NAME

CREATE_LANGUAGE − define a new procedural language

SYNOPSIS

CREATE [OR REPLACE] [TRUSTED] [PROCEDURAL] LANGUAGE *name*
HANDLER *call_handler* [INLINE *inline_handler*] [VALIDATOR *valfunction*]
CREATE [OR REPLACE] [TRUSTED] [PROCEDURAL] LANGUAGE *name*

DESCRIPTION

CREATE LANGUAGE registers a new procedural language with a PostgreSQL database. Subsequently, functions and procedures can be defined in this new language.

CREATE LANGUAGE effectively associates the language name with handler function(s) that are responsible for executing functions written in the language. Refer to Chapter 57 for more information about language handlers.

CREATE OR REPLACE LANGUAGE will either create a new language, or replace an existing definition. If the language already exists, its parameters are updated according to the command, but the language's ownership and permissions settings do not change, and any existing functions written in the language are assumed to still be valid.

One must have the PostgreSQL superuser privilege to register a new language or change an existing language's parameters. However, once the language is created it is valid to assign ownership of it to a non−superuser, who may then drop it, change its permissions, rename it, or assign it to a new owner. (Do not, however, assign ownership of the underlying C functions to a non−superuser; that would create a privilege escalation path for that user.)

The form of **CREATE LANGUAGE** that does not supply any handler function is obsolete. For backwards compatibility with old dump files, it is interpreted as **CREATE EXTENSION**. That will work if the language has been packaged into an extension of the same name, which is the conventional way to set up procedural languages.

PARAMETERS

TRUSTED
: TRUSTED specifies that the language does not grant access to data that the user would not otherwise have. If this key word is omitted when registering the language, only users with the PostgreSQL superuser privilege can use this language to create new functions.

PROCEDURAL
: This is a noise word.

name
: The name of the new procedural language. The name must be unique among the languages in the database.

HANDLER *call_handler*
: *call_handler* is the name of a previously registered function that will be called to execute the procedural language's functions. The call handler for a procedural language must be written in a compiled language such as C with version 1 call convention and registered with PostgreSQL as a function taking no arguments and returning the language_handler type, a placeholder type that is simply used to identify the function as a call handler.

INLINE *inline_handler*
: *inline_handler* is the name of a previously registered function that will be called to execute an anonymous code block (**DO** command) in this language. If no *inline_handler* function is specified, the language does not support anonymous code blocks. The handler function must take one argument of type internal, which will be the **DO** command's internal representation, and it will typically return void. The return value of the handler is ignored.

VALIDATOR *valfunction*
: *valfunction* is the name of a previously registered function that will be called when a new function in the language is created, to validate the new function. If no validator function is specified, then a new

function will not be checked when it is created. The validator function must take one argument of type oid, which will be the OID of the to−be−created function, and will typically return void.

A validator function would typically inspect the function body for syntactical correctness, but it can also look at other properties of the function, for example if the language cannot handle certain argument types. To signal an error, the validator function should use the **ereport()** function. The return value of the function is ignored.

NOTES

Use **DROP LANGUAGE** to drop procedural languages.

The system catalog pg_language (see Section 52.29) records information about the currently installed languages. Also, the psql command **\dL** lists the installed languages.

To create functions in a procedural language, a user must have the USAGE privilege for the language. By default, USAGE is granted to PUBLIC (i.e., everyone) for trusted languages. This can be revoked if desired.

Procedural languages are local to individual databases. However, a language can be installed into the template1 database, which will cause it to be available automatically in all subsequently−created databases.

EXAMPLES

A minimal sequence for creating a new procedural language is:

```
CREATE FUNCTION plsample_call_handler() RETURNS language_handler
  AS '$libdir/plsample'
  LANGUAGE C;
CREATE LANGUAGE plsample
  HANDLER plsample_call_handler;
```

Typically that would be written in an extension's creation script, and users would do this to install the extension:

```
CREATE EXTENSION plsample;
```

COMPATIBILITY

CREATE LANGUAGE is a PostgreSQL extension.

SEE ALSO

ALTER LANGUAGE (**ALTER_LANGUAGE**(7)), CREATE FUNCTION (**CREATE_FUNCTION**(7)), DROP LANGUAGE (**DROP_LANGUAGE**(7)), **GRANT**(7), **REVOKE**(7)