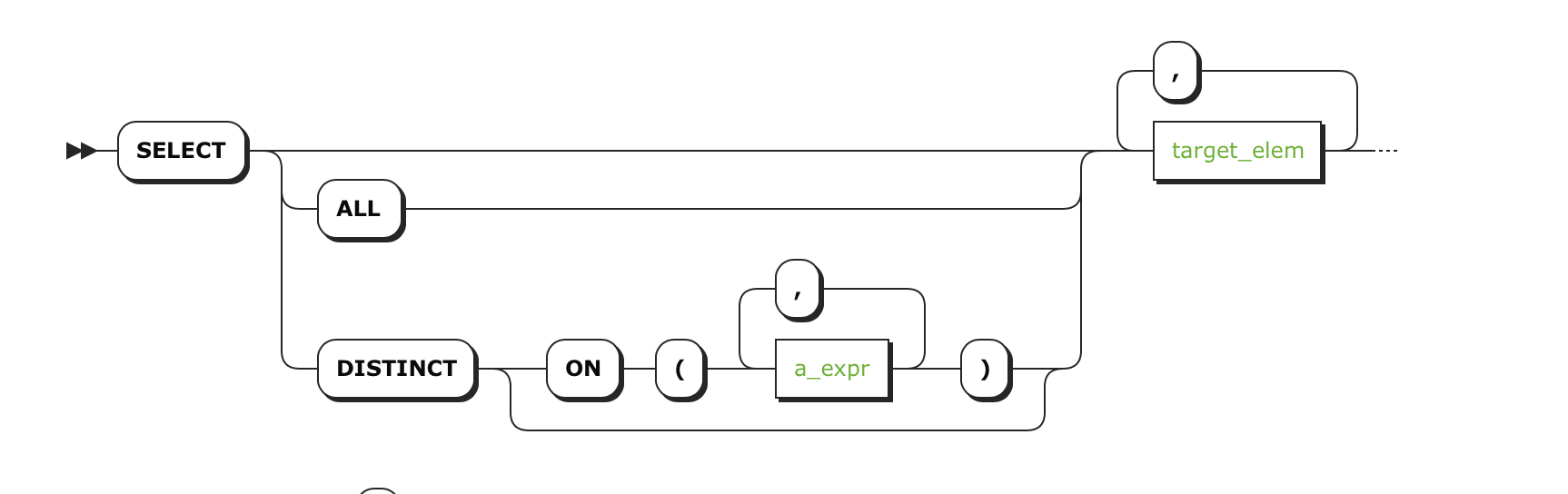
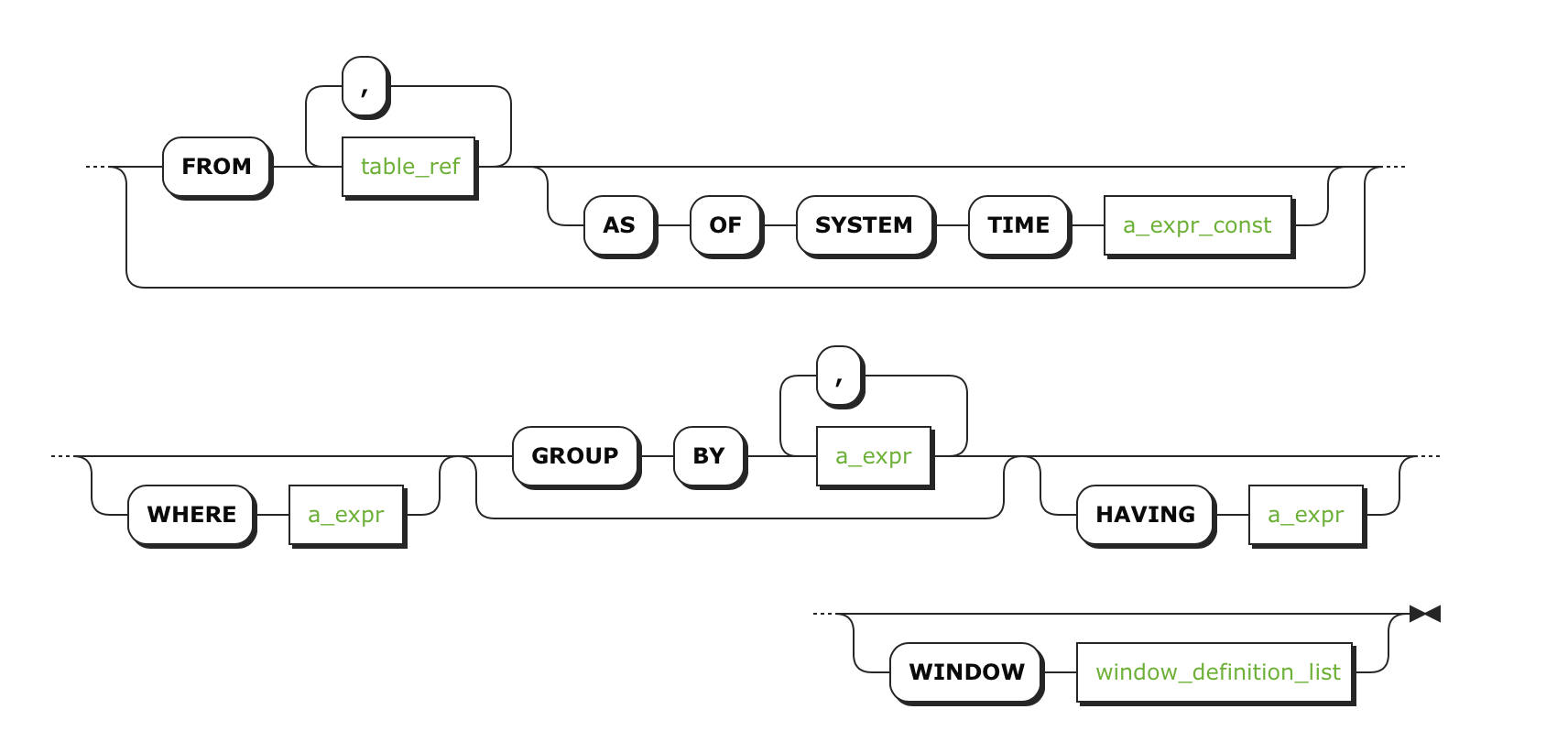
简单的 SELECT 是主要的用于读取和处理现有数据的SQL 语句。

当用作独立语句时， 简单的 SELECT 叫做 "the SELECT statement"。但是，它也能与其他结构组合，形成更复杂的[selection queries](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/selection-queries/)，叫做 [selection clause](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/selection-queries/#selection-clauses) 。

Synopsis

Tips: 简单 SELECT clause 还包含这里没有涉及的应用, 例如执行 [functions](http://doc.cockroachchina.baidu.com/#develop/sql-statements/SELECT/functions-and-operators.html) ，像 SELECT current\_timestamp();。

所需权限

用户必须有要操作表的 SELECT [权限](http://doc.cockroachchina.baidu.com/#deploy/access-management/privileges/) 。

参数

| **Parameter** | **Description** |
| --- | --- |
| DISTINCT or ALL | 具体信息请看： [Eliminate Duplicate Rows](http://doc.cockroachchina.baidu.com/#develop/sql-statements/SELECT/#eliminate-duplicate-rows). |
| DISTINCT ON后面的括号内跟着 ( a\_expr [, ...] ) | DISTINCT ON后面的括号内跟着[scalar expressions](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/)。 具体信息请看具体信息请看：[Eliminate Duplicate Rows](http://doc.cockroachchina.baidu.com/#develop/sql-statements/SELECT/#eliminate-duplicate-rows)。 |
| target\_elem | 用于计算每个结果行中的列的[scalar expression](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/)， 或者 \*会自动获取来自FROM子句的所有列。  如果 target\_elem包含一个 [aggregate function](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/#aggregate-functions)，则GROUP BY 子句会被使用来进一步控制这个聚合函数。。 |
| table\_ref | 我们需要从[table expression](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/table-expressions/)获取数据。  如果在 FROM子句中使用两个或多个逗号隔开的表达式，这和使用[CROSS JOIN](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/join-expressions/) 表达式是一样的。 |
| AS OF SYSTEM TIME timestamp | 可以检索timestamp时刻的数据。 **Note**: 因为 AS OF SYSTEM TIME 返回历史数据，所以返回的数据可能是陈旧的。 |
| WHERE a\_expr | 仅仅检索使表达式a\_expr返回TRUE的行，而且表达式a\_expr必须是使用列(例如 <column>=<value>）的返回布尔值的[scalar expression](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/)。 |
| GROUP BY a\_expr | 当在target\_elem 或者 HAVING 中使用[aggregate functions](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/#aggregate-functions)后，列出GROUP BY后的列分组。 |
| HAVING a\_expr | 仅仅检索a\_expr表达式返回TRUE的聚合函数分组，而a\_expr为使用聚合函数(例如， <aggregate function> = <value>)且返回布尔值的[scalar expression](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/)。  HAVING 和 WHERE 子句一样工作, 但是该子句是针对聚合函数的。 |
| WINDOW window\_definition\_list | 一些 [window functions definitions](http://doc.cockroachchina.baidu.com/#develop/window-functions/)。 |

消除重复行

DISTINCT子句主要是用于消除重复行。

默认情况下，或者使用ALL时，SELECT会返回所有行的数据，不会消除重复行信息。当使用 DISTINCT后，会消除重复行信息。

当DISTINCT后没有ON时，如果通过SELECT选出两行数据的所有结果完全一样，则判定这两行数据是重复的。 当DISTINCT后使用ON时，如果两行数据通过表达式 [scalar expressions](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/)计算出的值一样，则判定两行数据是重复的。当依据DISTINCT ON判定两行数据是重复时，那么通过 [ORDER BY](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/ordering-query-results/) 子句排序后，排在前面的一行数据将用于计算剩下的目标表达式。当没有使用ORDER BY时，随机选择任意一个重复行作为第一行。 当DISTINCT后使用ON时，如果通过SELECT选出两行数据通过表达式 [scalar expressions](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/)计算出的值一样，则判定两行数据是重复的。当依据DISTINCT ON判定两行数据是重复时，那么通过 [ORDER BY](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/ordering-query-results/) 子句排序后，排在前面的一行数据将用于计算剩下的目标表达式。当没有使用ORDER BY时，随机从两行数据选择一行数据。

例子

选择列

检索特定列

通过一个逗号隔开的列表检索特定列数据：

> **SELECT** id, name, balance

**FROM** accounts;

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

| id | name | balance |

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

| 1 | Bjorn Fairclough | 1200 |

| 2 | Bjorn Fairclough | 2500 |

| 3 | Arturo Nevin | 250 |

[ truncated ]

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

检索所有列

通过\*检索所有列的数据：

> **SELECT** \*

**FROM** accounts;

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

| id | name | balance | type | state\_opened |

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

| 1 | Bjorn Fairclough | 1200 | checking | AL |

| 2 | Bjorn Fairclough | 2500 | savings | AL |

| 3 | Arturo Nevin | 250 | checking | AK |

[ truncated ]

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

过滤数据

使用单独的条件过滤

在WHERE自己中使用返回布尔值，并且使用列的表达式过滤数据：

> **SELECT** name, balance

**FROM** accounts

**WHERE** balance < 300;

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

| name | balance |

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

| Arturo Nevin | 250 |

| Akbar Jinks | 250 |

| Andrea Maas | 250 |

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

多个条件过滤数据

在WHERE 子句中，使用AND或者OR联合多个条件。还可以使用NOT创建负过滤器。

> **SELECT** \*

**FROM** accounts

**WHERE** balance > 2500 **AND** **NOT** type = 'checking';

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

| id | name | balance | type | state\_opened |

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

| 4 | Tullia Romijnders | 3000 | savings | AK |

| 62 | Ruarc Mathews | 3000 | savings | OK |

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

选择无重复的行

没有[Primary Key](http://doc.cockroachchina.baidu.com/#develop/constraints/primary-key/) 或者[Unique](http://doc.cockroachchina.baidu.com/#develop/constraints/unique/) 约束的列可能存在相同的值：

> **SELECT** name

**FROM** accounts

**WHERE** state\_opened = 'VT';

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

| name |

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

| Sibylla Malone |

| Sibylla Malone |

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

使用DISTINCT可以去除检索数据中的重复数据。

> **SELECT** **DISTINCT** name

**FROM** accounts

**WHERE** state\_opened = 'VT';

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

| name |

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

| Sibylla Malone |

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

使用列表过滤

使用 WHERE <column> IN (<comma separated list of values>) 和使用OR组合列表中的数据是一样的。

> **SELECT** name, balance, state\_opened

**FROM** accounts

**WHERE** state\_opened **IN** ('AZ', 'NY', 'WA');

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

| name | balance | state\_opened |

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

| Naseem Joossens | 300 | AZ |

| Aygün Sanna | 900 | NY |

| Carola Dahl | 800 | NY |

| Edna Barath | 750 | WA |

| Edna Barath | 2200 | WA |

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

变更输出列的名字

在输出中，不是输出检索表中的列的名字，而是通过AS自定义名字：

> **SELECT** name **AS** NY\_accounts, balance

**FROM** accounts

**WHERE** state\_opened = 'NY';

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

| NY\_accounts | balance |

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

| Aygün Sanna | 900 |

| Carola Dahl | 800 |

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

这个操作*不会* 改变检索表中列的名字。如果要改变表中列的名字，可以使用[RENAME COLUMN](http://doc.cockroachchina.baidu.com/#develop/sql-statements/RENAME-COLUMN/)实现。

搜索字符串值

使用LIKE在列中搜索部分[string](http://doc.cockroachchina.baidu.com/#develop/data-types/STRING/) 匹配，支持以下通配符匹配：

* % 会匹配0或多个任意字符
* \_ 会匹配1个任意字符

例如:

> **SELECT** id, name, type

**FROM** accounts

**WHERE** name **LIKE** 'Anni%';

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

| id | name | type |

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

| 58 | Annibale Karga | checking |

| 59 | Annibale Karga | savings |

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

聚合函数

[Aggregate functions](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/#aggregate-functions)会对检索出来的行进行计算。

在整个列上进行聚合函数

通过使用聚合函数作为一个target\_elem，可以在整个列上进行计算。

> **SELECT** **MIN**(balance)

**FROM** accounts;

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

| MIN(balance) |

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

| 250 |

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

还可以把聚合函数检索到的值作为WHERE子句表达式的一部分。

> **SELECT** id, name, balance

**FROM** accounts

**WHERE** balance = (

**SELECT**

**MIN**(balance)

**FROM** accounts

);

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

| id | name | balance |

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

| 3 | Arturo Nevin | 250 |

| 10 | Henrik Brankovic | 250 |

| 26 | Odalys Ziemniak | 250 |

| 35 | Vayu Soun | 250 |

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

在检索出的行数据上进行聚合函数操作

通过select的过滤语句，在检索出的行数据上进行聚合函数操作：

> **SELECT** **SUM**(balance)

**FROM** accounts

**WHERE** state\_opened **IN** ('AZ', 'NY', 'WA');

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

| SUM(balance) |

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

| 4950 |

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

在聚合函数中加入过滤列

通过在 target\_elem中加入FILTER (WHERE <Boolean expression>)，可以过滤出需要被聚合函数处理的行，在FILTER表达式中返回FALSE 或者 NULL 的行数据将不会加入到聚合函数中：

> **SELECT** **count**(\*) **AS** unfiltered, **count**(\*) FILTER (**WHERE** balance > 1500) **AS** filtered **FROM** accounts;

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

| unfiltered | filtered |

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

| 84 | 14 |

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

创建聚合组

如果我们不想在所有的检索行进行聚合函数，可以将检索出的列进行分组，然后在每个分组进行聚合函数的计算。

创建聚合分组后，target\_elem的每列必须包含在 GROUP BY中。

例如：

> **SELECT** state\_opened **AS** state, **SUM**(balance) **AS** state\_balance

**FROM** accounts

**WHERE** state\_opened **IN** ('AZ', 'NY', 'WA')

**GROUP** **BY** state\_opened;

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

| state | state\_balance |

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

| AZ | 300 |

| NY | 1700 |

| WA | 2950 |

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

过滤聚合组

使用HAVING过滤聚合组，HAVING子句必须返回布尔值。

例如:

> **SELECT** state\_opened, **AVG**(balance) **as** **avg**

**FROM** accounts

**GROUP** **BY** state\_opened

**HAVING** **AVG**(balance) **BETWEEN** 1700 **AND** 50000;

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

| state\_opened | avg |

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

| AR | 3700.00 |

| UT | 1750.00 |

| OH | 2500.00 |

| AL | 1850.00 |

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

在Having子句中使用聚合函数

聚合函数不一定只能作为target\_elem参数，可以在HAVING子句中使用聚合函数：

例如:

> **SELECT** name, state\_opened

**FROM** accounts

**WHERE** state\_opened **in** ('LA', 'MO')

**GROUP** **BY** name, state\_opened

**HAVING** **COUNT**(name) > 1;

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

| name | state\_opened |

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

| Yehoshua Kleid | MO |

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

选择历史数据

CockraochDB允许你查找一个给定时间值的数据，这个时间值通过AS OF SYSTEM TIME子句设定，可支持不同的格式。更多相关信息，请看： [AS OF SYSTEM TIME](http://doc.cockroachchina.baidu.com/#develop/sql-statements/as-of-system-time/)。以下是几个例子：

SELECT 子句的高级用法

CockroachDB支持多种方法将SELECT语句的结果组合在一起。

更多相关信息，请看 [Selection Queries](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/selection-queries/) 。一些例子如下：

对结果进行排序与限制

为了对SELECT语句的结果进行排序与限制结果的行数，可以将 SELECT语句与 ORDER BY或者LIMIT /OFFSET进行组合形成一个新的[selection query](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/selection-queries/)或者[subquery](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/table-expressions/#subqueries-as-table-expressions)。

更多相关信息，请看See [Ordering Query Results](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/ordering-query-results/) 和[Limiting Query Results](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/limit-query-results/) 。

Note: 当一个查询中没有ORDER BY时，查询搜索出的结果不会按照任何一致性标准进行排序。CockroachDB按照协调节点接受到的顺序返回它们。  
  
而且，当CockroachDB使用ASC规则 进行排序时，会把NULL排在最前面，使用DESC规则时，排在最后面。而PostgreSQL恰好与CockroachDB相反。

组合多个查询的结果

两个或多个的查询结果可以按照如下方式组合在一起：

* 依据特定列上的条件，使用 [join expressions](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/join-expressions/) 组合结果。
* 依据包含/排除规则，用[set operations](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/selection-queries/#set-operations)组合数据。

See Also

* [Scalar Expressions](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/scalar-expressions/)
* [Selection Queries](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/selection-queries/)
* [Selection Clauses](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/selection-queries/#selection-clauses)
* [Set Operations](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/selection-queries/#set-operations)
* [Table Expressions](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/table-expressions/)
* [Ordering Query Results](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/ordering-query-results/)
* [Limiting Query Results](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/limit-query-results/)
* [SQL Performance Best Practices](http://doc.cockroachchina.baidu.com/#develop/performance-optimization/sql-best-practices/)