NAME

ALTER_AGGREGATE − change the definition of an aggregate function

SYNOPSIS

ALTER AGGREGATE *name* (*aggregate_signature*) RENAME TO *new_name*
ALTER AGGREGATE *name* (*aggregate_signature*)
OWNER TO { *new_owner* | CURRENT_ROLE | CURRENT_USER | SESSION_USER }
ALTER AGGREGATE *name* (*aggregate_signature*) SET SCHEMA *new_schema*

where *aggregate_signature* is:

* |
[*argmode*] [*argname*] *argtype* [, ...] |
[[*argmode*] [*argname*] *argtype* [, ...]] ORDER BY [*argmode*] [*argname*] *argtype* [, ...]

DESCRIPTION

ALTER AGGREGATE changes the definition of an aggregate function.

You must own the aggregate function to use **ALTER AGGREGATE**. To change the schema of an aggregate function, 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 aggregate function's schema. (These restrictions enforce that altering the owner doesn't do anything you couldn't do by dropping and recreating the aggregate function. However, a superuser can alter ownership of any aggregate function anyway.)

PARAMETERS

name
The name (optionally schema−qualified) of an existing aggregate function.

argmode
The mode of an argument: IN or VARIADIC. If omitted, the default is IN.

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

argtype
An input data type on which the aggregate function operates. To reference a zero−argument aggregate function, write * in place of the list of argument specifications. To reference an ordered−set aggregate function, write ORDER BY between the direct and aggregated argument specifications.

new_name
The new name of the aggregate function.

new_owner
The new owner of the aggregate function.

new_schema
The new schema for the aggregate function.

NOTES

The recommended syntax for referencing an ordered−set aggregate is to write ORDER BY between the direct and aggregated argument specifications, in the same style as in **CREATE AGGREGATE**. However, it will also work to omit ORDER BY and just run the direct and aggregated argument specifications into a single list. In this abbreviated form, if VARIADIC "any" was used in both the direct and aggregated argument lists, write VARIADIC "any" only once.

EXAMPLES

To rename the aggregate function myavg for type integer to my_average:

```
ALTER AGGREGATE myavg(integer) RENAME TO my_average;
```

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

```
ALTER AGGREGATE myavg(integer) OWNER TO joe;
```

To move the ordered−set aggregate mypercentile with direct argument of type float8 and aggregated argument of type integer into schema myschema:

```
ALTER AGGREGATE mypercentile(float8 ORDER BY integer) SET SCHEMA myschema;
```

This will work too:

```
ALTER AGGREGATE mypercentile(float8, integer) SET SCHEMA myschema;
```

COMPATIBILITY

There is no **ALTER AGGREGATE** statement in the SQL standard.

SEE ALSO

CREATE AGGREGATE (**CREATE_AGGREGATE**(7)), DROP AGGREGATE (**DROP_AGGREGATE**(7))