alter_extension

- Description
- Parameters
- Examples
- Compatibility
- See Also

ALTER EXTENSION

ALTER EXTENSION

7

SQL - Language Statements

ALTER EXTENSION

change the definition of an extension

ALTER EXTENSION

name

UPDATE [TO

new_version

] ALTER EXTENSION

name

SET SCHEMA

new_schema

ALTER EXTENSION

name

ADD

member_object

ALTER EXTENSION

name

DROP

member_object

where

member_object

is:

ACCESS METHOD

object_name

| AGGREGATE

aggregate_name

(

aggregate_signature

) | CAST (

source_type

AS

target_type

) | COLLATION

object_name

| CONVERSION

object_name

| DOMAIN

object_name

| EVENT TRIGGER

object_name

| FOREIGN DATA WRAPPER

object_name

| FOREIGN TABLE

object_name

| FUNCTION

function_name

[([[

argmode

] [

argname

]

argtype

[, ...]])] | MATERIALIZED VIEW

object_name

| OPERATOR

operator_name

(

left_type

,

right_type

) | OPERATOR CLASS

object_name

USING

index_method

| OPERATOR FAMILY

object_name

USING

index_method

| [PROCEDURAL] LANGUAGE

object_name

| PROCEDURE

procedure_name

[([[

argmode

] [

argname

]

argtype

[, ...]])] | ROUTINE

routine_name

[([[

argmode

] [

argname

]

argtype

[, ...]])] | SCHEMA

object_name

| SEQUENCE

object_name

| SERVER

object_name

| TABLE

object_name

| TEXT SEARCH CONFIGURATION

object_name

| TEXT SEARCH DICTIONARY

object_name

| TEXT SEARCH PARSER

object_name

| TEXT SEARCH TEMPLATE

object_name

| TRANSFORM FOR

type_name

LANGUAGE

lang_name

| TYPE

object_name

| VIEW

object_name

and

aggregate_signature

is:

* | [

argmode

] [

argname

]

argtype

[, ...] | [[

argmode

] [

argname

]

argtype

[, ...]] ORDER BY [

argmode

] [

argname

]

argtype

[, ...]

Description

`ALTER EXTENSION` changes the definition of an installed extension. There are several subforms:

`UPDATE`
This form updates the extension to a newer version. The extension must supply a suitable update script (or series of scripts) that can modify the currently-installed version into the requested version.

`SET SCHEMA`
This form moves the extension's objects into another schema. The extension has to be relocatable for this command to succeed.

`ADD member_object`
This form adds an existing object to the extension. This is mainly useful in extension update scripts. The object will subsequently be treated as a member of the extension; notably, it can only be dropped by dropping the extension.

`DROP member_object`
This form removes a member object from the extension. This is mainly useful in extension update scripts. The object is not dropped, only disassociated from the extension.

See ??? for more information about these operations.

You must own the extension to use `ALTER EXTENSION`. The `ADD`/`DROP` forms require ownership of the added/dropped object as well.

Parameters

<name>
The name of an installed extension.

<new_version>
The desired new version of the extension. This can be written as either an identifier or a string literal. If not specified, `ALTER EXTENSION UPDATE` attempts to update to whatever is shown as the default version in the extension's control file.

<new_schema>
The new schema for the extension.

<object_name>; <aggregate_name>; <function_name>; <operator_name>; <procedure_name>; <routine_name>
The name of an object to be added to or removed from the extension. Names of tables, aggregates, domains, foreign tables, functions, operators, operator classes, operator families, procedures, routines, sequences, text search objects, types, and views can be schema-qualified.

<source_type>
The name of the source data type of the cast.

<target_type>
The name of the target data type of the cast.

<argmode>
The mode of a function, procedure, or aggregate argument: `IN`, `OUT`, `INOUT`, or `VARIADIC`. If omitted, the default is `IN`. Note that `ALTER EXTENSION` does not actually pay any attention to `OUT` arguments, since only the input arguments are needed to determine the function's identity. So it is sufficient to list the `IN`, `INOUT`, and `VARIADIC` arguments.

<argname>
The name of a function, procedure, or aggregate argument. Note that `ALTER EXTENSION` does not actually pay any attention to argument names, since only the argument data types are needed to determine the function's identity.

<argtype>
The data type of a function, procedure, or aggregate argument.

<left_type>; <right_type>
The data type(s) of the operator's arguments (optionally schema-qualified). Write `NONE` for the missing argument of a prefix operator.

`PROCEDURAL`
This is a noise word.

<type_name>
The name of the data type of the transform.

<lang_name>
The name of the language of the transform.

Examples

To update the `hstore` extension to version 2.0:

```
ALTER EXTENSION hstore UPDATE TO '2.0';
```

To change the schema of the `hstore` extension to `utils`:

```
ALTER EXTENSION hstore SET SCHEMA utils;
```

To add an existing function to the `hstore` extension:

```
ALTER EXTENSION hstore ADD FUNCTION populate_record(anyelement, hstore);
```

Compatibility

`ALTER EXTENSION` is a PostgreSQL extension.

See Also

□□□ #2
□ □□□ □□□ 2026-10-31 14:07:11 UTC
□ □□□ □□□ 2026-10-31 14:07:12 UTC