 3)) AND (LO_QUANTITY < 25)
```

SAMPLE 子句

通过SAMPLE子句用户可以进行近似查询处理，近似查询处理仅能工作在MergeTree*类型的表中，并且在创建表时需要您指定采样表达式。

SAMPLE子句可以使用SAMPLE k来表示，其中k可以是0到1的小数值，或者是一个足够大的正整数值。

当k为0到1的小数时，查询将使用'k'作为百分比选取数据。例如，SAMPLE 0.1查询只会检索数据总量的10%。当k为一个足够大的正整数时，查询将使用'k'作为最大样本数。例如，SAMPLE 10000000查询只会检索最多10,000,000行数据。

示例：

```
SELECT
Title,
count() * 10 AS PageViews
FROM hits_distributed
SAMPLE 0.1
WHERE
CounterID = 34
AND toDate(EventDate) >= toDate('2013-01-29')
AND toDate(EventDate) <= toDate('2013-02-04')
AND NOT DontCountHits
```

```
AND NOT Refresh
AND Title != ''
GROUP BY Title
ORDER BY PageViews DESC LIMIT 1000;
```

上述例子中，查询将检索数据总量的0.1（10%）的数据。值得注意的是，查询不会自动校正聚合函数最终的结果，所以为了得到更加精确的结果，需要将count()的结果手动乘以10。

当使用像SAMPLE 10000000这样的方式进行近似查询时，由于没有了任何关于将会处理了哪些数据或聚合函数应该被乘以几的信息，所以这种方式不适合在这种场景下使用。

使用相同的采样率得到的结果总是一致的：如果我们能够看到所有可能存在在表中的数据，那么相同的采样率总是能够得到相同的结果（在建表时使用相同的采样表达式），换句话说，系统在不同的时间，不同的服务器，不同表上总以相同的方式对数据进行采样。

例如，我们可以使用采样的方式获取到与不进行采样相同的用户ID的列表。这将表明，您可以在IN子查询中使用采样，或者使用采样的结果与其他查询进行关联。

ARRAY JOIN 子句

ARRAY JOIN子句可以帮助查询并进行与数组和nested数据类型的连接。它有点类似arrayJoin函数，但它的功能更广泛。

ARRAY JOIN本质上等同于INNER JOIN数组。例如：

```
CREATE TABLE ck_array(id UInt8,scores Array(UInt8)) ENGINE = Memory;
```

```
:) CREATE TABLE ck_array(id UInt8,scores Array(UInt8)) ENGINE = Memory;

CREATE TABLE ck_array
(
    `id` UInt8,
    `scores` Array(UInt8)
)
ENGINE = Memory

Query id: ef0719fb-3396-426d-a456-93a436fac4df

Ok.

0 rows in set. Elapsed: 0.003 sec.
```

INSERT INTO ck_array VALUES (1,[89,91]),(2,[77,67]),(3,[79,89])

```
:) INSERT INTO ck_array VALUES (1,[89,91]),(2,[77,67]),(3,[79,89])

INSERT INTO ck_array VALUES

Query id: 9c971ae4-4cf8-4753-b626-7958dab7f515

Ok.

3 rows in set. Elapsed: 0.006 sec.
```

select * from ck_array;

```
:) select * from ck_array;

SELECT *
FROM ck_array

Query id: a54cf0d2-ea8f-4ce0-8075-bc9e14d9402c

┌─id─┬─scores─┐
│  1 │ [89,91] │
│  2 │ [77,67] │
│  3 │ [79,89] │
└────┴─────────┘

3 rows in set. Elapsed: 0.006 sec.
```

SELECT id, scores FROM ck_array ARRAY JOIN scores

```
:) SELECT id, scores FROM ck_array ARRAY JOIN scores

SELECT
    id,
    scores
FROM ck_array
ARRAY JOIN scores

Query id: c78148dd-2caa-46a8-b29a-bcbe7ba2dc77

┌─id─┬─scores─┐
│  1 │     89 │
│  1 │     91 │
│  2 │     77 │
│  2 │     67 │
│  3 │     79 │
│  3 │     89 │
└────┴────────┘

6 rows in set. Elapsed: 0.003 sec.
```

Join 语句Null的处理

请参考join_use_nulls、Nullable、NULL。

WHERE 子句

如果存在WHERE子句，则在该子句中必须包含一个UInt8类型的表达式。这个表达式通常是一个带有比较和逻辑的表达式。这个表达式将会在所有数据转换前用来过滤数据。

如果在支持索引的数据库表引擎中，这个表达式将被评估是否使用索引。

PREWHERE 子句

这个子句与WHERE子句的意思相同。主要的不同之处在于表数据的读取。当使用PREWHERE时，首先只读取PREWHERE表达式中需要的列。然后再根据PREWHERE执行的结果读取其他

需要的列。

如果在过滤条件中有少量不适合索引过滤的列,但是它们又可以提供很强的过滤能力。这时使用PREWHERE是有意义的,因为它将帮助减少数据的读取。

例如:在一个需要提取大量列的查询中为少部分列编写PREWHERE是很有作用的。

提示:PREWHERE仅支持*MergeTree系列引擎。在一个查询中可以同时指定PREWHERE和WHERE,在这种情况下,PREWHERE优先于WHERE执行。PREWHERE不适合用于已经存在于索引中的列,因为当列已经存在于索引中的情况下,只有满足索引的数据块才会被读取。如果将'optimize_move_to_prewhere'设置为1,并且在查询中不包含PREWHERE,则系统将自动的把适合PREWHERE表达式的部分从WHERE中抽离到PREWHERE中。

WITH TOTALS 修饰符

如果您指定了WITH TOTALS修饰符,将会在结果中得到一个被额外计算出的行。在这一行中将包含所有key的默认值(零或者空值),以及所有聚合函数对所有被选择数据行的聚合结果。

该行仅在JSON*, TabSeparated*, Pretty*输出格式中与其他行分开输出。