

Quick Links



[Online Demo](#)



[Download](#)



[FAQ](#)



[Latest Docs](#)



[Ask a
Question](#)



[Report a Bug](#)



[Screenshots](#)

This page in other versions: [Latest \(4.19\)](#) | [4.18](#) | [4.17](#) | [3.6](#) | [2.1](#) | [1.6](#) | [Development](#)

[pgAdmin 4 4.19 documentation](#) » [Managing Database Objects](#) » [previous](#) | [next](#) | [index](#)

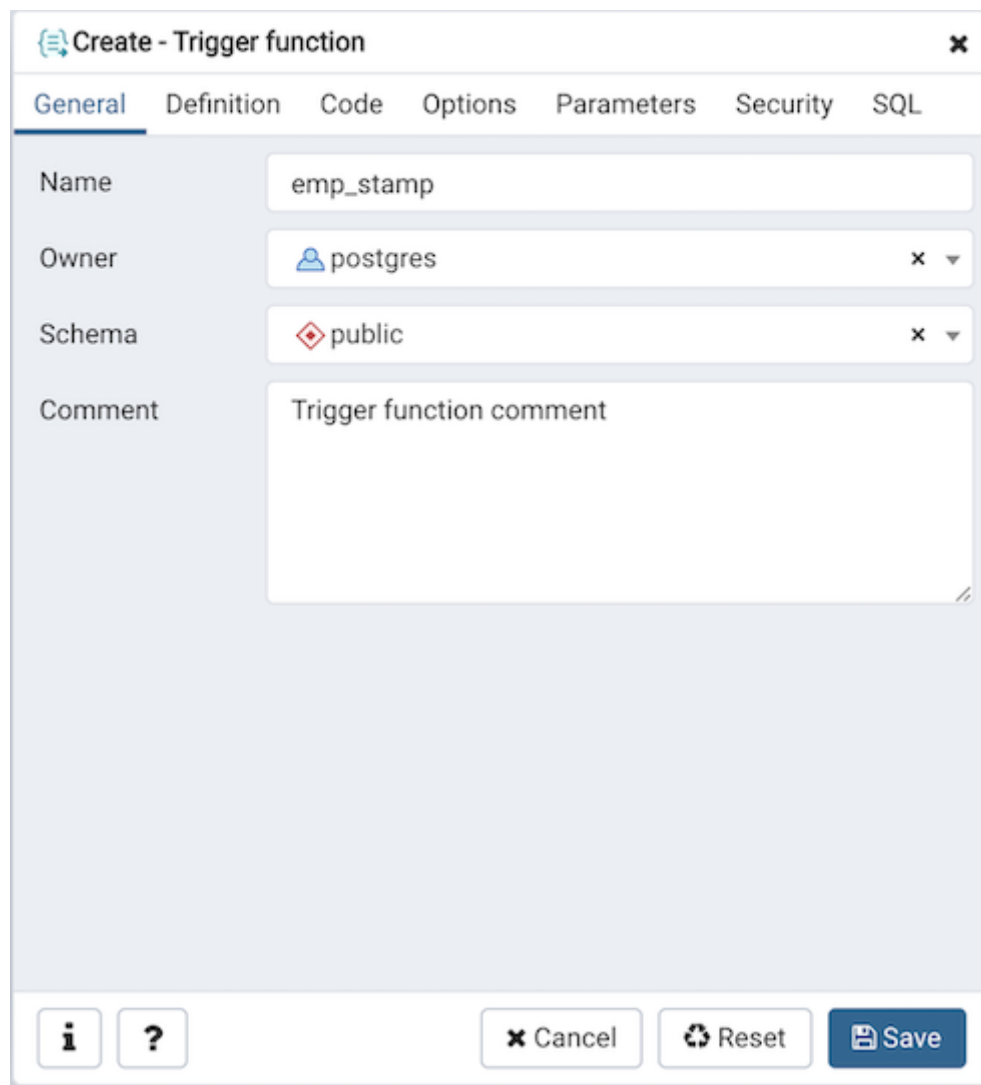
Warning: This documentation is for a pre-release version of pgAdmin 4

Contents

- [Trigger Function Dialog](#)
 - [Example](#)
- [Getting Started](#)
- [Connecting To A Server](#)
- [Managing Cluster Objects](#)
- [Managing Database Objects](#)
 - [Cast Dialog](#)
 - [Collation Dialog](#)
 - [Domain Dialog](#)
 - [Domain Constraints Dialog](#)
 - [Event Trigger Dialog](#)
 - [Extension Dialog](#)
 - [Foreign Data Wrapper Dialog](#)
 - [Foreign Server Dialog](#)
 - [Foreign Table Dialog](#)
 - [FTS Configuration Dialog](#)
 - [FTS Dictionary Dialog](#)
 - [FTS Parser Dialog](#)
 - [FTS Template Dialog](#)
 - [Function Dialog](#)
 - [Language Dialog](#)
 - [Materialized View Dialog](#)
 - [Package Dialog](#)
 - [Procedure Dialog](#)
 - [Schema Dialog](#)
 - [Sequence Dialog](#)
 - [Synonym Dialog](#)
 - [Trigger Function Dialog](#)
 - [Type Dialog](#)
 - [User Mapping Dialog](#)
 - [View Dialog](#)
- [Creating or Modifying a Table](#)
- [Management Basics](#)
- [Backup and Restore](#)
- [Developer Tools](#)
- [pgAgent](#)
- [pgAdmin Project Contributions](#)
- [Release Notes](#)
- [Licence](#)

Use the *Trigger function* dialog to create or manage a trigger_function. A trigger function defines the action that will be invoked when a trigger fires.

The *Trigger function* dialog organizes the development of a trigger function through the following dialog tabs: *General*, *Definition*, *Code*, *Options*, *Parameters* and *Security*. The *SQL* tab displays the SQL code generated by dialog selections.



The screenshot shows the 'Create - Trigger function' dialog box with the 'General' tab selected. The dialog has a title bar with a close button. Below the title bar are tabs for 'General', 'Definition', 'Code', 'Options', 'Parameters', 'Security', and 'SQL'. The 'General' tab contains the following fields:

- Name:** A text input field containing 'emp_stamp'.
- Owner:** A dropdown menu showing 'postgres' with a user icon and a close button.
- Schema:** A dropdown menu showing 'public' with a database icon and a close button.
- Comment:** A text area containing 'Trigger function comment'.

At the bottom of the dialog are three buttons: 'Cancel', 'Reset', and 'Save'.

Use the fields in the *General* tab to identify the trigger function:

- Use the *Name* field to add a descriptive name for the trigger function. The name will be displayed in the *pgAdmin* tree control. Please note that trigger functions will be invoked in alphabetical order.
- Use the drop-down listbox next to *Owner* to select the role that will own the trigger function.
- Select the name of the schema in which the trigger function will reside from the drop-down listbox in the *Schema* field.
- Store notes about the trigger function in the *Comment* field.

Click the *Definition* tab to continue.

The screenshot shows the 'Create - Trigger function' dialog box with the 'Definition' tab selected. The 'Return type' dropdown is set to 'trigger' and the 'Language' dropdown is set to 'plpgsql'. The bottom of the dialog features an information icon, a help icon, and buttons for 'Cancel', 'Reset', and 'Save'.

Use the fields in the *Definition* tab to define the trigger function:

- Use the drop-down listbox next to *Return type* to specify the pseudotype that is associated with the trigger function:
 - Select *trigger* if you are creating a DML trigger.
 - Select *event_trigger* if you are creating a DDL trigger.
- Use the drop-down listbox next to *Language* to select the implementation language. The default is *plpgsql*.

Click the *Code* tab to continue.

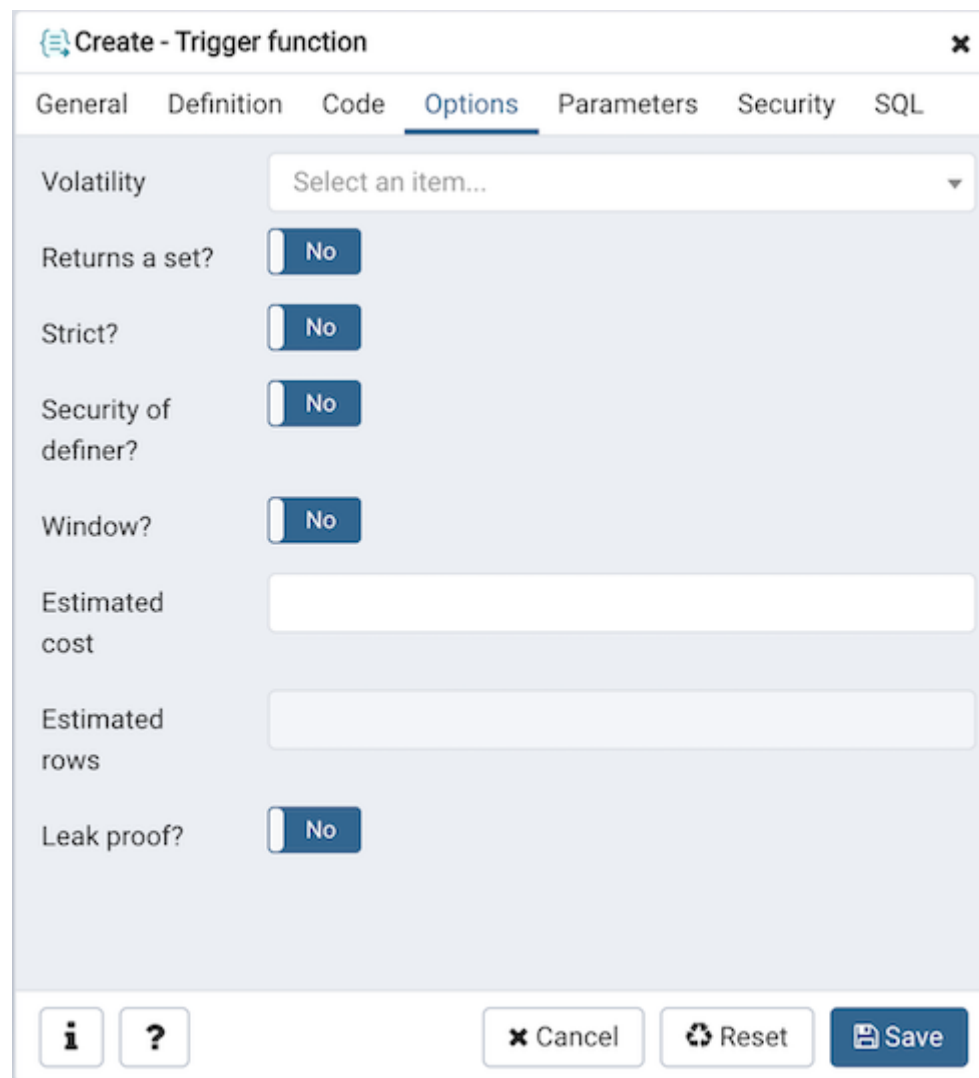
The screenshot shows the 'Create - Trigger function' dialog box with the 'Code' tab selected. The code editor contains the following SQL code:

```
1 BEGIN
2     SELECT 1;
3 END;
```

The bottom of the dialog features an information icon, a help icon, and buttons for 'Cancel', 'Reset', and 'Save'.

- Use the *Code* field to write the code that will execute when the trigger function is called.

Click the *Options* tab to continue.



The screenshot shows the 'Create - Trigger function' dialog box with the 'Options' tab selected. The dialog has tabs for General, Definition, Code, Options, Parameters, Security, and SQL. The Options tab contains the following fields and controls:

- Volatility:** A drop-down listbox with the text 'Select an item...' and a downward arrow.
- Returns a set?:** A toggle switch currently set to 'No'.
- Strict?:** A toggle switch currently set to 'No'.
- Security of definer?:** A toggle switch currently set to 'No'.
- Window?:** A toggle switch currently set to 'No'.
- Estimated cost:** An empty text input field.
- Estimated rows:** An empty text input field.
- Leak proof?:** A toggle switch currently set to 'No'.

At the bottom of the dialog, there are buttons for 'i' (info), '?' (help), 'Cancel', 'Reset', and 'Save'.

Use the fields in the *Options* tab to describe or modify the action of the trigger function:

- Use the drop-down listbox next to *Volatility* to select one of the following:
 - *VOLATILE* indicates that the trigger function value can change even within a single table scan.
 - *STABLE* indicates that the trigger function cannot modify the database, and that within a single table scan it will consistently return the same result for the same argument values.
 - *IMMUTABLE* indicates that the trigger function cannot modify the database and always returns the same result when given the same argument values.
- Move the *Returns a Set?* switch to indicate if the trigger function returns a set that includes multiple rows. The default is *No*.
- Move the *Strict?* switch to indicate if the trigger function always returns NULL whenever any of its arguments are NULL. If Yes, the function is not executed when there are NULL arguments; instead a NULL result is assumed automatically. The default is *No*.
- Move the *Security of definer?* switch to specify that the trigger function is to be executed with the privileges of the user that created it. The default is *No*.
- Move the *Window?* switch to indicate that the trigger function is a window function rather than a plain function. The default is *No*. This is currently only useful for trigger functions written in C.

- Use the *Estimated cost* field to specify a positive number representing the estimated execution cost for the trigger function, in units of `cpu_operator_cost`. If the function returns a set, this is the cost per returned row.
- Use the *Estimated rows* field to specify a positive number giving the estimated number of rows that the query planner should expect the trigger function to return. This is only allowed when the function is declared to return a set. The default assumption is 1000 rows.
- Move the *Leak proof?* switch to indicate whether the trigger function has side effects. The default is *No*. This option can only be set by the superuser.

Click the *Parameters* tab to continue.

	Name	Value
	array_nulls	False

Use the fields in the *Parameters* tab to specify settings that will be applied when the trigger function is invoked. Click the *Add* icon (+) to add a *Name/Value* pair to the table below.

- Use the drop-down listbox in the *Name* field to select a parameter.
- Use the *Value* field to specify the value that will be associated with the selected parameter. This field is context-sensitive.

Click the *Add* icon (+) to set additional parameters; to discard a parameter, click the trash icon to the left of the row and confirm deletion in the *Delete Row* popup.

Click the *Security* tab to continue.

Create - Trigger function

General Definition Code Options Parameters **Security** SQL

Privileges +

Grantee	Privileges	Grantor
postgres	<input checked="" type="checkbox"/> EXECUTE <input checked="" type="checkbox"/> WITH GRANT OPTION	postgres

Security labels +

Provider	Security label
----------	----------------

i ? Cancel Reset Save

Use the *Security* tab to assign privileges and define security labels.

Use the *Privileges* panel to assign usage privileges for the trigger function to a role. Click the *Add* icon (+) to add a role to the table.

- Select the name of the role from the drop-down listbox in the *Grantee* field.
- Click inside the *Privileges* field. Check the boxes to the left of one or more privileges to grant the selected privilege to the specified user.
- The current user, who is the default grantor for granting the privilege, is displayed in the *Grantor* field.

Click the *Add* icon (+) to assign additional privileges; to discard a privilege, click the trash icon to the left of the row and confirm deletion in the *Delete Row* popup.

Use the *Security Labels* panel to define security labels applied to the trigger function. Click the *Add* icon (+) to add each security label selection:

- Specify a security label provider in the *Provider* field. The named provider must be loaded and must consent to the proposed labeling operation.
- Specify a security label in the *Security Label* field. The meaning of a given label is at the discretion of the label provider. PostgreSQL places no restrictions on whether or how a label provider must interpret security labels; it merely provides a mechanism for storing them.

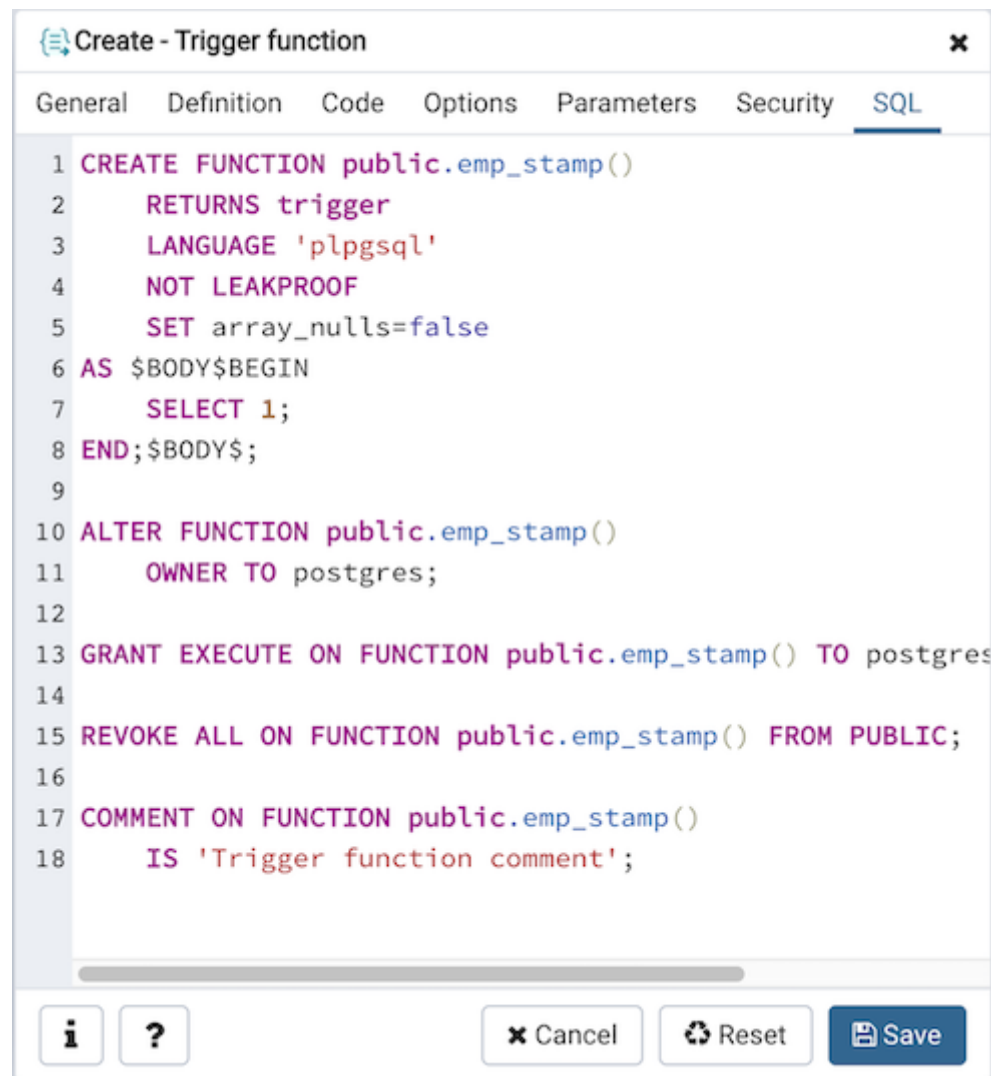
Click the *Add* icon (+) to assign additional security labels; to discard a security label, click the trash icon to the left of the row and confirm deletion in the *Delete Row* popup.

Click the *SQL* tab to continue.

Your entries in the *Trigger function* dialog generate a SQL command (see an example below). Use the SQL tab for review; revisit other tabs to modify the SQL command.

Example ¶

The following is an example of the sql command generated by user selections in the *Trigger function* dialog:



The screenshot shows a dialog box titled "Create - Trigger function" with a close button (X) in the top right corner. The dialog has several tabs: General, Definition, Code, Options, Parameters, Security, and SQL. The SQL tab is currently selected and highlighted. The SQL tab contains a text area with the following SQL code:

```
1 CREATE FUNCTION public.emp_stamp()  
2     RETURNS trigger  
3     LANGUAGE 'plpgsql'  
4     NOT LEAKPROOF  
5     SET array_nulls=false  
6 AS $BODY$BEGIN  
7     SELECT 1;  
8 END;$BODY$;  
9  
10 ALTER FUNCTION public.emp_stamp()  
11     OWNER TO postgres;  
12  
13 GRANT EXECUTE ON FUNCTION public.emp_stamp() TO postgres;  
14  
15 REVOKE ALL ON FUNCTION public.emp_stamp() FROM PUBLIC;  
16  
17 COMMENT ON FUNCTION public.emp_stamp()  
18     IS 'Trigger function comment';
```

At the bottom of the dialog, there are four buttons: an information button (i), a help button (?), a Cancel button (x), and a Save button (Save).

The example shown demonstrates creating a trigger function named *emp_stamp*.

- Click the *Info* button (i) to access online help.
- Click the *Save* button to save work.
- Click the *Cancel* button to exit without saving work.
- Click the *Reset* button to restore configuration parameters.