NAME

ALTER_OPERATOR − change the definition of an operator

SYNOPSIS

```
ALTER OPERATOR name ( { left_type | NONE } , right_type )
  OWNER TO { new_owner | CURRENT_ROLE | CURRENT_USER | SESSION_USER }

ALTER OPERATOR name ( { left_type | NONE } , right_type )
  SET SCHEMA new_schema

ALTER OPERATOR name ( { left_type | NONE } , right_type )
  SET ( {  RESTRICT = { res_proc | NONE }
      | JOIN = { join_proc | NONE }
      | COMMUTATOR = com_op
      | NEGATOR = neg_op
      | HASHES
      | MERGES
     } [, ... ] )
```

DESCRIPTION

ALTER OPERATOR changes the definition of an operator.

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

PARAMETERS

name
: The name (optionally schema−qualified) of an existing operator.

left_type
: The data type of the operator's left operand; write NONE if the operator has no left operand.

right_type
: The data type of the operator's right operand.

new_owner
: The new owner of the operator.

new_schema
: The new schema for the operator.

res_proc
: The restriction selectivity estimator function for this operator; write NONE to remove existing selectivity estimator.

join_proc
: The join selectivity estimator function for this operator; write NONE to remove existing selectivity estimator.

com_op
: The commutator of this operator. Can only be changed if the operator does not have an existing commutator.

neg_op
: The negator of this operator. Can only be changed if the operator does not have an existing negator.

HASHES
: Indicates this operator can support a hash join. Can only be enabled and not disabled.

MERGES
: Indicates this operator can support a merge join. Can only be enabled and not disabled.

NOTES

Refer to Section 36.14 and Section 36.15 for further information.

Since commutators come in pairs that are commutators of each other, ALTER OPERATOR SET COMMUTATOR will also set the commutator of the *com_op* to be the target operator. Likewise, ALTER OPERATOR SET NEGATOR will also set the negator of the *neg_op* to be the target operator. Therefore, you must own the commutator or negator operator as well as the target operator.

EXAMPLES

Change the owner of a custom operator a @@ b for type text:

```
ALTER OPERATOR @@ (text, text) OWNER TO joe;
```

Change the restriction and join selectivity estimator functions of a custom operator a && b for type int[]:

```
ALTER OPERATOR && (int[], int[]) SET (RESTRICT = _int_contsel, JOIN = _int_contjoinsel);
```

Mark the && operator as being its own commutator:

```
ALTER OPERATOR && (int[], int[]) SET (COMMUTATOR = &&);
```

COMPATIBILITY

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

SEE ALSO

CREATE OPERATOR (**CREATE_OPERATOR**(7)), DROP OPERATOR (**DROP_OPERATOR**(7))