NAME

dblink_get_result - gets an async query result

SYNOPSIS

```
dblink_get_result(text connname [, bool fail_on_error]) returns setof record
```

DESCRIPTION

dblink_get_result collects the results of an asynchronous query previously sent with **dblink_send_query**. If the query is not already completed, **dblink_get_result** will wait until it is.

ARGUMENTS

connname

Name of the connection to use.

fail_on_error

If true (the default when omitted) then an error thrown on the remote side of the connection causes an error to also be thrown locally. If false, the remote error is locally reported as a NOTICE, and the function returns no rows.

RETURN VALUE

For an async query (that is, an SQL statement returning rows), the function returns the row(s) produced by the query. To use this function, you will need to specify the expected set of columns, as previously discussed for **dblink**.

For an async command (that is, an SQL statement not returning rows), the function returns a single row with a single text column containing the command’s status string. It is still necessary to specify that the result will have a single text column in the calling FROM clause.

NOTES

This function *must* be called if **dblink_send_query** returned 1. It must be called once for each query sent, and one additional time to obtain an empty set result, before the connection can be used again.

When using **dblink_send_query** and **dblink_get_result**, dblink fetches the entire remote query result before returning any of it to the local query processor. If the query returns a large number of rows, this can result in transient memory bloat in the local session. It may be better to open such a query as a cursor with **dblink_open** and then fetch a manageable number of rows at a time. Alternatively, use plain **dblink()**, which avoids memory bloat by spooling large result sets to disk.

EXAMPLES

```
contrib_regression=# SELECT dblink_connect(’dtest1’, ’dbname=contrib_regression’);
```

```
 dblink_connect
----------------
 OK
(1 row)

contrib_regression=# SELECT * FROM
contrib_regression-# dblink_send_query('dtest1', 'select * from foo where f1 < 3') AS t1;
 t1
----
  1
(1 row)

contrib_regression=# SELECT * FROM dblink_get_result('dtest1') AS t1(f1 int, f2 text, f3 text[]);
 f1 | f2 |     f3
----+----+------------
  0 | a  | {a0,b0,c0}
  1 | b  | {a1,b1,c1}
  2 | c  | {a2,b2,c2}
(3 rows)

contrib_regression=# SELECT * FROM dblink_get_result('dtest1') AS t1(f1 int, f2 text, f3 text[]);
 f1 | f2 | f3
----+----+----
(0 rows)

contrib_regression=# SELECT * FROM
contrib_regression-# dblink_send_query('dtest1', 'select * from foo where f1 < 3; select * from foo where f1 > 6'
 t1
----
  1
(1 row)

contrib_regression=# SELECT * FROM dblink_get_result('dtest1') AS t1(f1 int, f2 text, f3 text[]);
 f1 | f2 |     f3
----+----+------------
  0 | a  | {a0,b0,c0}
  1 | b  | {a1,b1,c1}
  2 | c  | {a2,b2,c2}
(3 rows)
```

```
contrib_regression=# SELECT * FROM dblink_get_result('dtest1') AS t1(f1 int, f2 text, f3 text[]);
 f1 | f2 |      f3
----+----+---------------
  7 | h  | {a7,b7,c7}
  8 | i  | {a8,b8,c8}
  9 | j  | {a9,b9,c9}
 10 | k  | {a10,b10,c10}
(4 rows)

contrib_regression=# SELECT * FROM dblink_get_result('dtest1') AS t1(f1 int, f2 text, f3 text[]);
 f1 | f2 | f3
----+----+----
(0 rows)
```