fetch

- Description
- Parameters
- Outputs
- Notes
- Examples
- Compatibility
- See Also

FETCH

cursor

FETCH

FETCH

7

SQL - Language Statements

FETCH

retrieve rows from a query using a cursor

FETCH [

direction

] [FROM | IN]

cursor_name

where

direction

can be one of:

NEXT PRIOR FIRST LAST ABSOLUTE

count

RELATIVE

count

count

ALL FORWARD FORWARD

count

FORWARD ALL BACKWARD BACKWARD

count

BACKWARD ALL

Description

`FETCH` retrieves rows using a previously-created cursor.

A cursor has an associated position, which is used by `FETCH`. The cursor position can be before the first row of the query result, on any particular row of the result, or after the last row of the result. When created, a cursor is positioned before the first row. After fetching some rows, the cursor is positioned on the row most recently retrieved. If `FETCH` runs off the end of the available rows then the cursor is left positioned after the last row, or before the first row if fetching backward. `FETCH ALL` or `FETCH BACKWARD ALL` will always leave the cursor positioned after the last row or before the first row.

The forms `NEXT`, `PRIOR`, `FIRST`, `LAST`, `ABSOLUTE`, `RELATIVE` fetch a single row after moving the cursor appropriately. If there is no such row, an empty result is returned, and the cursor is left positioned before the first row or after the last row as appropriate.

The forms using `FORWARD` and `BACKWARD` retrieve the indicated number of rows moving in the forward or backward direction, leaving the cursor positioned on the last-returned row (or after/before all rows, if the <count> exceeds the number of rows available).

`RELATIVE 0`, `FORWARD 0`, and `BACKWARD 0` all request fetching the current row without moving the cursor, that is, re-fetching the most recently fetched row. This will succeed unless the cursor is

positioned before the first row or after the last row; in which case, no row is returned.

> This page describes usage of cursors at the SQL command level. If you are trying to use cursors inside a PL/pgSQL function, the rules are different — see ???.

Parameters

<direction>
<direction> defines the fetch direction and number of rows to fetch. It can be one of the following:

`NEXT`
Fetch the next row. This is the default if <direction> is omitted.

`PRIOR`
Fetch the prior row.

`FIRST`
Fetch the first row of the query (same as `ABSOLUTE 1`).

`LAST`
Fetch the last row of the query (same as `ABSOLUTE -1`).

`ABSOLUTE count`
Fetch the <count>'th row of the query, or the `abs(count)`'th row from the end if <count> is negative. Position before first row or after last row if <count> is out of range; in particular, `ABSOLUTE 0` positions before the first row.

`RELATIVE count`
Fetch the <count>'th succeeding row, or the `abs(count)`'th prior row if <count> is negative. `RELATIVE 0` re-fetches the current row, if any.

<count>
Fetch the next <count> rows (same as `FORWARD count`).

`ALL`
Fetch all remaining rows (same as `FORWARD ALL`).

`FORWARD`
Fetch the next row (same as `NEXT`).

`FORWARD count`
Fetch the next <count> rows. `FORWARD 0` re-fetches the current row.

`FORWARD ALL`
Fetch all remaining rows.

`BACKWARD`
Fetch the prior row (same as `PRIOR`).

`BACKWARD count`
Fetch the prior <count> rows (scanning backwards). `BACKWARD 0` re-fetches the current row.

`BACKWARD ALL`
Fetch all prior rows (scanning backwards).

<count>
<count> is a possibly-signed integer constant, determining the location or number of rows to fetch. For `FORWARD` and `BACKWARD` cases, specifying a negative <count> is equivalent to changing the sense of `FORWARD` and `BACKWARD`.

<cursor_name>
An open cursor's name.

Outputs

On successful completion, a `FETCH` command returns a command tag of the form

```
FETCH count
```

The <count> is the number of rows fetched (possibly zero). Note that in psql, the command tag will not actually be displayed, since psql displays the fetched rows instead.

Notes

The cursor should be declared with the `SCROLL` option if one intends to use any variants of `FETCH` other than `FETCH NEXT` or `FETCH FORWARD` with a positive count. For simple queries PostgreSQL will allow backwards fetch from cursors not declared with `SCROLL`, but this behavior is best not relied on. If the cursor is declared with `NO SCROLL`, no backward fetches are allowed.

`ABSOLUTE` fetches are not any faster than navigating to the desired row with a relative move: the underlying implementation must traverse all the intermediate rows anyway. Negative absolute

fetches are even worse: the query must be read to the end to find the last row, and then traversed backward from there. However, rewinding to the start of the query (as with `FETCH ABSOLUTE 0`) is fast.

`DECLARE` is used to define a cursor. Use `MOVE` to change cursor position without retrieving data.

Examples

The following example traverses a table using a cursor:

```
BEGIN WORK;

-- Set up a cursor:
DECLARE liahona SCROLL CURSOR FOR SELECT * FROM films;

-- Fetch the first 5 rows in the cursor liahona:
FETCH FORWARD 5 FROM liahona;

 code  |          title          | did | date_prod  |   kind   |  len
-------+-------------------------+-----+------------+----------+-------
 BL101 | The Third Man           | 101 | 1949-12-23 | Drama    | 01:44
 BL102 | The African Queen       | 101 | 1951-08-11 | Romantic | 01:43
 JL201 | Une Femme est une Femme | 102 | 1961-03-12 | Romantic | 01:25
 P_301 | Vertigo                 | 103 | 1958-11-14 | Action   | 02:08
 P_302 | Becket                  | 103 | 1964-02-03 | Drama    | 02:28

-- Fetch the previous row:
FETCH PRIOR FROM liahona;

 code  |  title  | did | date_prod  |  kind  |  len
-------+---------+-----+------------+--------+-------
 P_301 | Vertigo | 103 | 1958-11-14 | Action | 02:08

-- Close the cursor and end the transaction:
CLOSE liahona;
COMMIT WORK;
```

Compatibility

The SQL standard defines `FETCH` for use in embedded SQL only. The variant of `FETCH` described here returns the data as as if it were a `SELECT` result rather than placing it in host variables. Other than this point, `FETCH` is fully upward-compatible with the SQL standard.

The `FETCH` forms involving `FORWARD` and `BACKWARD`, as well as the forms `FETCH count` and `FETCH ALL`, in which `FORWARD` is implicit, are PostgreSQL extensions.

The SQL standard allows only `FROM` preceding the cursor name; the option to use `IN`, or to leave them out altogether, is an extension.

See Also

□□□ #2
□ □□□ □□□ 2026-10-31 14:12:20 UTC
□ □□□ □□□ 2026-10-31 14:12:21 UTC