NAME

ALTER_FUNCTION − change the definition of a function

SYNOPSIS

ALTER FUNCTION *name* [([[*argmode*] [*argname*] *argtype* [, ...]])]
action [...] [RESTRICT]
ALTER FUNCTION *name* [([[*argmode*] [*argname*] *argtype* [, ...]])]
RENAME TO *new_name*
ALTER FUNCTION *name* [([[*argmode*] [*argname*] *argtype* [, ...]])]
OWNER TO { *new_owner* | CURRENT_ROLE | CURRENT_USER | SESSION_USER }
ALTER FUNCTION *name* [([[*argmode*] [*argname*] *argtype* [, ...]])]
SET SCHEMA *new_schema*
ALTER FUNCTION *name* [([[*argmode*] [*argname*] *argtype* [, ...]])]
[NO] DEPENDS ON EXTENSION *extension_name*

where *action* is one of:

CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
IMMUTABLE | STABLE | VOLATILE
[NOT] LEAKPROOF
[EXTERNAL] SECURITY INVOKER | [EXTERNAL] SECURITY DEFINER
PARALLEL { UNSAFE | RESTRICTED | SAFE }
COST *execution_cost*
ROWS *result_rows*
SUPPORT *support_function*
SET *configuration_parameter* { TO | = } { *value* | DEFAULT }
SET *configuration_parameter* FROM CURRENT
RESET *configuration_parameter*
RESET ALL

DESCRIPTION

ALTER FUNCTION changes the definition of a function.

You must own the function to use **ALTER FUNCTION**. To change a function's schema, you must also have CREATE privilege on the new schema. To alter the owner, you must be able to SET ROLE to the new owning role, and that role must have CREATE privilege on the function's schema. (These restrictions enforce that altering the owner doesn't do anything you couldn't do by dropping and recreating the function. However, a superuser can alter ownership of any function anyway.)

PARAMETERS

name
The name (optionally schema−qualified) of an existing function. If no argument list is specified, the name must be unique in its schema.

argmode
The mode of an argument: IN, OUT, INOUT, or VARIADIC. If omitted, the default is IN. Note that **ALTER FUNCTION** does not actually pay any attention to OUT arguments, since only the input arguments are needed to determine the function's identity. So it is sufficient to list the IN, INOUT, and VARIADIC arguments.

argname
The name of an argument. Note that **ALTER FUNCTION** does not actually pay any attention to argument names, since only the argument data types are needed to determine the function's identity.

argtype
The data type(s) of the function's arguments (optionally schema−qualified), if any.

new_name
The new name of the function.

new_owner
The new owner of the function. Note that if the function is marked SECURITY DEFINER, it will subsequently execute as the new owner.

new_schema
The new schema for the function.

DEPENDS ON EXTENSION *extension_name*
NO DEPENDS ON EXTENSION *extension_name*
This form marks the function as dependent on the extension, or no longer dependent on that extension if NO is specified. A function that's marked as dependent on an extension is dropped when the extension is dropped, even if CASCADE is not specified. A function can depend upon multiple extensions, and will be dropped when any one of those extensions is dropped.

CALLED ON NULL INPUT
RETURNS NULL ON NULL INPUT
STRICT
CALLED ON NULL INPUT changes the function so that it will be invoked when some or all of its arguments are null. RETURNS NULL ON NULL INPUT or STRICT changes the function so that it is not invoked if any of its arguments are null; instead, a null result is assumed automatically. See CREATE FUNCTION (**CREATE_FUNCTION**(7)) for more information.

IMMUTABLE
STABLE
VOLATILE
Change the volatility of the function to the specified setting. See CREATE FUNCTION (**CREATE_FUNCTION**(7)) for details.

[EXTERNAL] SECURITY INVOKER
[EXTERNAL] SECURITY DEFINER
Change whether the function is a security definer or not. The key word EXTERNAL is ignored for SQL conformance. See CREATE FUNCTION (**CREATE_FUNCTION**(7)) for more information about this capability.

PARALLEL
Change whether the function is deemed safe for parallelism. See CREATE FUNCTION (**CREATE_FUNCTION**(7)) for details.

LEAKPROOF
Change whether the function is considered leakproof or not. See CREATE FUNCTION (**CREATE_FUNCTION**(7)) for more information about this capability.

COST *execution_cost*
Change the estimated execution cost of the function. See CREATE FUNCTION (**CREATE_FUNCTION**(7)) for more information.

ROWS *result_rows*
Change the estimated number of rows returned by a set−returning function. See CREATE FUNCTION (**CREATE_FUNCTION**(7)) for more information.

SUPPORT *support_function*
Set or change the planner support function to use for this function. See Section 36.11 for details. You must be superuser to use this option.

This option cannot be used to remove the support function altogether, since it must name a new support function. Use **CREATE OR REPLACE FUNCTION** if you need to do that.

configuration_parameter
value
Add or change the assignment to be made to a configuration parameter when the function is called. If *value* is DEFAULT or, equivalently, RESET is used, the function−local setting is removed, so that the

function executes with the value present in its environment. Use RESET ALL to clear all function−local settings. SET FROM CURRENT saves the value of the parameter that is current when **ALTER FUNCTION** is executed as the value to be applied when the function is entered.

See **SET**(7) and Chapter 19 for more information about allowed parameter names and values.

RESTRICT
Ignored for conformance with the SQL standard.

EXAMPLES

To rename the function sqrt for type integer to square_root:

```
ALTER FUNCTION sqrt(integer) RENAME TO square_root;
```

To change the owner of the function sqrt for type integer to joe:

```
ALTER FUNCTION sqrt(integer) OWNER TO joe;
```

To change the schema of the function sqrt for type integer to maths:

```
ALTER FUNCTION sqrt(integer) SET SCHEMA maths;
```

To mark the function sqrt for type integer as being dependent on the extension mathlib:

```
ALTER FUNCTION sqrt(integer) DEPENDS ON EXTENSION mathlib;
```

To adjust the search path that is automatically set for a function:

```
ALTER FUNCTION check_password(text) SET search_path = admin, pg_temp;
```

To disable automatic setting of *search_path* for a function:

```
ALTER FUNCTION check_password(text) RESET search_path;
```

The function will now execute with whatever search path is used by its caller.

COMPATIBILITY

This statement is partially compatible with the **ALTER FUNCTION** statement in the SQL standard. The standard allows more properties of a function to be modified, but does not provide the ability to rename a function, make a function a security definer, attach configuration parameter values to a function, or change the owner, schema, or volatility of a function. The standard also requires the RESTRICT key word, which is optional in PostgreSQL.

SEE ALSO

CREATE FUNCTION (**CREATE_FUNCTION**(7)), DROP FUNCTION (**DROP_FUNCTION**(7)), ALTER PROCEDURE (**ALTER_PROCEDURE**(7)), ALTER ROUTINE (**ALTER_ROUTINE**(7))