NAME

CREATE_TRANSFORM − define a new transform

SYNOPSIS

```
CREATE [ OR REPLACE ] TRANSFORM FOR type_name LANGUAGE lang_name (
  FROM SQL WITH FUNCTION from_sql_function_name [ (argument_type [, ...]) ],
  TO SQL WITH FUNCTION to_sql_function_name [ (argument_type [, ...]) ]
);
```

DESCRIPTION

CREATE TRANSFORM defines a new transform. **CREATE OR REPLACE TRANSFORM** will either create a new transform, or replace an existing definition.

A transform specifies how to adapt a data type to a procedural language. For example, when writing a function in PL/Python using the hstore type, PL/Python has no prior knowledge how to present hstore values in the Python environment. Language implementations usually default to using the text representation, but that is inconvenient when, for example, an associative array or a list would be more appropriate.

A transform specifies two functions:

- A “from SQL” function that converts the type from the SQL environment to the language. This function will be invoked on the arguments of a function written in the language.
- A “to SQL” function that converts the type from the language to the SQL environment. This function will be invoked on the return value of a function written in the language.

It is not necessary to provide both of these functions. If one is not specified, the language−specific default behavior will be used if necessary. (To prevent a transformation in a certain direction from happening at all, you could also write a transform function that always errors out.)

To be able to create a transform, you must own and have USAGE privilege on the type, have USAGE privilege on the language, and own and have EXECUTE privilege on the from−SQL and to−SQL functions, if specified.

PARAMETERS

type_name
: The name of the data type of the transform.

lang_name
: The name of the language of the transform.

from_sql_function_name[(*argument_type* [, ...])]
: The name of the function for converting the type from the SQL environment to the language. It must take one argument of type internal and return type internal. The actual argument will be of the type for the transform, and the function should be coded as if it were. (But it is not allowed to declare an SQL−level function returning internal without at least one argument of type internal.) The actual return value will be something specific to the language implementation. If no argument list is specified, the function name must be unique in its schema.

to_sql_function_name[(*argument_type* [, ...])]
: The name of the function for converting the type from the language to the SQL environment. It must take one argument of type internal and return the type that is the type for the transform. The actual argument value will be something specific to the language implementation. If no argument list is specified, the function name must be unique in its schema.

NOTES

Use **DROP TRANSFORM** to remove transforms.

EXAMPLES

To create a transform for type hstore and language plpython3u, first set up the type and the language:

```
CREATE TYPE hstore ...;

CREATE EXTENSION plpython3u;
```

Then create the necessary functions:

```
CREATE FUNCTION hstore_to_plpython(val internal) RETURNS internal
LANGUAGE C STRICT IMMUTABLE
AS ...;

CREATE FUNCTION plpython_to_hstore(val internal) RETURNS hstore
LANGUAGE C STRICT IMMUTABLE
AS ...;
```

And finally create the transform to connect them all together:

```
CREATE TRANSFORM FOR hstore LANGUAGE plpython3u (
  FROM SQL WITH FUNCTION hstore_to_plpython(internal),
  TO SQL WITH FUNCTION plpython_to_hstore(internal)
);
```

In practice, these commands would be wrapped up in an extension.

The contrib section contains a number of extensions that provide transforms, which can serve as real−world examples.

COMPATIBILITY

This form of **CREATE TRANSFORM** is a PostgreSQL extension. There is a **CREATE TRANSFORM** command in the SQL standard, but it is for adapting data types to client languages. That usage is not supported by PostgreSQL.

SEE ALSO

CREATE FUNCTION (**CREATE_FUNCTION**(7)), CREATE LANGUAGE (**CREATE_LANGUAGE**(7)), CREATE TYPE (**CREATE_TYPE**(7)), DROP TRANSFORM (**DROP_TRANSFORM**(7))