alter_opfamily

- Description
- Parameters
- Notes
- Examples
- Compatibility
- See Also

ALTER OPERATOR FAMILY

ALTER OPERATOR FAMILY

7

SQL - Language Statements

ALTER OPERATOR FAMILY

change the definition of an operator family

ALTER OPERATOR FAMILY

name

USING

index_method

ADD { OPERATOR

strategy_number

operator_name

(

op_type

,

op_type

) [FOR SEARCH | FOR ORDER BY

sort_family_name

] | FUNCTION

support_number

[(

op_type

[,

op_type

])]

function_name

[(

argument_type

[, ...])] } [, ...] ALTER OPERATOR FAMILY

name

USING

index_method

DROP { OPERATOR

strategy_number

(

op_type

[,

op_type

]) | FUNCTION

support_number

(

op_type

[,

op_type

]) } [, ...] ALTER OPERATOR FAMILY

name

USING

index_method

RENAME TO

new_name

ALTER OPERATOR FAMILY

name

USING

index_method

OWNER TO {

new_owner

| CURRENT_ROLE | CURRENT_USER | SESSION_USER } ALTER OPERATOR FAMILY

name

USING

index_method

SET SCHEMA

new_schema

Description

`ALTER OPERATOR FAMILY` changes the definition of an operator family. You can add operators and support functions to the family, remove them from the family, or change the family's name or owner.

When operators and support functions are added to a family with `ALTER OPERATOR FAMILY`, they are not part of any specific operator class within the family, but are just “loose” within the family. This indicates that these operators and functions are compatible with the family's semantics, but are not required for correct functioning of any specific index. (Operators and functions that are so required should be declared as part of an operator class, instead; see ???.) PostgreSQL will allow loose members of a family to be dropped from the family at any time, but members of an operator class cannot be dropped without dropping the whole class and any indexes that depend on it. Typically, single-data-type operators and functions are part of operator classes because they are needed to support an index on that specific data type, while cross-data-type operators and functions are made loose members of the family.

You must be a superuser to use `ALTER OPERATOR FAMILY`. (This restriction is made because an erroneous operator family definition could confuse or even crash the server.)

`ALTER OPERATOR FAMILY` does not presently check whether the operator family definition includes all the operators and functions required by the index method, nor whether the operators and functions form a self-consistent set. It is the user's responsibility to define a valid operator family.

Refer to ??? for further information.

Parameters

<name>
The name (optionally schema-qualified) of an existing operator family.

<index_method>
The name of the index method this operator family is for.

<strategy_number>
The index method's strategy number for an operator associated with the operator family.

<operator_name>
The name (optionally schema-qualified) of an operator associated with the operator family.

<op_type>
In an `OPERATOR` clause, the operand data type(s) of the operator, or `NONE` to signify a prefix operator. Unlike the comparable syntax in `CREATE OPERATOR CLASS`, the operand data types must always be specified.

In an `ADD FUNCTION` clause, the operand data type(s) the function is intended to support, if different from the input data type(s) of the function. For B-tree comparison functions and hash functions it is not necessary to specify <op_type> since the function's input data type(s) are always the correct ones to use. For B-tree sort support functions, B-Tree equal image functions, and all functions in GiST, SP-GiST and GIN operator classes, it is necessary to specify the operand data type(s) the function is to be used with.

In a `DROP FUNCTION` clause, the operand data type(s) the function is intended to support must be specified.

<sort_family_name>
The name (optionally schema-qualified) of an existing `btree` operator family that describes the sort ordering associated with an ordering operator.

If neither `FOR SEARCH` nor `FOR ORDER BY` is specified, `FOR SEARCH` is the default.

<support_number>
The index method's support function number for a function associated with the operator family.

<function_name>
The name (optionally schema-qualified) of a function that is an index method support function for the operator family. If no argument list is specified, the name must be unique in its schema.

<argument_type>
The parameter data type(s) of the function.

<new_name>
The new name of the operator family.

<new_owner>
The new owner of the operator family.

<new_schema>
The new schema for the operator family.

The `OPERATOR` and `FUNCTION` clauses can appear in any order.

Notes

Notice that the `DROP` syntax only specifies the “slot” in the operator family, by strategy or support number and input data type(s). The name of the operator or function occupying the slot is not mentioned. Also, for `DROP FUNCTION` the type(s) to specify are the input data type(s) the function is intended to support; for GiST, SP-GiST and GIN indexes this might have nothing to do with the actual input argument types of the function.

Because the index machinery does not check access permissions on functions before using them, including a function or operator in an operator family is tantamount to granting public execute permission on it. This is usually not an issue for the sorts of functions that are useful in an operator family.

The operators should not be defined by SQL functions. An SQL function is likely to be inlined into the calling query, which will prevent the optimizer from recognizing that the query matches an index.

Examples

The following example command adds cross-data-type operators and support functions to an operator family that already contains B-tree operator classes for data types `int4` and `int2`.

```
ALTER OPERATOR FAMILY integer_ops USING btree ADD

  -- int4 vs int2
  OPERATOR 1 < (int4, int2) ,
  OPERATOR 2 <= (int4, int2) ,
  OPERATOR 3 = (int4, int2) ,
  OPERATOR 4 >= (int4, int2) ,
  OPERATOR 5 > (int4, int2) ,
  FUNCTION 1 btint42cmp(int4, int2) ,

  -- int2 vs int4
  OPERATOR 1 < (int2, int4) ,
  OPERATOR 2 <= (int2, int4) ,
  OPERATOR 3 = (int2, int4) ,
  OPERATOR 4 >= (int2, int4) ,
  OPERATOR 5 > (int2, int4) ,
  FUNCTION 1 btint24cmp(int2, int4) ;
```

To remove these entries again:

```
ALTER OPERATOR FAMILY integer_ops USING btree DROP

  -- int4 vs int2
  OPERATOR 1 (int4, int2) ,
  OPERATOR 2 (int4, int2) ,
  OPERATOR 3 (int4, int2) ,
  OPERATOR 4 (int4, int2) ,
  OPERATOR 5 (int4, int2) ,
  FUNCTION 1 (int4, int2) ,

  -- int2 vs int4
  OPERATOR 1 (int2, int4) ,
  OPERATOR 2 (int2, int4) ,
  OPERATOR 3 (int2, int4) ,
  OPERATOR 4 (int2, int4) ,
  OPERATOR 5 (int2, int4) ,
  FUNCTION 1 (int2, int4) ;
```

Compatibility

There is no `ALTER OPERATOR FAMILY` statement in the SQL standard.

See Also

□□□ #2
□ □□□ □□□ 2026-10-31 14:07:27 UTC
□ □□□ □□□ 2026-10-31 14:07:28 UTC