

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

- [Procedure 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 *Procedure* dialog to create a procedure; procedures are supported by PostgreSQL v11+ and EDB Postgres Advanced Server. The *Procedure* dialog allows you to implement options of the CREATE PROCEDURE command.

The *Procedure* dialog organizes the development of a procedure through the following dialog tabs: *General*, *Definition*, *Options*, *Arguments*, *Parameters*, and *Security*. The *SQL* tab displays the SQL code generated by dialog selections.

The screenshot shows the 'Create - Procedure' dialog box with the following details:

- Title Bar:** Create - Procedure
- Tabs:** General (selected), Definition, Options, Arguments, Parameters, Security, SQL
- Name:** list_emp
- Owner:** enterisedb
- Schema:** public
- Comment:** This procedure returns list of employees.
- Buttons:** i, ?, Cancel, Reset, Save

Use the fields in the *General* tab to identify a procedure:

- Use the *Name* field to add a descriptive name for the procedure. The name will be displayed in the *pgAdmin* tree control.
- Use the drop-down listbox next to *Owner* to select a role.
- Select the name of the schema in which the procedure will reside from the drop-down listbox in the *Schema* field.
- Store notes about the procedure in the *Comment* field.

Click the *Definition* tab to continue.

Create - Procedure

General

Definition

Options

Arguments

Parameters

Security

SQL

Language

edbspl

Code

```
1 create table pem.test
2 (
3 coll integer,
4 primary key (coll)
5 )
```

i

?

Cancel

Reset

Save

Use the fields in the *Definition* tab to define the procedure:

- Use the drop-down listbox next to *Language* to select a language. The default is *edbspl*.
- Use the *Code* field to specify the code that will execute when the procedure is called.

Click the *Options* tab to continue.

Create - Procedure

General

Definition

Options

Arguments

Parameters

Security

SQL

Volatility

VOLATILE

Strict?

Yes

Security of definer?

Yes

Parallel

UNSAFE

Estimated cost

Leak proof?

Yes

i

?

Cancel

Reset

Save

Use the fields in the *Options* tab to describe or modify the behavior of the procedure:

- Use the drop-down listbox under *Volatility* to select one of the following. *VOLATILE* is the default value.

- *VOLATILE* indicates that the value can change even within a single table scan, so no optimizations can be made.
- *STABLE* indicates that the procedure cannot modify the database, and that within a single table scan it will consistently return the same result for the same argument values, but that its result could change across SQL statements.
- *IMMUTABLE* indicates that the procedure cannot modify the database and always returns the same result when given the same argument values.

- Move the *Strict?* switch to indicate if the procedure always returns NULL whenever any of its arguments are NULL. If Yes, the procedure 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 procedure is to be executed with the privileges of the user that created it. The default is *No*.
- Use the *Estimated cost* field to specify a positive number representing the estimated execution cost for the procedure, in units of `cpu_operator_cost`. If the procedure returns a set, this is the cost per returned row.
- Move the *Leak proof?* switch to indicate whether the procedure has side effects — it reveals no information about its arguments other than by its return value. The default is *No*.

Click the *Arguments* tab to continue.

Create - Procedure

General

Definition

Options

Arguments

Parameters

Security

SQL

Arguments

	Data Type	Mode	Argument Name	Default Value
	"char" x ▾	IN		

i

?

Cancel

Reset

Save

Use the fields in the *Arguments* tab to define an argument. Click *Add* to set parameters and values for the argument:

- Use the drop-down listbox next to *Data type* to select a data type.

- Use the drop-down listbox next to *Mode* to select a mode. Select *IN* for an input parameter; select *OUT* for an output parameter; select *INOUT* for both an input and an output parameter; or, select *VARIADIC* to specify a VARIADIC parameter.
- Write a name for the argument in the *Argument Name* field.
- Specify a default value for the argument in the *Default Value* field.

Click *Add* to define another argument; to discard an argument, click the trash icon to the left of the row and confirm deletion in the *Delete Row* popup.

Click the *Parameters* tab to continue.

Create - Procedure

General

Definition

Options

Arguments

Parameters

Security

SQL

Parameters

	Name	Value
<div></div>	<div>array_nulls</div>	<div>False</div>

i

?

Cancel

Reset

Save

Use the fields in the *Parameters* tab to specify settings that will be applied when the procedure is invoked:

- Use the drop-down listbox next to *Parameter Name* in the *Parameters* panel to select a parameter.
- Click the *Add* button to add the variable to *Name* field in the table.
- Use the *Value* field to specify the value that will be associated with the selected variable. This field is context-sensitive.

Click the *Security* tab to continue.

Create - Procedure

General

Definition

Options

Arguments

Parameters

Security

SQL

Privileges

	Grantee	Privileges	Grantor
	<div> <div></div> <div>enterisedb</div> </div>	<div> <div>EXECUTE</div> <div>WITH GRANT OPTION</div> </div>	<div> <div></div> <div>enterisedb</div> </div>

i

?

Cancel

Reset

Save

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

Use the *Privileges* panel to assign execute privileges for the procedure to a role:

- 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 *Add* 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 procedure. Click *Add* 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.

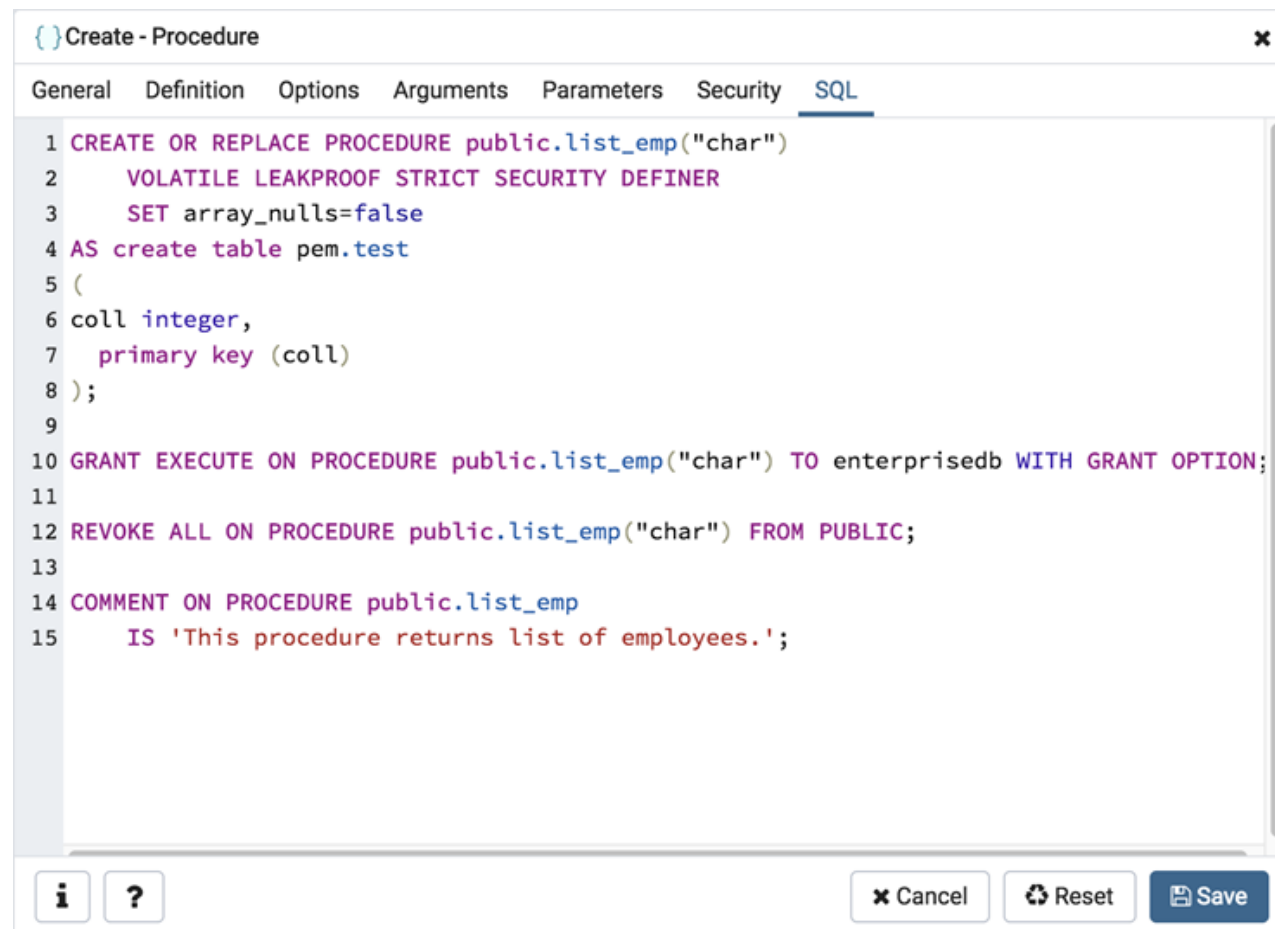
Click *Add* to assign additional security labels; to discard a security label, click the trash icon to the left of the row and confirm deletion in the *Delete Row* popup.

Click the *SQL* tab to continue.

Your entries in the *Procedure* 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 selections made in the *Procedure* dialog:



```
1 CREATE OR REPLACE PROCEDURE public.list_emp("char")
2     VOLATILE LEAKPROOF STRICT SECURITY DEFINER
3     SET array_nulls=false
4 AS create table pem.test
5 (
6 coll integer,
7     primary key (coll)
8 );
9
10 GRANT EXECUTE ON PROCEDURE public.list_emp("char") TO enterprisedb WITH GRANT OPTION;
11
12 REVOKE ALL ON PROCEDURE public.list_emp("char") FROM PUBLIC;
13
14 COMMENT ON PROCEDURE public.list_emp
15     IS 'This procedure returns list of employees.';
```

The example demonstrates creating a procedure that returns a list of employees from a table named *emp*. The procedure is a SECURITY DEFINER, and will execute with the privileges of the role that defined the procedure.

- 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.