

## 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](#) | [Development](#)

[pgAdmin 4 4.19 documentation](#) » [Creating or Modifying a Table](#) » [previous](#) | [next](#) | [index](#)

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

### Contents

- [Compound Trigger Dialog](#)
  - [Example](#)
- [Getting Started](#)
- [Connecting To A Server](#)
- [Managing Cluster Objects](#)
- [Managing Database Objects](#)
- [Creating or Modifying a Table](#)
  - [Check Dialog](#)
  - [Column Dialog](#)
  - [Compound Trigger Dialog](#)
  - [Exclusion Constraint Dialog](#)
  - [Foreign key Dialog](#)
  - [Index Dialog](#)
  - [Primary key Dialog](#)
  - [Rule Dialog](#)
  - [Table Dialog](#)
  - [Trigger Dialog](#)
  - [Unique Constraint Dialog](#)
- [Management Basics](#)
- [Backup and Restore](#)
- [Developer Tools](#)
- [pgAgent](#)
- [pgAdmin Project Contributions](#)
- [Release Notes](#)
- [Licence](#)

## Compound Trigger Dialog

Use the *Compound Trigger* dialog to create a compound trigger or modify an existing compound trigger. *Compound Trigger* is supported only for EPAS server 12 and above. A compound trigger executes a specified code when certain events occur.

The *Compound Trigger* dialog organizes the development of a compound trigger through the following dialog tabs: *General*, *Events*, and *Code*. The *SQL* tab displays the SQL code generated by dialog selections.

**Create - Compound Trigger**

General Events Code SQL

Name: test\_ct

Comment: This is a compound trigger

Buttons: [i] [?] [Cancel] [Reset] [Save]

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

- Use the *Name* field to add a descriptive name for the compound trigger. This must be distinct from the name of any other compound trigger for the same table. The name will be displayed in the *pgAdmin* tree control.
- Store notes about the compound trigger in the *Comment* field.

**test\_ct**

General Events Code SQL

Name: test\_ct

Trigger enabled?: Enable

Comment:

Buttons: [i] [?] [Cancel] [Reset] [Save]

- *Trigger enabled* field is available in compound trigger dialog once the trigger is created. You can select one of the four options available.

Click the *Events* tab to continue.

The screenshot shows a 'Create - Compound Trigger' dialog box with four tabs: General, Events (selected), Code, and SQL. The 'Events' tab contains a 'FOR Events' section with four event types: INSERT (switched to 'Yes'), UPDATE (switched to 'Yes'), DELETE (switched to 'No'), and TRUNCATE (switched to 'No'). Below this is a 'When' field with the condition '1 NEW.id < 100'. The 'Columns' field contains two selected columns: 'id' and 'name'. At the bottom, there are buttons for 'Cancel', 'Reset', and 'Save', along with information and help icons.

Use the fields in the *Events* tab to specify how and when the compound trigger fires:

- Select the type of event(s) that will invoke the compound trigger; to select an event type, move the switch next to the event to the *YES* position. The supported event types are *INSERT*, *UPDATE*, *DELETE* and *TRUNCATE*. Views cannot have *TRUNCATE* triggers.
- Use the *When* field to provide a boolean condition that will invoke the compound trigger.
- If defining a column-specific compound trigger, use the *Columns* field to specify the columns or columns that are the target of the compound trigger.

Click the *Code* tab to continue.

The screenshot shows a dialog box titled "Create - Compound Trigger" with a close button (X) in the top right corner. It has four tabs: "General", "Events", "Code" (which is selected and highlighted with a blue underline), and "SQL". The "Code" tab contains a text area with a light blue background and a vertical scrollbar on the right. The text in the code area is as follows:

```
1 -- Enter any global declarations below:
2
3 -- BEFORE STATEMENT block. Delete if not required.
4 BEFORE STATEMENT IS
5     -- Enter any local declarations here
6 BEGIN
7     -- Enter any required code here
8 END;
9
10 -- AFTER STATEMENT block. Delete if not required.
11 AFTER STATEMENT IS
12     -- Enter any local declarations here
13 BEGIN
14     -- Enter any required code here
15 END;
16
17 -- BEFORE EACH ROW block. Delete if not required.
18 BEFORE EACH ROW IS
19     -- Enter any local declarations here
20 BEGIN
21     -- Enter any required code here
```

At the bottom of the dialog, there are four buttons: an information icon (i), a question mark icon (?), a "Cancel" button with a close icon (X), a "Reset" button with a circular arrow icon, and a "Save" button with a floppy disk icon.

Use the *Code* field to specify the code for the five timing events *BEFORE STATEMENT*, *AFTER STATEMENT*, *BEFORE EACH ROW*, *AFTER EACH ROW*, *INSTEAD OF EACH ROW* that will be invoked when the compound trigger fires. Basic template is provided with place holders.

Click the *SQL* tab to continue.

Your entries in the *Compound Trigger* dialog generate a SQL command (see an example below). Use the *SQL* tab for review; revisit or switch tabs to make any changes to the SQL command.

#### Example ¶

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

Create - Compound Trigger

GeneralEventsCodeSQL

```
1 CREATE OR REPLACE TRIGGER test_ct
2   FOR INSERT
3   ON enterprisedb.test
4   WHEN (NEW.id < 100)
5   COMPOUND TRIGGER
6   -- Global declaration.
7   var varchar2(20) := 'Global_var';
8
9   BEFORE STATEMENT IS
10  BEGIN
11    DBMS_OUTPUT.PUT_LINE('Before Statement: ' || var);
12    var := 'BEFORE STATEMENT';
13  END;
14  END;
```

i?

CancelResetSave

The example demonstrates creating a compound trigger named *test\_ct*.

- 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.