## NAME

CREATE_EXTENSION − install an extension

## SYNOPSIS

CREATE EXTENSION [ IF NOT EXISTS ] *extension_name*
[ WITH ] [ SCHEMA *schema_name* ]
[ VERSION *version* ]
[ CASCADE ]

## DESCRIPTION

**CREATE EXTENSION** loads a new extension into the current database. There must not be an extension of the same name already loaded.

Loading an extension essentially amounts to running the extension's script file. The script will typically create new SQL objects such as functions, data types, operators and index support methods. **CREATE EXTENSION** additionally records the identities of all the created objects, so that they can be dropped again if **DROP EXTENSION** is issued.

The user who runs **CREATE EXTENSION** becomes the owner of the extension for purposes of later privilege checks, and normally also becomes the owner of any objects created by the extension's script.

Loading an extension ordinarily requires the same privileges that would be required to create its component objects. For many extensions this means superuser privileges are needed. However, if the extension is marked trusted in its control file, then it can be installed by any user who has CREATE privilege on the current database. In this case the extension object itself will be owned by the calling user, but the contained objects will be owned by the bootstrap superuser (unless the extension's script explicitly assigns them to the calling user). This configuration gives the calling user the right to drop the extension, but not to modify individual objects within it.

## PARAMETERS

IF NOT EXISTS

Do not throw an error if an extension with the same name already exists. A notice is issued in this case. Note that there is no guarantee that the existing extension is anything like the one that would have been created from the currently−available script file.

*extension_name*

The name of the extension to be installed. PostgreSQL will create the extension using details from the file *extension_name*.control, found via the server's extension control path (set by extension_control_path.)

*schema_name*

The name of the schema in which to install the extension's objects, given that the extension allows its contents to be relocated. The named schema must already exist. If not specified, and the extension's control file does not specify a schema either, the current default object creation schema is used.

If the extension specifies a schema parameter in its control file, then that schema cannot be overridden with a SCHEMA clause. Normally, an error will be raised if a SCHEMA clause is given and it conflicts with the extension's schema parameter. However, if the CASCADE clause is also given, then *schema_name* is ignored when it conflicts. The given *schema_name* will be used for installation of any needed extensions that do not specify schema in their control files.

Remember that the extension itself is not considered to be within any schema: extensions have unqualified names that must be unique database−wide. But objects belonging to the extension can be within schemas.

*version*

The version of the extension to install. This can be written as either an identifier or a string literal. The default version is whatever is specified in the extension's control file.

CASCADE

Automatically install any extensions that this extension depends on that are not already installed. Their dependencies are likewise automatically installed, recursively. The SCHEMA clause, if given, applies to all extensions that get installed this way. Other options of the statement are not applied to automatically−installed extensions; in particular, their default versions are always selected.

## NOTES

Before you can use **CREATE EXTENSION** to load an extension into a database, the extension's supporting files must be installed. Information about installing the extensions supplied with PostgreSQL can be found in Additional Supplied Modules.

The extensions currently available for loading can be identified from the pg_available_extensions or pg_available_extension_versions system views.

### Caution

Installing an extension as superuser requires trusting that the extension's author wrote the extension installation script in a secure fashion. It is not terribly difficult for a malicious user to create trojan−horse objects that will compromise later execution of a carelessly−written extension script, allowing that user to acquire superuser privileges. However, trojan−horse objects are only hazardous if they are in the *search_path* during script execution, meaning that they are in the extension's installation target schema or in the schema of some extension it depends on. Therefore, a good rule of thumb when dealing with extensions whose scripts have not been carefully vetted is to install them only into schemas for which CREATE privilege has not been and will not be granted to any untrusted users. Likewise for any extensions they depend on.

The extensions supplied with PostgreSQL are believed to be secure against installation−time attacks of this sort, except for a few that depend on other extensions. As stated in the documentation for those extensions, they should be installed into secure schemas, or installed into the same schemas as the extensions they depend on, or both.

For information about writing new extensions, see Section 36.17.

## EXAMPLES

Install the hstore extension into the current database, placing its objects in schema addons:

```
CREATE EXTENSION hstore SCHEMA addons;
```

Another way to accomplish the same thing:

```
SET search_path = addons;
CREATE EXTENSION hstore;
```

## COMPATIBILITY

**CREATE EXTENSION** is a PostgreSQL extension.

## SEE ALSO

ALTER EXTENSION (**ALTER_EXTENSION**(7)), DROP EXTENSION (**DROP_EXTENSION**(7))