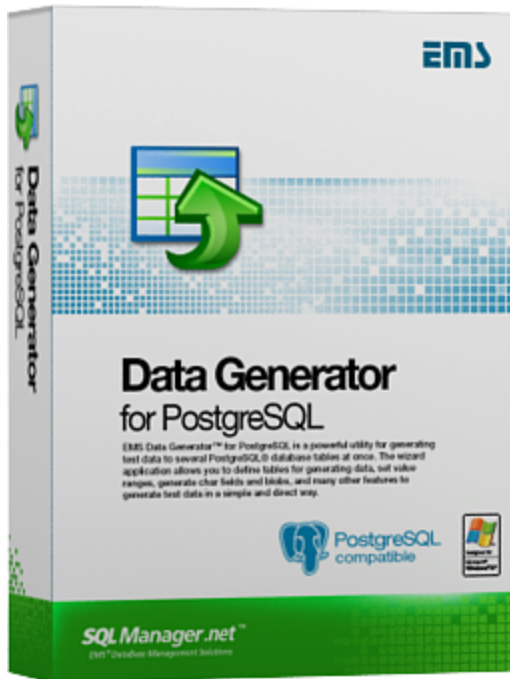


SQL Manager.net™

EMS® Software Development



Data Generator for PostgreSQL User's Manual

© 1999-2025 EMS Software Development

Data Generator for PostgreSQL

User's Manual

© 1999-2025 EMS Software Development

All rights reserved.

This manual documents EMS Data Generator for PostgreSQL, version 3.0.x.x

No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Use of this documentation is subject to the following terms: you may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way.

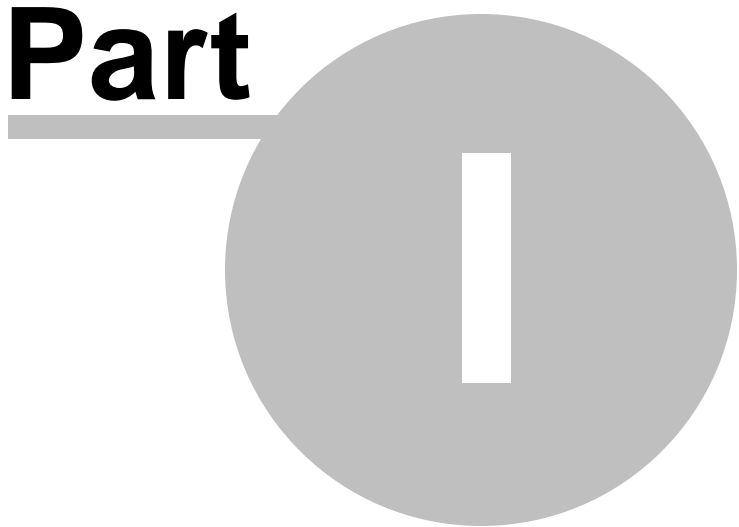
Document generated on: 26.06.2025

Table of Contents

Part I Welcome to EMS Data Generator!	6
What's new	7
System requirements	8
Installation	9
Registration	10
How to register Data Generator	12
EMS Data Generator FAQ	13
Other EMS Products	15
Part II Wizard Application	22
Using wizard application	23
Getting started	24
Step 1 - Setting connection properties	25
Selecting registered database.....	27
Step 2 - Selecting databases and tables	28
Step 3 - Specifying generation parameters	30
Setting type-specific properties.....	33
INTEGER column parameters.....	33
FLOAT column parameters.....	36
DATE column parameters.....	39
TIME column parameters.....	42
STRING column parameters.....	45
BLOB column parameters.....	49
ARRAY column parameters.....	52
GEOMETRIC column parameters.....	53
BOOLEAN column parameters.....	56
BIT column parameters.....	59
INTERVAL column parameters.....	62
JSON / JSONB column parameters.....	64
Viewing table DDL.....	68
Data Preview	69
Step 4 - Setting generation options	71
Step 5 - Start of data generation process	73
Step 6 - Editing generation script	75
Using configuration files	77
Saving configuration file	78
Save Template options.....	79
Loading configuration file	80
Setting program preferences	81
Setting general options	82
Setting default constraints	84
Numeric types.....	85
Character types.....	86

Date/Time types.....	88
JSON types.....	89
Setting program language	90
Part III Console Application	92
Using console application	93
Part IV Appendix	95
SSH tunneling options	95
HTTP tunneling options	96
Data generation mode	97
Configuration file format	98
Find Text dialog	101
Replace Text dialog	103

Part



1 Welcome to EMS Data Generator!

EMS Data Generator for PostgreSQL is a powerful utility for generating test data into one or several PostgreSQL database tables simultaneously, with script saving and editing capabilities. The wizard application allows you to define tables and columns for generating data, set value ranges, generate string values by mask, load values for BLOB columns directly from files, set lists of values manually, get sets of values from SQL queries and perform other operations with test data in a simple and direct way. The distribution of the utility also provides you with the console application which allows you to generate data in one-touch by using data generation templates.

Visit our web-site: <https://www.sqlmanager.net> for details.

Key features

- Unicode support
- Localizable user-friendly wizard interface
- Ability to save and edit data generation script, without actual script execution
- Generating data into several tables of different databases at one host
- Support for all PostgreSQL data types including arrays, network addresses and geometric types, BIT type, and user-defined types
- Various data generation modes for each data type including list, random, incremental data generation and more
- Ability to use SQL query results as the list of values for data generation
- Ability to get data from another column for data generation
- Ability to preview the data grid for each table
- Automatic control over referential integrity for data generation into linked tables
- Wide variety of generation parameters for each data type
- Ability to set NULL values for a specified percent of cases
- Ability to empty tables before data generation
- Possibility of saving all the generation parameters specified within the current wizard session
- The command-line utility to generate data using the configuration file

Product information

Homepage: <https://www.sqlmanager.net/products/postgresql/datagenerator>

Support Ticket <https://www.sqlmanager.net/support>

System:

Register online at: <https://www.sqlmanager.net/products/postgresql/datagenerator/buy>

1.1 What's new

Version

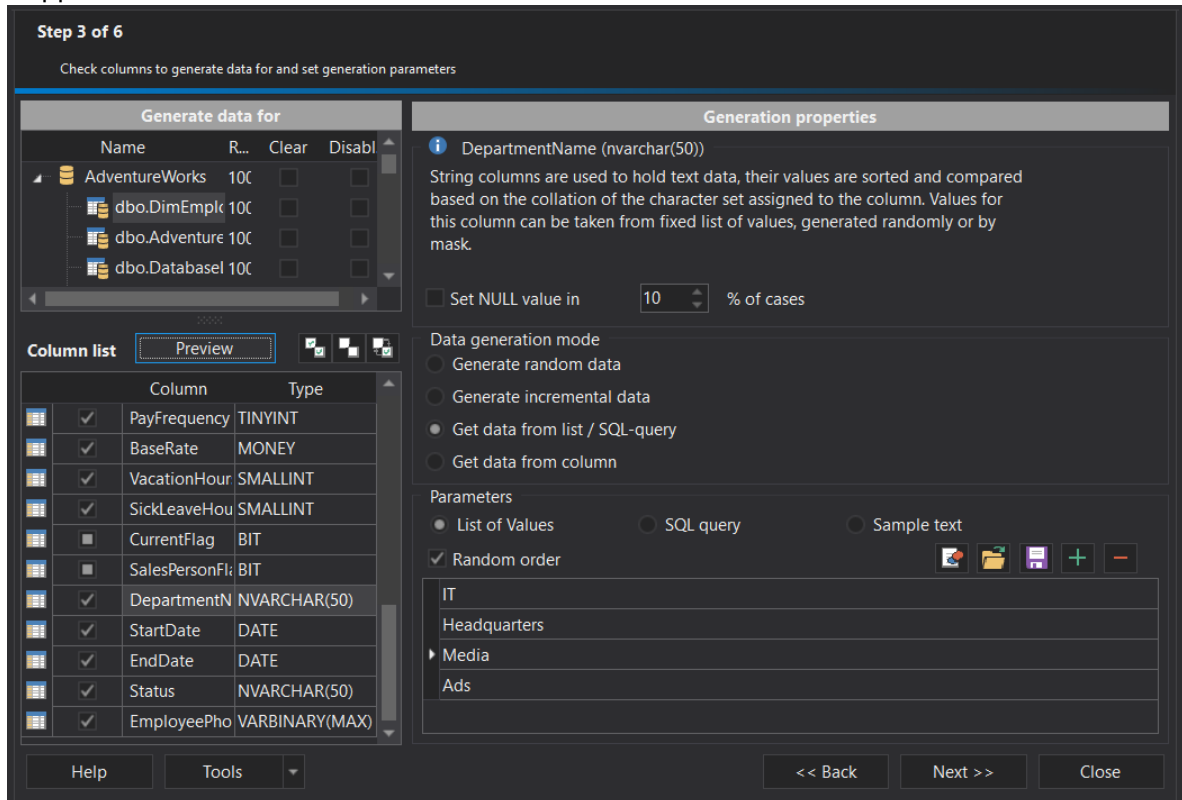
Data Generator for PostgreSQL 4.0

Release Date

November 21, 2023

What's new in EMS Data Generator 4.0?

- Support for dark visual theme added.



- Unicode names of objects are now supported.
- SSL implemented for PostgreSQL server versions.
- Updated SSH library with ECDSA, Ed25519 keys and Keyboard-interactive authentication method support.
- Support for the PostgreSQL 16 server version.
- JSON data type is now supported.
- Schema selection is now available.
- Encoding option for script file has been added.
- Files with test data are now supplied with the program installation pack.
- Many other improvements and fixes.

1.2 System requirements

System requirements for Data Generator for PostgreSQL

- Microsoft Windows XP, Microsoft Windows Server 2003, Microsoft Windows Server 2008, Microsoft Windows Server 2008 R2, Microsoft Windows Server 2012, Microsoft Windows Server 2012 R2, Microsoft Windows Server 2016, Microsoft Windows Vista, Microsoft Windows 7, Microsoft Windows 8/8.1, Microsoft Windows 10, Microsoft Windows 11, Microsoft Windows 11 ARM
- 50MB of available HD space for program installation
- Possibility to connect to any local or remote PostgreSQL server
- Supported PostgreSQL server versions: from 7.3 up to 17

1.3 Installation

If you are **installing Data Generator for PostgreSQL for the first time** on your PC:

- download the Data Generator for PostgreSQL distribution package from the [download page](#) available at our site;
- unzip the downloaded file to any local directory, e.g. *C:\unzipped*;
- run *PgDataGenSetup.exe* from the local directory and follow the instructions of the installation wizard;
- after the installation process is completed, find the Data Generator shortcut in the corresponding group of Windows Start menu.

If you want to **upgrade an installed copy of Data Generator for PostgreSQL** to the latest version:

- download the Data Generator for PostgreSQL distribution package from the [download page](#) available at our site;
- unzip the downloaded file to any local directory, e.g. *C:\unzipped*;
- close Data Generator application if it is running;
- run *PgDataGenSetup.exe* from the local directory and follow the instructions of the installation wizard.

See also:

[System requirements](#)

1.4 Registration

All purchases are provided by **PayPro Global** registration service. The **PayPro Global** order process is protected via a secure connection and makes on-line ordering by credit/debit card quick and safe.

PayPro Global is a global e-commerce provider for software and shareware sales via the Internet. It accepts payments in US Dollars, Euros, Pounds Sterling, Japanese Yen, Australian Dollars, Canadian Dollars or Swiss Franks by Credit Card (Visa, MasterCard/EuroCard, American Express, Diners Club), Bank/Wire Transfer.

If you want to review your order information, or you have questions about ordering or payments please visit our [PayPro Global Shopper Support](#), provided by **PayPro Global**.

Please note that all of our products are delivered via ESD (Electronic Software Delivery) only. After purchase you will be able to immediately download the registration keys. Also you will receive a copy of registration keys by email. Please make sure to enter a valid email address in your order. If you have not received the keys within 2 hours, please, contact us at sales@sqlmanager.net.

To obtain **MORE INFORMATION** on this product, visit us at <https://www.sqlmanager.net/products/postgresql/datagenerator>

Product distribution	PayPro Global
EMS Data Generator for PostgreSQL (Business license) + 1-Year Maintenance*	Register Now!
EMS Data Generator for PostgreSQL (Business license) + 2-Year Maintenance*	
EMS Data Generator for PostgreSQL (Business license) + 3-Year Maintenance*	
EMS Data Generator for PostgreSQL (Non-commercial license) + 1-Year Maintenance*	
EMS Data Generator for PostgreSQL (Non-commercial license) + 2-Year Maintenance*	
EMS Data Generator for PostgreSQL (Non-commercial license) + 3-Year Maintenance*	Download Now!
EMS Data Generator for PostgreSQL (Trial version)	

* **EMS Maintenance Program** provides the following benefits:

- Free software bug fixes, enhancements, updates and upgrades during the maintenance period
- Free unlimited communications with technical staff for the purpose of reporting Software failures
- Free reasonable number of communications for the purpose of consultation on operational aspects of the software

After your maintenance expires you will not be able to update your software or get technical support. To protect your investments and have your software up-to-date, you need to renew your maintenance.

You can easily reinitiate/renew your maintenance with our on-line, speed-through Maintenance Reinstatement/Renewal Interface. After reinitiating/renewal you will receive a confirmation e-mail with all the necessary information.

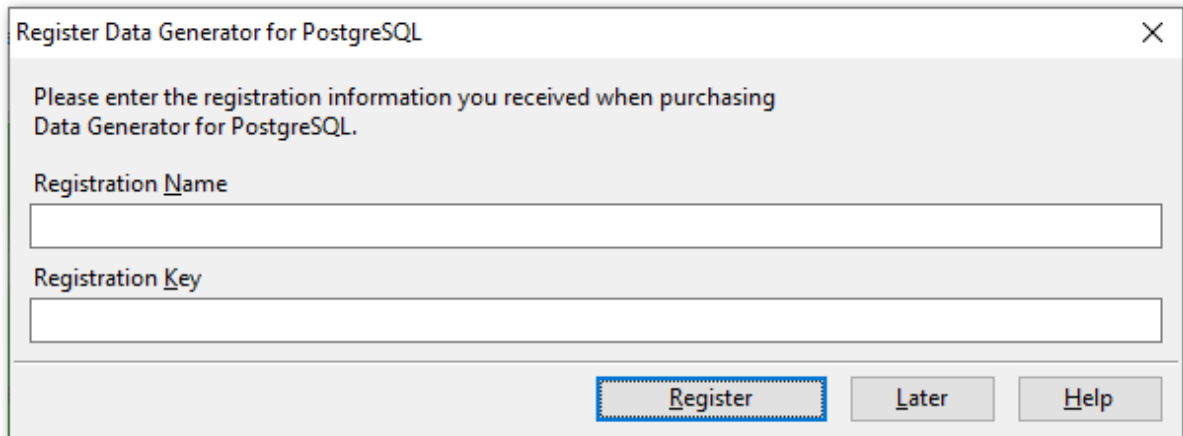
See also:

[How to register EMS Data Generator](#)

1.5 How to register Data Generator

To **register** your newly purchased copy of EMS **Data Generator for PostgreSQL**, perform the following:

- receive the notification letter from **PayPro Global** with the registration info;
- enter the **Registration Name** and the **Registration Key** from this letter;
- make sure that the registration process has been completed successfully – check the registration information at the [startup page](#).



The screenshot shows a dialog box titled "Register Data Generator for PostgreSQL" with a close button (X) in the top right corner. The main text inside the dialog reads: "Please enter the registration information you received when purchasing Data Generator for PostgreSQL." Below this text are two input fields. The first field is labeled "Registration Name" and the second field is labeled "Registration Key". At the bottom of the dialog, there are three buttons: "Register" (which is highlighted with a blue dashed border), "Later", and "Help".

See also:

[Registration](#)

1.6 EMS Data Generator FAQ

Please read this page attentively if you have questions about EMS Data Generator for PostgreSQL.

Table of contents

- [What is Data Generator for PostgreSQL?](#)
- [What do I need to start working with EMS Data Generator for PostgreSQL?](#)
- [What is the easiest way to configure the template files for the console application of Data Generator?](#)
- [How can I register the product?](#)
- [Are there any limitations implied in the trial version as compared with the full one?](#)

Question/answer list

Q: What is Data Generator for PostgreSQL?

A: Data Generator for PostgreSQL is a powerful utility for generating test data to several PostgreSQL database tables at once. The wizard application allows you to define tables for generating data, set value ranges, generate char and BLOB values, and many other features to generate test data in a simple and direct way. The utility also provides you with the console application which allows you to generate data in one touch by using generation templates.

Q: What do I need to start working with EMS Data Generator for PostgreSQL?

A: First of all you must have a possibility to connect to some local or remote PostgreSQL server to work with Data Generator for PostgreSQL. You can use the following link to download the server: <https://www.postgresql.org/download/>. Besides, you need your computer to satisfy the [system requirements](#) for Data Generator for PostgreSQL.

Q: What is the easiest way to configure the template files for the console application of Data Generator?

A: You can configure the template files for each table or export type visually using the PostgreSQL Data Generator wizard. Set the required generation options at [Step 4](#) of the wizard, click the 'Tools' button and select the 'Save Template' popup menu item. All the options will be saved to the template file which can be used later in the [console application](#).

Q: How can I register the product?

A: If you have already purchased Data Generator for PostgreSQL, you can register the product by entering the appropriate registration information. Please refer to [Registration](#) and [How to register EMS Data Generator](#) for details.

Q: Are there any limitations implied in the trial version as compared with the full one?

A: The trial version of the utility admits to the maximum of 100 records to be generated at a time. In all other respects it does not differ from the full version as far as the functionality is concerned. That is, you can test all the features implemented in Data Generator for PostgreSQL within the 30-day trial period.

[Scroll to top](#)

If you have any additional questions, contact us at our [Support Center](#).

1.7 Other EMS Products

Quick navigation

[MySQL](#)[Microsoft SQL Server](#)[PostgreSQL](#)[InterBase / FireBird](#)[Oracle](#)[IBM DB2](#)[Tools & components](#)

MySQL



[SQL Management Studio for MySQL](#)

EMS SQL Management Studio for MySQL is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!



[SQL Manager for MySQL](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



[Data Export for MySQL](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more.



[Data Import for MySQL](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.



[Data Pump for MySQL](#)

Migrate from most popular databases (MySQL, PostgreSQL, Oracle, DB2, InterBase/Firebird, etc.) to MySQL.



[Data Generator for MySQL](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



[DB Comparer for MySQL](#)

Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.



[DB Extract for MySQL](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



[SQL Query for MySQL](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.



[Data Comparer for MySQL](#)

Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

[Scroll to top](#)

Microsoft SQL Server



[SQL Management Studio for SQL Server](#)

EMS SQL Management Studio for SQL Server is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!



[EMS SQL Backup for SQL Server](#)

Perform backup and restore, log shipping and many other regular maintenance tasks on the whole set of SQL Servers in your company.



[SQL Administrator for SQL Server](#)

Perform administrative tasks in the fastest, easiest and most efficient way. Manage maintenance tasks, monitor their performance schedule, frequency and the last execution result.



[SQL Manager for SQL Server](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



[Data Export for SQL Server](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more



[Data Import for SQL Server](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.



[Data Pump for SQL Server](#)

Migrate from most popular databases (MySQL, PostgreSQL, Oracle, DB2, InterBase/Firebird, etc.) to Microsoft® SQL Server™.



[Data Generator for SQL Server](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



[DB Comparer for SQL Server](#)

Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.



[DB Extract for SQL Server](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



[SQL Query for SQL Server](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.



[Data Comparer for SQL Server](#)

Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

[Scroll to top](#)

PostgreSQL



[SQL Management Studio for PostgreSQL](#)

EMS SQL Management Studio for PostgreSQL is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!



[EMS SQL Backup for PostgreSQL](#)

Creates backups for multiple PostgreSQL servers from a single console. You can use automatic backup tasks with advanced schedules and store them in local or remote folders or cloud storages



[SQL Manager for PostgreSQL](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



[Data Export for PostgreSQL](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more



[Data Import for PostgreSQL](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.



[Data Pump for PostgreSQL](#)

Migrate from most popular databases (MySQL, SQL Server, Oracle, DB2, InterBase/Firebird, etc.) to PostgreSQL.



[Data Generator for PostgreSQL](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



[DB Comparer for PostgreSQL](#)

Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.



[DB Extract for PostgreSQL](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



[SQL Query for PostgreSQL](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.



[Data Comparer for PostgreSQL](#)

Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

[Scroll to top](#)

InterBase / Firebird



[SQL Management Studio for InterBase/Firebird](#)

EMS SQL Management Studio for InterBase and Firebird is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!



[SQL Manager for InterBase/Firebird](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



[Data Export for InterBase/Firebird](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more



[Data Import for InterBase/Firebird](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.



[Data Pump for InterBase/Firebird](#)

Migrate from most popular databases (MySQL, SQL Server, Oracle, DB2, PostgreSQL, etc.) to InterBase/Firebird.



[Data Generator for InterBase/Firebird](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



[DB Comparer for InterBase/Firebird](#)

Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.



[DB Extract for InterBase/Firebird](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



[SQL Query for InterBase/Firebird](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.



[Data Comparer for InterBase/Firebird](#)

Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

[Scroll to top](#)

Oracle



[SQL Management Studio for Oracle](#)

EMS SQL Management Studio for Oracle is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!



[SQL Manager for Oracle](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



[Data Export for Oracle](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more.



[Data Import for Oracle](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via

user-friendly wizard interface.



[Data Pump for Oracle](#)

Migrate from most popular databases (MySQL, PostgreSQL, MySQL, DB2, InterBase/Firebird, etc.) to Oracle



[Data Generator for Oracle](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



[DB Comparer for Oracle](#)

Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.



[DB Extract for Oracle](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



[SQL Query for Oracle](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.



[Data Comparer for Oracle](#)

Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

[Scroll to top](#)

IBM DB2



[SQL Manager for DB2](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



[Data Export for DB2](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more.



[Data Import for DB2](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.



[Data Pump for DB2](#)

Migrate from most popular databases (MySQL, PostgreSQL, Oracle, MySQL, InterBase/Firebird, etc.) to DB2



[Data Generator for DB2](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



[DB Extract for DB2](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



[SQL Query for DB2](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts

based on retrieved data quickly and more.

[Scroll to top](#)

Tools & components



[Advanced Data Export for RAD Studio VCL](#)

Advanced Data Export for RAD Studio VCL allows you to save your data in the most popular office programs formats.



[Advanced Data Export .NET](#)

Advanced Data Export .NET is a component for Microsoft Visual Studio .NET that will allow you to save your data in the most popular data formats for the future viewing, modification, printing or web publication. You can export data into MS Access, MS Excel, MS Word (RTF), PDF, TXT, DBF, CSV and more! There will be no need to waste your time on tiresome data conversion - Advanced Data Export will do the task quickly and will give the result in the desired format.



[Advanced Data Import for RAD Studio VCL](#)

Advanced Data Import for RAD Studio VCL will allow you to import your data to the database from files in the most popular data formats.



[Advanced PDF Generator for RAD Studio](#)

Advanced PDF Generator for RAD Studio gives you an opportunity to create PDF documents with your applications written on Delphi or C++ Builder.



[Advanced Query Builder for RAD Studio VCL](#)

Advanced Query Builder for RAD Studio VCL is a powerful component for Delphi and C++ Builder intended for visual building SQL statements for the SELECT, INSERT, UPDATE and DELETE clauses.



[Advanced Excel Report for RAD Studio](#)

Advanced Excel Report for RAD Studio is a powerful band-oriented generator of template-based reports in MS Excel.

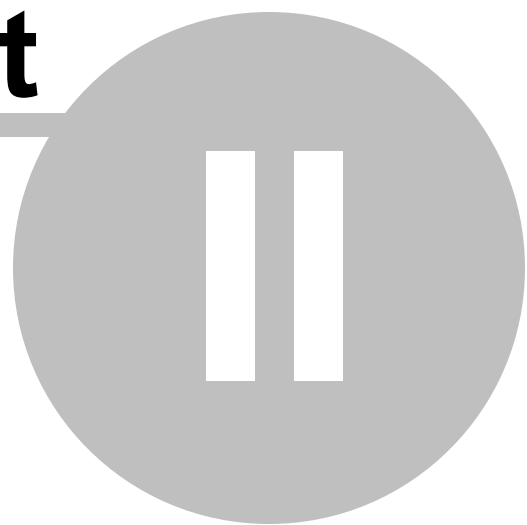


[Advanced Localizer for RAD Studio VCL](#)

Advanced Localizer for RAD Studio VCL is an indispensable component for Delphi for adding multilingual support to your applications.

[Scroll to top](#)

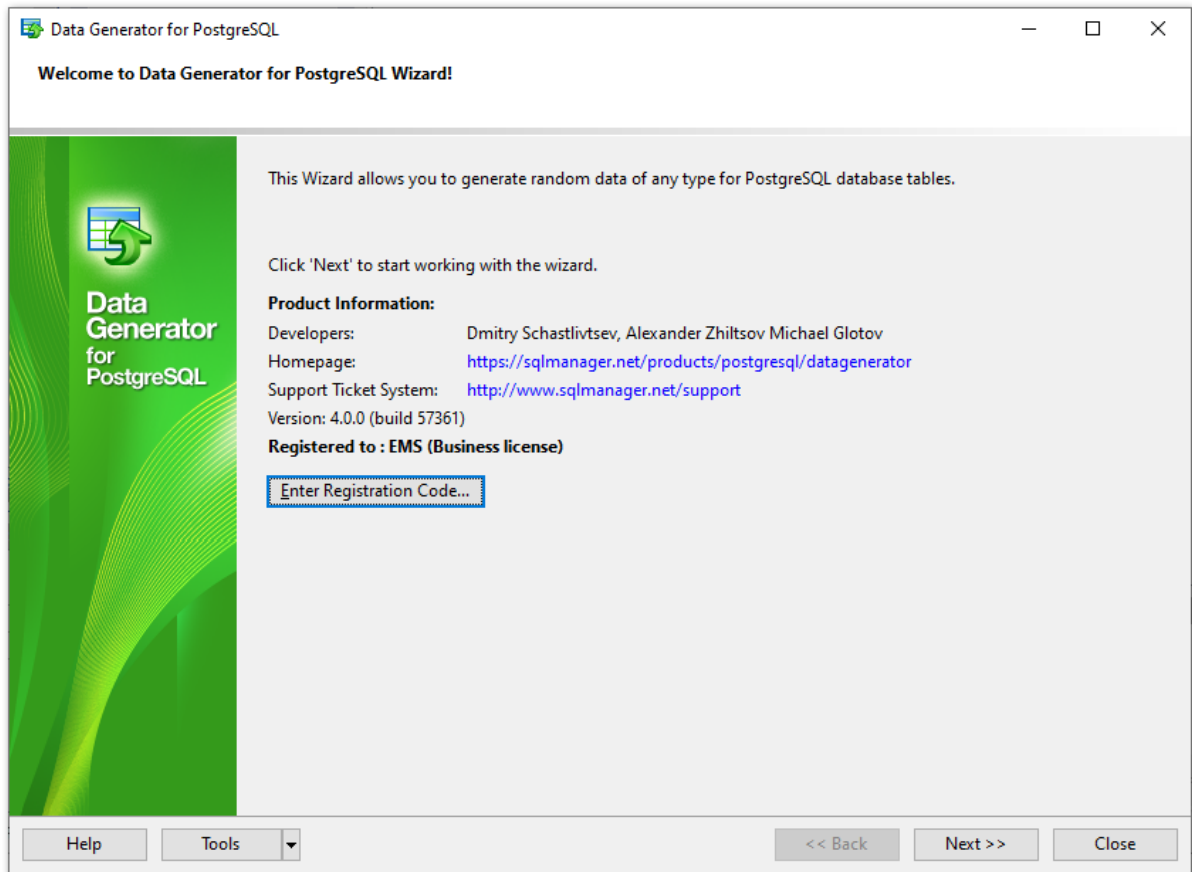
Part



2 Wizard Application

Data Generator for PostgreSQL wizard application provides easy-to-use wizard interface to set all data generation parameters visually.

- [Using wizard application](#)
- [Using configuration files](#)
- [Setting program preferences](#)



See also:

[Console Application](#)

2.1 Using wizard application

Go through the steps of the wizard and follow the wizard instructions to generate test data for your needs.

- [Getting started](#)
- [Step 1 - Setting connection properties](#)
- [Step 2 - Selecting databases and tables](#)
- [Step 3 - Specifying generation parameters](#)
- [Step 4 - Setting generation options](#)
- [Step 5 - Start of data generation process](#)
- [Step 6 - Editing generation script](#)

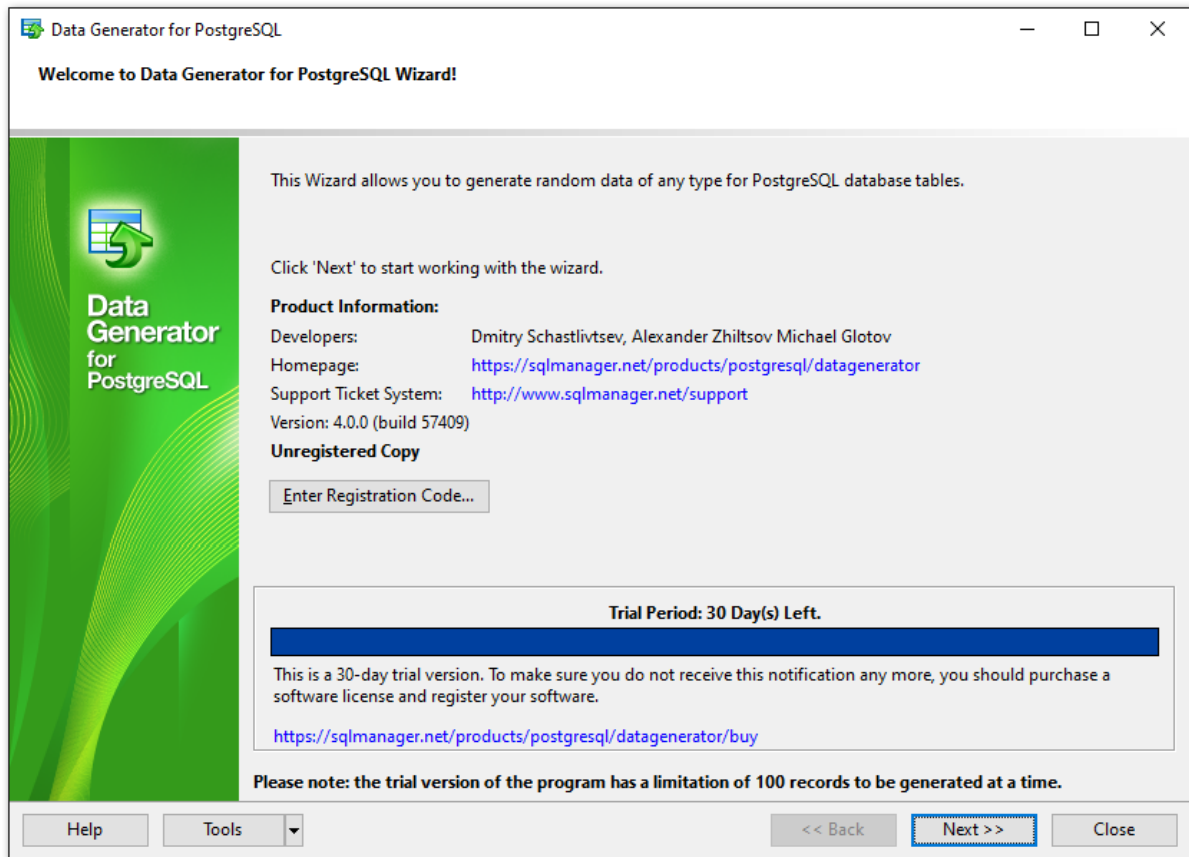
See also:

[Using console application](#)

2.1.1 Getting started

This is how Data Generator for PostgreSQL wizard application looks when you first start it.

This page allows you to view registration information. If you have not registered Data Generator for PostgreSQL yet, you can do it by pressing the **Enter Registration Code...** button and specifying your registration information.



When you are done, press the **Next** button to proceed to [Step 1](#).

See also:
[Registration](#)

2.1.2 Step 1 - Setting connection properties

At this step you should enter the necessary settings to establish connection to PostgreSQL server.

Data Generator for PostgreSQL

Step 1 of 6

Set PostgreSQL server connection properties

Host: testing-pg

Port: 54111

DB to connect: postgres

SSL mode: Disabled

Client certificate: C:\cert\V1\client.crt

Client key: C:\cert\V1\client.key

Root certificate: C:\cert\V1\root.crt

Revocation list:

☐ Don't use tunneling

☒ Connect through the Secure Shell (SSH) tunnel

SSH host: vadsrv

SSH port: 22

SSH login: tester

SSH password:

☒ Use Private Key for authentication

SSH key file: C:\SSHKeys\dsa_key.ppk

☐ Connect through the HTTP tunnel

URL: http://webserver_name/emsproxy.php

Help Tools << Back Next >> Close

For connection you should enter PostgreSQL host name in the **Host** field and enter PostgreSQL port to connect through in the **Connection port** field.

Afterwards you should specify *authorization* settings: **Login** and **Password**. The default superuser name is 'postgres' with the password specified during PostgreSQL server installation.

Please note that you should have sufficient privileges to write to the destination database on PostgreSQL server.

If you are using the EMS SQL Management Studio for PostgreSQL version of Data Generator for PostgreSQL then the **Select registered database** button is available. Click this button to pick a database already registered in the EMS SQL Management Studio in the [Select Host or Database](#) dialog.

SSL parameters

SSL preferences allows you to connect to the server via encrypted channel for increased security.

- Select the preferable **SSL mode**: *Disabled, Allow, Prefer, Require, Verify CA, Verify Full*.
- **Root certificate** - select the path to the client root.crt file.
- **Client certificate** - select the path to the client certificate.
- **Client key** - select the path to the client private key.
- **Revocation list** - select the file for Certificate Revocation List.

Tunneling settings

To setup the connection via **SSH tunnel**, input the following values in the corresponding fields:

- **SSH host name** is the name of the host where SSH server is running
- **SSH port** indicates the port where SSH server is activated
- **SSH user name** stands for the user on the machine where SSH server is running (
Note: it is a Linux/Windows user, not a user of PostgreSQL server)
- **SSH password** is the Linux/Windows user password

For details see [SSH tunneling options](#).

To use **HTTP tunneling**, just upload the tunneling script to the webserver where PostgreSQL server is located, or to any other webserver from which direct connections to your PostgreSQL server are allowed. This script exposes the PostgreSQL API as a set of web-services which is used by Data Generator for PostgreSQL.

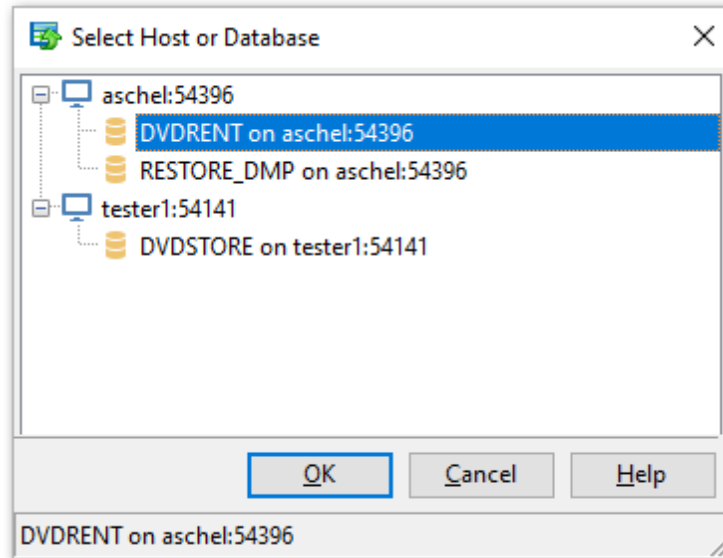
Note that the *emspoxy.php* script file is included into the distribution package and can be found in Data Generator installation directory.

For details see [HTTP tunneling options](#).

When you are done, press the **Next** button to proceed to the [next step](#).

2.1.2.1 Selecting registered database

Use this dialog to select a database for data generation. This dialog is available only in EMS SQL Management Studio version of Data Generator for PostgreSQL.



All databases registered in EMS SQL Management Studio for PostgreSQL are displayed in the list.

Select the necessary database and click the **OK** button.

Database registration information will be filled on the [first step](#) automatically.





2.1.3 Step 2 - Selecting databases and tables

At this step you should select tables for test data generation.

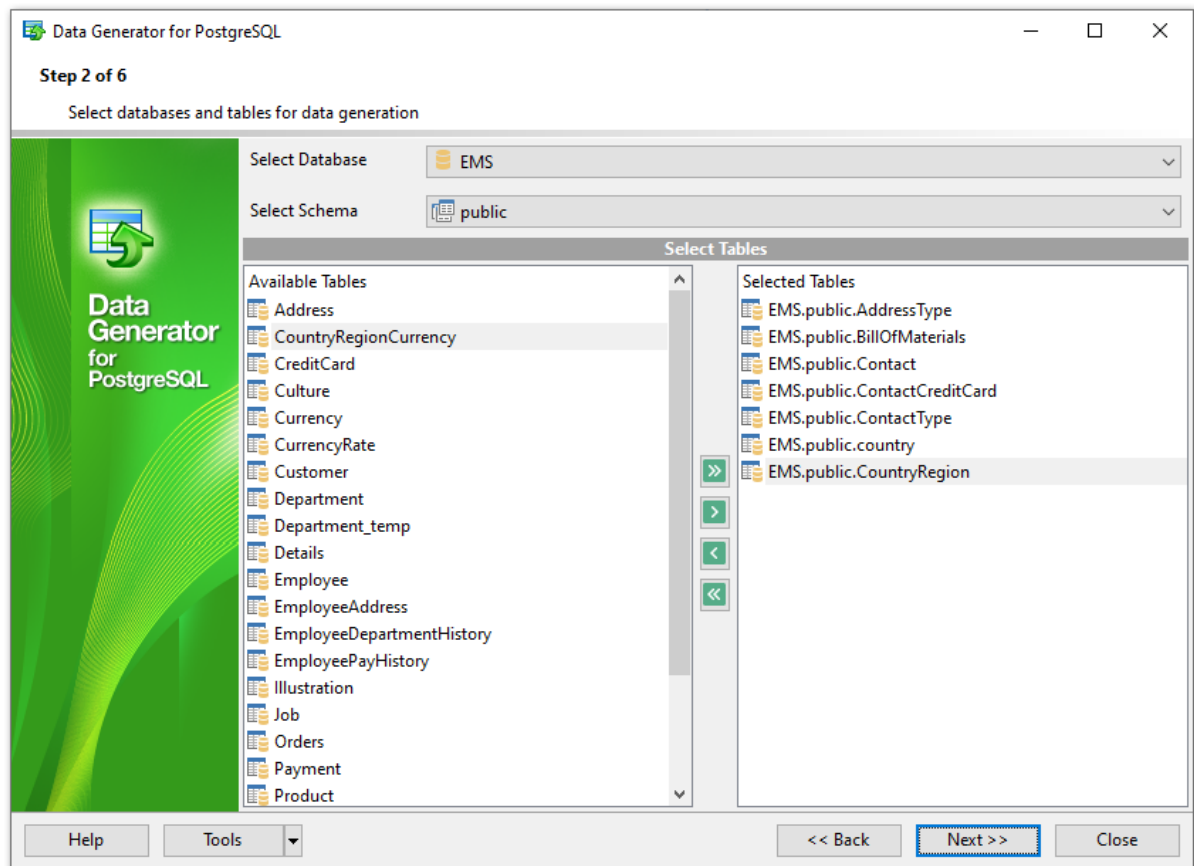
First you should select a database from the **Select Database** drop-down list at the top of the window.

Use the **Select Schema** drop-down list to select the schema in which the table should be created.

In the **Available Tables** list you can see all the tables belonging to the selected database. To select a table, you need to move it from the **Available Tables** list to the **Selected Tables** list. In this list tables are displayed with their full names: <database_name>.<table_name>. To cancel table selection, just remove it from the

Selected Tables list. Use the     buttons or drag-and-drop operations to move the tables from one list to another.

Hint: To select multiple tables, hold down the *Shift* or *Ctrl* key while selecting the table names.



Please note that the order of data generation for tables depends on their position in the **Selected Tables** list. This might be critical in case of generating data for linked tables. You can change their order by dragging tables across the list.

When you press the **Next** button at this step, Data Generator for PostgreSQL analyzes the order of data generation to avoid referential integrity conflicts and advises you to set a new order for data generation.

When you are done, press the **Next** button to proceed to the [next step](#).

2.1.4 Step 3 - Specifying generation parameters

At this step you can select columns for generating data and set various data generation parameters.

Selected tables are displayed in the **Generate data for** tree at the top-left side of the window. Table columns and their types are listed in the grid of the **Columns list** area below.

Records count

Set the number of data records to be generated for each table.

☒ Clear

Set this flag for a table to empty the table before data generation.

Preview

Click the **Preview** button to browse the selected table data in the [preview mode](#).

Data Generator for PostgreSQL

Step 3 of 6

Check columns to generate data for and set generation parameters

Generate data for		
Name	Records count	Clear
EMS	100	<input type="checkbox"/>
public.Address	350	<input checked="" type="checkbox"/>
public.Customer	1000	<input type="checkbox"/>
public.CreditCard	200	<input type="checkbox"/>
public.Orders	100	<input checked="" type="checkbox"/>

Column list	
Column	Type
<input checked="" type="checkbox"/> id	INTEGER
<input checked="" type="checkbox"/> auto_increment	INTEGER
<input checked="" type="checkbox"/> customer_number	VARCHAR(255)
<input checked="" type="checkbox"/> first_name	VARCHAR(255)
<input checked="" type="checkbox"/> last_name	VARCHAR(255)
<input checked="" type="checkbox"/> company	VARCHAR(255)
<input checked="" type="checkbox"/> password	VARCHAR(1024)
<input checked="" type="checkbox"/> legacy_password	VARCHAR(255)
<input checked="" type="checkbox"/> legacy_encoder	VARCHAR(255)
<input checked="" type="checkbox"/> email	VARCHAR(254)
<input checked="" type="checkbox"/> title	VARCHAR(100)

Generation properties

id (integer)
Integer columns are used to hold exact numeric data. Values for this column can be generated randomly, incremental or taken from fixed list of values.

☐ Set NULL value in % of cases

Data generation mode

☒ Generate random data

☐ Generate incremental data

☐ Get data from list / SQL-query

☐ Get data from column

Parameters

Min value

Max value

☐ Use formula

Help Tools << Back Next >> Close

When you select a table in the **Generate data for** tree, you can set data generation parameters for each of its columns within the **Generation properties** area at the right side of the window. Use the and the buttons to manage columns within the **Columns list** area. For each column you can set the following:

☒ **Set NULL value in ... % of cases**

Check this option and specify the percentage of NULL values for the column data, if necessary.

Other generation parameters vary according to the data type of the selected column:

- [FLOAT type parameters](#)
- [INTEGER type parameters](#)
- [DATE type parameters](#)
- [TIME type parameters](#)
- [STRING type parameters](#)
- [BLOB type parameters](#)
- [ARRAY type parameters](#)
- [GEOMETRIC type parameters](#)
- [BOOLEAN type parameters](#)
- [BIT type parameters](#)
- [INTERVAL type parameters](#)
- [JSON type parameters](#)

If a column is part of a foreign key, you can select one of the following options for this column:

☒ *Generate data from the dependent column*

Values for the column will be taken from the corresponding column(s) of the foreign table (s).

☒ *Generate data from list / SQL-query*

☒ *Ratio 1:N*

If this option is selected you should specify the *N* value using the spin-edit box below. Data will be generated into the column related as 1:N, i.e. *n* records will be generated into the foreign table for each record of the primary table.

Data generation mode

☐ Generate data from the dependent column
☐ Generate incremental data
☒ Get data from list / SQL-query
☐ Get data from column

Parameters

☐ List of Values ☐ SQL query ☒ Ratio 1:N

For each record of the primary table n records will be generated into the foreign table.(NOTE: Changing this value will result in changing record count of the current table)

1

When you are done, press the **Next** button to proceed to the [next step](#) of the Wizard.

2.1.4.1 Setting type-specific properties

2.1.4.1.1 INTEGER column parameters

Integer data types are used for exact numeric data storage. Values for this column can be generated *randomly*, *incrementally*, or they can be taken from a fixed *list* of values or *SQL query*, or from an existing table column of the same data type.

The **Generation properties** panel allows you to define preferences for generating values for integer types.

Generation properties

i MANAGER_ID (integer)

Integer columns are used to hold exact numeric data. Values for this column can be generated randomly, incremental or taken from fixed list of values.

☐ Set NULL value in 10 % of cases

Select [Data generation mode](#) as follows:

☒ **Generate random data**

The value is generated randomly within the defined interval (the minimum and the maximum values).

Data generation mode

☒ Generate random data

☐ Generate incremental data

☐ Get data from list / SQL-query

☐ Get data from column

Parameters

Min value -2147483648

Max value 2147483647

☒ Use formula

x*2+1

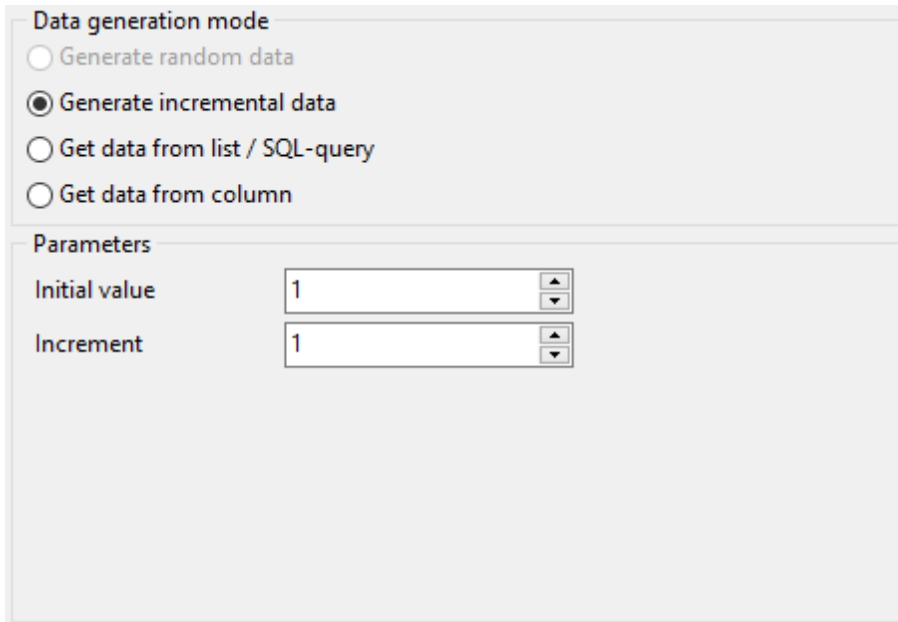
☒ **Use formula**

This option allows you to correct your data according to a formula; x is a randomly generated value here. Addition, subtraction, multiplication, dividing and exponentiation

operations (+, -, *, /, ^) can be used.


● **Generate incremental data**


Specify the **Initial value** and the **Increment** properties to generate an ordered incremented sequence of values.





● **Get Data from List / SQL query**

This panel allows you to define the list of values to generate integer data from. You can enter these values directly into the editor by selecting the **List of Values** option.

To add a single value, use the  **Add Value** button.

To load a list of values from an existing external file, use the  **Load from file** button.

To save the list to an external file, use the  **Save to file** button.

To remove a single value, use the  **Delete Value** button.

To remove all items from the list, use the  **Clear** button.






You can also specify whether the values are to be taken in **random order** or in the order they have been inputted.

Alternatively, you can set the **SQL Query** option and input an SQL query into the editor, and the resulting dataset will be used as the list for data generation.

Data generation mode
☐ Generate random data
☐ Generate incremental data
☒ Get data from list / SQL-query
☐ Get data from column

Parameters

☒ List of Values ☐ SQL query

☒ Random order     

1
2
▶ 3
4
5
6

● Get data from column

This option allows you to specify a column to generate data from: use the **Table** and **Column** drop-down lists to select the source table and column that will be used to take data for generation.

Data generation mode
☐ Generate random data
☐ Generate incremental data
☐ Get data from list / SQL-query
☒ Get data from column

Parameters

Table

characters

 ▼

Column

base_class

 ▼

2.1.4.1.2 FLOAT column parameters

Float data are used for approximate numeric data storage. Values for this column can be generated *randomly*, *incrementally*, or they can be taken from a fixed *list* of values or *SQL query*, or from an existing table column of the same data type.

The **Generation properties** panel allows you to define preferences for generating values for floating point numeric types.

Generation properties

i SALARY (double precision)

Float columns are used to hold approximate numeric data. Values for this column can be generated randomly, incremental or taken from fixed list of values.

☐ Set NULL value in % of cases

Select [Data generation mode](#) as follows:

☒ **Generate random data**

Here you can define the number of precision and the scale for the result randomly generated values.

Data generation mode

☒ Generate random data

☐ Generate incremental data

☐ Get data from list / SQL-query

☐ Get data from column

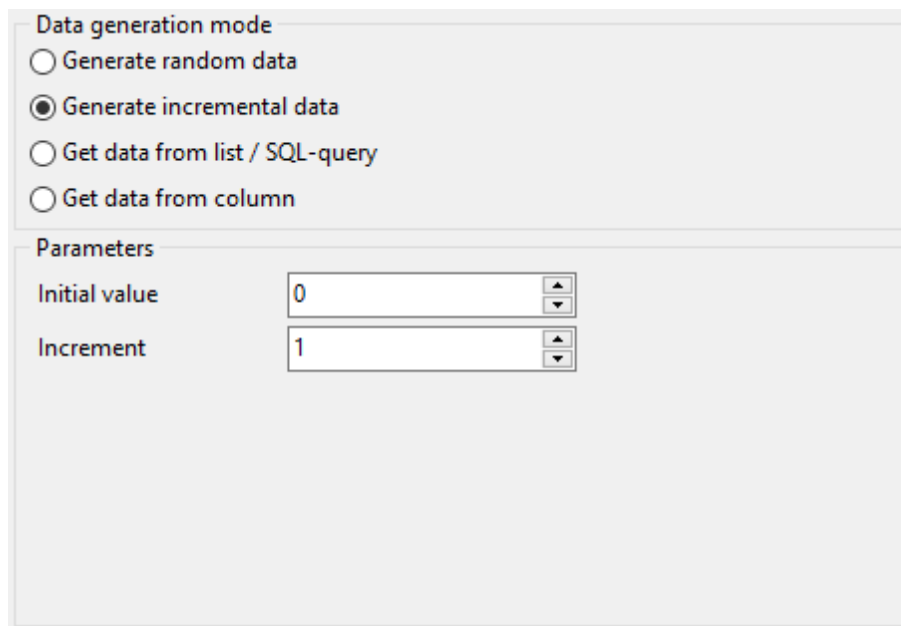
Parameters

Precision

Scale

☒ **Generate incremental data**


Specify the **Initial value** and the **Increment** properties to generate an ordered incremented sequence of values.




The screenshot shows a software interface for data generation. It has two main sections: 'Data generation mode' and 'Parameters'. In the 'Data generation mode' section, there are four radio buttons: 'Generate random data', 'Generate incremental data' (which is selected), 'Get data from list / SQL-query', and 'Get data from column'. In the 'Parameters' section, there are two input fields: 'Initial value' with the value '0' and 'Increment' with the value '1'. Both input fields have small up and down arrow buttons on their right sides.


Get data from List / SQL query


This panel allows you to define the list of values to generate floating point numeric data from. You can enter these values directly into the editor by selecting the **List of Values** option.

To add a single value, use the  **Add Value** button.

To load a list of values from an existing external file, use the  **Load from file** button.

To save the list to an external file, use the  **Save to file** button.

To remove a single value, use the  **Delete Value** button.

To remove all items from the list, use the  **Clear** button.

You can also specify whether the values are to be taken in **random order** or in the order they have been inputted.

Alternatively, you can set the **SQL Query** option and input an SQL query into the editor, and the resulting dataset will be used as the list for data generation.






Data generation mode

☐ Generate random data
☐ Generate incremental data
☒ Get data from list / SQL-query
☐ Get data from column

Parameters

☒ List of Values☐ SQL query

☒ Random order



-0,001
► -0,01
-0,1
0,001
0,01
0,1

● Get data from column

This option allows you to specify a column to generate data from: use the **Table** and **Column** drop-down lists to select the source table and column that will be used to take data for generation.

Data generation mode

☐ Generate random data
☐ Generate incremental data
☐ Get data from list / SQL-query
☒ Get data from column

Parameters

Tablepublishing▼

Columnsubbrand▼

2.1.4.1.3 DATE column parameters

Date are used for temporal values storage. Values for this type can be generated *randomly*, *incrementally*, or they can be taken from a fixed *list* of values or *SQL query*, or from an existing table column of the same data type.

The **Generation properties** panel allows you to define preferences for generating date values for date types.

Generation properties

i HireDate (date)

The date columns are used for holding temporal values. Values for this column can be generated randomly or taken from list of values.

☐ Set NULL value in % of cases

Select [Data generation mode](#) as follows:

☒ **Generate random data**

Set the date range by defining the minimum and the maximum values.

Data generation mode

☒ Generate random data

☐ Generate incremental data

☐ Get data from list / SQL-query

☐ Get data from column

Parameters

	Date
Min	<input type="text" value="8/1/2012"/>
Max	<input type="text" value="9/30/2012"/>


☒ **Generate incremental data**

Specify the **Start** and the **Increment** properties to generate an ordered incremented sequence of dates. The incremented value is day.


The screenshot shows a software interface for data generation. It has two main sections: 'Data generation mode' and 'Parameters'. In the 'Data generation mode' section, there are four radio buttons: 'Generate random data', 'Generate incremental data' (which is selected), 'Get data from list / SQL-query', and 'Get data from column'. The 'Parameters' section contains a 'Date' label, a 'Start' field with a dropdown menu showing '11.11.2011', and an 'Increment' field with a numeric input '1' and up/down arrow buttons.


● **Get data from list / SQL query**


This panel allows you to define the list of values to generate temporal data from. You can enter these values directly into the editor by selecting the **List of Values** option.

To add a single value, use the  **Add Value** button.

To load a list of values from an existing external file, use the  **Load from file** button.

To save the list to an external file, use the  **Save to file** button.

To remove a single value, use the  **Delete Value** button.

To remove all items from the list, use the  **Clear** button.

You can also specify whether the values are to be taken in **random order** or in the order they have been inputted.

Alternatively, you can set the **SQL Query** option and input an SQL query into the editor, and the resulting dataset will be used as the list for data generation.

Data generation mode

☐ Generate random data

☐ Generate incremental data

☒ Get data from list / SQL-query

☐ Get data from column

Parameters

☒ List of Values ☐ SQL query

☒ Random order

01.11.2011

02.11.2011

▶ 03.11.2011

04.11.2011

05.11.2011

06.11.2011

☒ **Get data from column**

This option allows you to specify a column to generate data from: use the **Table** and **Column** drop-down lists to select the source table and column that will be used to take data for generation.

Data generation mode

☐ Generate random data

☐ Generate incremental data

☐ Get data from list / SQL-query

☒ Get data from column

Parameters

Table

Column

2.1.4.1.4 TIME column parameters

Time data are used for temporal values storage. Values for this type can be generated *randomly*, *incrementally*, or they can be taken from a fixed *list* of values or *SQL query*, or from an existing table column of the same data type.

The **Generation properties** panel allows you to define preferences for generating time values for time types.

The screenshot shows a dialog box titled "Generation properties". Inside, there is an information icon and the text "LoginTime (time(0))". Below this, a descriptive text states: "The time columns are used for holding temporal values. Values for this column can be generated randomly or taken from list of values." At the bottom, there is a checkbox labeled "Set NULL value in" followed by a numeric input field containing "10" and a label "% of cases".

Select [Data generation mode](#) as follows:

☒ **Generate random data**

Set the time range by defining the minimum and the maximum values.

The screenshot shows a dialog box titled "Data generation mode". It contains four radio button options: "Generate random data" (which is selected), "Generate incremental data", "Get data from list / SQL-query", and "Get data from column". Below these options is a section titled "Parameters". Under "Parameters", there is a sub-section labeled "Time" which contains two rows: "Min" with a time input field set to "00:00:00", and "Max" with a time input field set to "23:59:59".


☒ **Generate incremental data**

Specify the **Start** value and the **Increment** properties to generate an ordered incremented sequence of values.


The screenshot shows a software interface for data generation. It has two main sections: 'Data generation mode' and 'Parameters'. In the 'Data generation mode' section, there are four radio buttons: 'Generate random data', 'Generate incremental data' (which is selected), 'Get data from list / SQL-query', and 'Get data from column'. The 'Parameters' section contains two time-related input fields. The first is labeled 'Start' and has a time value of '12:00:00' with up and down arrow buttons. The second is labeled 'Increment' and has a time value of '00:00:01' with up and down arrow buttons.


● **Get data from List / SQL query**


This panel allows you to define the list of values to generate temporal data from. You can enter these values directly into the editor by selecting the **List of Values** option.

To add a single value, use the  **Add Value** button.

To load a list of values from an existing external file, use the  **Load from file** button.

To save the list to an external file, use the  **Save to file** button.

To remove a single value, use the  **Delete Value** button.

To remove all items from the list, use the  **Clear** button.

You can also specify whether the values are to be taken in **random order** or in the order they have been inputted.

Alternatively, you can set the **SQL Query** option and input an SQL query into the editor, and the resulting dataset will be used as the list for data generation.

Data generation mode

☐ Generate random data
☐ Generate incremental data
☒ Get data from list / SQL-query
☐ Get data from column

Parameters

☒ List of Values ☐ SQL query

☒ Random order

01:00:00
02:00:00
▶ 03:00:00
04:00:00
05:00:00
06:00:00

☒ **Get data from column**

This option allows you to specify a column to generate data from: use the **Table** and **Column** drop-down lists to select the source table and column that will be used to take data for generation.

Data generation mode

☐ Generate random data
☐ Generate incremental data
☐ Get data from list / SQL-query
☒ Get data from column

Parameters

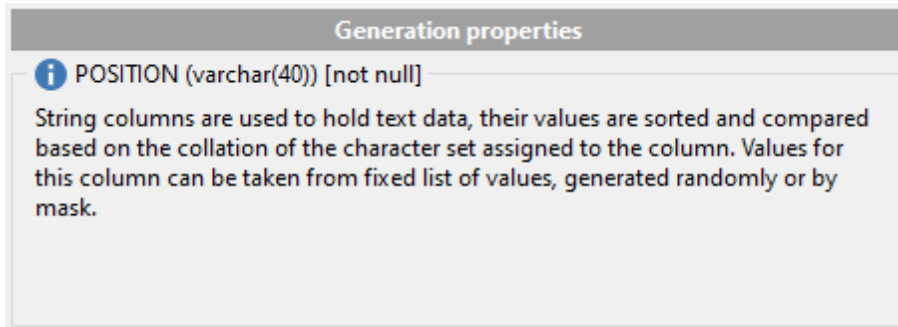
Table public.Employees ▼

Column HireDate ▼

2.1.4.1.5 STRING column parameters

Strings are used for text data storage. The values are sorted and compared on the basis of the collation of the character set assigned to the column. Values for this column can be generated *randomly* (with *constraints* or *mask* used), *incrementally*, or they can be taken from a fixed *list* of values or *SQL query*, or from an existing table column of the same data type.

The **Generation properties** panel allows you to define preferences for generating string values for string-based types.



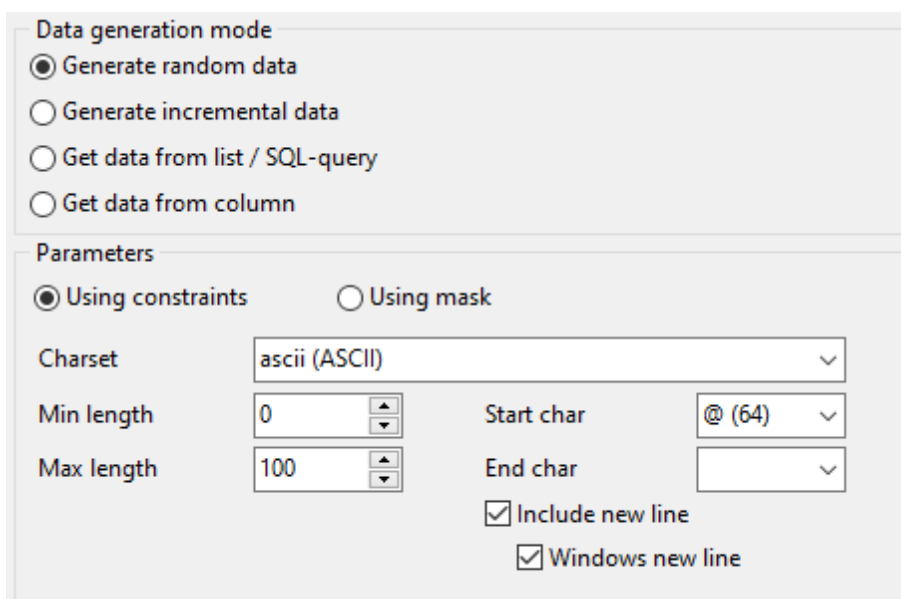
Select [Data generation mode](#) as follows:

☒ **Generate random data**

String random data can be generated in two ways - by using constraints or by typing the mask.

• **Using constraints**

Set the **Min length**, **Max length** values to define the minimum and the maximum length for generated values. You can also specify the **Start char** and the **End char** segments to be used for string values generation.

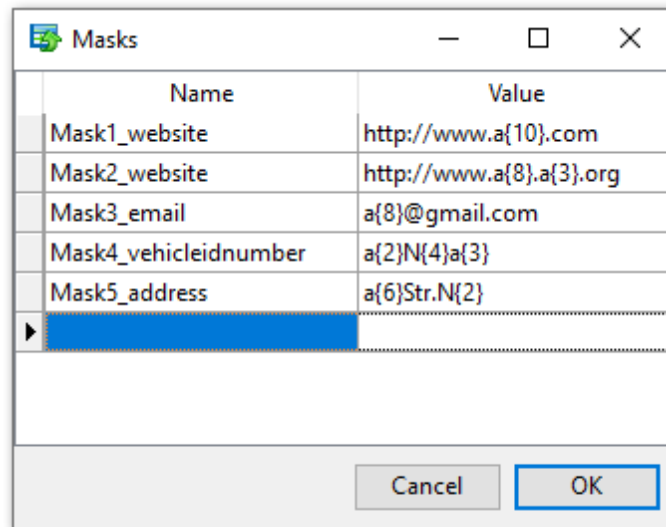


- **Using mask**

Check the option to generate values by mask. Use the **Masks** window to create and edit various masks for string data generation:

- the 'A' and the 'a' characters are replaced with a random letter (from 'A' to 'Z' and from 'a' to 'z');
- the 'N' character is replaced with a random digit;
- '{n}' results in iteration of the last sign n times;
- the character after the '\' symbol is interpreted as a common character.

All the rest of the mask characters will be moved to the result value without changes.



- **Generate incremental data**

Specify the **Initial value** and the **Increment** properties to generate an ordered incremented sequence of values.

Data generation mode

☐ Generate random data

☒ Generate incremental data

☐ Get data from list / SQL-query

☐ Get data from column

Parameters

Initial value

Increment


Start char

End char


Charset


☒ Get data from list / SQL query


This panel allows you to define the list of values to generate string data from. You can enter these values directly into the editor by selecting the **List of Values** option.

To add a single value, use the  **Add Value** button.

To load a list of values from an existing external file, use the  **Load from file** button.

To save the list to an external file, use the  **Save to file** button.

To remove a single value, use the  **Delete Value** button.

To remove all items from the list, use the  **Clear** button.

You can also specify whether the values are to be taken in **random order** or in the order they have been inputted.

Alternatively, you can set the **SQL Query** option and input an SQL query into the editor, and the resulting dataset will be used as the list for data generation.

Use of the **Sample text** option allows you to generate fragments of previously defined text into the string type column. To define sample text used by default, see the [Default constraints](#) section of the [Preferences](#) dialog.

Data generation mode

☐ Generate random data
☐ Generate incremental data
☒ Get data from list / SQL-query
☐ Get data from column

Parameters

☒ List of Values ☐ SQL query ☐ Sample text

☒ Random order

Alabama
Alaska
Arizona
Arkansas
California
Colorado

Get data from column

This option allows you to specify a column to generate data from: use the **Table** and **Column** drop-down lists to select the source table and column that will be used to take data for generation.

Data generation mode

☐ Generate random data
☐ Generate incremental data
☐ Get data from list / SQL-query
☒ Get data from column

Parameters

Table

Column

2.1.4.1.6 BLOB column parameters

A BLOB is a binary large object that can store a variable amount of data. You can generate values for this column *randomly* or choose to take them from a specified *list* of files or *SQL query*, or from an existing table column of the same data type.

The **Generation properties** panel allows you to define preferences for generating values for BLOB column type.

Select [Data generation mode](#) as follows:


☒ **Generate random data**


Set the **Min length**, **Max length** values to define the minimum and the maximum length for generated values.


☐ **Get data from files / SQL query**


This panel allows you to define the list of files to generate BLOB data from. You can specify the list of files to use their content as values for the BLOB column by selecting the **List of Files** option.

To add a file, use the  **Add Value** button.

To load a list of file paths from an existing external file, use the  **Load from file** button.

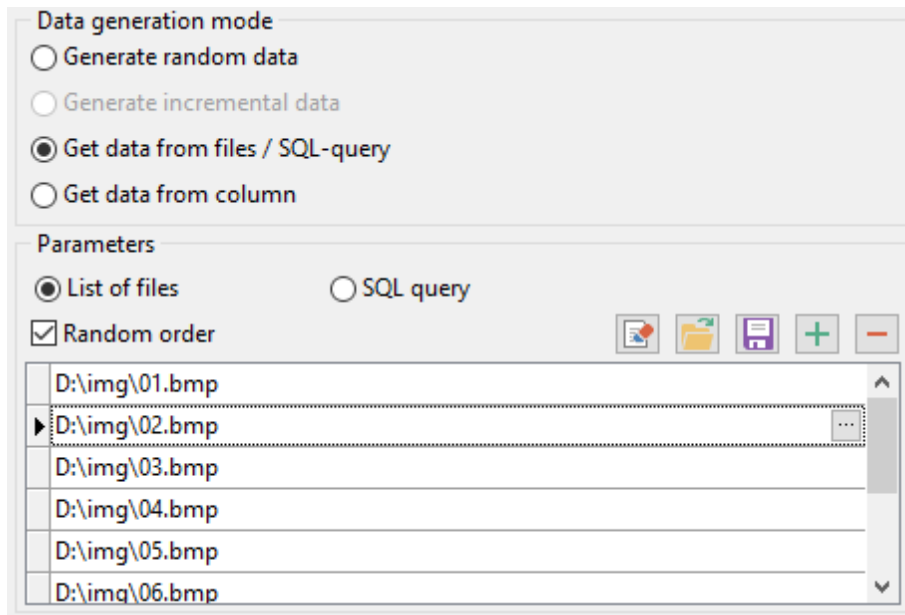
To save the list to an external file, use the  **Save to file** button.

To remove a single file, use the  **Delete Value** button.

To remove all items from the list, use the  **Clear** button.

You can also specify whether the files are to be taken in **random order** or in the order they have been inputted.

Alternatively, you can set the **SQL query** option and input an SQL query into the editor, and the resulting dataset will be used as the list for data generation.



The screenshot shows the 'Data generation mode' section with four radio buttons: 'Generate random data', 'Generate incremental data', 'Get data from files / SQL-query' (which is selected), and 'Get data from column'. Below this is the 'Parameters' section, which has two radio buttons: 'List of files' (selected) and 'SQL query'. There is a checkbox for 'Random order' which is checked. To the right of the checkbox are five icons: a red arrow pointing right, a folder, a save icon, a plus sign, and a minus sign. Below these is a list box containing six file paths: 'D:\img\01.bmp', 'D:\img\02.bmp' (which is highlighted with a mouse cursor), 'D:\img\03.bmp', 'D:\img\04.bmp', 'D:\img\05.bmp', and 'D:\img\06.bmp'. The list box has a scrollbar on the right.

☒ **Get data from column**

This option allows you to specify a column to generate data from: use the **Table** and **Column** drop-down lists to select the source table and column that will be used to take data for generation.

Data generation mode

☐ Generate random data

☐ Generate incremental data

☐ Get data from files / SQL-query

☒ Get data from column

Parameters

Table

Column

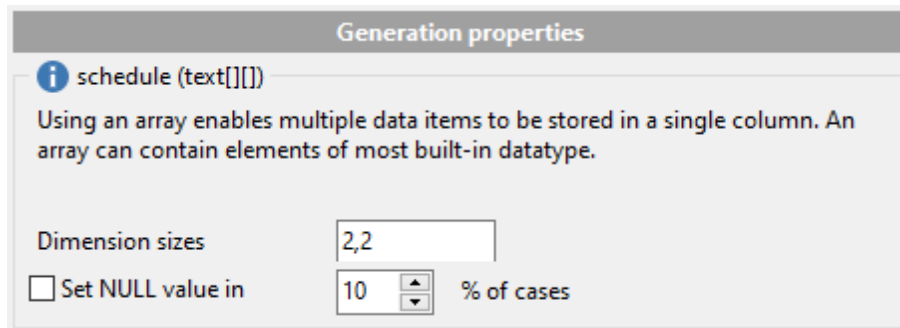
2.1.4.1.7 ARRAY column parameters

PostgreSQL allows columns of a table to be defined as variable-length multidimensional arrays. Values for this column can be generated *randomly*, *incrementally*, or they can be taken from a fixed *list* of values or *SQL query*.

The **Generation properties** panel allows you to define preferences for generating values for array column types.

Dimension sizes

Enter comma-separated list of array dimension sizes.



Generation properties

i schedule (text[][])

Using an array enables multiple data items to be stored in a single column. An array can contain elements of most built-in datatype.

Dimension sizes

☐ Set NULL value in % of cases

Select the preferable [Data generation mode](#) and specify generation parameters according to the data type used in the array.

2.1.4.1.8 GEOMETRIC column parameters

Geometric data types represent two-dimensional spatial objects. You can generate values for this columns *randomly* or choose to take them from a specified *list* of values or *SQL query*, or from an existing table column of the same data type.

The **Generation properties** panel allows you to define preferences for generating values for geometric column types.

The screenshot shows a window titled "Generation properties". Inside, there is a section for "Geom_circle (circle)". Below the title, a text box explains: "Geometric data types represent two-dimensional spatial objects. Values for this column can be generated randomly or taken from list." At the bottom, there is a checkbox labeled "Set NULL value in" followed by a numeric input field containing "10" and a label "% of cases".

Select [Data generation mode](#) as follows:


☒ **Generate random data**

The coordinates are generated randomly within the defined intervals (the minimum and the maximum values).


The screenshot shows two sections. The top section, "Data generation mode", contains four radio buttons: "Generate random data" (which is selected), "Generate incremental data", "Get data from list / SQL-query", and "Get data from column". The bottom section, "Parameters", contains six rows of input fields with spinners: "Min X" (-2147483648), "Max X" (2147483647), "Min Y" (-2147483648), "Max Y" (2147483647), "Min Radius" (0), and "Max Radius" (2147483647).


☒ **Get data from list / SQL query**


This panel allows you to define the list of values to generate geometric data from. You can enter these values directly into the editor by selecting the **List of Values** option.

To add a single value, use the  **Add Value** button.

To load a list of values from an existing external file, use the  **Load from file** button.

To save the list to an external file, use the  **Save to file** button.

To remove a single value, use the  **Delete Value** button.

To remove all items from the list, use the  **Clear** button.

You can also specify whether the values are to be taken in **random order** or in the order they have been inputted.

Alternatively, you can set the **SQL Query** option and input an SQL query into the editor, and the resulting dataset will be used as the list for data generation.






Data generation mode

☐ Generate random data
☐ Generate incremental data
☒ Get data from list / SQL-query
☐ Get data from column

Parameters

☒ List of Values ☐ SQL query

☒ Random order

<(100, 100), 10>
<(110, 100), 10>
<(100, 110), 10>
<(120, 100), 10>
<(100, 120), 10>
<(110, 120), 10>

☒ Get data from column

This option allows you to specify a column to generate data from: use the **Table** and **Column** drop-down lists to select the source table and column that will be used to take data for generation.

Data generation mode

☐ Generate random data

☐ Generate incremental data

☐ Get data from list / SQL-query

☒ Get data from column

Parameters

Table ▾

Column ▾

2.1.4.1.9 BOOLEAN column parameters

PostgreSQL Boolean columns can store either TRUE or FALSE values. You can generate values for this column *randomly* or choose to take them from a specified *list* of values or *SQL query*, or from an existing table column of the same data type.

The **Generation properties** panel allows you to define preferences for generating values for Boolean column type.

Generation properties

i IS_ACTIVE (boolean)

Boolean(bit) columns store either "1" or "0" values. This type of column is used for representing TRUE or FALSE, or YES or NO meanings. Values for this columns can be generated randomly or taken from list of values.

☐ Set NULL value in % of cases

Select [Data generation mode](#) as follows:

☒ **Generate random data**

Specify the ratio between True and False values in randomly generated data.

Data generation mode

☒ Generate random data

☐ Generate incremental data

☐ Get data from list / SQL-query

☐ Get data from column

Parameters


Count	False (25%)	True (75%)	Count
<input type="text" value="25"/>			<input type="text" value="75"/>


☒ **Get data from List / SQL query**


This panel allows you to define the list of values to generate boolean data from. You can enter these values directly into the editor by selecting the **List of Values** option.

To add a single value, use the **Add Value** button.

To load a list of values from an existing external file, use the  **Load from file** button.

To save the list to an external file, use the  **Save to file** button.

To remove a single value, use the  **Delete Value** button.

To remove all items from the list, use the  **Clear** button.

You can also specify whether the values are to be taken in **random order** or in the order they have been inputted.

Alternatively, you can set the **SQL Query** option and input an SQL query into the editor, and the resulting dataset will be used as the list for data generation.






Data generation mode

☐ Generate random data
☐ Generate incremental data
☒ Get data from list / SQL-query
☐ Get data from column

Parameters

☒ List of Values ☐ SQL query

☒ Random order

		<input checked="" type="checkbox"/>
		<input type="checkbox"/>
▶		<input checked="" type="checkbox"/>
		<input type="checkbox"/>

☒ **Get data from column**

This option allows you to specify a column to generate data from: use the **Table** and Column drop-down lists to select the source table and column that will be used to take data for generation.

Data generation mode

☐ Generate random data

☐ Generate incremental data

☐ Get data from list / SQL-query

☒ Get data from column

Parameters

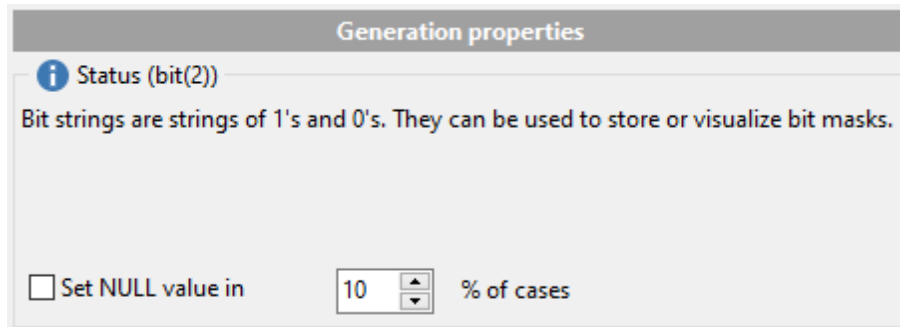
Table

Column

2.1.4.1.10 BIT column parameters

Bit strings are strings of 1's and 0's. They can be used to store or visualize bit masks. You can generate values for this column *randomly*, *incrementally* or choose to take them from a specified *list* of values or *SQL query*, or from an existing table column of the same data type.

The **Generation properties** panel allows you to define preferences for generating values for Bit column type.

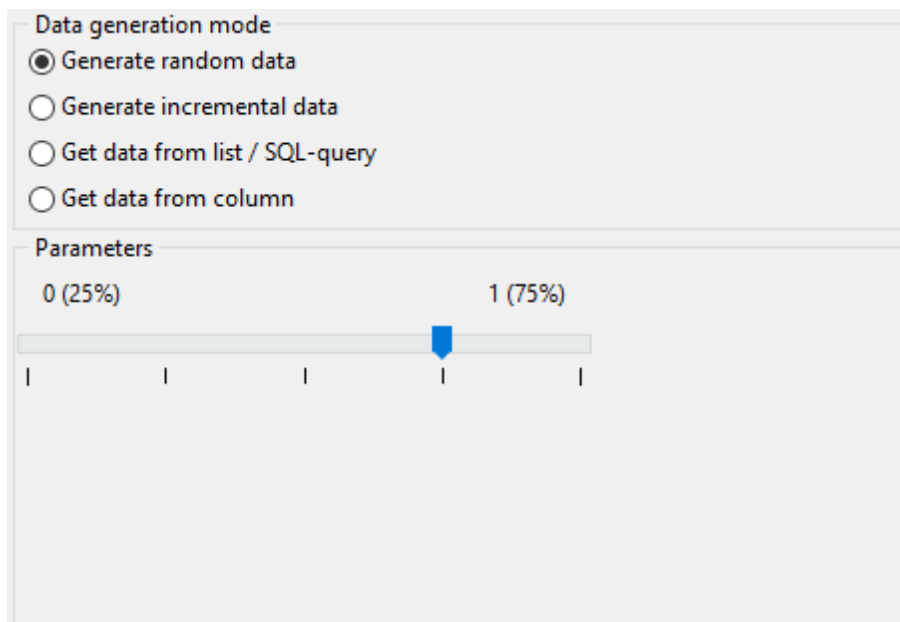


The screenshot shows a window titled "Generation properties". Inside, there is a section for "Status (bit(2))" with an information icon and a description: "Bit strings are strings of 1's and 0's. They can be used to store or visualize bit masks." Below this, there is a checkbox labeled "Set NULL value in" followed by a numeric input field containing "10" and a label "% of cases".

Select [Data generation mode](#) as follows:

☒ **Generate random data**

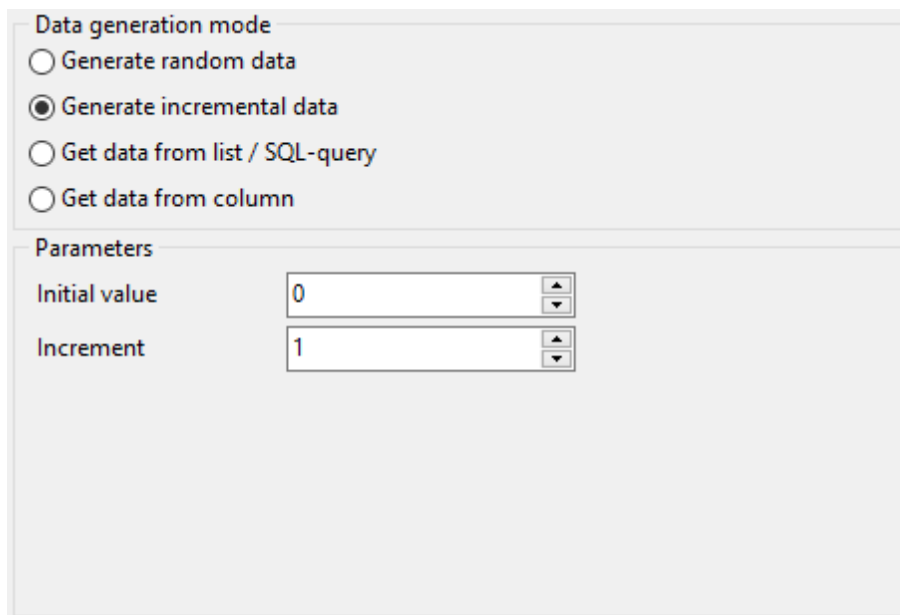
Specify the ratio between True and False values in randomly generated data.



The screenshot shows two sections. The first section, "Data generation mode", contains four radio buttons: "Generate random data" (which is selected), "Generate incremental data", "Get data from list / SQL-query", and "Get data from column". The second section, "Parameters", features a horizontal slider. The left end is labeled "0 (25%)" and the right end is labeled "1 (75%)". A blue arrow points to the slider at approximately the 75% mark.

☒ **Generate incremental data**


Specify the **Initial value** and the **Increment** properties to generate an ordered incremented sequence of values. The value is incremented bit-by-bit.




The screenshot shows a software interface for data generation. It has two main sections: 'Data generation mode' and 'Parameters'. In the 'Data generation mode' section, there are four radio buttons: 'Generate random data', 'Generate incremental data' (which is selected), 'Get data from list / SQL-query', and 'Get data from column'. In the 'Parameters' section, there are two input fields: 'Initial value' with the value '0' and 'Increment' with the value '1'. Both input fields have up and down arrow buttons next to them.


Get data from list / SQL query


This panel allows you to define the list of values to generate bit data from. You can enter these values directly into the editor by selecting the **List of Values** option.

To add a single value, use the  **Add Value** button.

To load a list of values from an existing external file, use the  **Load from file** button.

To save the list to an external file, use the  **Save to file** button.

To remove a single value, use the  **Delete Value** button.

To remove all items from the list, use the  **Clear** button.

You can also specify whether the values are to be taken in **random order** or in the order they have been inputted.






Alternatively, you can set the **SQL Query** option and input an SQL query into the editor, and the resulting dataset will be used as the list for data generation.

Data generation mode

☐ Generate random data
☐ Generate incremental data
☒ Get data from list / SQL-query
☐ Get data from column

Parameters

☒ List of Values ☐ SQL query
☒ Random order

11
▶ 10
00



● **Get data from column**

This option allows you to specify a column to generate data from: use the **Table** and **Column** drop-down lists to select the source table and column that will be used to take data for generation.

Data generation mode

☐ Generate random data
☐ Generate incremental data
☐ Get data from list / SQL-query
☒ Get data from column

Parameters

Table 
 Column 

2.1.4.1.11 INTERVAL column parameters

Interval type belongs to the date/time group of PostgreSQL data types and stores time intervals. This data type is useful for representing difference between two datetime values. You can generate values for this column *randomly* or choose to take them from an existing table column of the same data type.

The **Generation properties** panel allows you to define preferences for generating values for Interval column type.

Generation properties

i Term (interval(0))

INTERVAL type stores a period of time. This datatype is useful for representing the difference between two datetime values.
The values will be generated automatically depending on file format.

☐ Set NULL value in % of cases

Select [Data generation mode](#) as follows:

☒ **Generate random data**

Set the minimum and the maximum values in the appropriate spin editors. If **Min Day** value equals **Max Day** value then use the **Hours**, **Minutes** and **Seconds** to set the upper bound of a time interval (the lower bound defaults to '00:00:00').

Data generation mode

☒ Generate random data

☐ Generate incremental data

☐ Get data from list / SQL-query

☐ Get data from column

Parameters

Min Day

Max Day

Hours

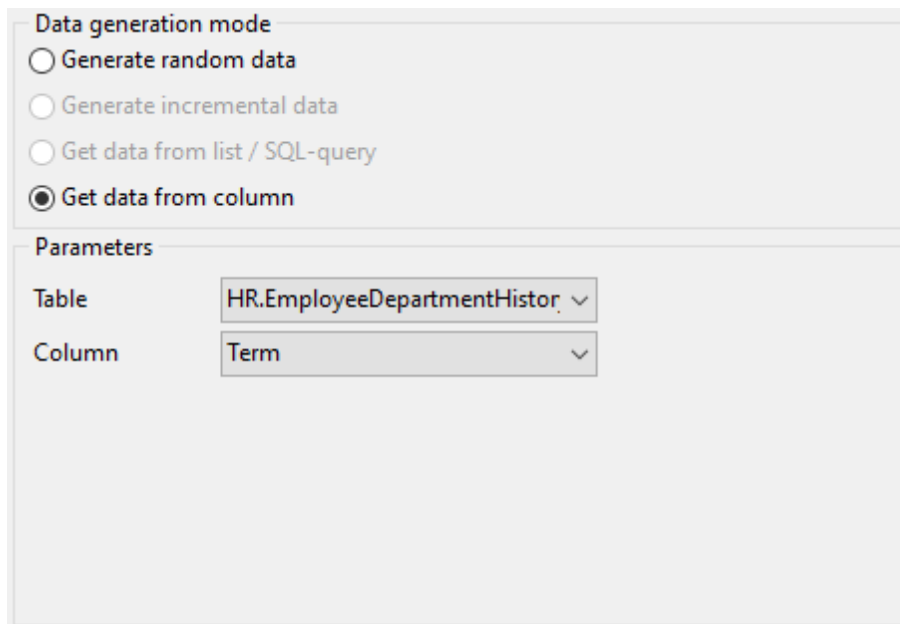
Minutes

Seconds

☒ **Get data from column**

This option allows you to specify a column to generate data from: use the **Table** and

Field drop-down lists to select the source table and field that will be used to take data for generation.



The screenshot displays a user interface for data generation. It is divided into two main sections: 'Data generation mode' and 'Parameters'.

Data generation mode

- ☐ Generate random data
- ☐ Generate incremental data
- ☐ Get data from list / SQL-query
- ☒ Get data from column

Parameters

Table: ▾

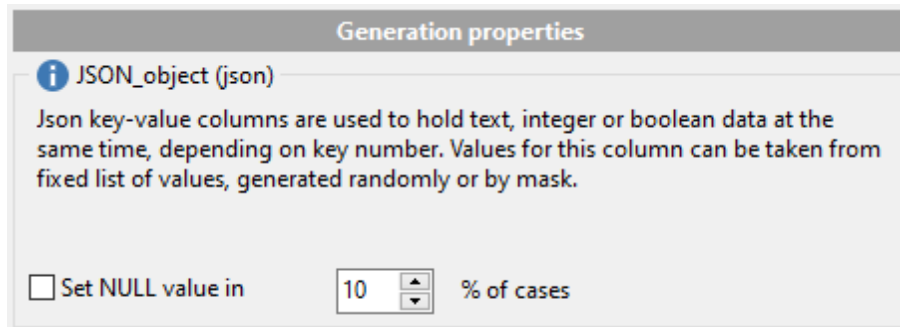
Column: ▾

2.1.4.1.12 JSON / JSONB column parameters

JSON type is used for representing structured data and it's mainly used for transmitting data in web applications. Values for this column can be generated *randomly* (with *constraints* or *mask* used) or they can be taken from a fixed *list* of values or *SQL query*, or from an existing table column of the same data type.

Generation properties

Adjust parameters for generating JSON data.



Generation properties

i JSON_object (json)

Json key-value columns are used to hold text, integer or boolean data at the same time, depending on key number. Values for this column can be taken from fixed list of values, generated randomly or by mask.

☐ Set NULL value in % of cases

Select [Data generation mode](#) as follows:

☒ **Generate random data**

Set the target number of each data type values in the **Integer values count**, **Boolean values count** and **String values count** options to be generated for JSON object.

Set minimum and maximum restrictions for numeric data in the **Min** and **Max** options.

Key value delimiter

Set the delimiter for key-value pair separator.

Text data for JSON can be generated randomly either using constraints or with mask templates.

- **Using constraints**

Set the **Min length**, **Max length** values to define the minimum and the maximum length for generated values. You can also specify the **Start char** and the **End char** segments to be used for string values generation.

Data generation mode

☒ Generate random data
☐ Generate incremental data
☐ Get data from list / SQL-query
☐ Get data from column

Parameters

Integer values count: 0 Min: -2147483648 Max: 2147483647
 Boolean values count: 0 Key-value pair delimiter: Win (CR LF)
 String values count: 1 Charset: ascii (ASCII)
☒ Using constraints ☐ Using mask
 Min length: 0 Start char: @ (64)
 Max length: 100 End char:

• Using mask

Check the option to generate values by mask. Use the **Masks** window to create and edit various masks for string data generation:

- the 'A' and the 'a' characters are replaced with a random letter (from 'A' to 'Z' and from 'a' to 'z');
- the 'N' character is replaced with a random digit;
- '{n}' results in iteration of the last sign n times;
- the character after the '\' symbol is interpreted as a common character.

All the rest of the mask characters will be moved to the result value without changes.


Masks

Name	Value
Mask1_website	http://www.a{10}.com
Mask2_website	http://www.a{8}.a{3}.org
Mask3_email	a{8}@gmail.com
Mask4_vehicleidnumber	a{2}N{4}a{3}
Mask5_address	a{6}Str.N{2}


Cancel OK


• Get data from list / SQL query


This panel allows you to define the list of values to generate string data from. You can enter these values directly into the editor by selecting the **List of Values** option.

To add a single value, use the  **Add Value** button.

To load a list of values from an existing external file, use the  **Load from file** button.

To save the list to an external file, use the  **Save to file** button.

To remove a single value, use the  **Delete Value** button.

To remove all items from the list, use the  **Clear** button.

You can also specify whether the values are to be taken in **random order** or in the order they have been inputted.

Alternatively, you can set the **SQL Query** option and input an SQL query into the editor, and the resulting dataset will be used as the list for data generation.






Data generation mode

☐ Generate random data
☐ Generate incremental data
☒ Get data from list / SQL-query
☐ Get data from column

Parameters

☒ List of Values ☐ SQL query

☒ Random order

{ "firstName": "Joe", "lastName": "Jackson", "gender": "male", "age": 28 }	^
{ "firstName": "Alex", "lastName": "Hunter", "gender": "male", "age": 23 }	
{ "firstName": "Rebecca", "lastName": "Smith", "gender": "female", "age": 22 }	
{ "firstName": "Adam", "lastName": "Freeman", "gender": "male", "age": 30 }	
▶ { "firstName": "Leon", "lastName": "Kennedy", "gender": "male", "age": 25 }	
{ "firstName": "Amanda", "lastName": "Mason", "gender": "female", "age": 18 }	▼

☒ **Get data from column**

This option allows you to specify a column to generate data from: use the **Table** and **Column** drop-down lists to select the source table and column that will be used to take data for generation.

Data generation mode

☐ Generate random data

☐ Generate incremental data

☐ Get data from list / SQL-query

☒ Get data from column

Parameters

Table

Column

2.1.4.2 Viewing table DDL

When you select a database in the **Generate data for** tree, you can select a table belonging to the database and view its DDL structure within the **Table Definition** area at the right side of the window.

Hint: If more convenient, you can see the DDL of the tables in the hint that popup when the mouse cursor is positioned over the table names within the **Generate Data for** area.

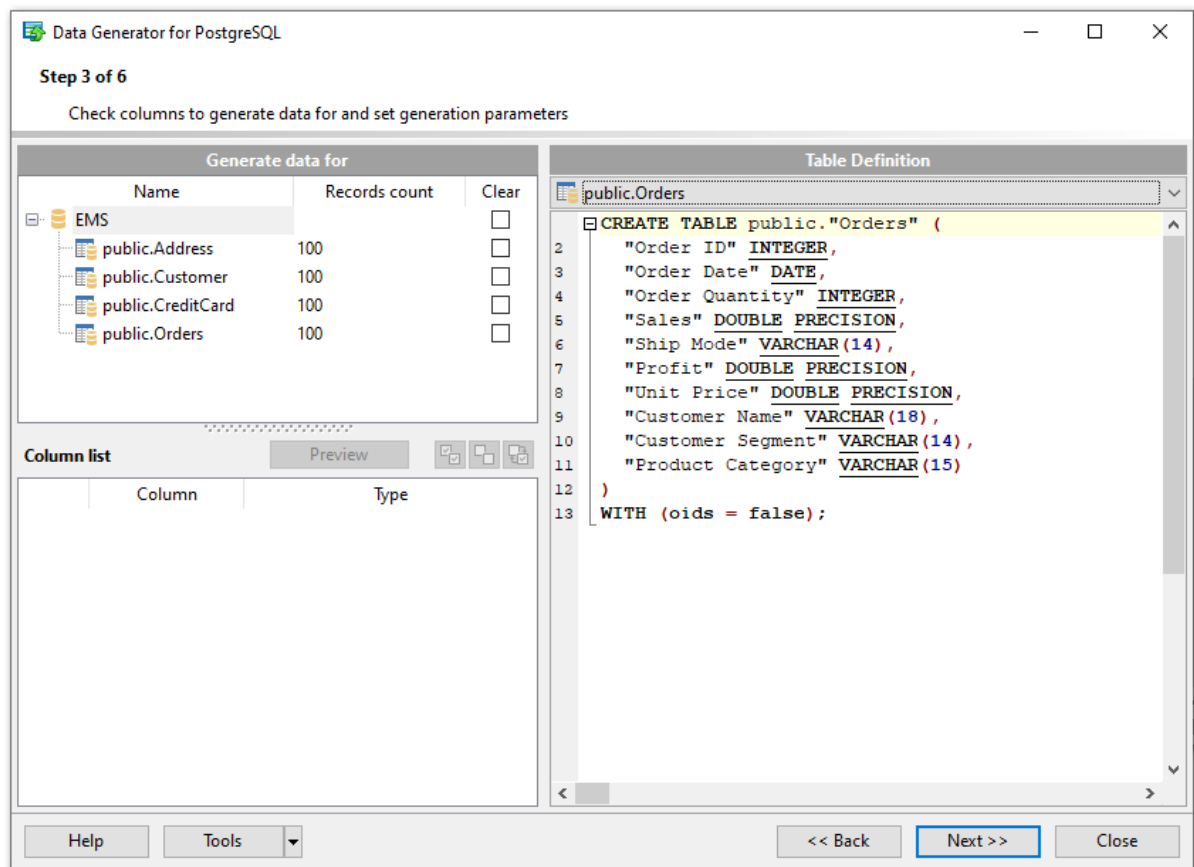


Table Definition

The drop-down list at the top contains the tables that were selected for data generation at [Step 2](#). Select a table to view its DDL.

Right-click within the **Table Definition** area to call the **popup menu** allowing you to *copy* the DDL of the table to Windows clipboard.

2.1.4.3 Data Preview

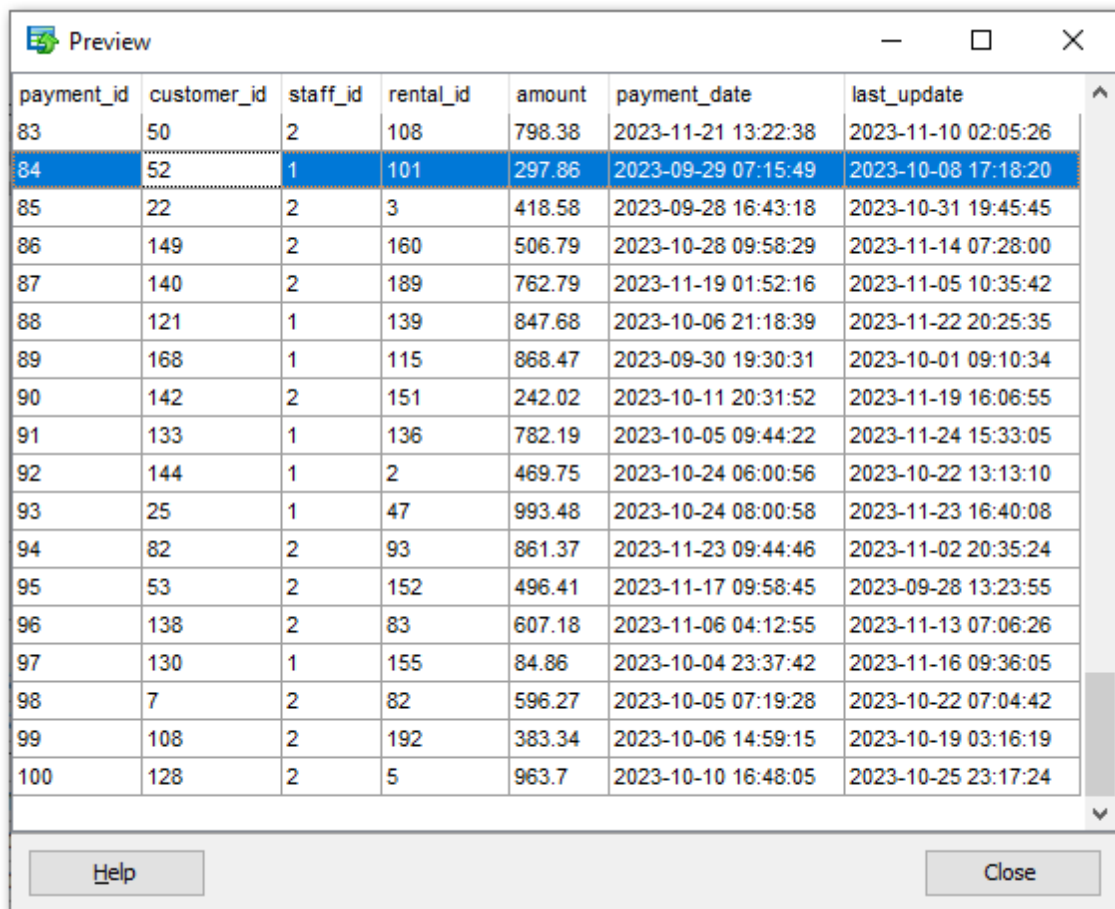
The **Preview** window allows you to browse the selected table data in the preview mode.

Note: The data in preview are selected randomly according to specified parameters and are not actually inserted into the table, i.e. a different set of values will be generated into the table.

To open the window, click the **Preview** button available at [Step 3](#).

The grid contains all selected columns with data that will be generated according to the parameters you have specified at [Step 3](#). If more convenient, you can **change the order** of the columns by dragging their headers horizontally.

Click a column caption to **sort** items by values of this column in the ascending or the descending mode.

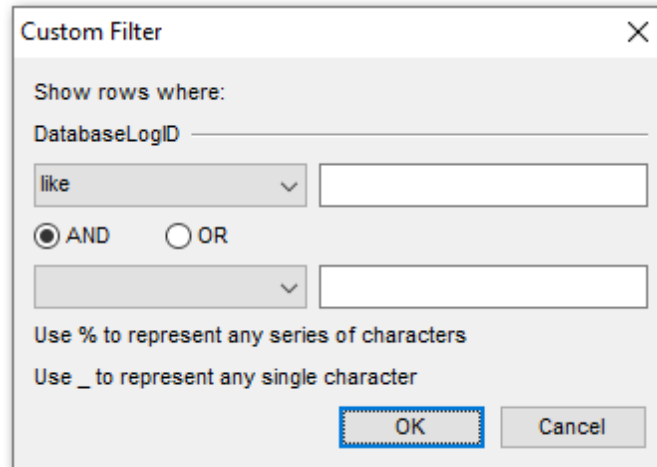


payment_id	customer_id	staff_id	rental_id	amount	payment_date	last_update
83	50	2	108	798.38	2023-11-21 13:22:38	2023-11-10 02:05:26
84	52	1	101	297.86	2023-09-29 07:15:49	2023-10-08 17:18:20
85	22	2	3	418.58	2023-09-28 16:43:18	2023-10-31 19:45:45
86	149	2	160	506.79	2023-10-28 09:58:29	2023-11-14 07:28:00
87	140	2	189	762.79	2023-11-19 01:52:16	2023-11-05 10:35:42
88	121	1	139	847.68	2023-10-06 21:18:39	2023-11-22 20:25:35
89	168	1	115	868.47	2023-09-30 19:30:31	2023-10-01 09:10:34
90	142	2	151	242.02	2023-10-11 20:31:52	2023-11-19 16:06:55
91	133	1	136	782.19	2023-10-05 09:44:22	2023-11-24 15:33:05
92	144	1	2	469.75	2023-10-24 06:00:56	2023-10-22 13:13:10
93	25	1	47	993.48	2023-10-24 08:00:58	2023-11-23 16:40:08
94	82	2	93	861.37	2023-11-23 09:44:46	2023-11-02 20:35:24
95	53	2	152	496.41	2023-11-17 09:58:45	2023-09-28 13:23:55
96	138	2	83	607.18	2023-11-06 04:12:55	2023-11-13 07:06:26
97	130	1	155	84.86	2023-10-04 23:37:42	2023-11-16 09:36:05
98	7	2	82	596.27	2023-10-05 07:19:28	2023-10-22 07:04:42
99	108	2	192	383.34	2023-10-06 14:59:15	2023-10-19 03:16:19
100	128	2	5	963.7	2023-10-10 16:48:05	2023-10-25 23:17:24

If necessary, you can filter records in the grid in either of the following ways:

- click the Arrow-Down button next to the column caption to display the drop-down list and select any of the column values to filter records by this value of the selected column;
- click the Arrow-Down button next to the column caption to display the drop-down list, then select the **Custom** item and build a simple filter using the **Custom Filter** dialog.

Select a logical operator for checking the column values (*like*, *is less than*, *is greater than*, etc.) and set a value to be checked by this operator in the corresponding box on the right.

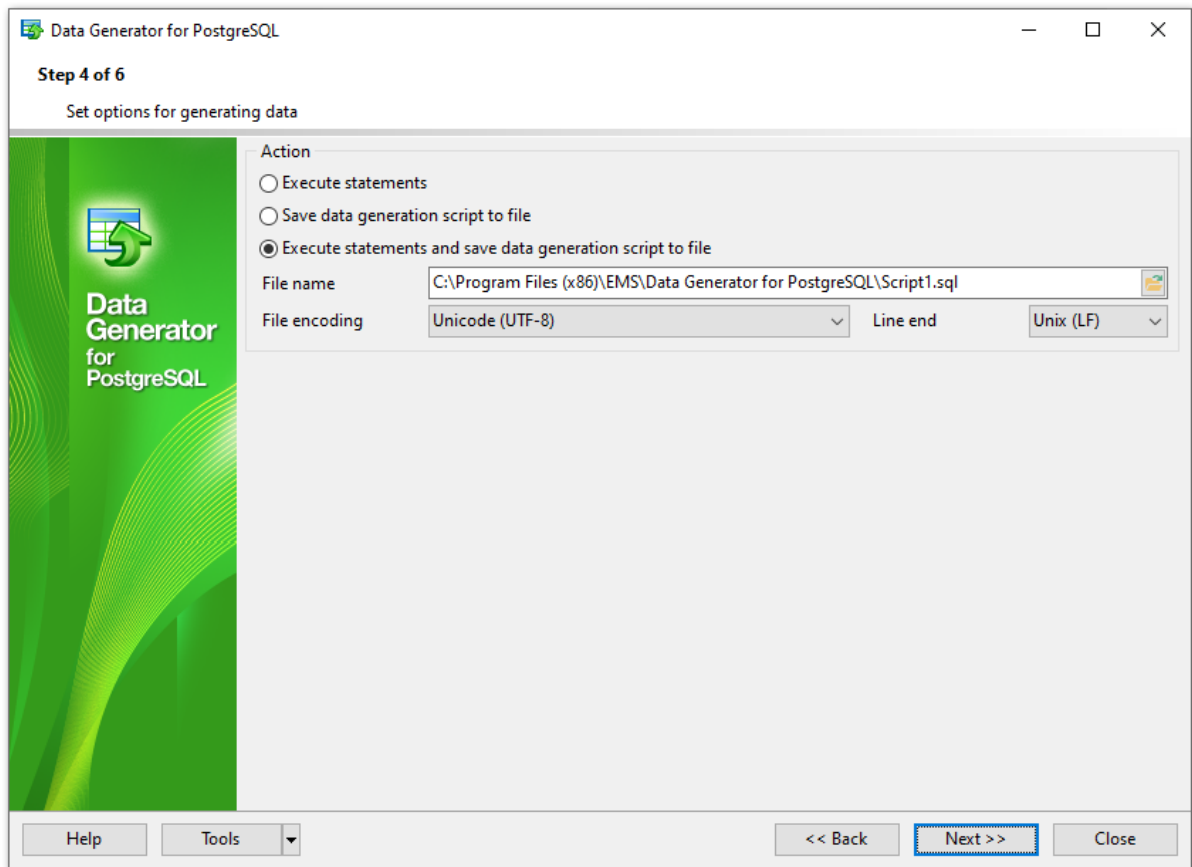


The image shows a 'Custom Filter' dialog box with a close button (X) in the top right corner. The main text is 'Show rows where:'. Below this is a text field containing 'DatabaseLogID'. Underneath is a dropdown menu currently showing 'like'. To the right of the dropdown is an empty text input box. Below the dropdown are two radio buttons: 'AND' (which is selected) and 'OR'. Below the radio buttons is another dropdown menu, currently empty, followed by another empty text input box. At the bottom, there are two lines of instructional text: 'Use % to represent any series of characters' and 'Use _ to represent any single character'. At the very bottom are two buttons: 'OK' and 'Cancel'. The 'OK' button is highlighted with a blue dashed border.

If necessary, you can set the second condition and specify the relation between the two conditions: whether both of them should be satisfied (*AND*) or just any of them (*OR*). Use the '_' character to represent any single symbol, and use the '%' character to represent any series of symbols in the condition string.

2.1.5 Step 4 - Setting generation options

At this step you can specify data generation options.



Action

Specify the action to be taken at the [next step](#) of the wizard:

☒ **Execute statements**

Select this option to execute the script for data generation.


☒ **Save data generation script to file**

Select this option if you only need to save the script for data generation to a file, without the script execution.

☒ **Execute statements and save data generation script to file**

Select this option to execute the script for data generation and save it to a file.

File name

This box is enabled if the *Save data generation script to file* or *Execute statements and save data generation script to file* option is selected. Type in or use the  **Explorer** button to specify the path to the *.sql file to store the SQL script.

File encoding

Select target encoding for the file.

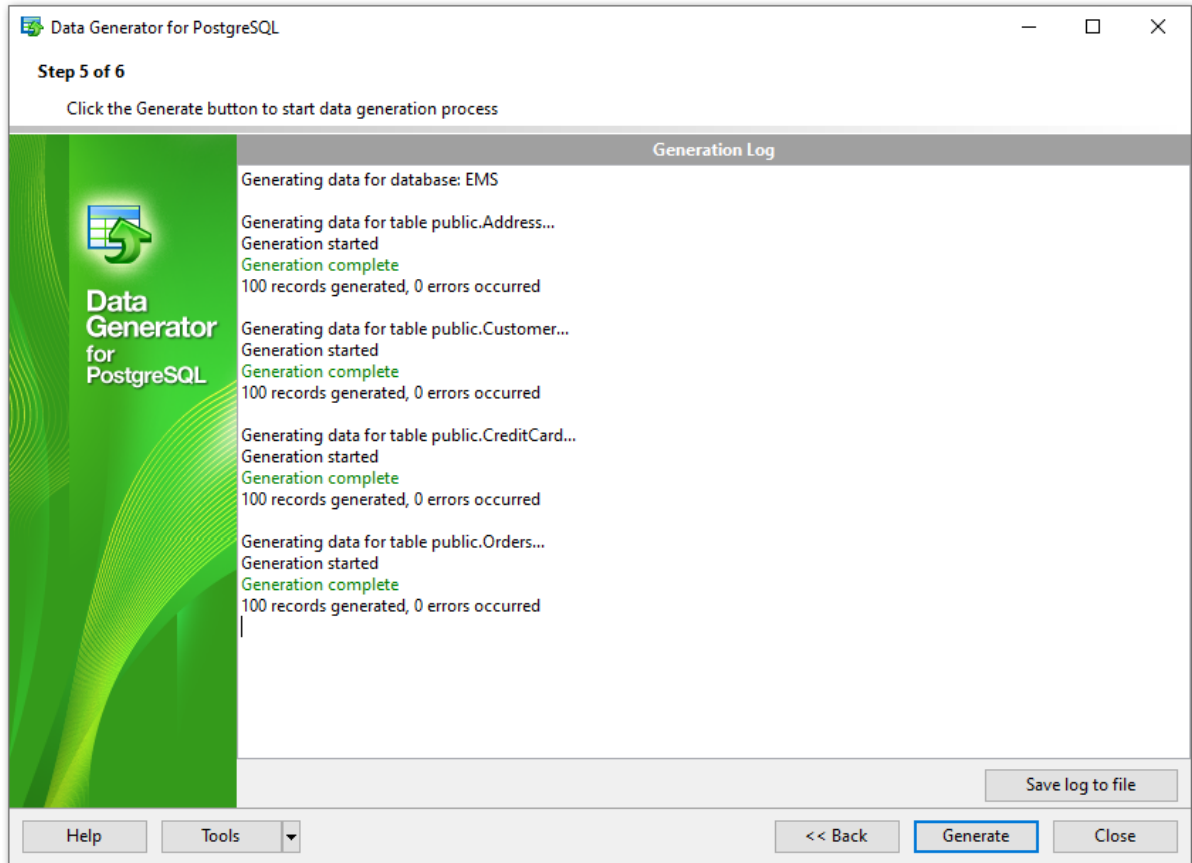
Line end

Select the required line end symbol for the file: CR LF for Windows OS, LF for Unix or CR for MacOS.

When you are done, press the **Next** button to proceed to the [next step](#) of the Wizard.

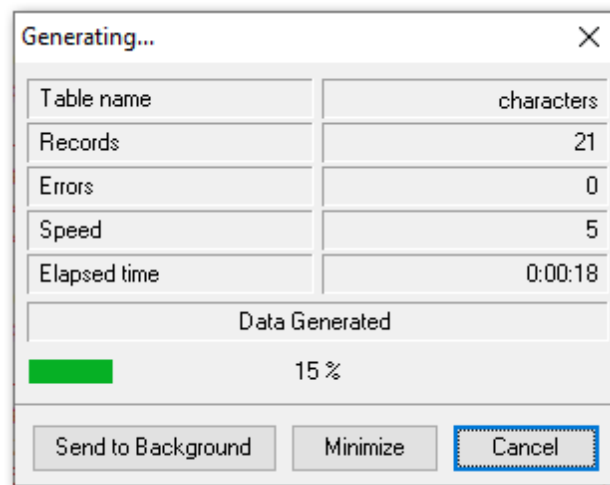
2.1.6 Step 5 - Start of data generation process

This step informs you that all data generation parameters are set, and you can start the generation process.



If everything is correct, press the **Generate** button to start the process. If you want to change something, you can return to any of the wizard steps using the **Back** button.

The **Generating...** dialog indicates the amount of generated *records*, elapsed *time*, the number of *errors* (if any) and visually represents the percentage of *data generated*.



Use the **Send to background** button to run the process in the background mode, the **Minimize** button to minimize the application to Windows Task bar, or the **Cancel** button to stop the generation process.

During data generation the **Generation Log** area displays the log of performed operations and errors (if any).

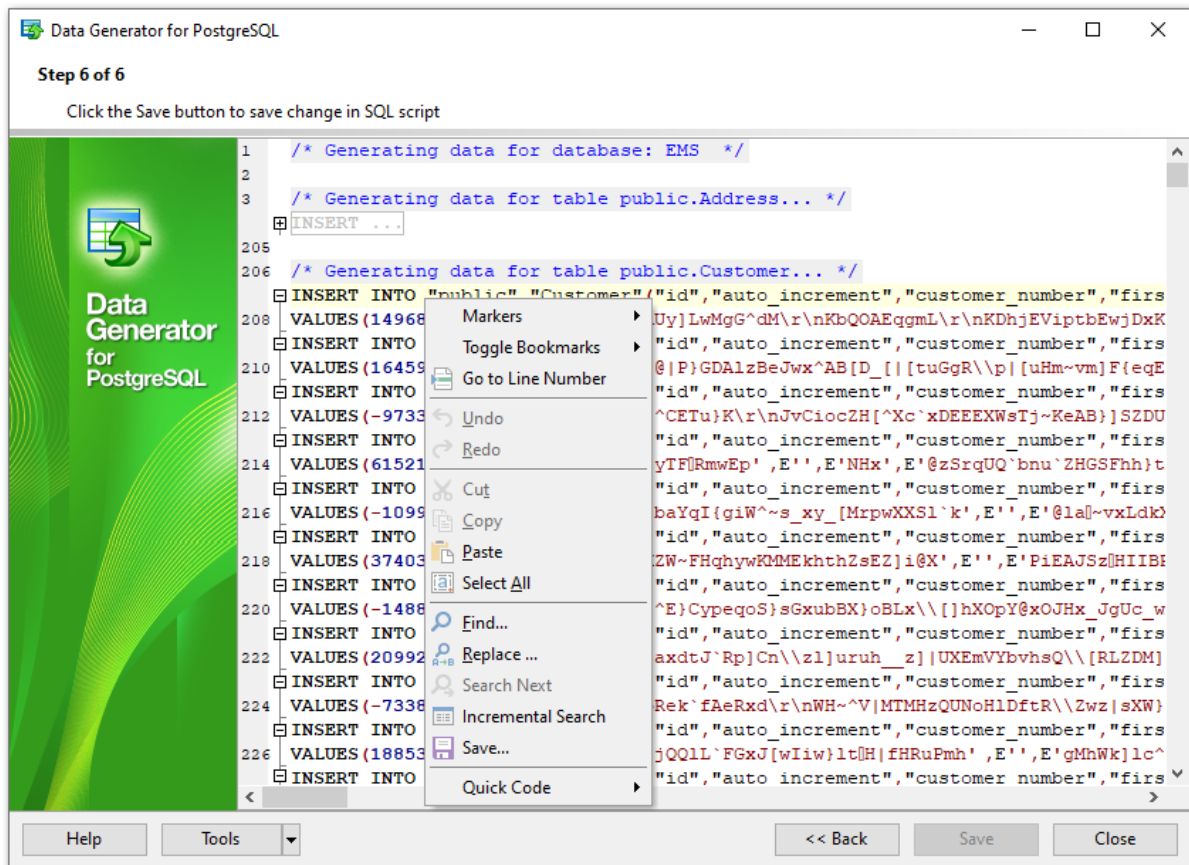
When the process is completed, you can use the **View SQL file in editor** and the **Save log to file** buttons to edit the data generation script at [Step 6](#) of the wizard and save the generation log content to an external text file respectively.

Do not forget to [save data generation options](#) if you need to repeat the generation process with the same (or similar) settings later.

2.1.7 Step 6 - Editing generation script

This step of the wizard allows you to edit the result script for data generation.

For your convenience the **code folding**, **syntax highlight** and a number of other features for efficient SQL editing are implemented.



The **context menu** of the editor area contains most of the standard text-processing functions (*Cut*, *Copy*, *Paste*, *Select All*) and functions for working with the script as a whole, e.g. you can *move the cursor to a particular line*, *set markers*, *toggle bookmarks*, etc.

Implementation of the [Find Text](#) and the [Replace Text](#) dialogs contributes to more efficient work with the SQL code.

Find the complete list of the context menu items below. The context menu allows you to:

- manage markers: *Drop Marker*, *Collect Marker*, *Swap Marker*;
- toggle bookmarks allowing you to navigate through the query text and jump to a line with a particular number;
- go to a line with specified number;
- perform editing operations: *Undo/Redo*, *Cut*, *Copy*, *Paste*, *Select all*;
- perform [search](#) and [replace](#) operations;
- save the script to an external *.sql file;
- format the selected code using *SQL Formatter* to make the code easier to read;

- use the Quick Code features: select a character, toggle a comment for a code fragment, toggle case of the selected text, indent/unindent lines in the script.

Press the **Back** button to return to any of the previous steps (the content of the editor area will not be lost).

If you press the **Save** button, the script will be saved to an external file.

Pressing the **Close** button will result in closing the application (before closing Data Generator will prompt for saving changes).

2.2 Using configuration files

Data Generator for PostgreSQL allows you to store its settings in external `*.gtm` files if you need to repeat data generation process several times.

You can load previously saved configuration settings to the application wizard if you need to make some changes before data generation, or you can run it with the [console application](#) for quicker generation.

- [Saving configuration file](#)
- [Loading configuration file](#)

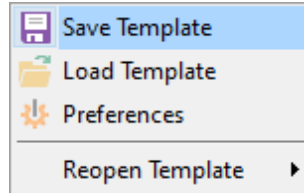
See also:

[Using wizard application](#)

[Setting program preferences](#)

2.2.1 Saving configuration file

Data Generator templates are saved within the **Save template options** dialog. To open this dialog, press the **Tools** button and select the **Save Template** popup menu item.




Please note that you can save data generation options at the steps from [Specifying generation parameters](#) to the [last step](#) of Data Generator for PostgreSQL wizard.

- [Save Template options](#)
- [Loading configuration file](#)

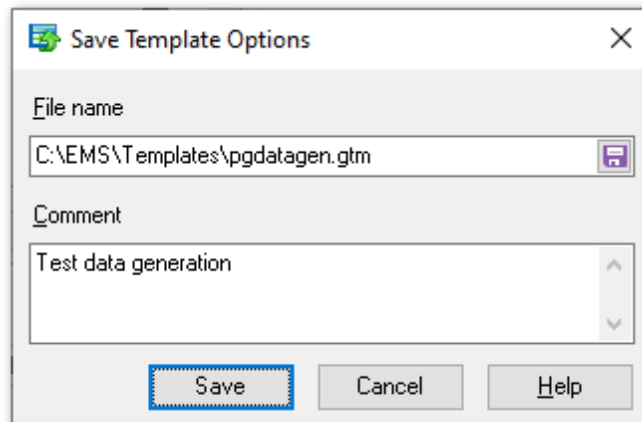
2.2.1.1 Save Template options

File name

Specify the template file name and select its location using the  button to open the **Save As...** dialog.

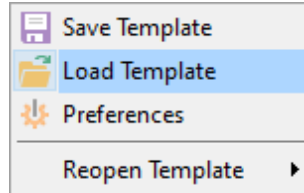
Comment

If necessary, set a comment for your template file in this field.



2.2.2 Loading configuration file

Data Generator templates are loaded within the **Open template** dialog. To open this dialog, press the **Tools** button and select the **Load template** popup menu item.



Please note that you can **reopen a template** at the steps from [starting](#) to [Specifying generation parameters](#) of the wizard using the corresponding popup menu item of the **Tools** menu.

- [Saving configuration file](#)
- [Save Template options](#)

2.3 Setting program preferences

Data Generator for PostgreSQL provides full customization of the program interface by setting various options within the **Preferences** dialog. This chapter is intended to inform you how to use these options.

General

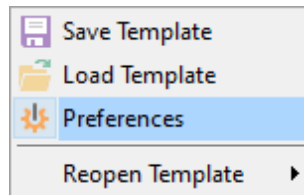
These options define general behavior of Data Generator for PostgreSQL

Defaults

This page allows you to set the constraints for data values used by default in the generation process.

Language

This page allows you to select a language to be applied for your copy of Data Generator for PostgreSQL.



See also:

[Using wizard application](#)

[Using configuration files](#)

2.3.1 Setting general options

This page allows you to define general options of the application.

Theme

Select the main color theme for the application: Light or Dark.

Number of records to generate

Sets the quantity of records generated by default.

Commit every ... records

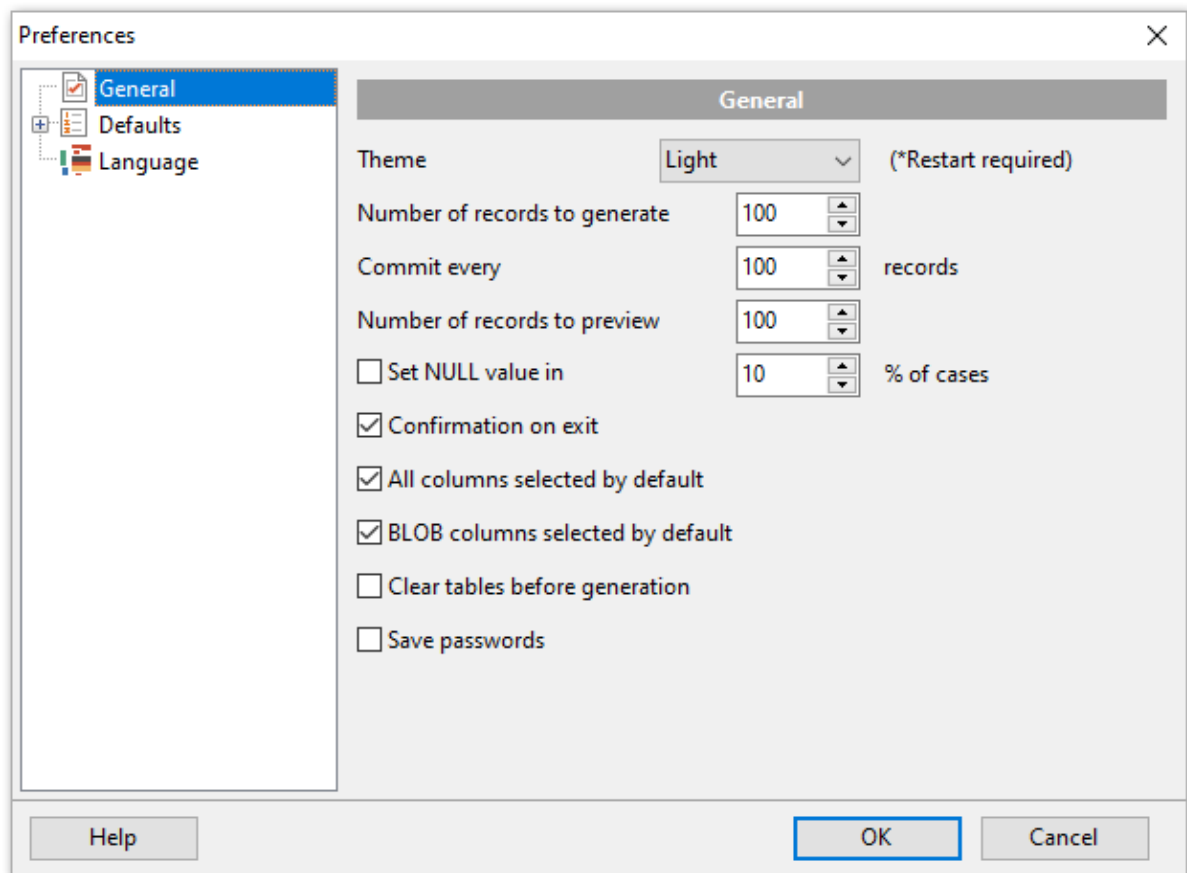
Specifies the number of records in each block of the generation script to be supplemented with the COMMIT statement.

Number of records to preview

Sets the quantity of records used in the [Data Preview](#) window by default.

☒ Set NULL value in ... % of cases

This option allows you to specify the percentage of records that will remain *NULL* by default.



☒ Confirmation on exit

Enables/disables confirmation upon exiting the program.

☒ **All columns selected by default**

Check this option to include all columns into the data generation process by default.

☒ **BLOB columns selected by default**

Uncheck this option if you need to exclude BLOB columns from the generation process by default.

☒ **Clear tables before generation**

Set this option to empty tables before data generation.

☒ **Save passwords**

Setting this option allows you to save passwords used for access to the databases automatically upon closing the application. Please note that checking this option saves the latest password used for connection to the database (including the SSH server password).

See also:

[Setting default constraints](#)

[Setting program language](#)

2.3.2 Setting default constraints

In this section you can define default constraints for all supported data types:

[Numeric types](#)

[Character types](#)

[Date/Time types](#)

[JSON types](#)

Default values are used as initially set values for data range and can be modified at [Step 3](#).

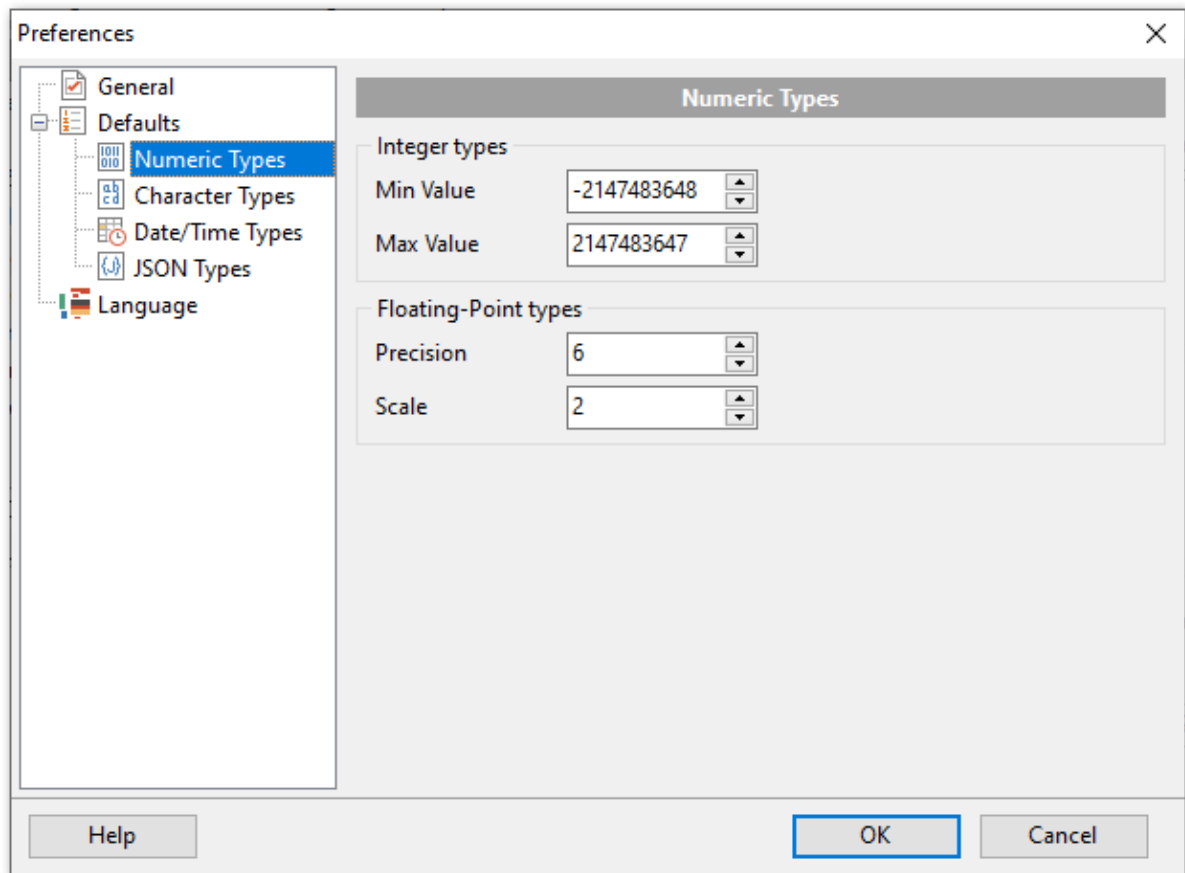
See also:

[Setting general options](#)

[Setting program language](#)

2.3.2.1 Numeric types

On this tab you can define default constraints for numeric types.



Integers types

Set the target minimum and maximum values set by default for integer types.

Floating-point types

Set the precision and scale values set by default for floating-point types.

2.3.2.2 Character types

On this tab you can define default constraints for character types.

Min Length

Set the minimum length of the generated value.

Max Length

Set the maximum length of the generated value.

Start Char

Select the start char for the range to be used for generation.

End Char

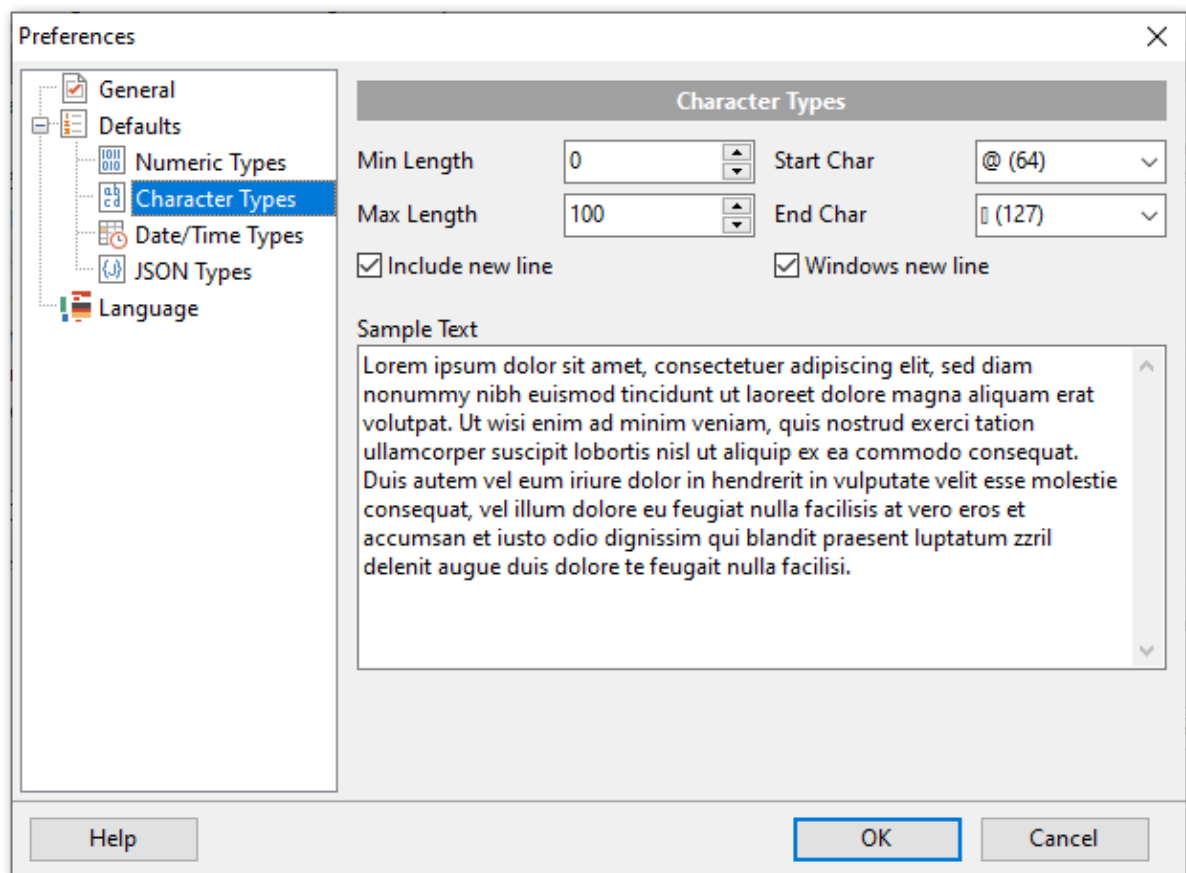
Select the end char for the range to be used for generation.

☒ Include new line

Select this option to allow line feeds in generated string values.

☒ Windows new line

Select this option to specify Windows-style line feeds.



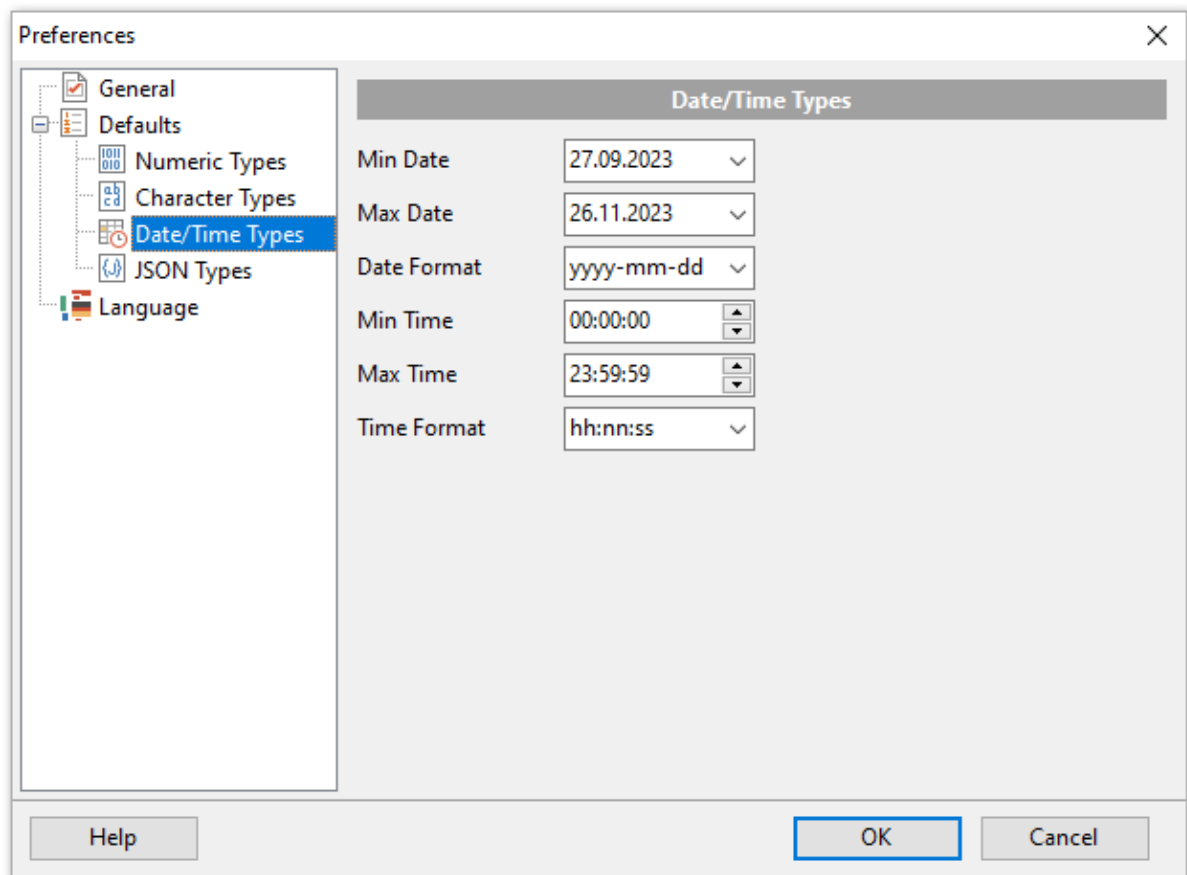
Sample Text

If more convenient, you can input any string that will be used as sample text. You can

choose to generate sample text when setting generation parameters for [strings](#).

2.3.2.3 Date/Time types

On this tab you can define default constraints for date and time types.

**Min Date**

Select the start date for generating date values.

Max Date

Select the end date for generating date values.

Date format

Type in or use the drop-down list to specify the preferable date format.

Min Time

Select the start range time for generating time values.

Max Time

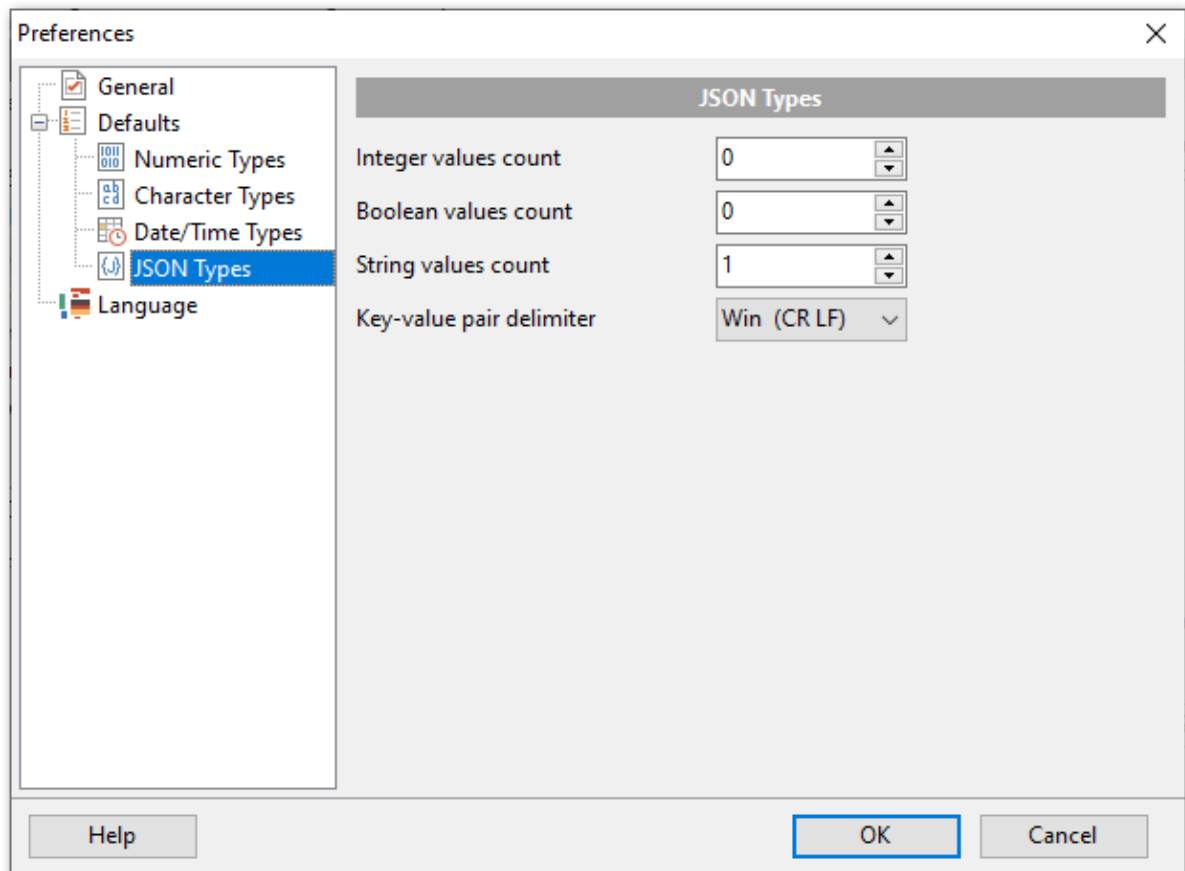
Select the end range time for generating time values.

Time format

Type in or use the drop-down list to specify the preferable time format.

2.3.2.4 JSON types

On this tab you can define default constraints for JSON data type.

**Integer values count**

Set the target number of integer values for generation.

Boolean values count

Set the target number of boolean type values for generation.

String values count

Set the target number of string values for generation.

Key-value pair delimiter

Select the default delimiter for key-value pair separator.

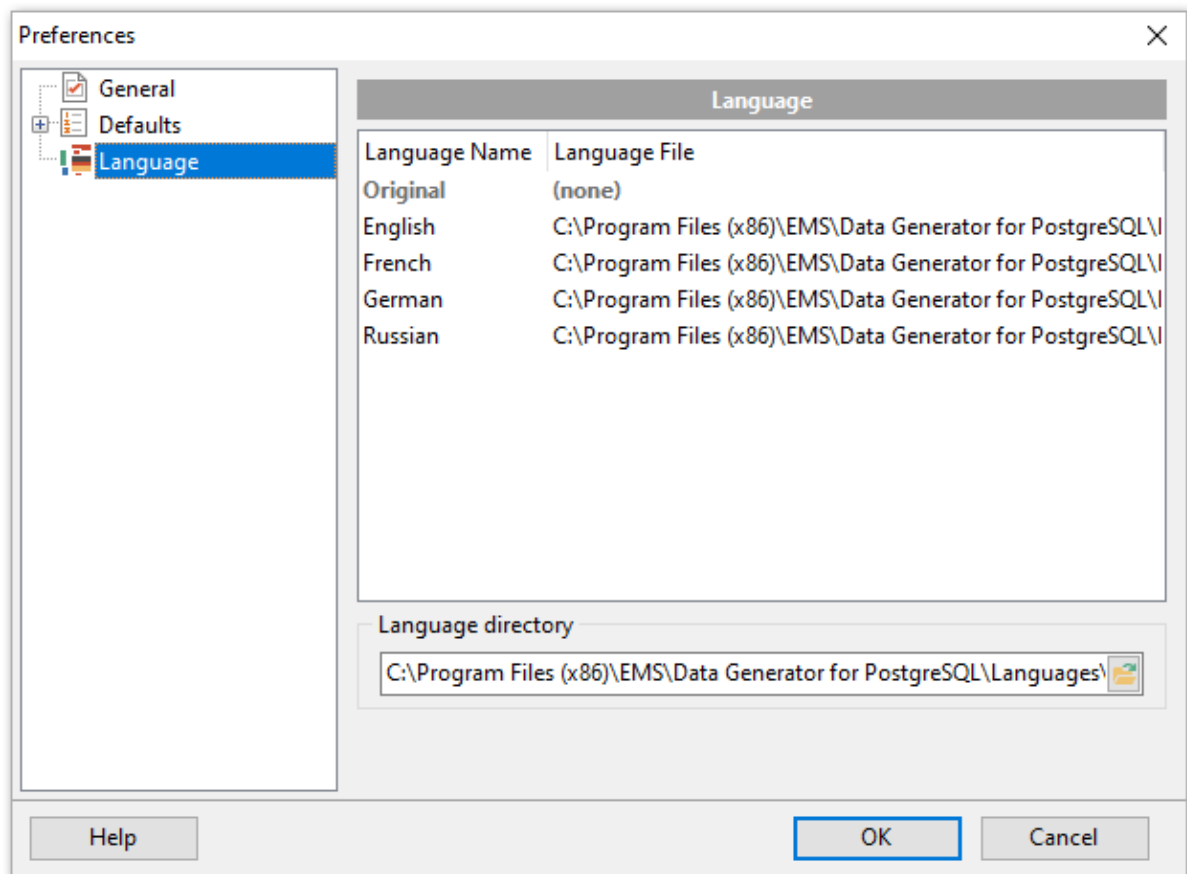
2.3.3 Setting program language

The **Language** page is provided for managing Data Generator localization files.

You can create your own *.lng files similar to those available in the %program_directory%\Languages folder, add them to the list of available languages and set the new language as the program interface language.

Available languages

Lists all the languages available for localization and the corresponding *.lng files. Double-click a language in the list to edit its name or the *.lng file.



Language directory

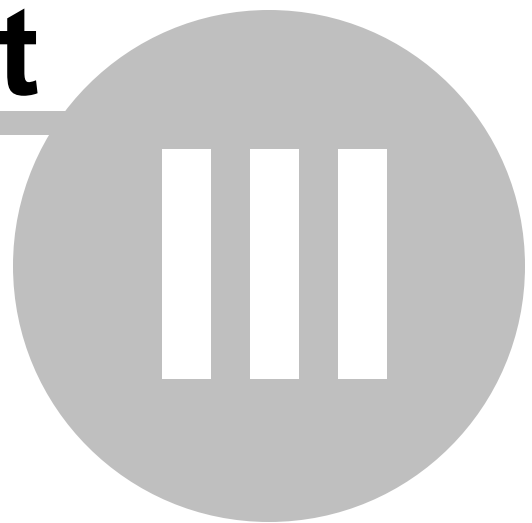
Use the  button to specify the directory where the *.lng files are stored by default.

See also:

[Setting general options](#)

[Setting default constraints](#)

Part



3 Console Application

Additionally to **the GUI version** which is implemented in the form of a wizard application, the installation package of Data Generator for PostgreSQL includes **the console version** which is intended for being run from Windows command line with a template file name used as the execution parameter.

```
C:\Program Files\EMS\Data Generator for PostgreSQL>PgDataGenC.exe_
```

Data Generator for PostgreSQL command line utility is intended for quick and powerful data generation to PostgreSQL tables.

- [Using console application](#)

See also:

[Wizard Application](#)

3.1 Using console application

All the generation options are set in **template** (*.gtm) files. A template can be also used in the **Console version** of Data Generator for PostgreSQL .

To create a template file, follow the instructions below:

- start Data Generator for PostgreSQL [Wizard Application](#);
- set all the required options in all steps of the wizard;
- test the generation process at the last step;
- [save all generation options in the template file](#).

The easiest way to start Data Generator for PostgreSQL console application is to double-click the generated *.gtm template file. The other way is to enter the command line and type the appropriate command.

Usage:

```
<path to Data Generator for PostgreSQL console application>\PgDataGenC.exe  
TemplateFile [-L] [-B]
```

TemplateFile

Stands for the *.gtm template file to be used as the console version execution parameter

[-L]

Applies the current [localization](#) selected in [Wizard Application](#) (GUI)

[-B]

Use this parameter in the command line to run the console version of **Data Generator for PostgreSQL** in background mode

Example:

```
"C:\Program Files\EMS\Data Generator for PostgreSQL\PgDataGenC.exe" "C:  
\EMS\Templates\DataGenerator\Example.gtm" -L
```

Note: The following exit codes can be returned by Data Generator for PostgreSQL to the operating system after performing the latest task:

- 0 - successful completion;
- 1 - error(s) occurred during task performing;
- 2 - fatal error occurred. The task was not performed.

See also:

[Using wizard application](#)

[Configuration file format](#)

Part

IV

4 Appendix

4.1 SSH tunneling options

To setup the connection via **SSH tunnel**, input the following values in the corresponding fields:

- **SSH host** is the name of the host where SSH server is running
- **SSH port** indicates the port where SSH server is activated
- **SSH login** stands for the user on the machine where SSH server is running (**Note:** it is a Linux/Windows user, not a user of PostgreSQL server)
- **SSH password** is the Linux/Windows user password

Please note that PostgreSQL **host name** should be set relatively to the SSH server in this case. For example, if both PostgreSQL and SSH servers are located on the same computer, you should specify *localhost* as **host name** instead of the server external host name or IP address.

☒ **Use Private Key for authentication**

If the SSH encryption is enabled on the SSH server, a user can generate a pair of cryptographic keys (the **Private key** and the **Public key**). The **Public key** is placed on the SSH server, and the **Private key** is the part you keep secret inside a secure box that can only be opened with the correct passphrase (or an empty string as the passphrase). When you wish to access the remote system, you open the secure box with your passphrase (if any), and use the private key to authenticate yourself with the Public key on the remote Linux computer.

SSH Key file

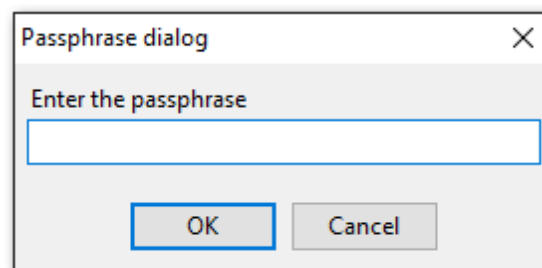
Specify the location (the secure box) of the **Private key** file on your local machine. Supported Private Key file formats are:

OpenSSH

Putty

SSH.com

Note that you need to trust your local machine not to scrape your passphrase or a copy of your Private key file while it is out of its secure box.



4.2 HTTP tunneling options

To use **HTTP tunneling**, just upload the tunneling script to the webserver where PostgreSQL server is located, or to any other webserver from which direct connections to your PostgreSQL server are allowed. This script exposes the PostgreSQL API as a set of web-services used by Data Generator for PostgreSQL.

In case of using this connection method the response will be slower as compared to the direct connection or the SSH Tunneling method, since the data are XML encoded and HTTP is stateless by nature. However, all the features of Data Generator for PostgreSQL are available.

Note that the *emspoxy.php* script file is included into the distribution package and can be found in Data Generator installation directory.

4.3 Data generation mode

This option defines data generation mode - *random data generation*, *incremented values generation* or getting data from *list or SQL query*, or getting data from an existing table column.

Select the **Generate random data** option to generate random data within the defined range.

Another mode - **Generate incremental data** - allows one to set the initial value and the increment for generated values.

Select the **Get Data from List / SQL query** option to generate data by getting values from the user-defined list randomly or in the fixed order. This can be:

- a *list of values* (for numeric, string, date/time, boolean data types);
- a *list of files* (for BLOB data type);
- an *SQL query*;
- previously defined *sample text* (for string data types).

Select the **Get data from column** option to specify a column to generate data from: use the *Table* and *Column* drop-down lists to select the source table and column that will be used to take data for generation.

4.4 Configuration file format

The **configuration (template) file** used by Data Generator for PostgreSQL is divided into several sections, each corresponding to a particular group of settings specified at different steps of the [GUI application](#) wizard.

[#General#]

This section stores general information about the utility:

<i>Parameter</i>	<i>Description</i>
Product	internal product name
Version	major version

[#Comment#]

This section stores the template file comment as specified optionally in the [Save template options](#) dialog:

<i>Parameter</i>	<i>Description</i>
Line<N>	comment text

where *N* stands for the comment line identifier

Example:

Line0=Data Generator for PostgreSQL

Line1=Template file

Line2=Data generation #1

[CONNECTION]

This section stores connection parameters to PostgreSQL. The parameters correspond to the values entered at [Step 1](#) of the [Wizard application](#) and are obligatory.

<i>Parameter</i>	<i>Description</i>
Host	host where the database resides (if Remote = 1)
Port	port on which PostgreSQL is listening
Remote	0 = local connection 1 = remote connection
Login	PostgreSQL login
Password	password to identify the login (encrypted)
Charset	client character set specified for the connection
FontCharset	font character set used by the application
Major	major version used to encrypt the passwords (the value must not be changed)
Minor	minor version used to encrypt the passwords (the value must not be changed)
SSLMode	SSL mode: 0 - Disabled 1 - Allow 2 - Prefer 3 - Require 4 - Verify CA 5 - Verify Full

SSLCert	path to SSL certificate file
SSLKey	path to SSL private key file
SSLCA	path to Certificate Authority (CA)
SSLCertRL	path to Certificate Revocation List
Passphrase	SSL passphrase

[TUNNELING]

The section contains parameters required for connection with tunneling used; the values correspond to the settings specified at [Step 1](#) of the [Wizard application](#) for connection via [SSH Tunnel](#) or [HTTP Tunnel](#) (if used).

<i>Parameter</i>	<i>Description</i>
TunnelType	indicates the tunneling type being used: SSH, HTTP, or none (TunnelType = <i>ttNotUse</i>)
SSHHostName	name of the host where SSH server is running
SSHPort	port on which SSH server is activated
SSHUserName	user on the machine where SSH server is running
SSHPassword	password to identify SSH server user (encrypted)
SSHKeyFile	path to the Private key used for the SSH connection (if SSHUseKeyFile = <i>True</i>)
SSHUseKeyFile	<i>True</i> = SSH Private key is used <i>False</i> = SSH Private key is not used
PassPhrase	passphrase for Private key (if SSHUseKeyFile = <i>True</i>)
HTTPUrl	URL to the <i>emspoxy.php</i> script file uploaded to your web-server (for HTTP tunneling)

[ADDITIONAL]

The section contains additional settings specified at [Step 2](#) and [Step 4](#) of the [Wizard application](#).

<i>Parameter</i>	<i>Description</i>
TablesCount	number of tables selected for data generation
SqlExecute	corresponds to the <i>Execute statements</i> option of the Action radio group available at Step 4 : <i>1</i> = enabled <i>0</i> = disabled
SqlSave	corresponds to the <i>Save data generation script to file</i> option of the Action radio group available at Step 4 : <i>1</i> = enabled <i>0</i> = disabled
SaveFile	path to the script file (if SqlSave = <i>1</i>)
BlobFile	the parameter is not used by Data Generator for PostgreSQL
ExportBlobType	the parameter is not used by Data Generator for PostgreSQL

[TABLE_XX]

The section is repeated for all tables; the settings are specified at [Step 3](#) of the [Wizard application](#).

<i>Parameter</i>	<i>Description</i>
Database	indicates the name of the database where the table is located
TableName	indicates the name of the table to generate data into

RecordCount number of records to be generated
ClearBeforeGeneration 1 leads to emptying table
 0 leaves the table as it was before data generation

[TABLE_XX_FIELD_YY]

The section is created for each field of each table.

<i>Parameter</i>	<i>Description</i>
DoGenerate	0 indicates that the field is excluded 1 indicates that the field is included
IncludeNulls	1 specifies that the NULL values are set for certain percent of cases 0 disables this option
NumNulls	the percentage of field values to be set to NULL
GenMethod	defines Data generation mode : 0 stands for incremental data generation 1 = random data generation 2 refers to Get data from list / SQL query option 3 = from another field
GenFromSQL	0 = the direct list of values is taken for data generation 1 = SQL query is used
SQL	text of the SQL query from which a list of values is taken for generation (as the result of the SQL query execution)
UsingMask	0 = no mask is used for string field values 1 = generation of string field values by mask
Mask	mask for string field values generation
MinInt	minimal value for integer fields
MaxInt	maximal value for integer fields
UseFormula	0 = no formula 1 = a formula is applied for data generation
Formula	formula for data generation, e.g. $x*2+1$
Digits	digits quantity for float fields
Precision	precision value for float fields
MinDate	minimum value for date fields
MaxDate	maximum value for date fields
IncludeTime	indicates whether time is added (for <i>DATETIME</i> fields)
MinTime	minimum value for time fields
MaxTime	maximum value for time fields
MinLength	minimum length for string fields
MaxLength	maximum length for string fields
StartChar	first char code for generating strings
EndChar	last char code for generating strings
Charset	field character set
InitialValue	the initial value for data generation into the field
IncrementStep	specifies the step to increment values (for GenMethod = 0)
UseNewLine	1 = a line feed is used for a new line 0 = no line feeds used
WinNewLineStyle	style to be applied to line feeds: 1 = Windows style 0 = Unix style
SampleText	sample text to be generated for a string field

4.5 Find Text dialog

The **Find Text** dialog is provided for quick and flexible searching for specified text within the working area of the script editor.

Text to find

Enter a search string in this box. The Arrow-Down button which can be found next to the input box allows you to select any of the previously entered search strings.

Options

☒ **Case sensitive**

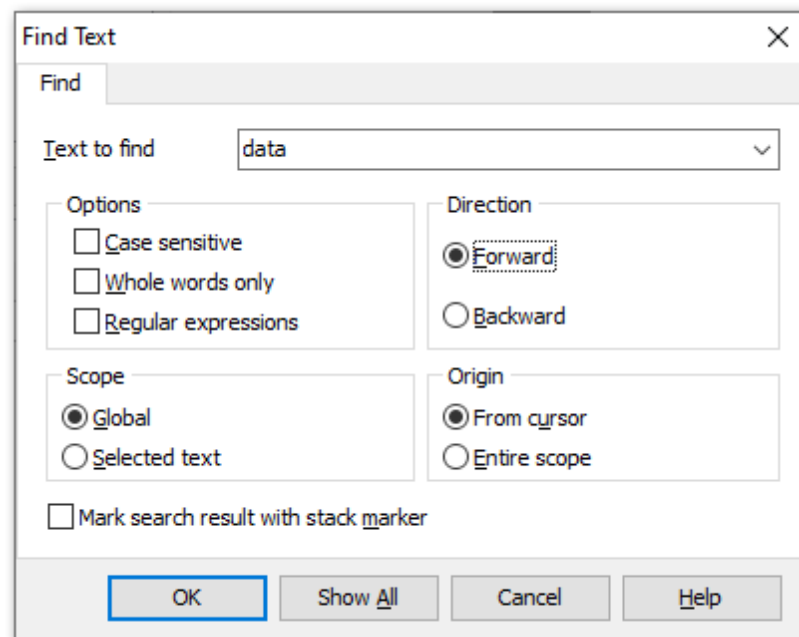
This option can be used to differentiate uppercase characters from lowercase ones during the search process.

☒ **Whole words only**

Use this option to search for words only (with this option off, the search string might be found within longer words.)

☒ **Regular expressions**

Recognizes regular expressions in the search string.



Direction

☒ **Forward**

Searches from the current position to the end of the working area.

☐ **Backward**

Searches from the current position to the beginning of the working area.

Scope

☒ **Global**

Searches within the entire working area, in the direction specified by the *Direction*

setting.

☒ **Selected text**

Searches only within the currently selected text, in the direction specified by the *Direction* setting. You can use the mouse or block commands to select a block of text.

Origin

☒ **From cursor**

The search starts at the cursor's current position, and then proceeds either forward to the end of the scope, or backward to the beginning of the scope depending on the *Direction* setting.

☒ **Entire scope**

The search covers either the entire block of selected text or the entire script (no matter where the cursor is in the Editor area) depending upon the *Scope* options.

☒ **Mark search result with stack marker**

The option toggles marking search results. If this option is selected, stack markers are set at all search positions - this makes it possible to jump from one marker (search result) to another within the text.

Click the **Show All** button to highlight every occurrence of the search string.

4.6 Replace Text dialog

The **Replace Text** dialog is provided for searching and replacing text within the working area of the script editor.

Text to find

Enter a search string in this box. The Arrow-Down button which can be found next to the input box allows you to select any of the previously entered search strings.

Text to replace

This box allows you to enter a string to replace the search string. The Arrow-Down button which can be found next to the input box allows you to select any of the previously entered strings. To replace the search string with an empty string, leave this input box blank.

Options

☒ **Case sensitive**

This option can be used to differentiate uppercase characters from lowercase ones during the search process.

☒ **Whole words only**

Use this option to search for words only (with this option off, the search string might be found within longer words.)

☒ **Regular expressions**

Recognizes regular expressions in the search string.

☒ **Replace with template**

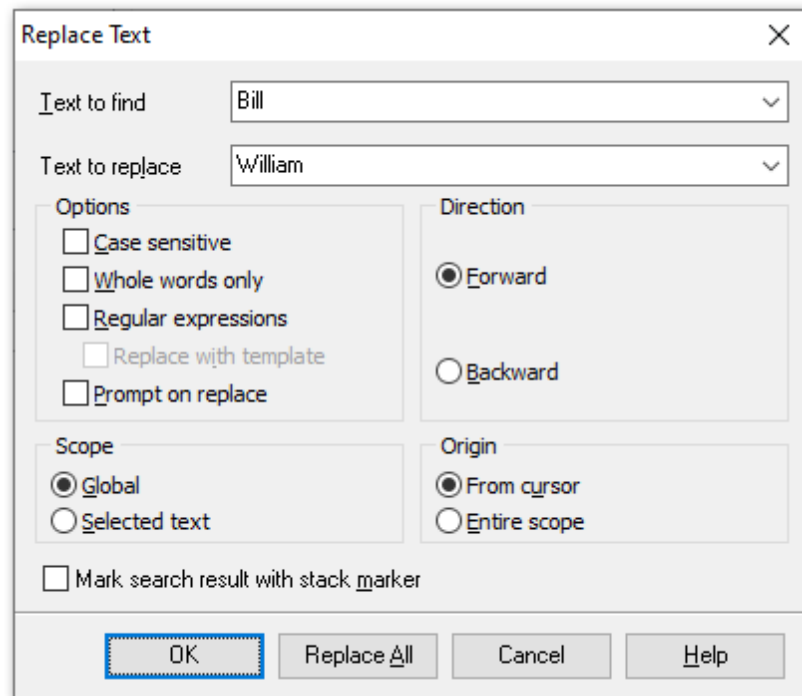
This option requires the **Regular expressions** option selection.

Enable this option to use regular expressions in the **Text to replace** field. Expression used in this field will be applied to each string that matches the **Text to find** expression.

Note: The syntax of regular expressions that can be used in the Text to find and the Text to replace fields is similar to that used in Perl regular expressions. Comprehensive information about it can be found at <http://perldoc.perl.org/perlre.html#Regular-Expressions>.

☒ **Prompt on replace**

Check this option if you wish to be prompted before replacing upon each occurrence of the search string. When this option is off, the search string is replaced automatically.



Direction

☒ **Forward**

Searches and replaces from the current position to the end of the working area.

☐ **Backward**

Searches and replaces from the current position to the beginning of the working area.

Scope

☒ **Global**

Searches and replaces within the entire working area, in the direction specified by the *Direction* setting.

☐ **Selected text**

Searches and replaces only within the currently selected text, in the direction specified by the *Direction* setting. You can use the mouse or block commands to select a block of text.

Origin

☒ **From cursor**

The search and replace process starts at the cursor's current position, and then proceeds either forward to the end of the scope, or backward to the beginning of the scope depending on the *Direction* setting.

☐ **Entire scope**

The search and replace process covers either the entire block of selected text or the entire script (no matter where the cursor is in the Editor area) depending upon the *Scope* options.

☒ **Mark search result with stack marker**

The option toggles marking search results. If this option is selected, stack markers are set at all search positions - this makes it possible to jump from one marker (search result) to another within the text.

Click the **Replace All** button to replace every occurrence of the search string. If you have checked the **Prompt on replace** option, the confirmation dialog box appears upon each occurrence of the search string.

Credits

Software Developers:

Sergey Sviridov

Dmitry Schastlivtsev

Alexey Butalov

Alexander Zhiltsov

Technical Writers:

Dmitry Doni

Olga Ryabova

Semyon Slobodenyuk

Cover Designer:

Tatyana Makurova

Translators:

Anna Shulkina

Serge Fominikh

Team Coordinators:

Dmitry Schastlivtsev

Alexander Chelyadin

Roman Tkachenko