

NAME

`dblink` – executes a query in a remote database

SYNOPSIS

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

`dblink(text connstr, text sql [, bool fail_on_error])` returns setof record

`dblink(text sql [, bool fail_on_error])` returns setof record

DESCRIPTION

dblink executes a query (usually a **SELECT**, but it can be any SQL statement that returns rows) in a remote database.

When two text arguments are given, the first one is first looked up as a persistent connection's name; if found, the command is executed on that connection. If not found, the first argument is treated as a connection info string as for **dblink_connect**, and the indicated connection is made just for the duration of this command.

ARGUMENTS

connname

Name of the connection to use; omit this parameter to use the unnamed connection.

connstr

A connection info string, as previously described for **dblink_connect**.

sql

The SQL query that you wish to execute in the remote database, for example `select * from foo`.

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

The function returns the row(s) produced by the query. Since **dblink** can be used with any query, it is declared to return record, rather than specifying any particular set of columns. This means that you must specify the expected set of columns in the calling query — otherwise PostgreSQL would not know what to expect. Here is an example:

```
SELECT *
FROM dblink('dbname=mydb options=-csearch_path=',
            'select prname, prosrc from pg_proc')
AS t1(prname name, prosrc text)
WHERE prname LIKE 'bytea%';
```

The “alias” part of the FROM clause must specify the column names and types that the function will return. (Specifying column names in an alias is actually standard SQL syntax, but specifying column types is a PostgreSQL extension.) This allows the system to understand what `*` should expand to, and what prname in the WHERE clause refers to, in advance of trying to execute the function. At run time, an error will be thrown if the actual query result from the remote database does not have the same number of columns shown in the FROM clause. The column names need not match, however, and **dblink** does not insist on exact type matches either. It will succeed so long as the returned data strings are valid input for the column type declared in the FROM clause.

NOTES

A convenient way to use **dblink** with predetermined queries is to create a view. This allows the column type information to be buried in the view, instead of having to spell it out in every query. For example,

```
CREATE VIEW myremote_pg_proc AS
SELECT *
FROM dblink('dbname=postgres options=-csearch_path=');
```

```

        'select proname, prosrc from pg_proc')
    AS t1(proname name, prosrc text);

```

```
SELECT * FROM myremote_pg_proc WHERE proname LIKE 'bytea%';
```

EXAMPLES

```

SELECT * FROM dblink('dbname=postgres options=-csearch_path=',
    'select proname, prosrc from pg_proc')
    AS t1(proname name, prosrc text) WHERE proname LIKE 'bytea%';
proname | prosrc
-----+-----

```

```

byteacat | byteacat
byteaeq  | byteaeq
bytealt  | bytealt
byteale  | byteale
byteagt  | byteagt
byteage  | byteage
byteane  | byteane
byteacmp | byteacmp
bytealike| bytealike
byteanlike| byteanlike
byteain  | byteain
byteaout | byteaout
(12 rows)

```

```

SELECT dblink_connect('dbname=postgres options=-csearch_path=');
dblink_connect
-----

```

```

OK
(1 row)

```

```

SELECT * FROM dblink('select proname, prosrc from pg_proc')
    AS t1(proname name, prosrc text) WHERE proname LIKE 'bytea%';
proname | prosrc
-----+-----

```

```

byteacat | byteacat
byteaeq  | byteaeq
bytealt  | bytealt
byteale  | byteale
byteagt  | byteagt
byteage  | byteage
byteane  | byteane
byteacmp | byteacmp
bytealike| bytealike
byteanlike| byteanlike
byteain  | byteain
byteaout | byteaout
(12 rows)

```

```

SELECT dblink_connect('myconn', 'dbname=regression options=-csearch_path=');
dblink_connect
-----

```

```

OK
(1 row)

```

```
SELECT * FROM dblink('myconn', 'select proname, prosrc from pg_proc')
  AS t1(proname name, prosrc text) WHERE proname LIKE 'bytea%';
```

proname	prosrc
bytearecv	bytearecv
byteasend	byteasend
byteale	byteale
byteagt	byteagt
byteage	byteage
byteane	byteane
byteacmp	byteacmp
bytealike	bytealike
byteanlike	byteanlike
byteacat	byteacat
byteaeq	byteaeq
bytealt	bytealt
byteain	byteain
byteaout	byteaout

(14 rows)