NAME

CREATE_OPERATOR − define a new operator

SYNOPSIS

```
CREATE OPERATOR name (
    {FUNCTION|PROCEDURE} = function_name
    [, LEFTARG = left_type ] [, RIGHTARG = right_type ]
    [, COMMUTATOR = com_op ] [, NEGATOR = neg_op ]
    [, RESTRICT = res_proc ] [, JOIN = join_proc ]
    [, HASHES ] [, MERGES ]
)
```

DESCRIPTION

CREATE OPERATOR defines a new operator, *name*. The user who defines an operator becomes its owner. If a schema name is given then the operator is created in the specified schema. Otherwise it is created in the current schema.

The operator name is a sequence of up to NAMEDATALEN−1 (63 by default) characters from the following list:

```
+ − * / < > = ~ ! @ # % ^ & | ` ?
```

There are a few restrictions on your choice of name:

- −− and /* cannot appear anywhere in an operator name, since they will be taken as the start of a comment.
- A multicharacter operator name cannot end in + or −, unless the name also contains at least one of these characters:

  ```
  ~ ! @ # % ^ & | ` ?
  ```

  For example, @− is an allowed operator name, but *− is not. This restriction allows PostgreSQL to parse SQL−compliant commands without requiring spaces between tokens.
- The symbol => is reserved by the SQL grammar, so it cannot be used as an operator name.

The operator != is mapped to <> on input, so these two names are always equivalent.

For binary operators, both LEFTARG and RIGHTARG must be defined. For prefix operators only RIGHTARG should be defined. The *function_name* function must have been previously defined using **CREATE FUNCTION** and must be defined to accept the correct number of arguments (either one or two) of the indicated types.

In the syntax of CREATE OPERATOR, the keywords FUNCTION and PROCEDURE are equivalent, but the referenced function must in any case be a function, not a procedure. The use of the keyword PROCEDURE here is historical and deprecated.

The other clauses specify optional operator optimization attributes. Their meaning is detailed in Section 36.15.

To be able to create an operator, you must have USAGE privilege on the argument types and the return type, as well as EXECUTE privilege on the underlying function. If a commutator or negator operator is specified, you must own those operators.

PARAMETERS

name

The name of the operator to be defined. See above for allowable characters. The name can be schema−qualified, for example CREATE OPERATOR myschema.+ (...). If not, then the operator is created in the current schema. Two operators in the same schema can have the same name if they operate on different data types. This is called overloading.

function_name
The function used to implement this operator.

left_type
The data type of the operator's left operand, if any. This option would be omitted for a prefix operator.

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

com_op
The commutator of this operator.

neg_op
The negator of this operator.

res_proc
The restriction selectivity estimator function for this operator.

join_proc
The join selectivity estimator function for this operator.

HASHES
Indicates this operator can support a hash join.

MERGES
Indicates this operator can support a merge join.

To give a schema−qualified operator name in *com_op* or the other optional arguments, use the OPERATOR() syntax, for example:

```
COMMUTATOR = OPERATOR(myschema.===) ,
```

NOTES

Refer to Section 36.14 and Section 36.15 for further information.

When you are defining a self−commutative operator, you just do it. When you are defining a pair of commutative operators, things are a little trickier: how can the first one to be defined refer to the other one, which you haven't defined yet? There are three solutions to this problem:

- One way is to omit the COMMUTATOR clause in the first operator that you define, and then provide one in the second operator's definition. Since PostgreSQL knows that commutative operators come in pairs, when it sees the second definition it will automatically go back and fill in the missing COMMUTATOR clause in the first definition.
- Another, more straightforward way is just to include COMMUTATOR clauses in both definitions. When PostgreSQL processes the first definition and realizes that COMMUTATOR refers to a nonexistent operator, the system will make a dummy entry for that operator in the system catalog. This dummy entry will have valid data only for the operator name, left and right operand types, and owner, since that's all that PostgreSQL can deduce at this point. The first operator's catalog entry will link to this dummy entry. Later, when you define the second operator, the system updates the dummy entry with the additional information from the second definition. If you try to use the dummy operator before it's been filled in, you'll just get an error message.
- Alternatively, both operators can be defined without COMMUTATOR clauses and then **ALTER OPERATOR** can be used to set their commutator links. It's sufficient to **ALTER** either one of the pair.

In all three cases, you must own both operators in order to mark them as commutators.

Pairs of negator operators can be defined using the same methods as for commutator pairs.

It is not possible to specify an operator's lexical precedence in **CREATE OPERATOR**, because the parser's precedence behavior is hard−wired. See Section 4.1.6 for precedence details.

The obsolete options SORT1, SORT2, LTCMP, and GTCMP were formerly used to specify the names of

sort operators associated with a merge−joinable operator. This is no longer necessary, since information about associated operators is found by looking at B−tree operator families instead. If one of these options is given, it is ignored except for implicitly setting MERGES true.

Use **DROP OPERATOR** to delete user−defined operators from a database. Use **ALTER OPERATOR** to modify operators in a database.

EXAMPLES

The following command defines a new operator, area−equality, for the data type box:

```
CREATE OPERATOR === (
   LEFTARG = box,
   RIGHTARG = box,
   FUNCTION = area_equal_function,
   COMMUTATOR = ===,
   NEGATOR = !==,
   RESTRICT = area_restriction_function,
   JOIN = area_join_function,
   HASHES, MERGES
);
```

COMPATIBILITY

CREATE OPERATOR is a PostgreSQL extension. There are no provisions for user−defined operators in the SQL standard.

SEE ALSO

ALTER OPERATOR (**ALTER_OPERATOR**(7)), CREATE OPERATOR CLASS (**CREATE_OPERATOR_CLASS**(7)), DROP OPERATOR (**DROP_OPERATOR**(7))