NAME

ALTER_TYPE − change the definition of a type

SYNOPSIS

ALTER TYPE *name* OWNER TO { *new_owner* | CURRENT_ROLE | CURRENT_USER | SESSION_USER }
ALTER TYPE *name* RENAME TO *new_name*
ALTER TYPE *name* SET SCHEMA *new_schema*
ALTER TYPE *name* RENAME ATTRIBUTE *attribute_name* TO *new_attribute_name* [CASCADE | RESTRICT]
ALTER TYPE *name action* [, ...]
ALTER TYPE *name* ADD VALUE [IF NOT EXISTS] *new_enum_value* [{ BEFORE | AFTER } *neighbor_enum_value*]
ALTER TYPE *name* RENAME VALUE *existing_enum_value* TO *new_enum_value*
ALTER TYPE *name* SET (*property* = *value* [, ...])

where *action* is one of:

ADD ATTRIBUTE *attribute_name data_type* [COLLATE *collation*] [CASCADE | RESTRICT]
DROP ATTRIBUTE [IF EXISTS] *attribute_name* [CASCADE | RESTRICT]
ALTER ATTRIBUTE *attribute_name* [SET DATA] TYPE *data_type* [COLLATE *collation*] [CASCADE | RESTRICT]

DESCRIPTION

ALTER TYPE changes the definition of an existing type. There are several subforms:

OWNER
: This form changes the owner of the type.

RENAME
: This form changes the name of the type.

SET SCHEMA
: This form moves the type into another schema.

RENAME ATTRIBUTE
: This form is only usable with composite types. It changes the name of an individual attribute of the type.

ADD ATTRIBUTE
: This form adds a new attribute to a composite type, using the same syntax as **CREATE TYPE**.

DROP ATTRIBUTE [IF EXISTS]
: This form drops an attribute from a composite type. If IF EXISTS is specified and the attribute does not exist, no error is thrown. In this case a notice is issued instead.

ALTER ATTRIBUTE ... SET DATA TYPE
: This form changes the type of an attribute of a composite type.

ADD VALUE [IF NOT EXISTS] [BEFORE | AFTER]
: This form adds a new value to an enum type. The new value's place in the enum's ordering can be specified as being BEFORE or AFTER one of the existing values. Otherwise, the new item is added at the end of the list of values.

  If IF NOT EXISTS is specified, it is not an error if the type already contains the new value: a notice is issued but no other action is taken. Otherwise, an error will occur if the new value is already present.

RENAME VALUE
: This form renames a value of an enum type. The value's place in the enum's ordering is not affected. An error will occur if the specified value is not present or the new name is already present.

SET (*property* = *value* [, ...])
: This form is only applicable to base types. It allows adjustment of a subset of the base−type properties that can be set in **CREATE TYPE**. Specifically, these properties can be changed:

- RECEIVE can be set to the name of a binary input function, or NONE to remove the type's binary input function. Using this option requires superuser privilege.
- SEND can be set to the name of a binary output function, or NONE to remove the type's binary output function. Using this option requires superuser privilege.
- TYPMOD_IN can be set to the name of a type modifier input function, or NONE to remove the type's type modifier input function. Using this option requires superuser privilege.
- TYPMOD_OUT can be set to the name of a type modifier output function, or NONE to remove the type's type modifier output function. Using this option requires superuser privilege.
- ANALYZE can be set to the name of a type−specific statistics collection function, or NONE to remove the type's statistics collection function. Using this option requires superuser privilege.
- SUBSCRIPT can be set to the name of a type−specific subscripting handler function, or NONE to remove the type's subscripting handler function. Using this option requires superuser privilege.
- STORAGE can be set to plain, extended, external, or main (see Section 66.2 for more information about what these mean). However, changing from plain to another setting requires superuser privilege (because it requires that the type's C functions all be TOAST−ready), and changing to plain from another setting is not allowed at all (since the type may already have TOASTed values present in the database). Note that changing this option doesn't by itself change any stored data, it just sets the default TOAST strategy to be used for table columns created in the future. See ALTER TABLE (**ALTER_TABLE**(7)) to change the TOAST strategy for existing table columns.

See CREATE TYPE (**CREATE_TYPE**(7)) for more details about these type properties. Note that where appropriate, a change in these properties for a base type will be propagated automatically to domains based on that type.

The ADD ATTRIBUTE, DROP ATTRIBUTE, and ALTER ATTRIBUTE actions can be combined into a list of multiple alterations to apply in parallel. For example, it is possible to add several attributes and/or alter the type of several attributes in a single command.

You must own the type to use **ALTER TYPE**. To change the schema of a type, 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 type's schema. (These restrictions enforce that altering the owner doesn't do anything you couldn't do by dropping and recreating the type. However, a superuser can alter ownership of any type anyway.) To add an attribute or alter an attribute type, you must also have USAGE privilege on the attribute's data type.

PARAMETERS

name
The name (possibly schema−qualified) of an existing type to alter.

new_name
The new name for the type.

new_owner
The user name of the new owner of the type.

new_schema
The new schema for the type.

attribute_name
The name of the attribute to add, alter, or drop.

new_attribute_name
The new name of the attribute to be renamed.

data_type

The data type of the attribute to add, or the new type of the attribute to alter.

new_enum_value
The new value to be added to an enum type's list of values, or the new name to be given to an existing value. Like all enum literals, it needs to be quoted.

neighbor_enum_value
The existing enum value that the new value should be added immediately before or after in the enum type's sort ordering. Like all enum literals, it needs to be quoted.

existing_enum_value
The existing enum value that should be renamed. Like all enum literals, it needs to be quoted.

property
The name of a base−type property to be modified; see above for possible values.

CASCADE
Automatically propagate the operation to typed tables of the type being altered, and their descendants.

RESTRICT
Refuse the operation if the type being altered is the type of a typed table. This is the default.

NOTES

If **ALTER TYPE ... ADD VALUE** (the form that adds a new value to an enum type) is executed inside a transaction block, the new value cannot be used until after the transaction has been committed.

Comparisons involving an added enum value will sometimes be slower than comparisons involving only original members of the enum type. This will usually only occur if BEFORE or AFTER is used to set the new value's sort position somewhere other than at the end of the list. However, sometimes it will happen even though the new value is added at the end (this occurs if the OID counter “wrapped around” since the original creation of the enum type). The slowdown is usually insignificant; but if it matters, optimal performance can be regained by dropping and recreating the enum type, or by dumping and restoring the database.

EXAMPLES

To rename a data type:

```
ALTER TYPE electronic_mail RENAME TO email;
```

To change the owner of the type email to joe:

```
ALTER TYPE email OWNER TO joe;
```

To change the schema of the type email to customers:

```
ALTER TYPE email SET SCHEMA customers;
```

To add a new attribute to a composite type:

```
ALTER TYPE compfoo ADD ATTRIBUTE f3 int;
```

To add a new value to an enum type in a particular sort position:

```
ALTER TYPE colors ADD VALUE 'orange' AFTER 'red';
```

To rename an enum value:

```
ALTER TYPE colors RENAME VALUE 'purple' TO 'mauve';
```

To create binary I/O functions for an existing base type:

```
CREATE FUNCTION mytypesend(mytype) RETURNS bytea ...;
CREATE FUNCTION mytyperecv(internal, oid, integer) RETURNS mytype ...;
```

```
ALTER TYPE mytype SET (
   SEND = mytypesend,
   RECEIVE = mytyperecv
);
```

COMPATIBILITY

The variants to add and drop attributes are part of the SQL standard; the other variants are PostgreSQL extensions.

SEE ALSO

CREATE TYPE (**CREATE_TYPE**(7)), DROP TYPE (**DROP_TYPE**(7))