pgAdmin3

Contents:

- Introduction
- Using pgAdmin III
 - pgAdmin Main Window
 - Connect to server
 - Change Password
 - Control Server
 - Query tool
 - pgAdmin Debugger
 - pgAdmin Data Export
 - Edit Data
 - Maintain a database object
 - Backup
 - Supported File Formats
 - The Backup Dialog
 - Restore
 - The Restore Dialog
 - Grant Wizard
 - Report Tool
 - Database Server Status
 - pgAdmin Options
 - Guru Hints
 - Command Line Parameters
- pgAgent
 - pgAgent Installation
 - pgAgent Jobs
 - pgAgent Schedules
 - pgAgent Steps
- Slony-I support
 - Slony-I with pgAdmin III overview

- Slony-I administration with pgAdmin III: installation
- Creating paths and listens
- Creating sets and subscriptions
- Execute DDL scripts with Slony-I
- Slony-I tasks
- Slony-I example
- Extended features
- Appendices
 - Bug Reporting
 - The pgAdmin Development Team
 - Translation team
 - The PostgreSQL Licence
 - The MIT Kerberos Licence
 - The OpenSSL Licence

pgAdmin3 LTS 1.23.0a documentation »

next | index

Introduction

pgAdmin III is a comprehensive PostgreSQL database design and management system for Unix and Windows systems. It is freely available under the terms of the The PostgreSQL Licence and may be redistributed provided the terms of the licence are adhered to. The project is managed by the The pgAdmin Development Team.

This software was written as a successor to the original pgAdmin and pgAdmin II products, which though popular, had limitations in the design that prevented them being taken to the 'next level'. pgAdmin III is written in C++ and uses the excellent wxWidgets (formerly wxWindows) cross platform toolkit. Connection to PostgreSQL is made using the native libpq library.

pgAdmin3 LTS 1.23.0a documentation »

previous | next | index

Using pgAdmin III

This section explains how you can use pgAdmin to maintain your PostgreSQL databases. pgAdmin supports database server versions 7.3 and up. Versions older than 7.3 are not supported, please use pgAdmin II for these.

Contents:

- pgAdmin Main Window
 - Getting started
- Connect to server
 - Connection errors
- Change Password
 - Good practice
- Control Server
- Query tool
 - Graphical Query builder
 - pgAdmin Data Export
 - Query Tool Macros
 - pgScript Scripting Language Reference
- pgAdmin Debugger
- pgAdmin Data Export
- Edit Data
 - View Data Options
- Maintain a database object
 - VACUUM
 - ANALYZE
 - REINDEX
- Backup
- Supported File Formats
- The Backup Dialog
- Restore
- The Restore Dialog

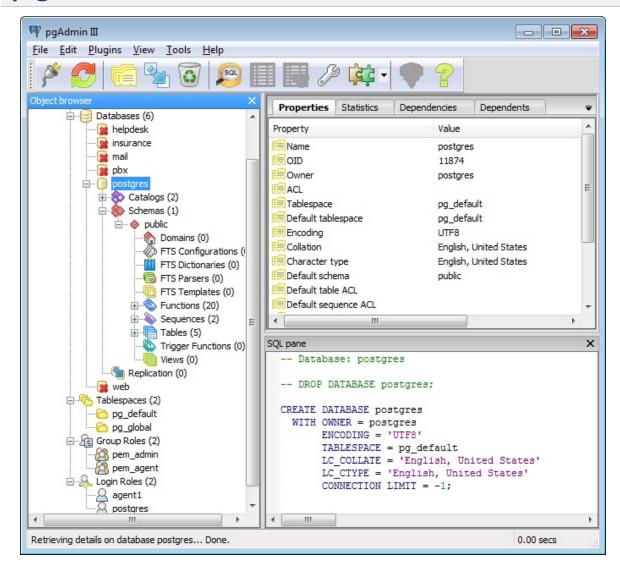
- Grant Wizard
- Report Tool
 - Default XSL Stylesheet
- Database Server Status
- pgAdmin Options
 - pgAdmin Browser Options
 - pgAdmin Query tool Options
 - pgAdmin Database Designer Options
 - pgAdmin Server Status Options
 - pgAdmin Miscellaneous Options
- Guru Hints
- Command Line Parameters

When editing the properties of a database object, pgAdmin will support you with help about the underlying PostgreSQL SQL commands, if you press the F1 function key. In order for this to work, the SQL helpsite setting in the options dialog must be set correctly.

pgAdmin3 LTS 1.23.0a documentation »

previous | next | index

pgAdmin Main Window



In the main window, the structure of the databases is displayed. You can create new objects, delete and edit existing objects if the privileges of the user that you are using on the current connection to the database allow this.

The left side of the main window shows a tree with all servers, and the objects they contain.

The upper right side shows details of the object currently selected in the tree. Some objects might have statistics in addition to their properties, these can be shown if you select the Statistics tab.

The lower right side contains a reverse engineered SQL script. You can copy this to any editor using cut & paste, or save it to a file using Save definition... from the File menu, or use it as a template if you select the Query Tool. If the Copy SQL from main window to query tool option is selected, the SQL query will be copied automatically to the tool.

The status line will show you some status information, as well as the time the last action took pgAdmin III to complete.

You can resize the main window, and change the sizes of the three main regions as you prefer. These adjustments will be preserved when you exit the program.

pgAdmin is bandwidth friendly. The status of objects in the browser is only refreshed on request or after changes made with the built-in tools. Be aware that this does not cover changes made via manual SQL or from other users or other clients. It is generally advisable to manually refresh objects before working on them in such environments.

Getting started

After you have added the desired server(s) to the tree on the left side using the Add server menu or toolbar button, each server will show up under the top node "Servers".

To open a connection to a server, select the desired server in the tree, and double click on it or use Connect from the Tools menu. The connection will be established, and the properties of the top level objects are retrieved from the database server. If you've been connected to that database previously, pgAdmin III will restore the previous selection of database and schema for you. The current situation is saved when exiting the program, so that pgAdmin III is able to restore the previous environment.

Using the menu or toolbar buttons, you can create new objects, delete objects and edit properties of existing objects if the user that you entered when adding the server connection has sufficient privileges. You may find some options grayed out if displaying properties. This means, that the database server you're currently using doesn't support the feature, or that this property can't be changed by design, or that your user privileges won't allow you to change it.

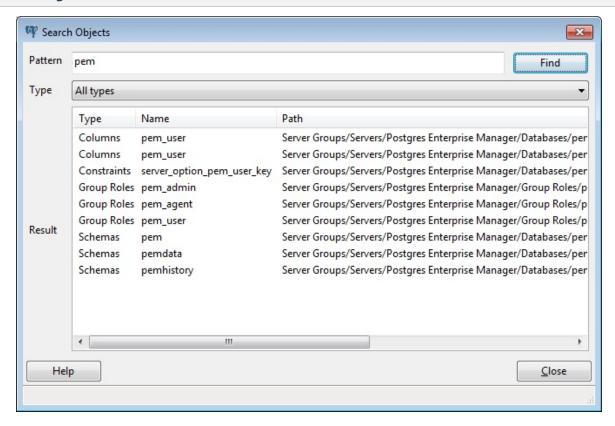
You can search for objects in the database using the Search Tool

Contents:

Object search

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » pgAdmin previous | next | index Main Window »

Object search



With this dialog, you can search for almost any kind of objects in a database.

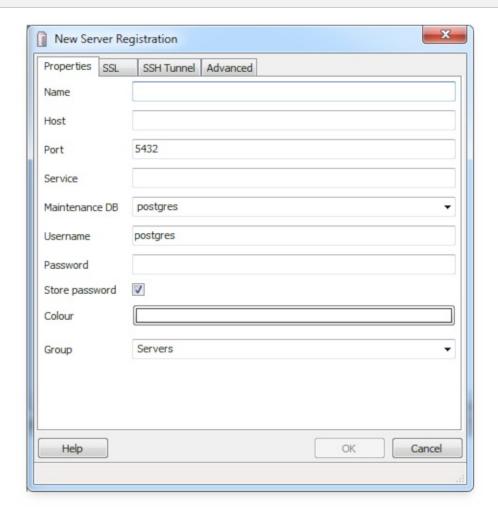
You can access it by right clicking a database and select "Search objects" or by hitting CTRL-G.

The minimum pattern length are 3 characters except for operators. The search performed is non-case sensitive and will find all objets whose name contains the pattern. You can only search for object names.

The result is presented in the grid with object type, object name and the object tree path. You can click on a result row to select the object in the browser. If the object is grey, this means that you don't have enabled those object types in the Browser settings, so you can't click on it.

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » pgAdmin previous | next | index Main Window »

Connect to server



Use this dialog to add a new server connection to the pgAdmin tree.

If you're experiencing connection problems, check the connection problems.

The name is a text to identify the server in the pgAdmin tree.

The host is the IP address of the machine to contact, or the fully qualified domain name. On Unix based systems, the address field may be left blank to use the default PostgreSQL Unix Domain Socket on the local machine, or be set to an alternate path containing a PostgreSQL socket. If a path is entered, it must begin

with a "/". The port number may also be specified.

The service field is the name of a service configured in the pg_service.conf file. For details, see the pg_service documentation.

The maintenance DB field is used to specify the initial database that pgAdmin connects to, and that will be expected to have the pgAgent schema and adminpack objects installed (both optional). On PostgreSQL 8.1 and above, the maintenance DB is normally called 'postgres', and on earlier versions 'template1' is often used, though it is preferrable to create a 'postgres' database for this purpose to avoid cluttering the template database.

If you select "Store password", pgAdmin stores passwords you enter in the ~/.pgpass file under Unix or :file:%APPDATA%postgresqlpgpass.conf under Win32 for later reuse. For details, see pgpass documentation. It will be used for all libpq based tools. If you want the password removed, you can select the server's properties and uncheck the selection any time.

The colour field allows you to set a specific colour for this server. This colour will be used in the background of the tree where each object of this server is displayed.

The Group field is used to push your server in a specific group. You can have a production group, and a test group. Or LAN specific groups. It's completely up to you. But it helps when you have lors of server to register.

The second tab has all the SSL specific options: what kind of SSL connection you want, your root certificate file, your server CRL, your client certificate file, and finally your client key file.

You can use the "SSH Tunneling" tab to connect pgAdmin (through an intermediary proxy host) to a server that resides on a network to which the client may not be able to connect directly.

- Check the box next to "Use SSH tunneling?" to specify thatpgAdmin should use an SSH tunnel when connecting to the specified server.
- Specify the name or IP address of the SSH host (through which client connections will be forwarded) in the *Tunnel host* field.
- Specify the name of a user with connection privileges for the SSH host in the *Username* field.
- Specify the type of authentication that will be used when connecting to the SSH host in the *Authentication* field.
 - Select the radio button next to Password to specify that pgAdmin will use a password for authentication by the SSH host. This is the default.
 - Select the radio button next to *Identity file* to specify that pgAdmin will use a private key file when connecting.
- If the SSH host is expecting a private key file for authentication, use the *Identity file* field to specify the location of the key file.
- If the SSH host is expecting a password, use the Password/Passphrase field to specify a the password, or if an identity file is being used, specify the passphrase.

The fourth tab contains some advanced options that are seldomly used.

The "Connect now?" field makes pgAdmin attempt a connection as soon as you hit the OK button.

You can unckeck the "Connect now" checkbox if you don't want the connection to the server being established immediately, but only registered for later use. In this case, the connection parameters won't be validated.

The "Restore env?" option determines whether or not pgAdmin will attempt to restore the browser environment when you reconnect to this server. If you regularly use different databases on the same server you might want to turn this option off.

The Rolename field allows you to connect as a role, and then get the permissions of another one (the one you specified in this field). The connection role must be a member of the rolename.

The DB restriction field allows you to enter an SQL restriction that will be used against the pg_database table to limit the databases that you see. For example, you might enter: 'live_db', 'test_db' so that only live_db and test_db are shown in the pgAdmin browser. Note that you can also limit the schemas shown in the database from the Database properties dialogue by entering a restriction against pg_namespace.

The "Service ID" field specifies parameters to control the database service process. Its meaning is operating system dependent.

If pgAdmin is running on a Windows machine, it can control the postmaster service if you have enough access rights. Enter the name of the service. In case of a remote server, it must be prepended by the machine name (e.g. PSE1pgsql-8.0). pgAdmin will automatically discover services running on your local machine.

If pgAdmin is running on a Unix machine, it can control processes running on the local machine if you have enough access rights. Enter a full path and needed options to access the pg_ctl program. When executing service control functions, pgAdmin will append status/start/stop keywords to this. Example:

sudo /usr/local/pgsql/bin/pg_ctl -D /data/pgsql

This dialog can be launched at a later time to correct or add parameters by executing "properties" when the server is selected. You should not be connected to it if you want to make changes on its properties.

Contents:

- Connection errors
 - could not connect to Server: Connection refused
 - FATAL: no pg_hba.conf entry

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III »

previous | next | index

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Connect to previous | next | index server »

Connection errors

When connecting to a PostgreSQL server, you might get error messages that need some more explanations. If you encounter them, please read the following information carefully, we're sure they will help you.

- notrunning
- no-hba

could not connect to Server: Connection refused



If this message appears, there are two possible reasons for this: either the server isn't running at all. Simply start it.

The other non-trivial cause for this message is that the server isn't configured to accept TCP/IP requests on the address shown.

For security reasons, a PostgreSQL server "out of the box" doesn't listen on TCP/IP ports. Instead, it has to be enabled to listen for TCP/IP requests. This can be done by adding **tcpip = true** to the postgresql.conf file for Versions 7.3.x and 7.4.x, or **listen_addresses='*** for Version 8.0.x and above; this will make the

server accept connections on any IP interface.

For further information, please refer to the PostgreSQL documentation about runtime configuration.

FATAL: no pg_hba.conf entry



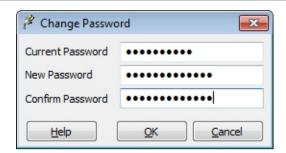
If this message appears, your server can be contacted correctly over the network, but isn't configured to accept your connection. Your client isn't detected as a legal user for the database.

You will have to add an entry in the form **host template1 postgres 192.168.0.0/24 md5** for IPV4 or **host template1 postgres ::ffff:192.168.0.0/120 md5** for IPV6 networks to the pg hba.conf file.

For further information, please refer to the PostgreSQL documentation about client authentication.

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Connect to previous | next | index server »

Change Password



To change your password, follow the following steps:

- 1. Enter your current password in the *Current Password* textbox.
- 2. Enter your desired password in the *New Password* textbox.
- 3. Enter your desired password again, this time in the *Confirm Password* textbox.
- 4. Click the OK button.

Good practice

It is good policy to set a password to protect your data, even in 'safe' environments such as at home. In the workplace, failure to apply an appropriate password policy could leave you in breach of Data Protection laws in some circumstances.

pgAdmin does not enforce any password restrictions, however we do recommend that you consider the following guidelines when selecting passwords. This is not an exhaustive list and **will not guarantee security**.

- Ensure that passwords are of adequate length. 6 characters should be the absolute minimum.
- Ensure that passwords are not easily guessable by others, or open to dictionary attacks. Use a mixture of upper and lower case letters and numerics, and avoid using words or names.
 Consider using the first letter from each word in a phrase that you will remember easily but others will not guess.
- Ensure that your password is change regularly at least every ninety days.

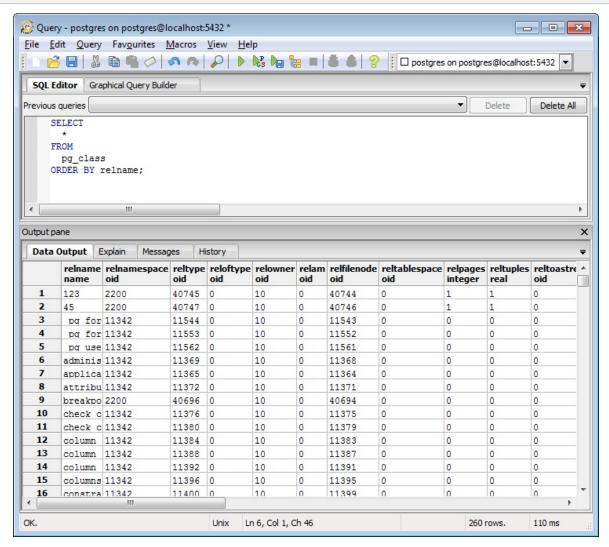
Control Server

If you entered correct data in the Server property dialogue service field, pgAdmin will check if the service is running, and allows to start and stop the service.

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III »

previous | next | index

Query tool



The Query Tool enables you to execute arbitrary SQL commands.

The upper part of the Query Tool contains the SQL Editor where you type your commands. You may read the query from a file, or write it out to a file. When writing a file, the encoding of the file is determined by the file suffix: if it is *.sql, a 8 bit local character set is used. If *.usql is selected, the file will be written in UTF-8, which enables the storage of virtually all characters used throughout the world, according to the coding rules of the Unicode Consortium. If you select *.*, the setting from the read/write unicode option is used.

The edit entry window also contains autocompletion code to help you write queries. To use autocomplete, simply start typing your query and then press Control+Space to see a list of possible object names to insert. For example, type "*SELECT * FROM* " (without the quotes, but with the trailing space), and then hit Control and Space together to see a popup menu of autocomplete options. The autocomplete system is based on code from *psql*, the PostgreSQL command line interpreter and will generally be able to offer autocomplete options in the same places that it can in *psql*.

The editor also offers a number of features for helping with code formatting:

- The auto-indent feature will automatically indent text to the same depth as the previous line when you press return.
- Block indent text by selecting two or more lines and pressing the Tab key.
- Block outdent text that has been indented using the current tab settings using Shift+Tab.
- Comment out SQL by selecting some text and pressing the Control+K.
- Uncomment SQL by selecting some text starting with a comment and pressing Control+Shift+K.
- Shift the selected text to upper case by pressing Control+U.
- Shift the selected text to lower case by pressing Control+Shift+U.

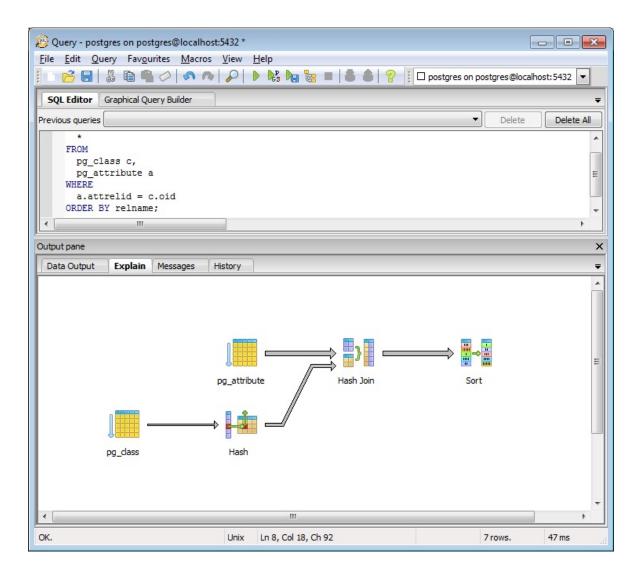
If you prefer to build your queries graphically, you can use the Graphical Query builder to generate the SQL for you.

To execute a query, select Execute from the Query menu, press the execute toolbar button, or press the F5 function key. The complete contents of the window will be sent to the database server, which executes it. You may also execute just a part of the text, by selecting only the text that you want the server to execute.

Options on the *Query* menu allow you to control COMMIT and ROLLBACK behavior for transactions:

- Check the box next to *Auto-Rollback* to instruct the server to automatically roll back a transaction if an error occurs during the transaction.
- Check the box next to *Auto-Commit* to instruct the server to automatically commit each transaction. Any changes made by the transaction will be visible to others, and are guaranteed to be durable in the event of a crash. By default, auto-commit behavior is enabled.

Explain from the Query menu, or F7 function key will execute the EXPLAIN command. The database server will analyze the query that's sent to it, and will return the results.



The result is displayed as text in the Data Output page, and graphically visualized on the Explain page. This enables you to find out how the query is parsed, optimized and executed. You can modify the degree of inspection by changing the Explain options for this in the Query menu. Please note that "EXPLAIN VERBOSE" can not be displayed graphically.

In case the query you sent to the server using the Execute or Explain command takes longer than you expect, and you would like to abort the execution, you can select Cancel from the Query menu, press the Cancel toolbar button or use Alt-Break function key to abort the execution.

You can run pgScript scripts by selecting Execute pgScript from the Query menu instead of Execute, or you press the Execute pgScript toolbar button, or you press the F6 function key. The complete contents of the edit entry window will be sent to the pgScript engine, which interprets it. pgScript scripts are composed of regular SQL commands but add some enhancements:

- Control-of-flow language (IF and WHILE structures)
- Local variables (SET @VARIABLE = 5)
- Random data generators (INTEGER(10, 20) Or REFERENCE(table, column))

For more information on the pgScript language, please look at the pgScript scripting language reference.

If you want to have help about a SQL command you want to execute, you can mark a SQL keyword and select SQL Help from the Help menu, the SQL Help toolbar button or simply press the F1 key. pgAdmin III will try to locate the appropriate information in the PostgreSQL documentation for you.

The result of the database server execution will be displayed in the lower part of the Query Tool. If the last command in the chain of SQL command sent to the server was as command returning a result set, this will be shown on the Data Output page. All rowsets from previous commands will be discarded.

To save the data in the Data Output page to a file, you can use the Export dialog.

Information about all commands just executed will go to the Messages page. The History page will remember all commands executed and the results from this, until you use Clear History from the Query menu to clear the window. If you want to retain the history for later inspection, you can save the contents of the History page to

a file using the Save history option from the Query menu.

The status line will show how long the last query took to complete. If a dataset was returned, not only the elapsed time for server execution is displayed, but also the time to retrieve the data about your current selection.

In the toolbar combobox you can quickly change your database connection from one database to another, without launching another instance of the query tool. Initially, only one database will be available, but by selecting <new connection> from the combobox, you can add another connection to it.



In the options dialog, you can specify a default limit for the rowset size to retrieve. By default, this value will be 100. If the number of rows to retrieve from the server exceeds this value, a message box will appear asking what to do to prevent retrieval of an unexpected high amount of data. You may decide to retrieve just the first rows, as configured with the max rows setting, or retrieving the complete rowset regardless of the setting, or abort the query, effectively retrieving zero rows.

If you have queries that you execute often, you can add these to the favourites menu, and have them automatically put in the buffer when you select them from the menu. The Manage Favourites menu option may be used to organise your favourites. Alternatively, you can store them as Query Tool Macros.

The Query Tool also includes a powerful Find and Replace tool. In addition to offering the normal options found in most tools, a Regular Expression search mode is included which allows you to perform extremely powerful search (and replace) operations. Unless your copy of pgAdmin was built against a non-standard build of wxWidgets, a built in version of Henry Spencer's regular expression library is used based on the 1003.2 spec and some (not quite all) of the Perl5 extensions.

For more details of the regular expression syntax offered, please see the wxWidgets documentation.

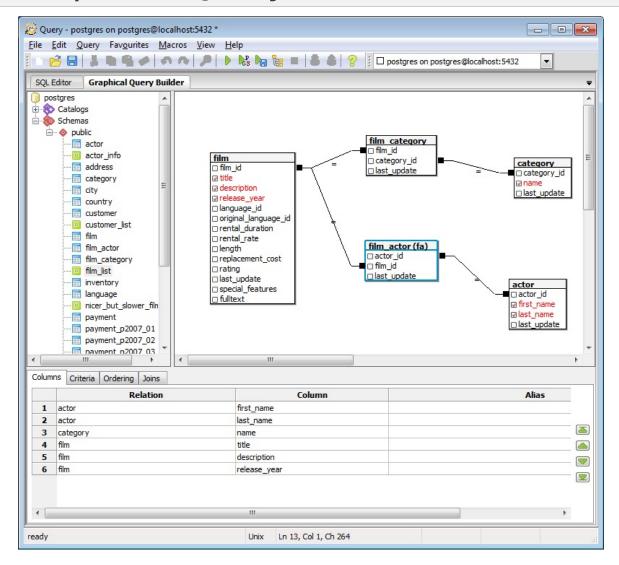
Contents:

- Graphical Query builder
- pgAdmin Data Export
- Query Tool Macros
- pgScript Scripting Language Reference
 - Overview
 - Examples
 - SQL Commands
 - Variables
 - Control-of-flow structures
 - Additional functions and procedures
 - Random data generators

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Query tool previous | next | index

×

Graphical Query builder



The Graphical Query Builder (GQB) is part of the Query Tool which allows you to build simple SQL queries visually.

There are three main parts of the GQB. The relation browser on the left hand side allows you to select catalogs, classes and views to include in your queries. Along the bottom are a set of tabs which allow you to specify selection criteria, output format and sorting. The third section is the canvas on which you draw the relationships between the relations in your tables. You can adjust the relative size of each section by dragging the join (or sash) between the panes.

To add relations to the query, either double click them in the relation browser, or drag them from the browser onto the canvas. You can arrange relations on the canvas by selecting the relation name with the mouse and moving the relation to the desired position.

To create joins between relations, drag a column from one relation onto another. A line will be drawn between them to indicate the join. Right-clicking the join will present a popup menu from where you can select the join operator or delete it.

To select the columns that will appear in the query results, check the desired columns within the relation on the canvas. Each column selected will be added to the list on the Columns tab on which you can adjust the ordering, and specify column aliases.

To specify selection criteria, add rows to the grid on the Criteria tab. The restricted value can be set to a column name, or a constant value. The Operator column allows you to select a simple operator to be used to compare the restricted value with the value specified in the Value column. The Connector specifies how the criteria will be joined to the next (if any).

The query results may be ordered on the Ordering tab. Select the columns required on the left, and use the buttons in the middle to add them to the ordering list on the right. You can also remove columns in the same way. For each column added to the ordering list, you can specify the sorting direction. You can also adjust the priority of each ordering column by selecting it, and moving it up or down the list using the buttons on the right.

When you have designed your query, you can use the Execute, Execute to File, or Explain options on the toobar (and Query menu) to generate and execute the query.

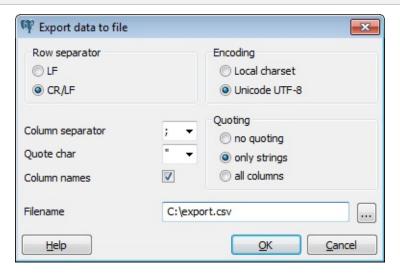
pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Query tool previous | next | index

×

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Query tool previous | next | index

×

pgAdmin Data Export



Data is written to a file, using the selected options.

The row separator option selects the character used to separate rows of data. On Unix systems, usually a linefeed character (LF) is used, on Windows system a carriage return/linefeed (CR/LF) combination.

The separating character between columns can be selected between colon, semicolon and a vertical bar.

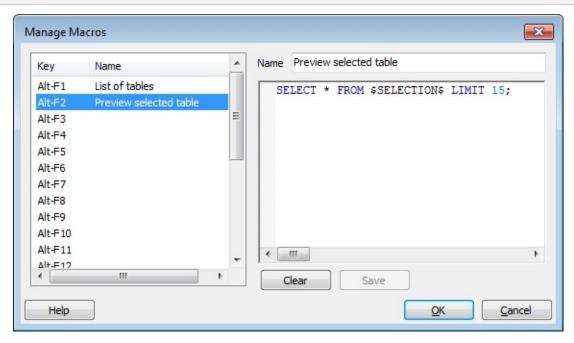
The individual columns can be enclosed in quotes. Quoting can be applied to string columns only (i.e. numeric columns will not be quoted) or all columns regardless of data type. The character used for quoting can be a single quote or a double quote.

If the "Column names" option is selected, the first row of the file will contain the column names.

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Query tool previous | next | index

×

Query Tool Macros



Query Tool Macros enable you to execute pre-defined SQL queries with single key press. Pre-defined queries can contain the placeholder \$SELECTION\$. Upon macro execution, the placeholder will be replaced with the currently selected text in the SQL pane of the Query Tool.

To create a macro, select the Manage Macros option from the Macros menu on the Query Tool. Select the key you wish to use, enter the name of the macro, and the query, optionally including the selection placeholder, and then click the Save button to store the macro.

To clear a macro, select the macro on the Manage Macros dialogue, and then click the Clear button.

To execute a macro, simply press the appropriate shortcut keys, or select it from the Macros menu.

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Query tool previous | next | index

>>

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Query tool previous | next | index

>>

pgScript Scripting Language Reference

Overview

pgScript is composed of pgScript commands:

```
pgScript command
: Regular PostgreSQL SQL Command (SELECT INSERT CREATE ...)
| Variable declaration or assignment (DECLARE SET)
| Control-of-flow structure (IF WHILE)
| Procedure (ASSERT PRINT LOG RMLINE)
```

Command names (**SELECT**, **IF**, **SET**, ...) are case-insensitive and must be ended with a semi-column ;. Identifiers are case-sensitive.

Examples

Batch table creations

```
DECLARE @I, @T; -- Variable names begin with a @
SET @I = 0; -- @I is an integer
WHILE @I < 20
BEGIN
SET @T = 'table' + CAST (@I AS STRING); -- Casts @I
CREATE TABLE @T (id integer primary key, data text);

SET @I = @I + 1;
END
```

Insert random data

```
DECLARE @I, @J, @T, @G;
SET @I = 0;
SET @G1 = INTEGER(10, 29, 1); /* Random integer generator
                            Unique numbers between 10 and 29 *
SET @G2 = STRING(10, 20, 3); /* Random string generator
                             3 words between 10 and 20 characte
WHILE @I < 20
BEGIN
    SET @T = 'table' + CAST (@I AS STRING);
SET @J = 0;
   WHILE @J < 20
   BEGIN
        INSERT INTO @T VALUES (@G1, '@G2');
        SET @J = @J + 1;
    END
SET @I = @I + 1;
END
```

Batch table deletions

```
DECLARE @I, @T; -- Declaring is optional
```

```
SET @I = 0;
WHILE 1 -- Always true
BEGIN
    IF @I >= 20
        BREAK; -- Exit the loop if @I > 20

SET @T = 'table' + CAST (@I AS STRING);
    DROP TABLE @T;

SET @I = @I + 1;
END
```

Print information on screen

```
SET @PROGR@M#TITLE = 'pgScript';
PRINT '';
PRINT @PROGR@M#TITLE + ' features:';
PRINT '';
PRINT ' * Regular PostgreSQL commands';
PRINT ' * Control-of-flow language';
PRINT ' * Local variables';
PRINT ' * Random data generators';
```

SQL Commands

You can run ANY PostgreSQL query from a pgScript EXCEPT those ones:

```
BEGIN;
END;
```

This is because **BEGIN** and **END** are used for delimiting blocks. Instead use:

```
BEGIN TRANSACTION;
END TRANSACTION;
```

For a list of PostgreSQL commands: http://www.postgresql.org/docs/8.3/interactive/sql-commands.html

Variables

There are two main types of variables: simple variables and records (result sets composed of lines and columns).

Variable names begin with a @ and can be composed of letters, digits, $_$, #, @.

Variable type is guessed automatically according to the kind of value it contains. This can be one of: number (real or integer), string, record.

Simple variables

Simple variable declaration

Declaring simple variable is optional:

```
DECLARE @A, @B;
DECLARE @VAR1;
```

Simple variable affectation

This is done with the **SET** command. The variable type depends on the value assigned to this variable:

```
SET @A = 1000, @B = 2000; -- @A and @B are <strong>integer nu SET @C = 10e1, @D = 1.5; -- @C and @D are <strong>real numbe SET @E = 'ab', @F = 'a''b'; -- @E and @F are <strong>strings** SET @G = "ab", @H = "a\"b"; -- @G and @H are <strong>strings**
```

An uninitialized variable defaults to an empty string. It is possible to override variables as many times as wanted:

```
PRINT @A; -- Prints an empty string

SET @A = 1000; -- @A is initialized an integer

PRINT @A; -- Prints 1000

SET @A = 'ab'; -- @A becomes a string

PRINT @A; -- Prints ab
```

Data generators

Data generators allows users to generate random values. There are various types of generators, each one producing different type of data. A variable initialized with a data generator behaves like a regular simple variable except that it has a different value each time it is used:

```
SET @A = INTEGER(100, 200);
PRINT @A; -- Prints an integer between 100 and 200
PRINT @A; -- Prints another integer between 100 and 200
```

A variable can contain a generator but its type is one of: number (real or integer), string. For a list of available generators and their associated type, see generators.

Records

Record declaration

Declaring a record is **required**. A name for each column must be specified even if they will not be used anymore afterwards:

```
DECLARE @R1 { @A, @B }, @R2 { @A, @C }; -- Two records with two DECLARE @R3 { @A, @B, @C, @D }; -- One record with fou
```

The number of lines is dynamic: see the next section.

Record affectation

To access a specific location in a record, one must use the line number (starts at 0) and can use either the column name (between quotes) or the column number (starts at 0). This specific location behaves like a simple variable. Note that a record cannot contain a record:

```
SET @R1[0]['@A'] = 1; -- First line & p; first column
SET @R1[0][0] = 1; -- Same location
SET @R1[4]['@B'] = 1; -- Fifth line & p; second column
SET @R1[0][1] = 1; -- Same location
```

In the above example, three empty lines are automatically inserted between the first and the fifth. Using an invalid column number or name results in an exception.

Specific location can be used as right values as well. A specific line can also be used as right value:

```
SET @R1[0][0] = @R3[0][1], @A = @R2[0][0]; -- Behaves like simp
SET @A = @R1[1]; -- @A becomes a record which is the first line
```

Remember that SET @R1[0][0] = @R2 is impossible because a record cannot contain a record.

It is possible to assign a record to a variable, in this case the variable does not need to be declared:

```
SET @A = @R3; -- @A becomes a record because it is assigned a r
```

SQL queries

Any SQL query executed returns a record. If the query is a SELECT query then it returns the results of the query. If it is something else then it returns a one-line record (true) if this is a success otherwise a zero-line record (false):

```
SET @A = SELECT * FROM table; -- @A is a record with the resu

SET @B = INSERT INTO table ...; -- @B is a one-line record if t
```

Record functions

See function2.

Cast

It is possible to convert a variable from one type to another with the cast function:

```
SET @A = CAST (@B AS STRING);
SET @A = CAST (@B AS REAL);
SET @A = CAST (@B AS INTEGER);
SET @A = CAST (@B AS RECORD);
```

When a record is converted to a string, it is converted to its flat representation. When converted to a number, the record is first converted to a string and then to a number (see string conversion for more details).

When a number is converted to a string, it is converted to its string representation. When converted to a record, it is converted to a one-line-one-column record whose value is the number.

When a string is converted to a number, if the string represents a number then this number is returned else an exception is thrown. When converted to a record, either the program can find a **record pattern** in the string or it converts it to a one-line-one-column record whose value is the string. A record pattern is:

```
SET @B = '(1, "abc", "ab\\"")(1, "abc", "ab\\"")'; -- @B is a s
SET @B = CAST (@B AS RECORD); @B becomes a two-line-three-colum
```

Remember a string is surrounded by simple quotes. Strings composing a record must be surrounded by double quotes which are escaped with \(\infty\) (we double the slash because it is already a special character for the enclosing simple quotes).

Operations

Operations can only be performed between operands of the same type. Cast values in order to conform to this criterion.

Comparisons result in a number which is 0 or 1.

Strings

```
Comparisons: = \langle \rangle \rangle \langle \langle \rangle \rangle AND OR
```

Concatenation: +

```
SET @B = @A + 'abcdef'; -- @A must be a string and @B will be a
```

Boolean value: non-empty string is true, empty string is false

Inverse boolean value: NOT

Case-insensitive comparison: ~=

Numbers

```
Comparisons: = \langle \rangle \rangle \langle \langle \rangle \rangle AND OR
```

Arithmetic: + - * / %

```
SET @A = CAST ('10' AS INTEGER) + 5; -- '10' string is converte
```

Boolean value: 0 is false, anything else is true

Inverse boolean value: NOT (note that NOT NOT 10 = 1)

An arithmetic operation involving at least one real number gives a real number as a result:

```
SET @A = 10 / 4.; -- 4. is a real so real division: @A = 2.5

SET @A = 10 / 4; -- 4 is an integer so integer division: @A =
```

Records

Comparisons: = <> > < <= >= AND OR

Boolean value: zero-line record is false, anything else is true

Inverse boolean value: NOT

Comparisons for records are about inclusion and exclusion. Order of lines does not matter. <= means that each row in the left operand has a match in the right operand. >= means the opposite. = means that <= and >= are both true at the same time...

Comparisons are performed on strings: even if a record contains numbers like 10 and 1e1 we will have '10' <> '1e1'.

Control-of-flow structures

Conditional structure

```
IF condition
BEGIN

pgScript commands

END

ELSE
BEGIN

pgScript commands

END
```

pgScript commands are optional. **BEGIN** and **END** keywords are optional if there is only one pgScript command.

Loop structure

```
WHILE condition
BEGIN
pgScript commands
END
```

pgScript commands are optional. **BEGIN** and **END** keywords are optional if there is only one pgScript command.

BREAK ends the enclosing **WHILE** loop, while **CONTINUE** causes the next iteration of the loop to execute. **RETURN** behaves like **BREAK**:

```
WHILE condition1
BEGIN
IF condition2
BEGIN
BREAK;
END
```

Conditions

Conditions are in fact results of operations. For example the string comparison 'ab' = 'ac' will result in a number which is false (the equality is not true):

```
IF 'ab' ~= 'AB' -- Case-insensitive comparison which result in
BEGIN
    -- This happens
END
IF 0 -- false
BEGIN
    -- This does **not** happen
END
ELSE
BEGIN
   -- This happens
END
WHILE 1
    -- Infinite loop: use BREAK for exiting
END
4
```

It is possible to the result of a SQL SELECT query directly as a condition. The query needs to be surrounded by parenthesis:

Additional functions and procedures

Procedures

Procedures do not return a result. They must be used alone on a line and cannot be assigned to a variable.

Print

Prints an expression on the screen:

```
PRINT 'The value of @A is' + CAST (@A AS STRING);
```

Assert

Throws an exception if the expression evaluated is false:

```
ASSERT 5 > 3 AND 'a' = 'a';
```

Remove line

Removes the specified line of a record:

```
RMLINE(@R[1]); -- Removes @R second line
```

Functions

Functions do return a result. Their return value can be assigned to a variable, like the CAST operation.

Trim

Removes extra spaces surrounding a string:

```
SET @A = TRIM(' a '); -- @A = 'a'
```

Lines

Gives the number of lines in a record:

```
IF LINES(@R) > 0
BEGIN
    -- Process
END
```

Columns

Gives the number of columns in a record:

```
IF COLUMNS(@R) > 0
BEGIN
-- Process
END
```

Random data generators

Overview of the generators

One can assign a variable (**SET**) with a random data generators. This means each time the variable will be used it will have a different value.

However the variable is still used as usual:

```
SET @G = STRING(10, 20, 2);
SET @A = @G; -- @A will hold a random string
SET @B = @G; -- @B will hold another random string
PRINT @G, -- This will print another third random string
```

Sequence and seeding

Common parameters for data generators are sequence and seed.

sequence means that a sequence of values is generated in a random order, in other words each value appears only once before the sequence starts again: this is useful for columns with a UNIQUE constraint. For example, this generator:

```
SET @G = INTEGER(10, 15, 1); -- 1 means generate a sequence
```

It can generate such values:

```
14 12 10 13 11 15 14 12 10 13 11
```

Where each number appears once before the sequence starts repeating.

sequence parameter must be an integer: if it is 0 then no sequence

is generated (default) and if something other than 0 then generate a sequence.

seed is an integer value for initializing a generator: two generators with the same parameters and the same seed will generate **exactly** the same values.

seed must be an integer: it is used directly to initialize the random data generator.

Data generators

Optional parameters are put into brackets:

```
Generator
: INTEGER ( min, max, [sequence], [seed] );
| REAL ( min, max, precision, [sequence], [seed] );
| DATE ( min, max, [sequence], [seed] );
| TIME ( min, max, [sequence], [seed] );
| DATETIME ( min, max, [sequence], [seed] );
| STRING ( min, max, [nb], [seed] );
| REGEX ( regex, [seed] );
| FILE ( path, [sequence], [seed], [encoding] );
| REFERENCE ( table, column, [sequence], [seed] );
```

Integer numbers

```
INTEGER ( min, max, [sequence], [seed] );
INTEGER ( -10, 10, 1, 123456 );
```

min is an integer, max is an integer, sequence is an integer and seed is an integer.

Real numbers

```
REAL ( min, max, precision, [sequence], [seed] );
REAL ( 1.5, 1.8, 2, 1 );
```

min is a number, max is a number, precision is an integer that indicates the number of decimals (should be less than 30), sequence is an integer and seed is an integer.

Dates

```
DATE ( min, max, [sequence], [seed] );
DATE ( '2008-05-01', '2008-05-05', 0 );
```

min is a string representing a date, max is a string representing a date, sequence is an integer and seed is an integer.

Times

```
TIME ( min, max, [sequence], [seed] );
TIME ( '00:30:00', '00:30:15', 0 );
```

min is a string representing a time, max is a string representing a time, sequence is an integer and seed is an integer.

Timestamps (date/times)

```
DATETIME ( min, max, [sequence], [seed] );
DATETIME ( '2008-05-01 14:00:00', '2008-05-05 15:00:00', 1 );
```

min is a string representing a timestamp, max is a string representing a timestamp, sequence is an integer and seed is an integer.

Strings

```
STRING ( min, max, [nb], [seed] );
STRING ( 10, 20, 5 );
```

min is an integer representing the minimum length of a word, max is an integer representing the maximum length of a word, nb is an

integer representing the number of words (default: 1) and seed is an integer.

In the above example we generate 5 words (separated with a space) whose size is between 10 and 20 characters.

Strings from regular expressions

```
REGEX ( regex, [seed] );
REGEX ( '[a-z]{1,3}@[0-9]{3}' );
```

regex is a string representing a simplified regular expressions and seed is an integer.

Simplified regular expressions are composed of:

- Sets of possible characters like [a-z_.] for characters between
 a and z + _ and .
- Single characters

It is possible to specify the minimum and maximum length of the preceding set or single character:

- {min, max} like {1,3} which stands for length between 1 and 3
- {min} like {3} which stands for length of 3
- Default (when nothing is specified) is length of 1

Note: be careful with spaces because 'a {3}' means one a followed by three spaces because the 3 is about the last character or set of characters which is a space in this example.

If you need to use $[\] \setminus \{ \text{ or } \}$, they must be escaped because they are special characters. Remember to use **double backslash**: $' \setminus \{ \{ \} \} '$ for three $[\ .$

Strings from dictionary files

```
FILE ( path, [sequence], [seed], [encoding] );
FILE ( 'file.txt', 0, 54321, 'utf-8' );
```

path is a string representing the path to a text file, sequence is an integer, seed is an integer and encoding is a string representing the file character set (default is system encoding).

This generates a random integer between 1 and the number of lines in the file and then returns that line. If the file does not exist then an exception is thrown.

encoding supports the most known encoding like utf-8, utf-16le, utf-16be, iso-8859-1, ...

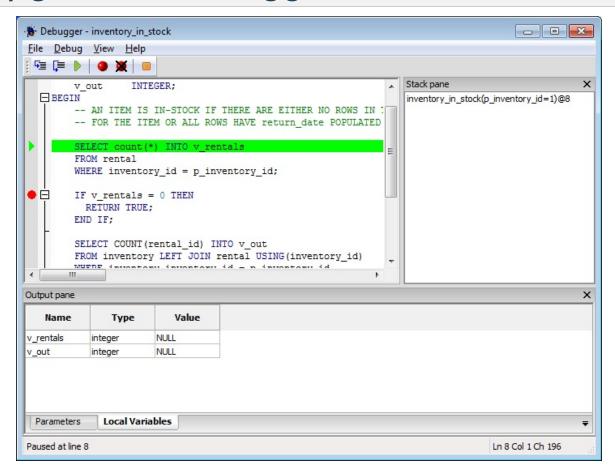
Reference to another field

```
REFERENCE ( table, column, [sequence], [seed] );
REFERENCE ( 'tab', 'col', 1 );
```

table is a string representing a table, column is a string representing a column of the table, sequence is an integer and seed is an integer.

This is useful for generating data to put into foreign-key-constrained columns.

pgAdmin Debugger



The debugger may be used to debug pl/pgsql functions in PostgreSQL, as well as EDB-SPL functions, stored procedures and packages in EnterpriseDB.

Note: The debugger may only be used by roles with 'superuser' privileges.

In order to use the debugger, a plugin is required on your server. This is included by default with EnterpriseDB, and is available for download on pgFoundry. It is installed as a contrib module with the Windows distribution of PostgreSQL 8.3 and above.

The debugger may be used for both in-context and direct debugging. To debug an object in-context, right click it in the pgAdmin browser

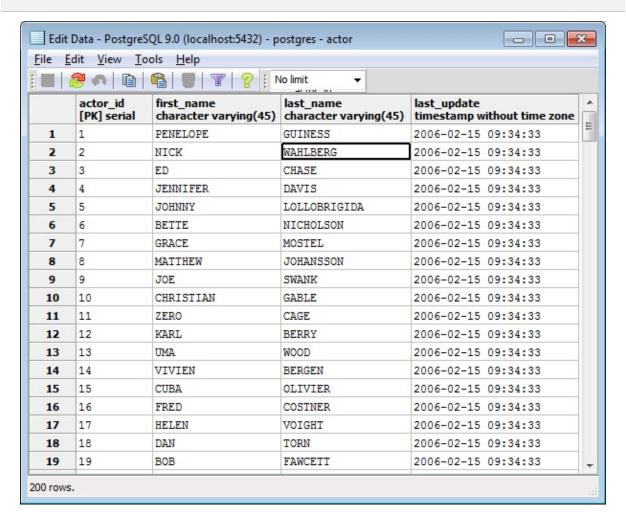
treeview, and select the "Global breakpoint" option. The debugger will then wait for the next session to execute the object, and break on the first line of executable code. To directly debug an object, right click it and select the "Debug" option. The debugger will prompt you for any parameter values that may be required, invoke the object, and break on the first line of executable code.

When entering parameter values, type the value into the appropriate cell on the grid, or, leave the cell empty to represent NULL, enter " (two single quotes) to represent an empty string, or to enter a literal string consisting of just two single quotes, enter ". PostgreSQL 8.4 and above supports variadic function parameters. These may be entered as a comma-delimited list of values, quoted and/or cast as required.

Once the debugger session has started, you can step through the code using the menu options, keyboard shortcuts or toolbar buttons. Breakpoints may be set or cleared by clicking in the margin of the source window, or by clicking on the desired code line and using the "Toggle breakpoint" button or menu option. If you step into other functions, the Stack pane may be used to navigate to different stack frames - simply select the frame you wish to view.

When the debugger has reached the end of the executable code, if running in-context it will wait for the next call to the function, otherwise it will prompt for parameter values again and restart execution. You may exit the debugger at any time.

Edit Data



The Edit Grid tool allows you to view the data that's stored in a table or view, and to edit the content if technically possible.

The header of the grid will show the name of each column, together with the data type. A column that's part of the primary key will additionally be marked with [PK].

Important: In order to edit the data, each row in the table must be uniquely identifiable. This can be done using the OID, or a primary key. If none of them exist, the table is read only. Note that views can't be edited and are read only by design; updatable views (using rules) are not supported at this time.

To change the value of a field, you select that field and enter the new text. The Store toolbar button allows the data to be written back to the server. The row will be written automatically, if you select a different row.

To enter a new row into the table, you enter the data into the last row that has a row number marked with an asterisk. As soon as you store the data, the row will get a row number, and a fresh empty asterisk line is created.

If an SQL NULL is to be written to the table, simply leave the field empty. If you store a new row, this will let the server fill in the default value for that column. If you store a change to an existing row, the value NULL will explicitly be written.

To enter a newline into a field, press Ctrl-Enter.

If you want pgAdmin III to write an empty string to the table, you enter the special string " (two single quotes) in the field. If you want to write a string containing solely two single quotes to the table, you need to escape these quotes, by typing "

To delete a row, press the Delete toolbar button.

The Refresh toolbar button allows to reread the contents of the table, refreshing the display.

You can select one or more rows, and copy them with Ctrl-C or the Copy toolbar button to the clipboard.

The Sort/Filter toolbar button will open the View Data Options dialog.

The context menu can also be used to quick-sort or quick-filter the data set. When a cell in the grid is right-clicked, the following options allow the user to selectively view the data:

- Filter by Selection: When selected, refreshes the data set and displays only those rows whose column value matches the value in the cell currently selected.
- Exclude by Selection: When selected, refreshes the data set and excludes those rows whose column value matches the value in the cell currently selected.
- Remove Filter: When selected, removes all selection / exclusion filter conditions.
- Sort Ascending: When selected, refreshes the data set and displays the currently selected rows in the ascending order of the selected column values. If a sorting preference is already present for this data set, this sorting preference is appended to the current sort order.
- Sort Descending: When selected, refreshes the data set and displays the currently selected rows in the descending order of the selected column values. If a sorting preference is already present for this data set, this sorting preference is appended to the current sort order.
- Remove Sorting: When selected, removes all sorting preferences for this data set.

Note: If a column is already selected for sorting, it's position in the sort list will remain the same, only the direction will be changed.

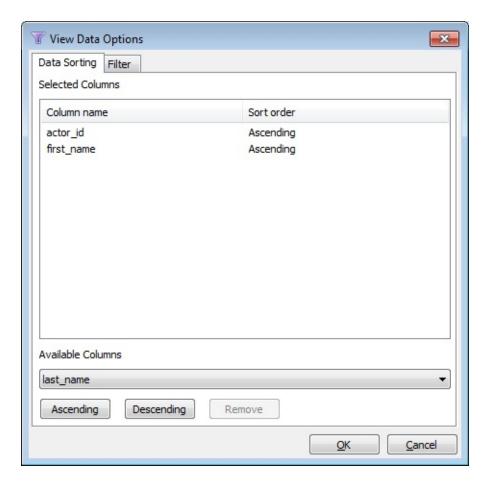
Contents:

View Data Options

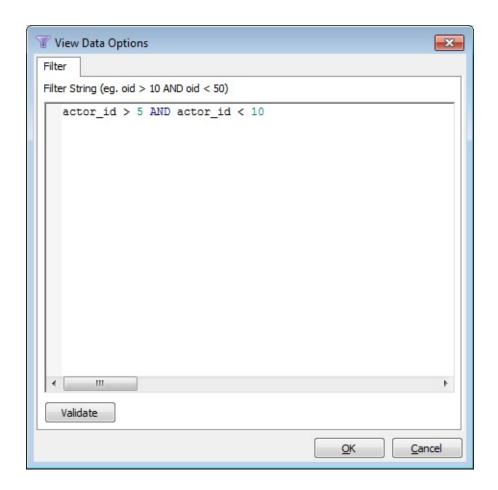
pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Edit Data » previous | next | index

View Data Options

When "Apply Filter and View Data" is selected from the main menu, or the "Sort/Filter" toolbar button is pressed in the data viewer, this dialogue appears. It allows changing the sort order of the data, and applying a filter to narrow the displayed data.



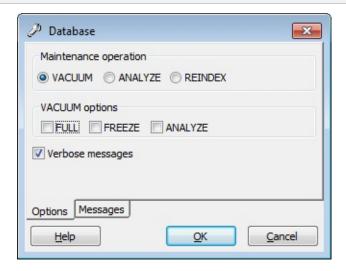
You can select one or more columns and the sort direction which should be used to retrieve the data from the database.



You can enter an arbitrary WHERE clause, to restrict the result set which is retrieved from the database. You can click "Validate" to have pgAdmin III test whether the WHERE clause is valid.

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Edit Data » previous | next | index

Maintain a database object



This tool allows to maintain the database in total, or only a selected table, or a selected index.

Maintenance comes in three flavors.

VACUUM

VACUUM will scan the database or table for rows, that are not in use any more. If a row is updated or deleted, the previous content isn't replaced, but rather marked invalid. The new data is inserted freshly into the database. You need to perform a garbage collection regularly, to insure that your database doesn't contain too much unused data, wasting disk space and ultimately degrading performance.

Please press the Help button to see the PostgreSQL help about the VACUUM command to learn more about the options.

The output of the database server is displayed in the messages page as they arrive. If Verbose is selected, the server will send very detailed info about what it did.

While this tool is very handy for ad-hoc maintenance purposes, you are encouraged to install an automatic job, that performs a VACUUM job regularly to keep your database in a neat state.

ANALYZE

ANALYZE investigates statistical values about the selected database or table. This enables the query optimizer to select the fastest query plan, to give optimal performance. Every time your data is changing radically, you should perform this task. It can be included in a VACUUM run, using the appropriate option.

REINDEX

REINDEX rebuilds the indexes in case these have degenerated caused by unusual data patterns inserted. This can happen for example if you insert many rows with increasing index values, and delete low index values.

The RECREATE option doesn't call the REINDEX SQL command internally, instead it drops the existing table and recreates it according to the current index definition. This doesn't lock the table exclusively, as REINDEX does, but will lock write access only.

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III »

previous | next | index

Backup

The *Backup* dialog provides an easy-to-use interface for the pg_dump pg_dump command line utility. Easy to use options allow you to backup a database, a schema definition, data, or an database object.

By selecting fields on the *Backup* dialog, you specify which options should be included in a customized pg_dump command. The pg_dump command writes an archive that you can use with the pgAdmin's *Restore* dialog, the psql client, or pg_restore to recreate the objects backed up by the archive.

If you choose to create a plain-text backup, you can review the SQL commands that build the selected object to better help you understand how the object will be recreated. You can also optionally modify the content to create new database objects before restoring.

Supported File Formats

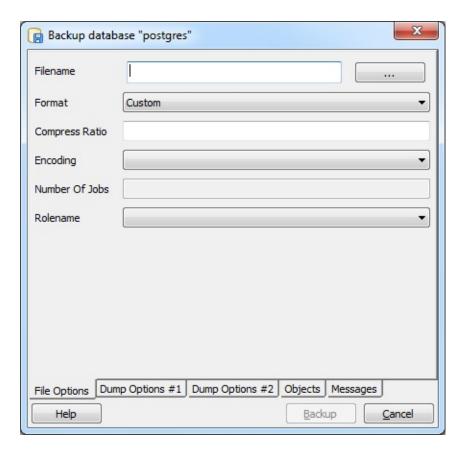
The drop-down listbox in the *Format* field allows you to select an archive format. Each format has advantages and disadvantages; select the format that is best suited for your application:

- Select *Plain* to generate a plain-text script file containing SQL statements and commands that you can execute at the psql command line or with pg_dump to recreate the database objects and load the table data. A plain-text backup file can easily be edited in a text editor if desired before restoring the database objects with the psql program. Plain-text format is normally recommended for smaller databases.
- Select Custom to generate a pg_dump formatted binary file that allows for restoration of all or only selected database objects from the backup file. You can use pgAdmin to restore from a custom archive backup file. A custom archive backup file cannot be edited, but you can use pgAdmin to select which database objects to restore from the backup file. Custom archive format is recommended for medium to large databases from which you may want to select the database objects to restore from the backup file.
- Select *Tar* to generate a tar archive file that allows for restoration of all or only selected database objects from the backup file. You can use pgAdmin to restore from a tar archive backup file.
- Select *Directory* to generate a directory-format archive suitable for use with pgAdmin's Restore dialog or pg_restore. This file format creates a directory with one file for each table and blob being dumped, plus a Table of Contents file describing the dumped objects in a machinereadable format that pg_restore can read. A directory

format archive can be manipulated with standard Unix tools; for example, files in an uncompressed archive can be compressed with the gzip tool. This format is compressed by default and supports parallel dumps.

The Backup Dialog

To open the *Backup* dialog, right click on the name of a database or a named object in the tree control and select *Backup*... from the context menu. The *Backup* dialog opens:

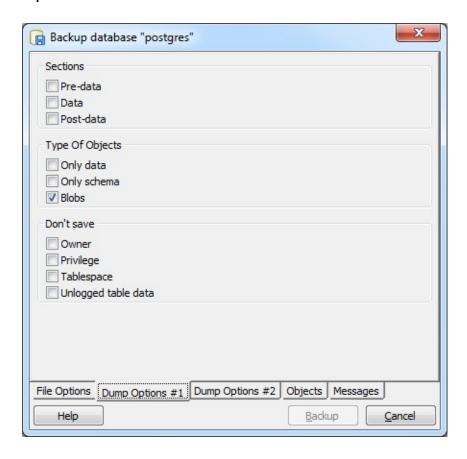


Use the fields on the *File Options* tab to specify general information about the backup.

- Enter the name of the backup file in the *Filename* field. Optionally, use the browser button to navigate into a directory and select a file that will contain the archive.
- Use the drop-down listbox in the *Format* field to select the file format for the backup.
- Use the *Compress Ratio* field to select a compression level for the backup. Specify a value of zero to mean use no

- compression; specify a maximum compression value of 9. Please note that tar archives do not support compression.
- Use the *Encoding* drop-down listbox to select the encoding that should be used for the archive.
- Use the *Number of Jobs* field (when applicable) to specify the number of tables that will be dumped simultaneously in a parallel backup.

When you've completed the *File Options* tab, navigate to the *Dump Options #1* tab:

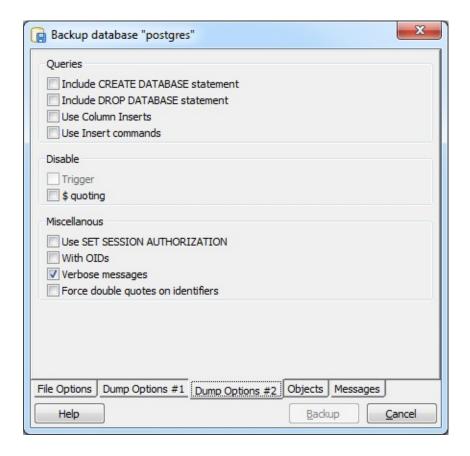


Use the fields on the *Dump Options #1* tab to specify details about the type of objects that will be backed up:

• Use the checkboxes in the *Sections* box to select a portion of the object that will be backed up. By default, a backup will include all sections.

- Check the box next to *Pre-data* to include all data definition items not included in the data or post-data item lists.
- Check the box next to Data to backup actual table data, large-object contents, and sequence values.
- Check the box next to Post-data to include definitions of indexes, triggers, rules, and constraints other than validated check constraints.
- Use the checkboxes in the *Type of Objects* box to select the objects that will be included in the backup. By default, all objects will be included in the backup.
 - Check the box next to Only data to back up only the data.
 - Check the box next to *Only schema* to back up only the schema (the data definitions).
 - Check the box next to *Blobs* to include large objects in the backup.
- Use the checkboxes in the *Don't Save* box to select the objects that will not be included in the backup.
 - Check the box next to Owner to omit commands that set object ownership.
 - Check the box next to *Privilege* to omit commands that create access privileges.
 - Check the box next to *Tablespace* to omit tablespaces.
 - Check the box next to *Unlogged* table data to omit the contents of unlogged tables.

When you've completed the *Dump Options #1* tab, select the *Dump Options #2* tab:

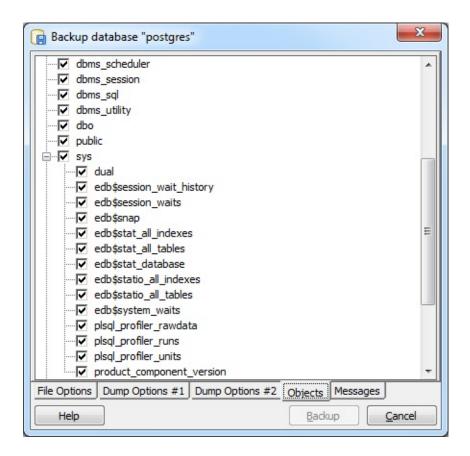


Use the fields on the *Dump Options #2* tab to specify details about the statements used within the backup file.

- Use the checkboxes in the *Queries* box to specify the type of statements that should be included in the backup.
 - Check the box next to *Include CREATE DATABASE* statement to include a command in the backup that creates a new database when restoring from the backup.
 - Check the box next to *Include DROP DATABASE* statement to include a command in the backup that drops any existing database before restoring from the backup.
 - Check the box next to Use Column Inserts to dump the data in the form of INSERT statements, with explicit column names. Please note: this may make restoration from backup slow.
 - Check the box next to Use Insert commands to dump the data in the form of INSERT statements rather than using a

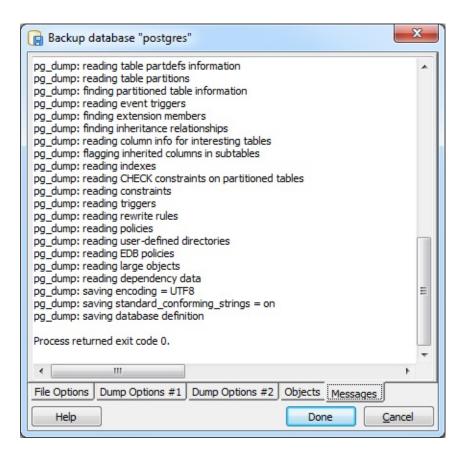
- COPY command. Please note: this may make restoration from backup slow.
- Use the checkboxes in the *Disable* box to specify the type of statements that should be excluded from the backup.
 - Check the box next to *Trigger* (active when creating a dataonly backup) to include commands that will disable triggers on the target table while the data is being loaded.
 - Check the box next to \$ quoting to disable dollar quoting within function bodies; if disabled, the function body will be quoted using SQL standard string syntax.
- Use the checkboxes in the *Miscellaneous* box to specify miscellaneous backup options.
 - SET **SESSION** Check the box Use next to AUTHORIZATION to include a statement that will use a SET SESSION AUTHORIZATION command to determine object ownership (instead of an ALTER **OWNER** command).
 - Check the box next to *With OIDs* to include object identifiers as part of the table data for each table.
 - Check the box next to Verbose messages to instruct pg_dump to use verbose messages.
 - Check the box next to Force double quotes on identifiers to force the quoting of all identifiers.

When you've completed the *Dump Options #2* tab, select the *Objects* tab:



Use the fields on the *Objects* tab to select the objects that will be included in the backup; by default, when performing a database backup, all objects will be selected for inclusion in the archive. Deselect an object name to exclude that object from the archive.

When you've specified the details that will be incorporated into the pg_dump command, click the *Backup* button to build and execute a command based on those preferences; the result will be displayed on the *Messages* tab:



If the backup is successful, the Messages tab will display:

Process returned exit code 0.

Scroll up to review the pg_dump command used to generate the archive, or to view any error messages that were returned during the backup. When you're finished, click *Done* to exit the *Backup* dialog.

Restore

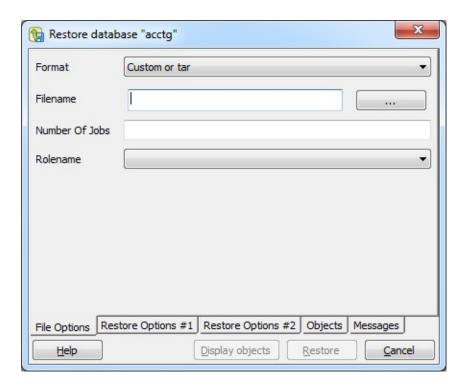
By selecting fields on the *Restore* dialog, you specify which options should be included in a customized pg_restore command. The pg_restore command plays back an archive that recreates the database, database object, or data described by commands within the archive.

If your archive is in a Custom, Tar, or Directory format, you can use the Restore dialog to restore from the archive. If you have saved the backup in Plain format, use the psql client to restore.

Please Note: If you are restoring into an existing database, you must ensure that any objects that might create conflicts because of pre-existing constraints or dependencies are dropped or truncated; use the *DROP CASCADE* or *TRUNCATE CASCADE* options on the context menu to clean up existing conflicts before performing a restore.

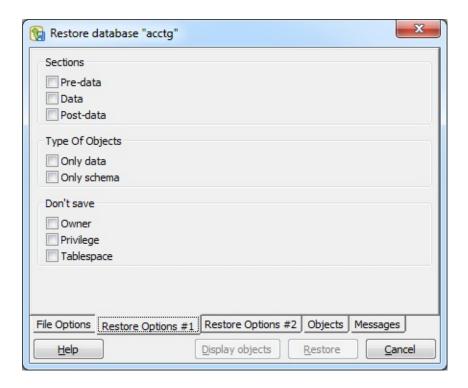
The Restore Dialog

To open the *Restore* dialog, right click on the name of an object in the tree control and select Restore from the context menu. The *Restore* dialog opens:



Use the fields on the *File Options* tab to general information about the backup archive. * Use the *Format* field to select the file format of the archive you are restoring. pgAdmin can restore from a Custom file (pg_dump format), a Tar file, or a Directory format file. * Use the *Filename* field to specify the name of the backup archive that will be used for the restore; optionally, use the file browser to navigate to and select the file. * Use the *Number of Jobs* field to instruct the server to use multiple concurrent jobs for the restore. The optimal value for this option depends on the hardware setup of the server, of the client, and of the network. This option is supported only by the Custom and Directory archive formats. * Use the *Rolename* field to specify the name of the role that will be used when invoking pg_restore.

When you've completed the *File Options* tab, navigate to the *Restore Options #1* tab:

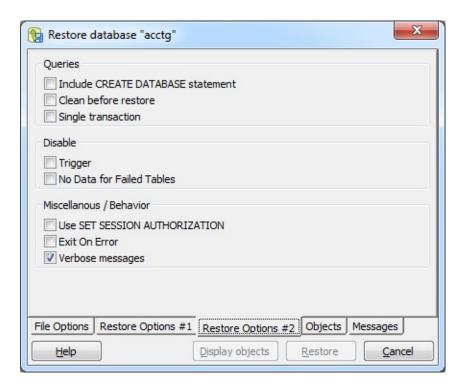


Use the fields on the *Restore Options #1* tab to specify details about the type of objects that will be restored.

- Use the checkboxes in the Sections box to select a portion of the object that will be restored. By default, a restore will include all sections.
 - Check the box next to *Pre-data* to include all data definition items not included in the data or post-data item lists.
 - Check the box next to Data to include actual table data, large-object contents, and sequence values.
 - Check the box next to Post-data to include definitions of indexes, triggers, rules, and constraints other than validated check constraints.
- Use the checkboxes in the Type of Objects box to select the objects that will be restored. By default, all objects will be included.
 - Check the box next to Only data to restore only the data.

- Check the box next to Only schema to restore only the schema (the data definitions).
- Use the checkboxes in the *Don't Save* box to select the objects that will not be included.
 - Check the box next to Owner to omit commands that set object ownership.
 - Check the box next to *Privilege* to omit commands that create access privileges.
 - Check the box next to *Tablespace* to omit tablespaces.

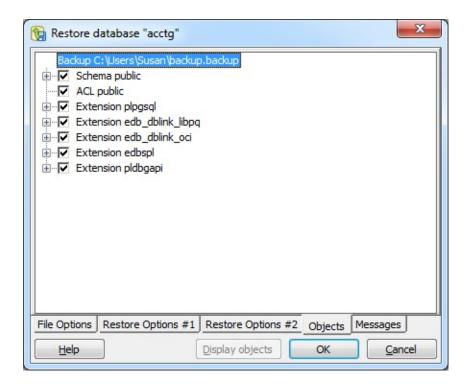
When you've completed the *Restore Options #1* tab, select the *Restore Options #2* tab:



- Use the checkboxes in the *Queries* box to specify how the restore process should handle statements.
 - Check the box next to *Include CREATE DATABASE* statement to include a command in the backup that creates a new database when restoring from the backup.
 - Check the box next to *Clean before restore* to instruct the server to drop an object before restoring it from the archive.

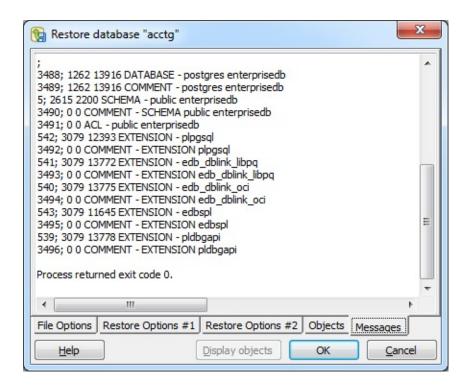
- Please note that this option does not remove all dependencies for all objects; manual cleanup may be required before restoring into an existing database.
- Check the box next to Single transaction to execute the restore as a single transaction; this ensures that all commands will complete successfully before the changes are applied.
- Use the checkboxes in the *Disable* box to specify trigger and table loading preferences.
 - Check the box next to *Trigger* (when creating a data-only backup) to include commands that will disable triggers on the target table while the data is being loaded.
 - Check the box next to No Data for Failed Tables to instruct pg_restore to not load table data if the create command for a table fails. Specifying this option prevents duplicate or obsolete data from being loaded into an existing table.
- Use the checkboxes in the *Miscellaneous Behavior* box to specify additional restore options.
 - Check the box next to Use SET SESSION AUTHORIZATION to use SQL-standard SET SESSION AUTHORIZATION commands instead of ALTER OWNER commands to determine object ownership.
 - Check the box next to Exit on Error to exit the restore if an error is encountered while sending SQL commands to the database.
 - Check the box next to Verbose messages to instruct pg_restore to use verbose messages.

When you've completed the *Restore Options #2* tab, navigate to the Objects tab:



Click the *Display objects* button to populate the tree control on the *Objects* tab; when the list of objects is displayed, check the box to the left of an object name to include that object from the restore.

When you've completed the tabs, navigate to the *Messages* tab, and press *OK*.



When the restoration completes, the *Messages* tab displays details about the restoration process:

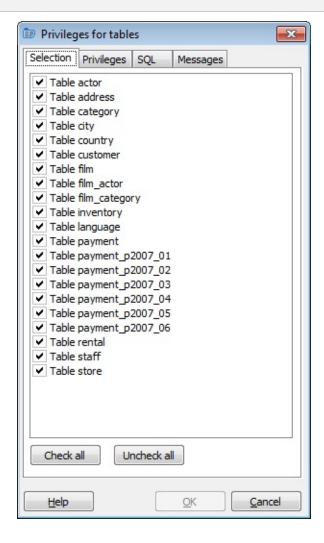
If the restore was successful, the Messages tab will display:

Process returned exit code 0.

If you receive an exit code other than 0, scroll through the *Messages* window to locate the problem; after correcting the problem, you can repeat the process.

Scroll to the top of the Messages dialog to view the executed pg_restore command. When you're finished, click *Done* to exit the *Restore* dialog.

Grant Wizard



The Grant Wizard allows assignment of a set of privileges to groups and users to multiple objects (tables, sequences, views and functions) in a convenient way.

The Selection tab will show all applicable objects under the currently selected object. Check all objects you like to grant privileges on.

The security tab defines the privileges and groups/users that should be granted. Assigning no privileges to a group or user will result in revoking all rights for them.

The SQL tab shows the sql commands that are generated from the

selected grant targets.

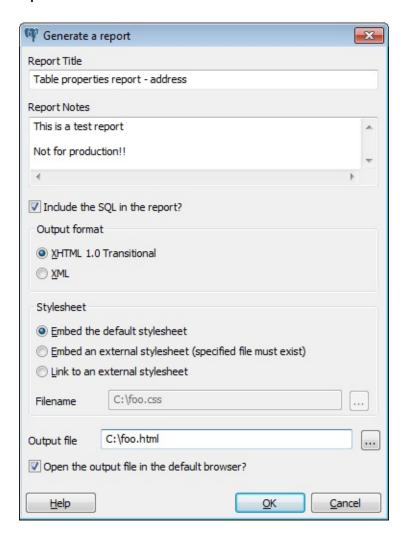
When clicking OK, the sql commands will be executed; any result will be displayed in the Messages tab.

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III »

previous | next | index

Report Tool

pgAdmin includes a simple reporting engine which allows you to quickly generate reports from the queries you write in the Query Tool, or from objects or collections of objects in the main application Window - for example, you can create a report of the properties of any object, or a list of functions in a schema. To create a report, select a node in the browser treeview, and select the report to create from the context menu, or from the Reports submenu of the Tools menu. To create a report in the Query Tool, select the Quick Report option from the File menu.

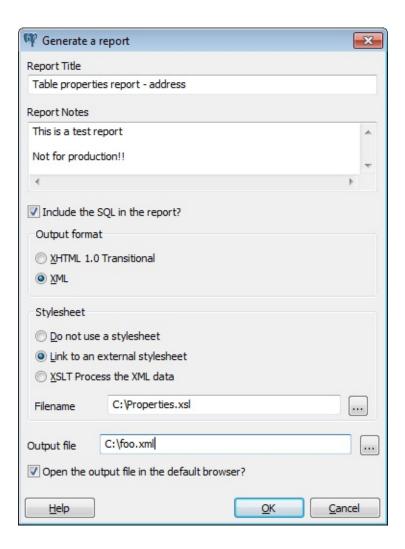


Each report contains a title and optional notes that can be modified

before the report is produced. In addition, you may select whether or not to include any SQL that may be relevant to the report you have selected.

Reports are generated internally in XML, however pgAdmin can apply an XML stylesheet to the report at processing time to format the output as required. A default stylesheet is built-in which produces HTML output. With the HTML output option selected, you can opt to embed the default CSS stylesheet (which will render the report in the same colors as the pgAdmin website), to embed an external stylesheet into the report, or to link to an external stylesheet. The following class/object ID's are used:

- #ReportHeader: This div contains the report header.
- #ReportNotes: This div contains the option report notes.
- #ReportDetails: This div contains the main body of the report.
- #ReportFooter: This div contains the report footer.
- .ReportSQL: This class is used by the <PRE></PRE> blocks containing SQL.
- .ReportDetailsOddDataRow: This class is applied to the odd numbered rows of tables.
- .ReportDetailsEvenDataRow: This class is applied to the even numbered rows of tables.
- ReportTableHeaderCell: This class is applied to table header cells.
- .ReportTableValueCell: This class is applied to table data cells.
- .ReportTableInfo: This class is applied to table footnotes.



When generating reports in XML format, you can opt to output plain XML, XML linked to an external XSL stylesheet, or to process the XML using an external stylesheet and save the resulting output. This allows complete flexibility to format reports in any way.

The default XSL stylesheet used to render XHTML output can be used as a starting point for your own, and sample XML data may also be reviewed if required.

Contents:

- Default XSL Stylesheet
 - Sample XML Data

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Report Toolprevious | next | index

×

Default XSL Stylesheet

For reference, and as a starting point for your own stylesheets, the builtin stylesheet that pgadmin uses to render XHTML report output from XML report data is included below. This stylesheet includes the default HTML stylesheet that will be embedded into a report to give it the pgadmin look and feel.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999</pre>
<xsl:output method="xml" doctype-system="http://www.w3.org/TR/x</pre>
<xsl:template match="/report">
<html>
  <head>
    <xsl:if test="header/title != ''">
      <title><xsl:value-of select="header/title" /></title>
    </xsl:if>
    <meta http-equiv="Content-Type" content="text/html; charset</pre>
    <style type="text/css">
      body { font-family: verdana, helvetica, sans-serif; marq
      h1 { font-weight: bold; font-size: 150%; border-bottom-st
      h2 { font-size: 130%; padding-bottom: 0.5ex; color: #4767
      h3 { font-size: 110%; padding-bottom: 0.5ex; color: #0000
      th { text-align: left; background-color: #47678E; color:
      #ReportHeader { padding: 10px; background-color: #47678E;
      #ReportHeader th { width: 25%; white-space: nowrap; verti
      #ReportHeader td { vertical-align: top; color: #EEEEEE; }
      #ReportNotes { padding: 10px; background-color: #EEEEEE;
      .ReportSQL { margin-bottom: 10px; padding: 10px; display:
      #ReportDetails { margin-left: 10px; margin-right: 10px; m
      #ReportDetails td, th { font-size: 80%; margin-left: 2px;
      #ReportDetails th { border-bottom-color: #777777; border-
      .ReportDetailsOddDataRow { background-color: #dddddd; }
      .ReportDetailsEvenDataRow { background-color: #eeeeee; }
      .ReportTableHeaderCell { background-color: #dddddd; color
      .ReportTableValueCell { vertical-align: top; font-size: 8
      .ReportTableInfo { font-size: 80%; font-style: italic; }
      #ReportFooter { font-weight: bold; font-size: 80%; text-a
      #ReportFooter a { color: #EBA525; text-decoration: none;
    </style>
  </head>
```

```
<body>
  <div id="ReportHeader">
  <xsl:if test="header/title != ''">
     <h1><xsl:value-of select="header/title" /></h1>
  </xsl:if>
  <xsl:if test="header/generated != ''">
     <b>Generated: </b><xsl:value-of select="header/generated"</pre>
  </xsl:if>
  <xsl:if test="header/server != ''">
     <b>Server: </b><xsl:value-of select="header/server" /><br</pre>
  </xsl:if>
  <xsl:if test="header/database != ''">
     <b>Database: </b><xsl:value-of select="header/database" /</pre>
  </xsl:if>
  <xsl:if test="header/catalog != ''">
     <b>Catalog: </b><xsl:value-of select="header/catalog" /><</pre>
  </xsl:if>
  <xsl:if test="header/schema != ''">
     <b>Schema: <xsl:value-of select="header/schema" /><br</pre>
  </xsl:if>
  <xsl:if test="header/table != ''">
     <b>Table: <b>Table: />/>/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////p/////////////////////////////////////////////////////////////////////////////
  </xsl:if>
  <xsl:if test="header/job != ''">
     <b>Job: </b><xsl:value-of select="header/job" /><br />
  </xsl:if>
  </div>
  <xsl:if test="header/notes != ''">
     <div id="ReportNotes">
     <b>Notes: </b><br /><br />
     <xsl:call-template name="substitute">
         <xsl:with-param name="string" select="header/notes" />
     </xsl:call-template>
     </div>
  </xsl:if>
  <div id="ReportDetails">
```

```
<xsl:apply-templates select="section" >
        <xsl:sort select="@number" data-type="number" order="as</pre>
      </xsl:apply-templates>
    </div>
    <div id="ReportFooter">
Report generated by <a href="http://www.enterprisedb.com/">Post
    </div>
    <br />
  </body>
</html>
</xsl:template>
<xsl:template match="section">
  <xsl:if test="../section[@id = current()/@id]/@name != ''">
    <h2><xsl:value-of select="../section[@id = current()/@id]/@
  </xsl:if>
  <xsl:if test="count(../section[@id = current()/@id]/table/col
    <div style=overflow:auto>
      <xsl:apply-templates select="../section[@id = current</pre>
            <xsl:sort select="@number" data-type="number" order</pre>
            <xsl:with-param name="count" select="count(../secti</pre>
          </xsl:apply-templates>
        <xsl:apply-templates select="../section[@id = current()</pre>
          <xsl:sort select="@number" data-type="number" order="</pre>
          <xsl:with-param name="column-meta" select="../section</pre>
        </xsl:apply-templates>
     </div>
    <br />
    <xsl:if test="../section[@id = current()/@id]/table/info !=</pre>
      <xsl:value-of select="../secti</pre>
    </xsl:if>
  </xsl:if>
  <xsl:if test="../section[@id = current()/@id]/sql != ''">
    <xsl:call-template name="substitute">
         <xsl:with-param name="string" select="../section[@id =</pre>
      </xsl:call-template>
```

```
</xsl:if>
</xsl:template>
<xsl:template match="column">
  <xsl:param name="count" />
  <xsl:attribute name="width"><xsl:value-of select="100 div $co</pre>
    <xsl:call-template name="substitute">
      <xsl:with-param name="string" select="@name" />
   </xsl:call-template>
  </xsl:template>
<xsl:template match="*" mode="rows">
  <xsl:param name="column-meta" />
  <xsl:choose>
  <xsl:when test="position() mod 2 != 1">
    <xsl:attribute name="class">ReportDetailsOddDataRow</xsl:at</pre>
  </xsl:when>
  <xsl:otherwise>
    <xsl:attribute name="class">ReportDetailsEvenDataRow</xsl:a</pre>
  </xsl:otherwise>
  </xsl:choose>
   <xsl:apply-templates select="$column-meta" mode="cells">
     <xsl:with-param name="row" select="." />
   </xsl:apply-templates>
  </xsl:template>
<xsl:template match="*" mode="cells">
  <xsl:param name="row" />
    <xsl:choose>
     <xsl:when test="$row/@*[name() = current()/@id]|$row/*[na
       <xsl:call-template name="substitute">
         <xsl:with-param name="string" select="$row/@*[name()</pre>
       </xsl:call-template>
     </xsl:when>
     <xsl:otherwise>
       <xsl:text> </xsl:text>
     </xsl:otherwise>
   </xsl:choose>
  </td>
</xsl:template>
<xsl:template name="substitute">
```

```
<xsl:param name="string" />
   <xsl:param name="from" select="'</pre>
" />
   <xsl:param name="to">
      <br />
   </xsl:param>
   <xsl:choose>
      <xsl:when test="contains($string, $from)">
         <xsl:value-of select="substring-before($string, $from)</pre>
         <xsl:copy-of select="$to" />
         <xsl:call-template name="substitute">
            <xsl:with-param name="string" select="substring-aft</pre>
            <xsl:with-param name="from" select="$from" />
            <xsl:with-param name="to" select="$to" />
         </xsl:call-template>
      </xsl:when>
      <xsl:otherwise>
         <xsl:value-of select="$string" />
      </xsl:otherwise>
   </xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

Sample XML Data

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Report Toolprevious | next | index

>>

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Report Toolprevious | next | index » Default XSL Stylesheet »

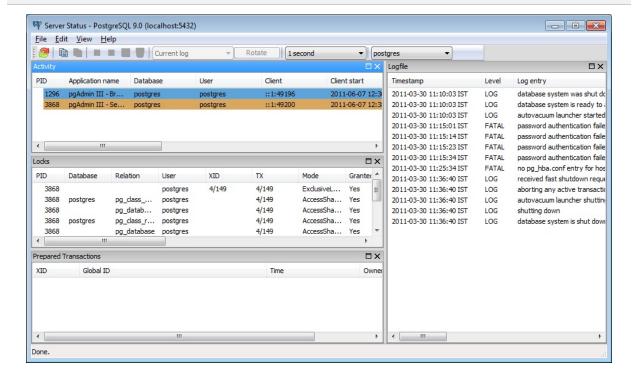
Sample XML Data

The sample report below demonstrates the XML format that the pgadmin Report Tool will produce.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="C:\stylesheet.xsl" ?>
<report>
 <header>
   <!-- All the values in the header section are optional, apa
   <title>Sample XML file</title>
   <notes>This is a sample XML file.</notes>
   <generated>10/05/2006 11:21:23
   <server>localhost:5432</database>
   <database>postgres</database>
   <schema>public</schema>
   pg_ts_cfg
   <job></job>
 </header>
 <!-- Multiple report sections may be present, and must have a
  <section id="s1" number="1" name="Query results">
   <!-- A table is optional within a section. If present, the
   <!-- of columns should be appropriate for the attributes sp
   <!-- in each row. A row needn't specify every value however
   <columns>
       <column id="c1" number="1" name="oid" />
       <column id="c2" number="2" name="ts_name" />
       <column id="c3" number="3" name="prs_name" />
       <column id="c4" number="4" name="locale" />
     </columns>
     <rows>
       <row id="r1" number="1" c1="17108" c2="default" c3="def</pre>
       <row id="r2" number="2" c1="17109" c2="default russian"</pre>
       <row id="r3" number="3" c1="17110" c2="simple" c3="defa</pre>
     </rows>
```

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » Report Toolprevious | next | index » Default XSL Stylesheet »

Database Server Status



The Server Status dialogue displays the current connections to each database, the user that is connected, the process ID, the client address and start time (on PostgreSQL 8.1 and above), the current query being executed (if any), and the query start time (where appropriate) on PostgreSQL 7.4 and above on the Status tab.

The Cancel button allows cancelling the query running on a specific backend. Terminate will shutdown the backend. **Attention:** Both functions should be used carefully, as the interrupt the client's work ungracefully. Particularly, the terminate function might disturb the function of the complete server, force it to restart its services and thus interrupt all user connections. You should use this function only if the server is seriously injured by a backend you can't control otherwise.

The Locks tab shows the current locks outstanding in the PostgreSQL Lock Manager. This information can be useful when

attempting to debug or track down deadlocks on your server. Not all information is necessarily shown for each lock. In particular, the Relation name may be shown as an OID instead of by it's name, if the relation is in a different database to that being monitored.

Note: When the pg_locks view is accessed as is the case whenever this dialgue is open, PostgreSQL's internal lock manager data structures are momentarily locked, and a copy is made for the dialogue to display. This ensures that the dialogue displays a consistent set of results, while not blocking normal lock manager operations longer than necessary. Nonetheless there could be some impact on database performance if this view is read often.

On a PostgreSQL server running version 8.1 or newer, the Transaction tab allows you to view outstanding prepared transactions. Prepared transactions are an aspect of Two Phase Commit (2PC), used in distributed transaction managers. Usually, prepared transactions are handled by the transaction manager. In case of a failure, it might be necessary to commit or rollback a transaction manually; you can use the 'Commit' or 'Rollback' buttons to do this.

The Logfile tab shows server log files, if configured in postgresql.conf (redirect_stderr or logging_collector = true, log_destination = 'stderr' and log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log' on PostgreSQL or 'enterprisedb-%Y-%m-%d_%H%M%S.log' on EnterpriseDB's Advanced Server). pgAdmin will extract a time stamp from the logfile in a separate column, if the log_line_prefix is configured accordingly. We recommend using '%t:' as format, because more complicate formats might not be interpretable correctly.

The combobox allows you to select historic logfiles or the current one. If "current" is selected, pgAdmin will correctly detect logfile rotation and continue to display them.

The "Rotate" button will force the server to rotate its server logfile. This function is currently not implemented on 8.0 servers; if you think this is valuable for you please contact us.

Please note that displaying the logfile requires additional functions loaded on the server side, which are available for 8.x servers only.

To refresh the display click the *Refresh* button. The display will also be automatically refreshed based on the refresh interval specified. Note that you have one refresh rate per tab.

You can hide panes by clicking on their close button or by clicking on the appropriate menu item in the View menu.

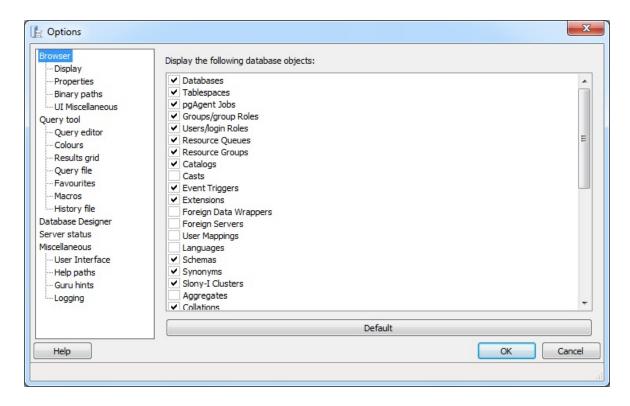
You can also copy some lines on the tabs' list. Select the lines you want to copy and click on the Copy button of the toolbar.

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III »

previous | next | index

pgAdmin Options

pgAdmin has a selection of configuration options that you can use to personalize your workspace. pgAdmin is usually configured by using the *Options* dialog from within the pgAdmin application; to open the *Options* dialogue, select *Options* from the *File* menu.



The navigation panel on the *Options* dialog contains a tree control; each node of the tree control provides access to customizable options that are related to the selected node. For information about the configurable options on each node of the pgAdmin tree control, please select from the links shown below.

Contents:

- pgAdmin Browser Options
- pgAdmin Query tool Options
- pgAdmin Database Designer Options

- pgAdmin Server Status Options
- pgAdmin Miscellaneous Options

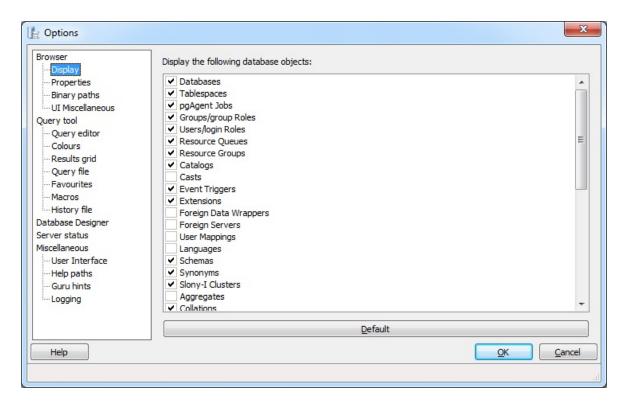
pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III »

previous | next | index

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » pgAdmin previous | next | index Options »

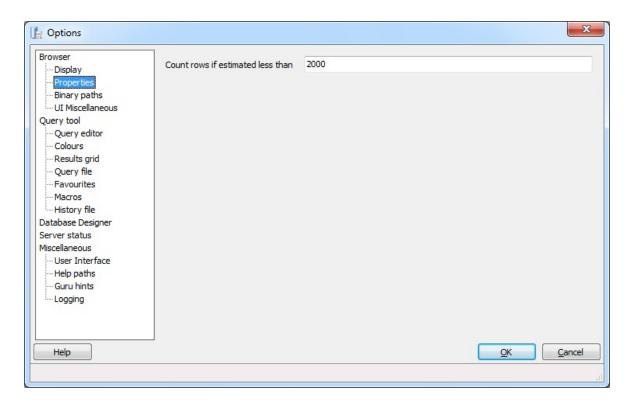
pgAdmin Browser Options

Use the options that are located under the *Browser* node of the tree control to personalize your workspace.



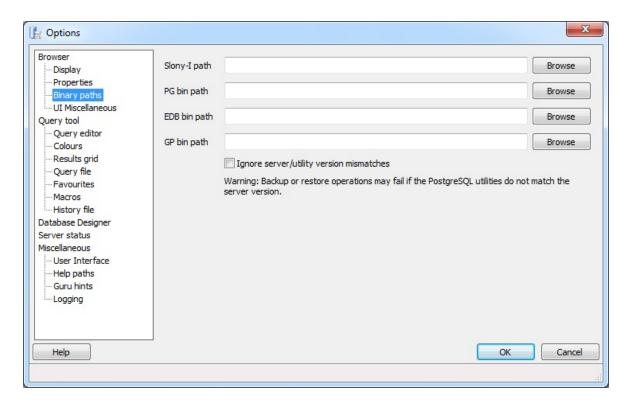
Use the options on the *Display* dialog to specify general display preferences:

- **Display the following database objects** Check the box next to a database object type to instruct pgAdmin to display the selected object type in the pgAdmin tree control. By default, pgAdmin displays only the most commonly used object types.
- Default You can reduce the number of object types displayed to increase the speed of pgAdmin when querying the system catalogs. Use the *Default* button to reset the list to its default settings.



Use the options on the *Properties* dialog to specify display properties:

• Count rows if estimated less than - Include a value in the Count rows if estimated less than field to perform a SELECT count(*) if the estimated number of rows in a table (as read from the table statistics) is below the specified limit. After performing the SELECT count(*), pgAdmin will display the row count.

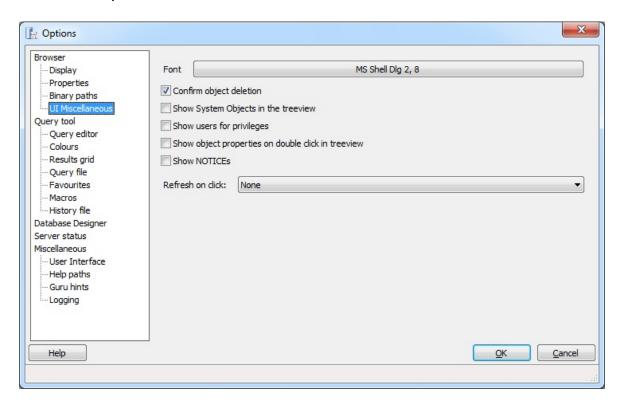


Use the options on the *Binary paths* dialog to specify binary file locations:

- **Slony-I path** To enable pgAdmin to create Slony-I replication clusters from scratch, you must provide the path to the Slony-I script installation directory in this field. If no valid scripts can be found, you will only be allowed to join existing clusters.
- PG bin path Use this field to specify the path to the standard PostgreSQL pg_dump, pg_restore and pg_dumpall utilities. pgAdmin will use these utilities when backing up and restoring your PostgreSQL databases. If this path is not set, pgAdmin will attempt to locate the utilities in its installation directory, the 'hint' directory set by a local PostgreSQL installation (on Windows) and in the system path.
- EDB bin path Use this field to specify the path to the EnterpriseDB pg_dump, pg_restore and pg_dumpall utilities. pgAdmin will use these utilities when backing up and restoring your EnterpriseDB databases. If this path is not set, pgAdmin will attempt to find the utilities in standard locations used by

EnterpriseDB.

• **GP bin path** - Use this field to specify the path to the Greenplum pg_dump, pg_restore and pg_dumpall utilities. pgAdmin will use these utilities when backing up and restoring your Greenplum databases. If this path is not set, pgAdmin will attempt to find the utilities in standard locations used by Greenplum.



Use the options on the *UI Miscellaneous* dialog to specify miscellaneous pgAdmin preferences:

- **Font** Use the *Font* selector to specify the font used as the standard font throughout pgAdmin's main user interface. This may be useful for countries with multibyte character sets (where standard fonts may not display well), or if you wish to use a font different from the one configured on your system.
- Confirm object deletion Select this option to instruct pgAdmin to require confirmation before deleting an object

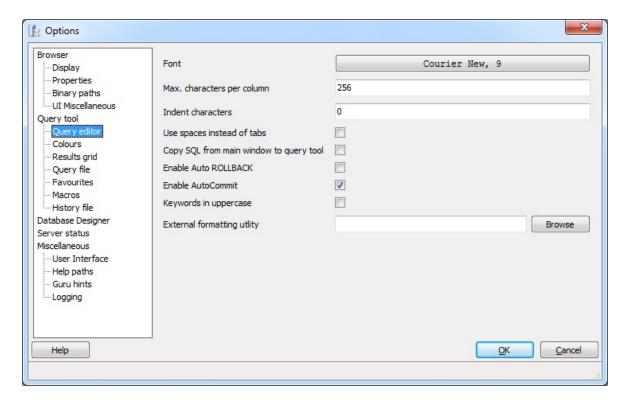
(recommended).

- Show System Objects in the treeview Check the box next to Show System Objects in the treeview to instruct pgAdmin to display objects such as system schemas (e.g. pg_temp*) and system columns (e.g. xmin, ctid) in the tree control.
- Show users for privileges Select this option to include usernames as well as groups in the subject list when viewing the *Privileges* tab of an object that supports privileges. Disabling this option is useful on large systems with many users that are organised in groups.
- Show object properties on double click in treeview Select this option to specify that double clicking an object in the treeview should open the *Properties* dialogue for that object. Disable this option to browse the treeview using a double-click.
- **Show NOTICEs** Check the box next to *Show NOTICEs* to instruct pgAdmin to display notices from the server.
- Refresh on click Use the Refresh on click drop-down listbox to specify that the displayed properties of the selected objects should be updated in the tree control when the object is selected. Select from:
 - None to update no objects.
 - Refresh object on click to refresh the selected object.
 - Refresh object and children on click to refresh the selected object, and any objects that reside under the selected object in the tree control.

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III » pgAdmin previous | next | index Options »

pgAdmin Query tool Options

Use the options that are located under the *Query tool* node of the tree control to personalize the behavior of the Query tool.



Use the fields on the *Query editor* dialog to specify workspace preferences for the Query editor:

- **Font** Use the *Font* selector to specify the font used in the Query tool interface.
- Max characters per column Use the *Max characters per column* field to specify the maximum length of data displayed in a single column.
- **Indent characters** Use the *Indent* characters field to specify the number of characters to indent with in SQL text boxes.
- Use spaces instead of tabs Check the box next to Use

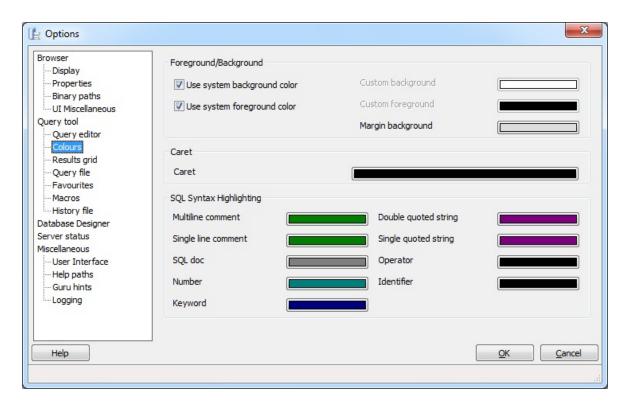
spaces instead of tabs to instruct pgAdmin to insert a space when you press the tab key in an SQL text box.

- Copy SQL from main window to SQL dialogue Check the box next to Copy SQL from main window to SQL dialogue to instruct pgAdmin to copy any object definition displayed on the main object browser for use as a default query in the Query Tool when the Query Tool opens.
- Enable Auto ROLLBACK Check the box next to Enable Auto ROLLBACK to instruct the query tool to execute a ROLLBACK if a query fails.
- **Enable Auto COMMIT** Check the box next to *Enable Auto COMMIT* to instruct the query tool to enable auto commit after executing the query.
- **Keywords in uppercase** Check the box next to *Keywords in uppercase* to instruct the Query tool to convert any keywords entered to an uppercase font.
- External formatting utility Use the External formatting utility field to provide the name of a formatting tool that can be used to make queries more readable. Use of a tool that provides support for UTF-8 multi-byte characters is not required, but is preferable. The utility must:
 - Accept a STDIN stream and write the result set to STDOUT
 - Provide a result in less than 3 seconds
 - Exit with a code 0 on success

To use an external formatting tool in the Query Editor:

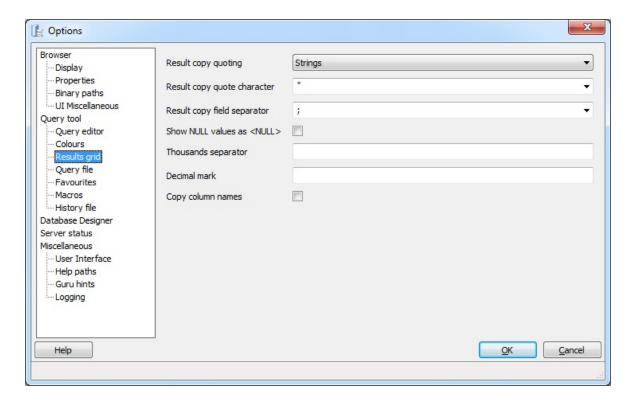
- Provide the name of the formatting tool in the *External* formatting utility field of the *Options* dialogue.
- In the Query Editor, select a block of text for formatting, and

Press Ctrl+Shift+f, or navigate through the *Edit* menu to the *Format* menu, and select *External Format*.



Use the fields displayed on the *Colours* dialog to select colours for items displayed in the Query tool:

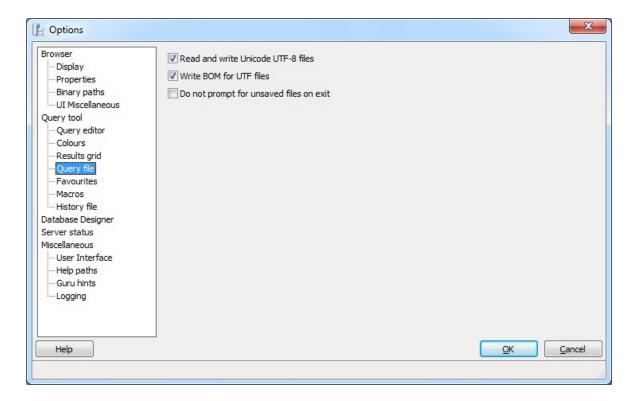
- Foreground/Background Use the color selectors in the Foreground/Background box to specify background and foreground colors for the Query tool.
- Caret Use the color selector in the Caret box to specify a color for the caret (or cursor).
- SQL Syntax Highlighting Use the color selectors in the SQL Syntax Highlighting box to specify display colors for syntax components.



Use the fields on the *Results Grid* dialog to specify display preferences for the results grid.

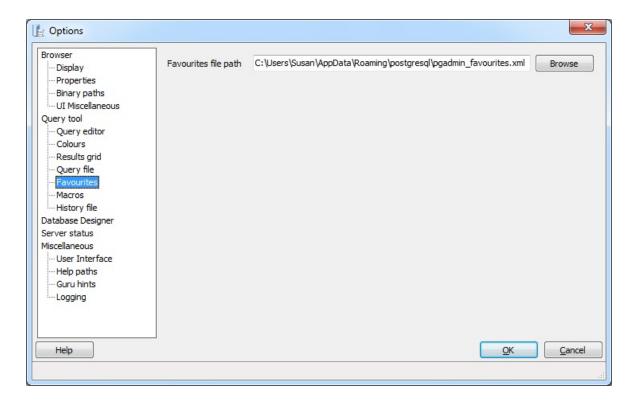
- Result copy quoting Use the Result copy quoting drop-down listbox to specify how fields in a result set should be quoted when copied to the clipboard from the result list. Please Note Disabling this option may prevent copying and pasting rows in the edit grid from functioning properly if the result set contains special characters like line breaks. Specify:
 - None to omit quotes when copied.
 - Strings to quote string values when copied (numeric columns will not be quoted).
 - All to quote all copied values (regardless of data type).
- Result copy quote character Use the Result copy quote character drop-down listbox to select a character that will be used to quote the values when copied to the clipboard from the result list. It has no effect if Result copy quoting is set to None.

- Result copy field separator Use the Result copy field separator drop-down listbox to select a character that will be used to separate the fields copied to the clipboard from the result list.
- Show NULL values as <NULL> Check the box next to Show NULL values as <NULL> to cause NULL values to be shown as <NULL> in the Query Tool's results grid. Note that this option does not affect data that is exported from the grid, only what is displayed.
- **Thousands separator** Use the *Thousands separator* field to specify a delimiter character that will be used to break a value into groups of three digits, counting left from the decimal mark.
- **Decimal mark** Use the *Decimal mark* field to specify a delimiter character that will be used to break a number into an integer portion and a fractional portion.
- **Copy column names** Check the box next to *Copy column names* to specify that when copying from the result grid, the column name should be included as the first row.



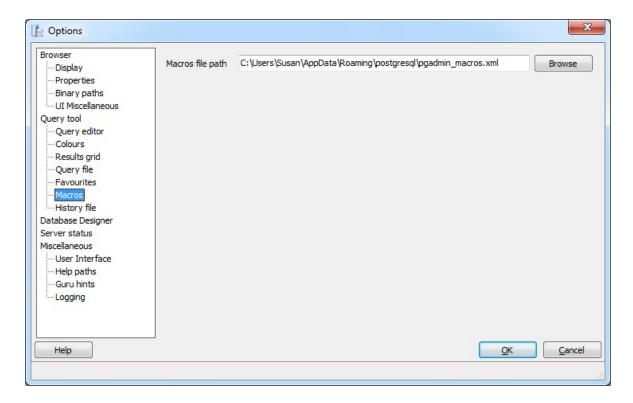
Use the fields on the *Query File* dialog to specify UTF-8 details, and file saving preferences:

- Read and write Unicode UTF-8 files Check the box next to Read and write Unicode UTF-8 files to write Unicode UTF-8 files by default.
- Write BOM for UTF files Check the box next to Write BOM for UTF files to instruct the query tool to write a byte order mark at the beginning of each file written.
- **Do not prompt for unsaved files on exit** Check the box next to *Do not prompt for unsaved files on exit* to prevent pgAdmin from prompting you to save SQL queries upon exit.



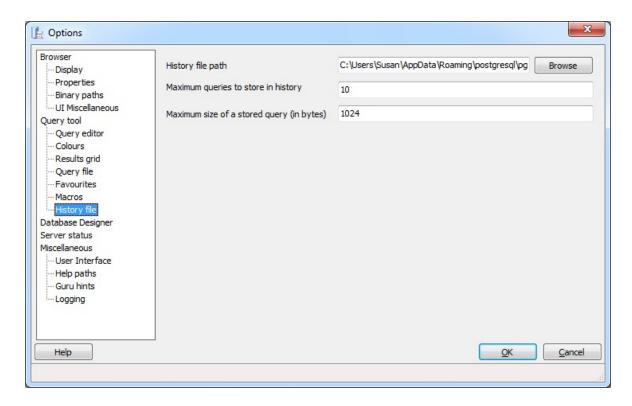
Use the fields on the *Favourites* dialog to specify a location for the Query tool's *Favourites* file:

• Favourites file path - The Favourites file path field specifies the path to the Query tool's Favourites file. The default file name is pgadmin_favourites.xml; you can use the Browse button to open a file browser and modify the location of the file.



Use the fields on the *Macros* dialog to specify a location for the Query tool's *Macros* file:

• Macros file path - The Macros file path field specifies the path to the Query tool's Macros file. The default file name is pgadmin_macros.xml; you can use the Browse button to open a file browser and modify the location of the file.

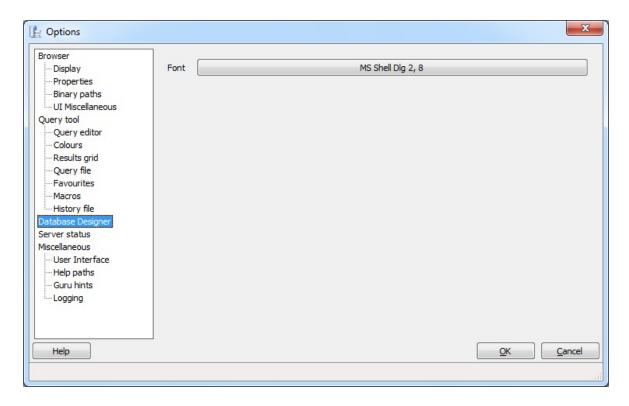


Use the fields on the *History File* dialog to specify user preferences for the Query tool's history file:

- History file path The History file path specifies the path to the Query tool's history file. The default file name is pgadmin_histoqueries.xml; you can use the Browse button to open a file browser and modify the location of the file.
- Maximum queries to store in history Use the *Maximum* queries to store in history field to specify the number of queries stored by pgAdmin. When the maximum number of queries is reached, the oldest query will be deleted, and a new query will be stored.
- Maximum size of a stored query (in bytes) Use the
 Maximum size of a stored query (in bytes) field to specify the
 size (in bytes) of the largest query that will be stored in the
 history file; any query that exceeds the specified size will not be
 saved in the history file.

pgAdmin Database Designer Options

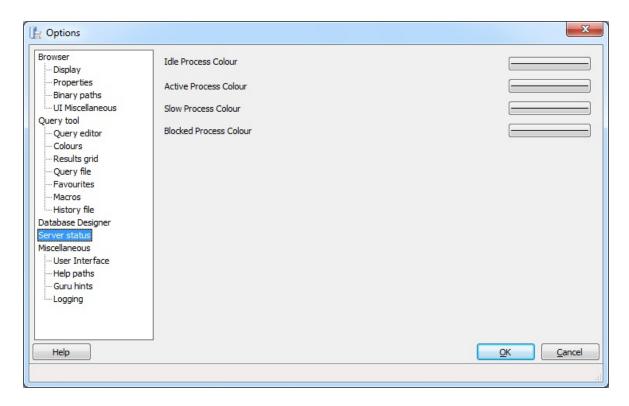
Use the options that are located under the *Database Designer* node of the tree control to specify options for the Database Designer tool.



• Font - Open the font selector to specify the fonts displayed within the Database Designer graphical interface.

pgAdmin Server Status Options

Use the options that are located under the *Server Status* node of the tree control to customize server status display preferences.



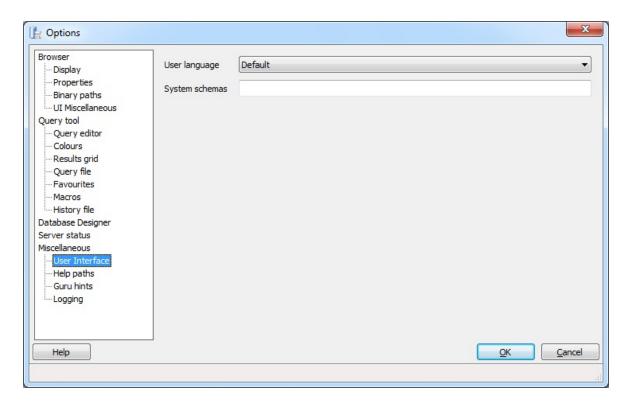
Use the fields on the *Server Status* dialog to specify the colors displayed to indicate the status of a server:

- Idle Process Colour Use the color selector to specify the display color of an idle process. A process is considered idle when the server is waiting for a new command.
- Active Process Colour Use the color selector to specify the display color of an active process. A process is considered active when the server is executing a query.
- **Slow Process Colour** Use the color selector to specify the display color of a slow process. A process is considered slow when a query exceeds 10 seconds.
- Blocked Process Colour Use the color selector to specify the

display color of a blocked process. A process is considered blocked an active query is waiting because of a lock.

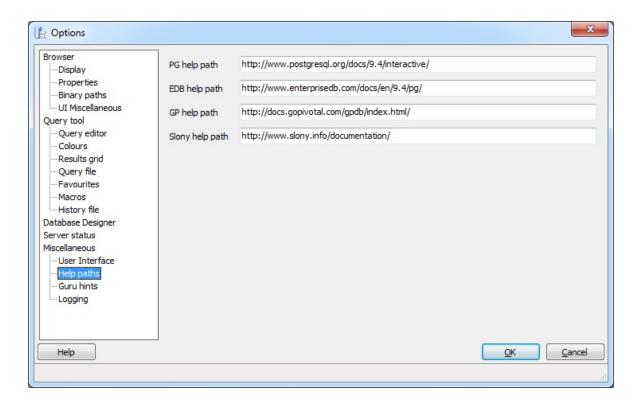
pgAdmin Miscellaneous Options

Use the options that are located under the *Miscellaneous* node of the tree control to personalize the behavior of your workspace.



Use the fields on the *User Interface* dialog to specify user interface preferences:

- **User language** Use the *User language* drop-down listbox to select the language displayed in the pgAdmin user interface.
- System schemas pgAdmin will consider any schema (or schemas) listed in the System schemas field to be a system schema. To specify that a schema is a system schema, include the schema name in a list of comma-separated strings in the System schemas field.

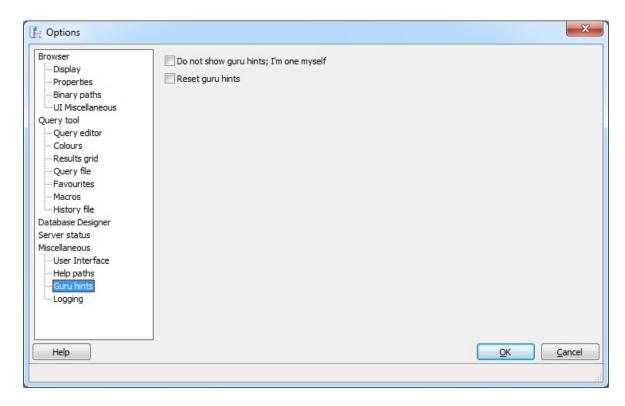


Use the fields on the *Help paths* dialog to specify the locations of help files. The path specified may be a URL or directory containing the HTML format help files (note that some browsers may require file:///path/to/local/files/ notation, while others require /path/to/local/files/ notation), or the path to a Compiled HTML Help (.chm) file (on Windows), an HTML Help project (.hhp) file, or a Zip archive containing the HTML files and the HTML Help project file.

- PG help path Use the PG help path field to specify the path to the PostgreSQL help files. PostgreSQL Help files are displayed when PostgreSQL HELP is selected from the Help menu, or when accessing the help files through many database objects on PostgreSQL servers.
- EDB help path Use the EDB help path option to specify the path to the EnterpriseDB help files. EnterpriseDB Help files are displayed when EnterpriseDB Help is selected from the Help menu, or when accessing the help files through many database objects on EnterpriseDB servers.
- **GP help path** Use the *GP help path* option to specify the path

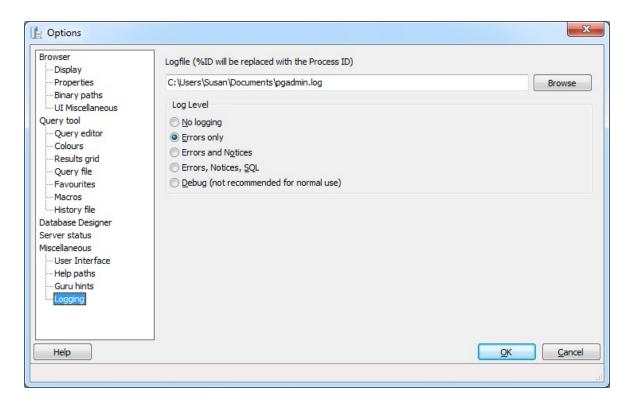
to the Greenplum help files. Greenplum Help files are displayed when *Greenplum Database Help* is selected from the *Help* menu, or when accessing the help files through many database objects on Greenplum servers.

• **Slony help path** - Use the *Slony help path* option to specify the path to the Slony help files. Slony Help files are displayed when *Slony Help* is selected from the *Help* menu.



By default, if pgAdmin suspects you might need support, it displays a guru hint window. Use the fields on the *Guru hints* dialog to suppress or display guru hints:

- **Do not show guru hints; I'm one myself** Check the box next to *Do not show guru hints; I'm one myself* to suppress this behaviour.
- Reset guru hints Check the box next to Reset guru hints to instruct pgAdmin to display all guru hints until suppressed again.



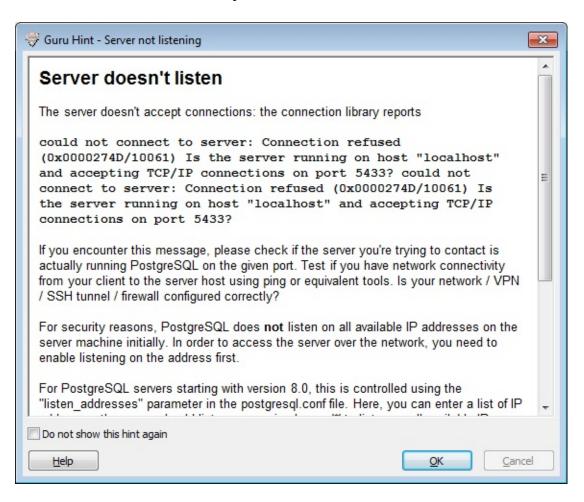
Use the fields on the *Logging* dialog to specify the location and content of pgAdmin log files:

- **Logfile** Select this option to specify the name of the file in which pgAdmin will store any logging data. If you include %ID in the specified file name, it will be replaced with the process ID, allowing multiple instances of pgAdmin to run with logging to individual files.
- No logging No logging will be performed at all.
- Errors only pgAdmin will log errors to the log file, but nothing else.
- Errors and Notices pgAdmin will log errors and notice messages from PostgreSQL to the log file, but nothing else.
- Errors, Notices, SQL Error messages, notice messages from PostgreSQL and SQL queries will be logged to the log file.
- Debug Error messages, notice messages from PostgreSQL, SQL queries and debug messages will be logged to the log file. This option may significantly slow down pgAdmin, and is therefore not recommended for normal use.

Guru Hints

pgAdmin III has some functions included which should help to implement healthy and good performing databases. They are the result of the long experience of the developers worldwide, and the design problems commonly observed.

If pgAdmin III thinks it has a valuable hint for you, it will pop up a guru hint unless you have chosen to suppress that individual hint or all hints (see options). You can bring up the hint by pressing the hint button in the toolbar at any time, if enabled.



This is an example of a guru hint, as it appears when trying to connect to a non-running or misconfigured PostgreSQL server.

If you check 'do not show this hint again', the automatic popup of the individual hint will be suppressed in the future, until it is reset in the options. Error messages will still appear, but without the extended information as provided by the hints.

pgAdmin3 LTS 1.23.0a documentation » Using pgAdmin III »

previous | next | index

Command Line Parameters

There are a number of command line options that may be passed to the pgAdmin executable when starting it.

- -h Display command line option help.
- -s <server description> Auto connect to the specified server.
 <server description> is the description entered when you added the server to the treeview in pgAdmin, quoted if required. If multiple servers are registered with the same description, the first found will be auto-connected.
- -S Open a Server Status window upon connection. If used together with the -s <server description> option, the server status window will automatically be connected to this server. If used alone, a dialog box will appear prompting for the connection information. This switch cannot be used together with the -Sc <connection string> option.
- -Sc <connection string> Open a Server Status window upon connection. The server status window will automatically be connected to the server specified in the connection string. This option cannot be used together with the -S switch.
 - **NOTE:** The "password" connection string option will not be recognised for security reasons. Please use a PGPASSFILE if you need to store the connection password.
- -q Open a Query Tool window upon connection. If used together with the -s <server description> option, the query tool will automatically be connected to this server. If used alone, a dialog box will appear prompting for the connection information. This switch cannot be used together with the -qc <connection string> option.

• -qc <connection string> Open a Query Tool window upon connection. The query tool will automatically be connected to the server specified in the connection string. This option cannot be used together with the -q switch.

NOTE: The "password" connection string option will not be recognised for security reasons. Please use a PGPASSFILE if you need to store the connection password.

- -f <SQL script> Automatically open the specified SQL script in the Query Tool. Only useful with -q or -qc.
- -ch <hba file> Edit the specified pg_hba.conf config file.
- -cm <config file> Edit the specified postgresql.conf config file.
- -cp <pgpass file> Edit the specified .pgpass/pgpass.conf config file.
- -c <dir> Edit the config files in specified cluster directory.
- **-t** Start in dialogue translation test mode. This mode allows each dialogue to be viewed without running the application to assist with string translation.

pgAgent

Introduced in pgAdmin III v1.4, pgAgent is a job scheduling agent for PostgreSQL, capable of running multi-step batch/shell and SQL tasks on complex schedules.

From pgAdmin 1.9 onwards, pgAgent is distributed independently of pgAdmin. It may be downloaded from the download area.

Contents:

- pgAgent Installation
 - Database setup
 - Daemon installation on Unix
 - Service installation on Windows
 - Security concerns
- pgAgent Jobs
- pgAgent Schedules
- pgAgent Steps

pgAdmin3 LTS 1.23.0a documentation »

previous | next | index

pgAgent Installation

pgAgent runs as a daemon on Unix systems, and a service on Windows systems. In the majority of cases it will run on the database server itself - it is for this reason that pgAgent is not automatically setup when pgAdmin is installed. In some cases however, it may be preferable to run pgAgent on multiple systems, against the same database - individual jobs may be targetted at a particular host, or left for execution by any host. Locking prevents execution of the same instance of a job by multiple hosts.

Database setup

pgAgent stores its configuration in the 'postgres' database in your cluster. This database exists by default in PostgreSQL 8.1 versions, for earlier versions you need to create the database yourself. The database must have the pl/pgsql procedural language installed - PostgreSQL's 'createlang' program can do this if required.

Connect to the "postgres" database (once created if needed), and open the SQL tool. If the server is 9.1 or later, and pgAgent 3.4.0 or later, simply paste in this query and click the 'Run' button:

CREATE EXTENSION pgagent;

This command will create a number of tables and other objects in a schema called 'pgagent'.

For earlier versions of PostgreSQL or pgAgent, select the File -> Open option from the menu and find the 'pgagent.sql' script installed with pgAdmin. The installation location for this file varies from operating system to operating system, however it will normally be found under 'C:Program filespgAdmin III' on Windows systems (or 'C:Program filesPostgreSQL8.xpgAdmin III' if installed with the PostgreSQL server installer), or '/usr/local/pgadmin3/share/pgadmin3' or '/usr/share/pgadmin3' on Unix systems. Once the file is loaded, click the 'Run' button to execute the script.

The script will create a number of tables and other objects in a schema called 'pgagent'.

Daemon installation on Unix

To install the pgAgent daemon on a Unix system, you will normally need to have root privileges to modify the system startup scripts - doing so is quite system specific so you should consult your system documentation for further information.

The program itself takes few command line options - most of which are only needed for debugging or specialised configurations:

```
Usage:
   /path/to/pgagent [options] <connect-string>

options:
   -f run in the foreground (do not detach from the terminal)
   -t <poll time interval in seconds (default 10)>
   -r <retry period after connection abort in seconds (>=10, def
   -s <log file (messages are logged to STDOUT if not specified)
   -l <logging verbosity (ERROR=0, WARNING=1, DEBUG=2, default 0</pre>
```

The connect string required is a standard PostgreSQL libpq connection string (see the PostgreSQL documentation on the connection string for further details). For example, the following command lilne will run pgAgent against a server listening on the localhost, using a database called 'pgadmin', connecting as the user 'postgres':

```
/path/to/pgagent hostaddr=127.0.0.1 dbname=postgres user=postgr
```

Service installation on Windows

pgAgent is able to self-install itself as a service on Windows systems. The command line options available are similar to those on Unix systems, but include an additional parameter to tell the service what to do:

The service may be quite simply installed from the command line as follows (adjusting the path as required):

```
"C:\Program Files\pgAdmin III\pgAgent" INSTALL pgAgent -u postg
```

The service may then be started from the command line using *net* start pgAgent, or from the Services control panel applet. Any logging output or errors will be reported in the Application event log. The DEBUG mode may be used to run pgAgent from a command prompt. When run this way, log messages will output to the command window.

Security concerns

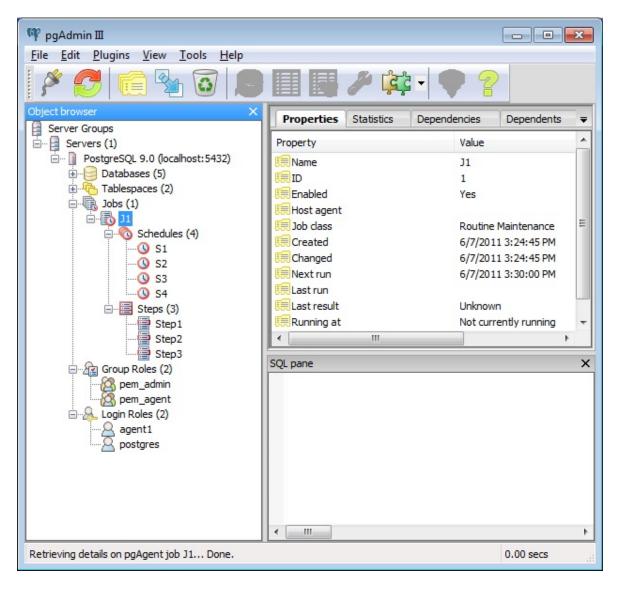
pgAgent is a very powerful tool, but does have some security considerations that you should be aware of:

Database password - *DO NOT* be tempted to include a password in the pgAgent connection string - on Unix systems it may be visible to all users in 'ps' output, and on Windows systems it will be stored in the registry in plain text. Instead, use a libpq ~/.pgpass file to store the passwords for every database that pgAgent must access. Details of this technique may be found in the PostgreSQL documentation on .pgpass file.

System/database access - all jobs run by pgAgent will run with the security privileges of the pgAgent user. SQL steps will run as the user that pgAgent connects to the database as, and batch/shell scripts will run as the operating system user that the pgAgent service or daemon is running under. Because of this, it is essential to maintain control over the users that are able to create and modify jobs. By default, only the user that created the pgAgent database objects will be able to do this - this will normally be the PostgreSQL superuser.

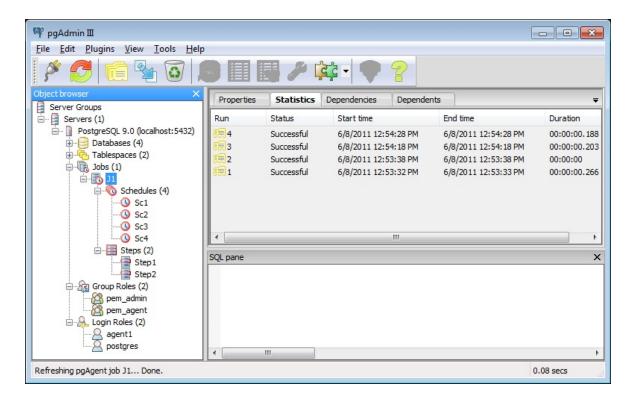
pgAgent Jobs

pgAgent runs 'jobs', each of which consists of pgAgent Steps and pgAgent Schedules. To configure jobs on your server, browse to the server on which the pgAgent database objects were created. A *Jobs* node will be displayed, under which individual jobs are shown. New jobs may be created, and existing jobs modified in the same way as any other object in pgAdmin.



The properties tab in the main pgAdmin window will display the details of the selected job, and the Statistics tab will show the details

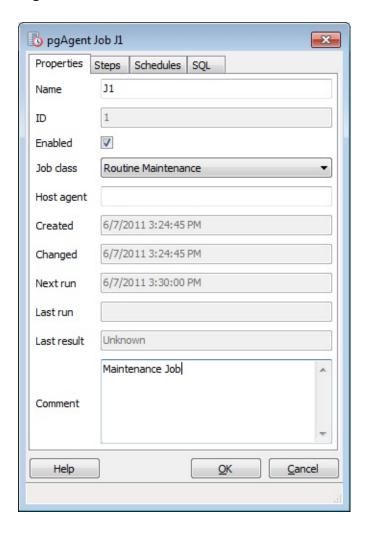
of each run of the job.



Job themselves are very simple - the complexity is in the steps and schedules. When configuring a job, we specify the job name, whether or not it is enabled, a job class or category, and optionally the hostname of a specific machine running pgAgent, if this job should only run on that specific server. If left empty (which is normally the case), any server may run the job. This is not normally needed for SQL-only jobs, however any jobs with batch/shell steps may need to be targetted to a specific server. A comment may also be included.

Note: It is not always obvious what value to specify for the Host Agent in order to target a job step to a specific machine. With pgAgent running on the required machines and connected to the scheduler database, the following query may be run to view the hostnames as reported by each agent:

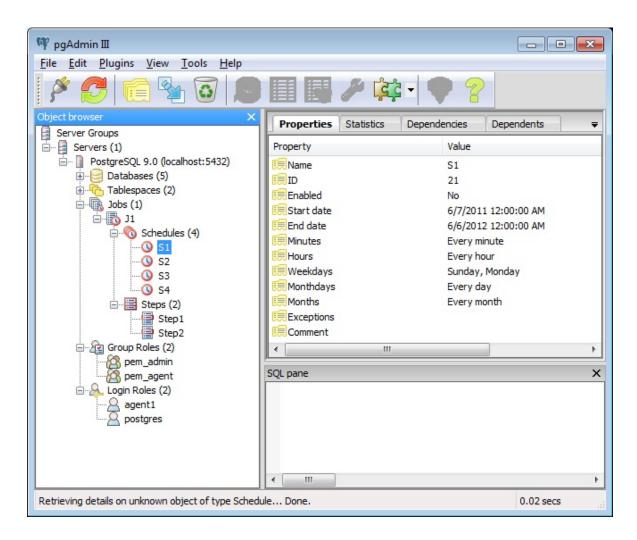
Use the hostname exactly as reported by the query in the Host Agent field.



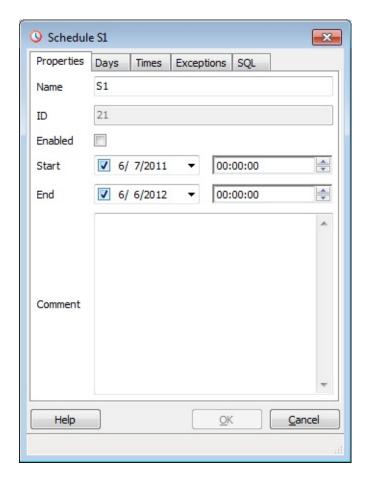
pgAgent Schedules

Each Job is executed according to one or more schedules. Each time the job or any of its schedules are altered, the next runtime of the job is re-calculated. Each instance of pgAgent periodically polls the database for jobs with the next runtime value in the past. By polling at least once every minute, all jobs will normally start within one minute of the specified start time. If no pgAgent instance is running at the next runtime of a job, it will run as soon as pgAgent is next started, following which it will return to the normal schedule.

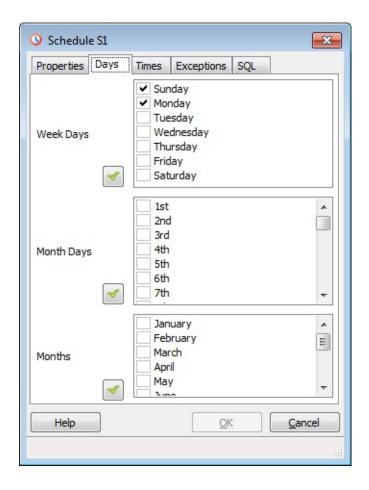
Schedules may be added to a job through the job properties dialogue, or added as a sub-object. The *Properties* tab of the main pgAdmin window will display details of the selected schedule.



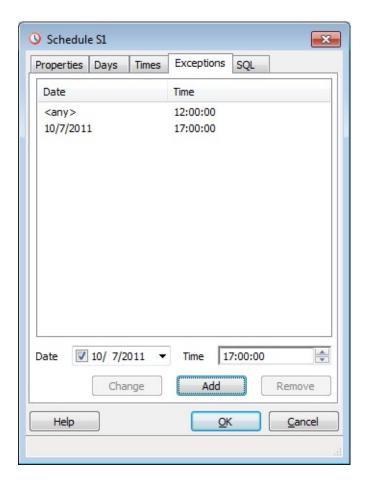
Each schedule consists of the basic details such as a name, whether or not it is enable and a comment. In addition, a start date and time is specified (before which the schedule has no effect), and optionally an end date and time (after which the schedule has no effect).



Schedules are specified using a cron-style format. For each selected time or date element, the schedule will execute. For example, to execute at 5 minutes past every hour, simply tick '5' in the *Minutes* list box. Values from more than one field may be specified in order to further control the schedule. For example, to execute at 12:05 and 14:05 every Monday and Thursday, you would tick minute 5, hours 12 and 14, and weekdays Monday and Thursday. For additional flexibility, the *Month Days* check list includes an extra *Last Day* option. This matches the last day of the month, whether it happens to be the 28th, 29th, 30th or 31st.



On occasion it may be desirable to specify an exception for a schedule - for example, you may not want a schedule to fire on a particular national holiday. To achieve this, each schedule may have a list of date and/or time exceptions attached to it. If a schedule lands on an exception, that instance will be skipped, and the following occurance will become the next runtime.



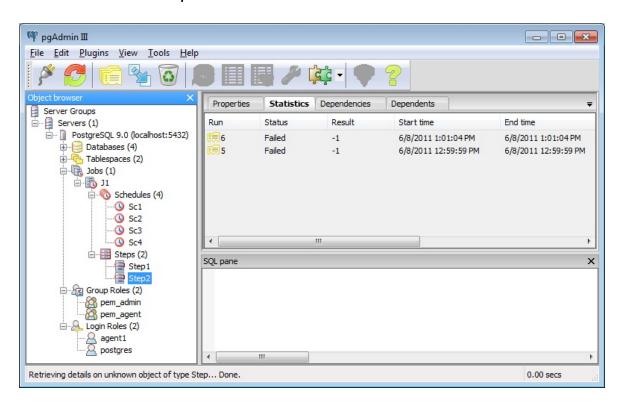
pgAdmin3 LTS 1.23.0a documentation » pgAgent »

previous | next | index

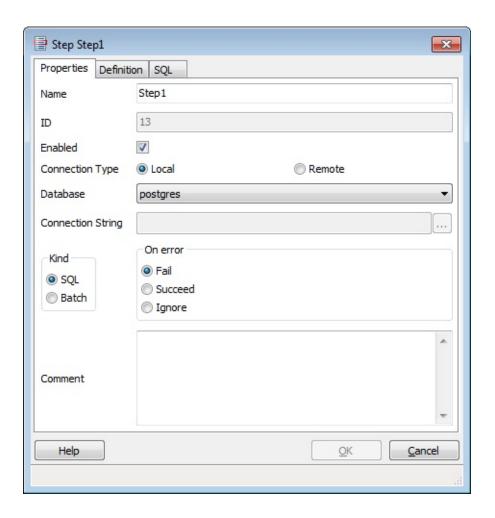
pgAgent Steps

Each Job consists of a number of steps, each of which may be an SQL script, or an operating system batch/shell script. Each step in a given job is run in turn, in alphanumeric name order.

Steps may be added to a job through the job properties dialogue, or added as a sub-object. The *Properties* tab of the main pgAdmin window will display details of the selected step, and the *Statistics* tab will display details of each run of the step, including and output or errors from the script.



Each step consists of the details shown on the screenshot below, most of which are self-explanatory. If *Kind* is set to SQL, then it goes without saying that a database against which to run the script must be selected. If set to *Batch*, the database/connection string should be left blank. The *On Error* option controls how failure of this step will affect the status of the overall job.



The *Definition* tab contains a single text box into which the step script should be entered. For SQL steps, this should be a series of one or more SQL statements. For batch jobs, when running on a Windows server, standard batch file syntax must be used, and when running on a *nix server*, any shell script may be used, provided that a suitable interpreter is specified on the first line (e.g. *#!/bin/sh).

Slony-I support

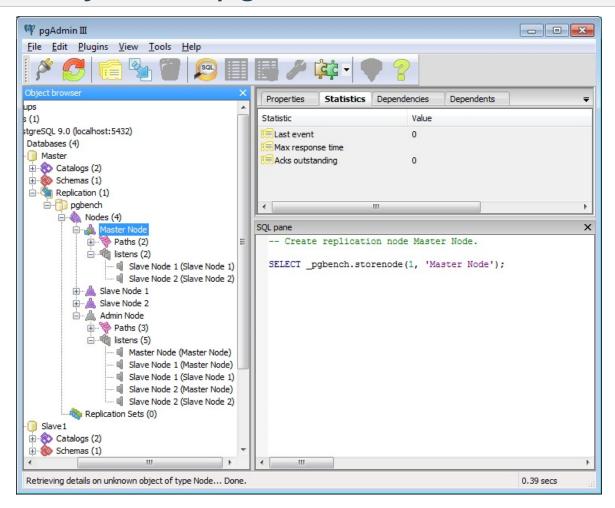
pgAdmin III includes a frontend to Slony-I, the most popular masterslave replication solution for PostgreSQL. pgAdmin III makes maintaining the replication setup easier, and features health information to monitor the state of the cluster.

Contents:

- Slony-I with pgAdmin III overview
- Slony-I administration with pgAdmin III: installation
 - Prerequisites
 - Overview
 - Create cluster and first node
 - Join additional nodes to cluster
 - Upgrade node to new cluster software
- · Creating paths and listens
 - Create paths to other nodes
 - Create listens to other nodes
- Creating sets and subscriptions
 - Create replication set
 - Define replicated table
 - Define replicated sequence
 - Subscribe a replication set
- Execute DDL scripts with Slony-I
- Slony-I tasks
- Slony-I example

For further information, please refer to the official *Slony-I documentation* http://slony.info/documentation/> which is embedded in this help.

Slony-I with pgAdmin III overview



The Slony-I objects are integrated into pgAdmin's main object tree browser, allowing a single interface to both database object and replication administration.

The statistics tab for the nodes collection as well as for individual nodes show the status of the replication event queue, and allows monitoring of the functionality of the slony cluster.

As an example, the situation shown above displays the status of a node that hasn't been responsive for about an hour, with 381 events pending to be replicated to that node.

Slony-I administration with pgAdmin III: installation

Prerequisites

As a prerequisite to running Slony-I on PostgreSQL, the slony modules xxid and slony1_funcs must be present on all servers that have to run a Slony-I replication node. This is usually done by the Slony-I installation routine.

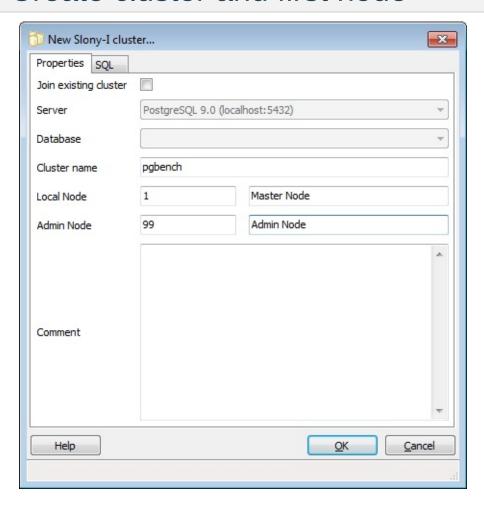
Overview

Setting up Slony-I for the first time can be a daunting task. The following sections of the pgAdmin helpfile will help guide you through creation of your first cluster.

- Step 1: Create the cluster on the master node.
- Step 2: Add slave nodes to the cluster.
- Step 3: Setup paths on each node to all other nodes.
- Step 4: Setup listens on each node to all other nodes (**Note**: Not required with Slony-I v1.1 or later).
- Step 5: Create a replication set.
- Step 6: Add tables and sequences to the set.
- Step 7: Subscribe the slaves to the set.

Note At this point, the slon daemons (service engines on Windows) should be started.

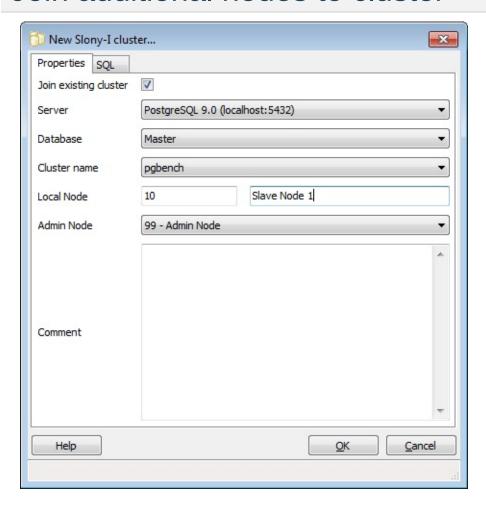
Create cluster and first node



To install a Slony-I cluster on the first database, the "New Slony-I Cluster" dialog is used. It executes the official Slony-I cluster creation SQL scripts, which are located in the directory configured in the Options dialog.

pgAdmin III needs to store information how to contact each individual node in the cluster. To achieve this, pgAdmin III uses the concept of "Administrative nodes".

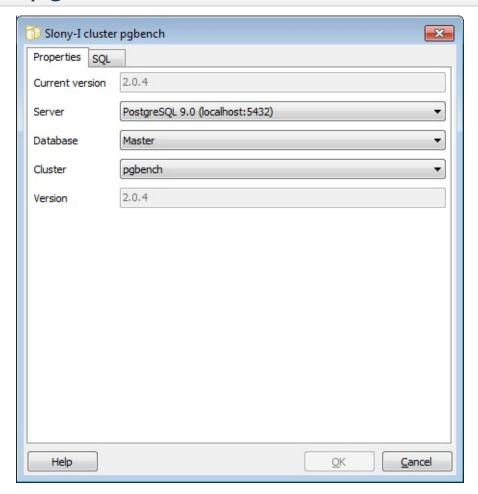
Join additional nodes to cluster



After the first node in the Slony-I replication cluster has been successfully created, all subsequent nodes take their configuration and procedures from the first nodes. This process is called "Joining a cluster" in pgAdmin III. Usually, you should also select an existing node as admin node, to insure later accessibility from pgAdmin III.

After you added a new node to the Slony-I cluster, you need to set up replication paths between the nodes, to enable communication between the nodes.

Upgrade node to new cluster software



When a cluster is to be upgraded to a new version of the Slony-I clustering software, the upgrade process has to be run on all nodes of the cluster. For each node, the slon daemon needs to be stopped, then the upgrade dialog is started and a node with the new software is selected (pgAdmin III will extract all software from that node), and finally the slon daemon is started again.

Currently, pgAdmin III does **not** support upgrading from slony creation scripts. Instead, create an intermediate cluster from the creation scripts, use it as a source for the upgrade dialog, and drop the cluster after usage. You may also use the slonik tool to upgrade the first node, and then use it as source for subsequent node

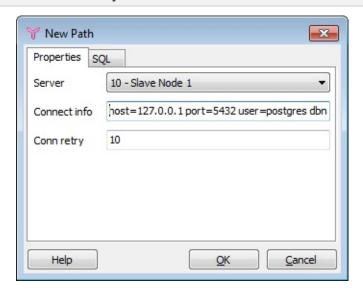
upgrades.

pgAdmin3 LTS 1.23.0a documentation » Slony-I support »

previous | next | index

Creating paths and listens

Create paths to other nodes



Slony-I needs path information, that defines how a slon process can communicate to other nodes. The conninfo string takes a connect string as described in the libpq connection documentation. Usually, you will need to specify host, dbname and username, while the password should be stored in the .pgpass file.

You must create a path to every other node, on each node. For example, in a two node cluster you need to create a path to the slave on the master, and one to the master on the slave.

Create listens to other nodes



After the communication path has been defined, the slon processes need to be advised to listen to events from other nodes.

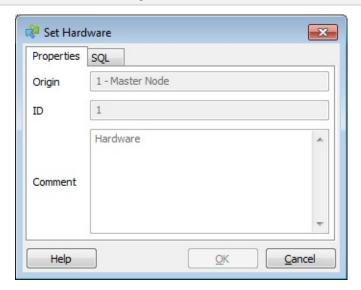
Note: This step is not necessary for Slony-I v1.1 and later, because listen information is generated automatically when paths are defined.

pgAdmin3 LTS 1.23.0a documentation » Slony-I support »

previous | next | index

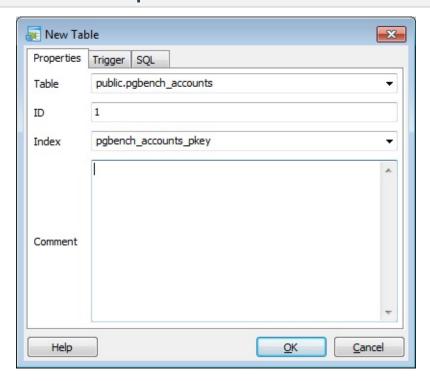
Creating sets and subscriptions

Create replication set



Slony-I groups tables and sequences it has to replicate from a master to slaves into replication sets. The set is created on the source node of the data.

Define replicated table

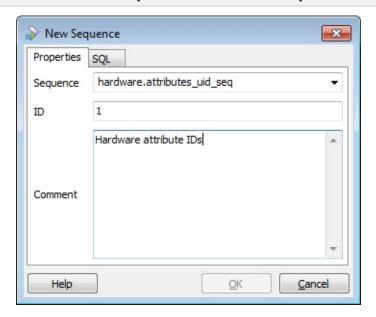


If the source table has triggers defined on it, these have to be disabled on replication target nodes. But in replication environments, the master and slave roles might exchange, so it is necessary to enable and disable triggers in those situations. The 'Trigger' page allows selection of triggers that Slony-I should enable and disable if necessary.

Attention: If a table you'd like to have replicated doesn't appear in the table combobox, this usually means that the table lacks a unique index. Slony-I requires that each row in tables that are to be replicated must be uniquely identifyable. Usually, this should be done with a primary key, but for replication purposes a unique key is satisfactory.

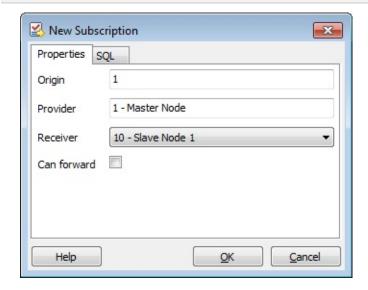
While Slony-I has a helper function to define intermediate unique keys, this is not supported with tables added to replication sets with pgAdmin III. We strongly recommend defining a primary key on the tables to be replicated.

Define replicated sequence



The sequence allows adding sequences to a replication set.

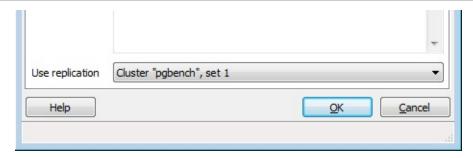
Subscribe a replication set



After a replication set has been defined, it can be subscribed. Subscriptions have to be created on the source node (note: on Slony-1 before v1.1, this had to be performed on the target node).

After a set has been subscribed, its table and sequence definition can't be changed any more. To add more tables, you can create a set that includes the additional tables and sequences you'd like to have added to the first replication set, then subscribe exactly the same receiver nodes to it, and finally use Merge to merge both sets into one.

Execute DDL scripts with Slony-I



Most schema changes that can be executed with pgAdmin III are replication-aware, i.e. you can advise pgAdmin III to apply the change (create/alter) using the replication facilities, simply by selecting a replication set that is subscribed to all nodes which should receive the PostgreSQL object.

pgAdmin3 LTS 1.23.0a documentation » Slony-I support »

previous | next | index

Slony-I tasks

In the appropriate context, pgAdmin III can trigger several maintenance tasks.

- **Restart node** Under some conditions, it may be necessary to restart a slon process on a node, initialize as if freshly started and make it reload its configuration. This can be triggered remotely with the restart node function.
- **Lock set** Disables updates to a set to be replicated. This is necessary for clean switch over the source of a replication from one node to another.
- Unlock set Re-enables updates to a set to be replicated after a previous lock.
- **Merge set** Joins two sets, originating from the same node and subscribed by the same nodes, into one. This can be used as a workaround to the fact that a subscribed set can't be extended.
- Move set Moves a replication set from one node to another,
 i.e. making the target node the new source. The old node will
 become subscriber to the new provider node. This is the normal
 way how to switch over gracefully the master role from one node
 to another.

Slony-I example

In this example, a master server is setup with two direct slaves. This example was written and tested using Slony-I v1.2.11 and PostgreSQL 8.2.5, running on a single Windows XP machine. The PostgresSQL pgbench utility is used to generate the test schema and workload.

- 1. Create 3 databases, master, slave1 and slave2 and ensure pl/pgsql is setup in each.
- 2. Create a pgbench schema in the master database:

```
> pgbench -i -U postgres master
```

- 3. Add a primary key called history_pkey to the history table on the tid, bid and aid columns
- 4. Create a schema-only dump of the master database, and load it into slave1 and slave2:

```
> pg_dump -s -U postgres master > schema.sql
> psql -U postgres slave1 < schema.sql
> psql -U postgres slave2 < schema.sql</pre>
```

5. Create Slony config files for each slon engine (daemon on Unix). The files should contain just the following two lines:

```
cluster_name='pgbench'
conn_info='host=127.0.0.1 port=5432 user=postgres dbname=ma

♪
```

Create a file for each database, adjusting the dbname parameter as required and adding any other connection options that may be needed.

6. (Windows only) Install the Slony-I service:

```
> slon -regservice Slony-I
```

7. Register each of the engines (this is only necessary on Windows - on Unix the slon daemons may be started individually and given the path to the config file on the command line using the -f option):

```
> slon -addengine Slony-I C:\slony\master.conf
> slon -addengine Slony-I C:\slony\slave1.conf
> slon -addengine Slony-I C:\slony\slave2.conf
```

8. In pgAdmin under the Slony Replication node in the master database, create a new Slony-I cluster using the following options:

```
Join existing cluster: Unchecked
Cluster name: pgbench
Local node: 1 Master node
Admin node: 99 Admin node
```

9. Under the Slony Replication node, create a Slony-I cluster in each of the slave databases using the following options:

```
Join existing cluster: Checked
Server:
                      <Select the server containing the ma</pre>
Database:
                      master
Cluster name:
                    pgbench
                              Slave node 1
Local node:
                     10
Admin node:
                      99 - Admin node
Join existing cluster: Checked
Server:
                      <Select the server containing the ma</pre>
Database:
                      master
Cluster name:
                      pgbench
Local node:
                               Slave node 2
                      20
Admin node:
                      99 - Admin node
```

10. Create Paths on the master to both slaves, and on each slave

back to the master. Create the paths under each node on the master, using the connection strings specified in the slon config files. Note that future restructuring of the cluster may require additional paths to be defined.

11. Create a Replication Set on the master using the following settings:

```
ID: 1
Comment: pgbench set
```

12. Add the tables to the replication set using the following settings:

```
Table:
                      public.accounts
ID:
Index:
                      accounts_pkey
Table:
                      public.branches
ID:
                      branches_pkey
Index:
Table:
                      public.history
ID:
                      history_pkey
Index:
Table:
                      public.tellers
ID:
Index:
                      tellers_pkey
```

13. On the master node, create a new subscription for each slave using the following options:

```
Origin: 1
Provider: 1 - Master node
Receiver: 10 - Slave node 1

Origin: 1
Provider: 1 - Master node
Receiver: 20 - Slave node 2
```

14. Start the slon service (or daemons on Unix):

> net start Slony-I

Initial replication should begin and can be monitored on the statistics tab in pgAdmin for each node. The pgbench utility may be run against the master database to generate a test workload.

pgAdmin3 LTS 1.23.0a documentation » Slony-I support »

Extended features

pgAdmin III's functionality can be further enhanced using the adminpack contrib module for PostgreSQL. This is a library of additional functions that may be installed on your server which allows pgAdmin to offer additional features, such as:

- The sizes on disk of tablespaces, databases, tables and indexes are displayed on the Statistics tab in the main window.
- The Status dialogue should be able to display the server logfiles (if logging is appropriately configured).
- Running queries from other users may be cancelled from the Status dialogue.
- The server's configuration files postgresql.conf and pg_hba.conf may be editted, and reloaded remotely.

The adminpack is installed and activated by default if you are running the 'official' pgInstaller distribution of PostgreSQL for Windows and is included as a contrib module with all versions of PostgreSQL 8.2 and above. However, if you are running any other version of PostgreSQL you will need to manually install it. To do so, simply copy the appropriate adminpack source code to the /contrib directory of your pre-configured PostgreSQL source tree. You can download the adminpacks here.

Run the following commands (substituting *admin* for *admin81* if required):

```
# cd $PGSRC/contrib/admin
# make all
# make install
```

The module should now be built, and installed into your existing PostgreSQL installation. To be used, you must now install the module into the database you use as pgAdmin's *Maintenance DB*

(normally *postgres*, though you may wish to use a different database). To do so, locate the *admin.sql* or *admin81.sql* file installed in the previous step - normally this may be found in /usr/local/pgsql/share/contrib/ and run it against your database. You will normally need to do this as the PostgreSQL superuser:

\$ psql postgres < /usr/local/pgsql/share/contrib/admin.sql</pre>

The next time you connect to the server with pgAdmin, it will detect the presence of the adminpack and offer full functionality.

Since PostgreSQL 9.1, you don't need to use psql. You can manually add adminpack extension or make pgAdmin do the same by clicking on the "Fix It!" button of the guru hint.

pgAdmin3 LTS 1.23.0a documentation »

Appendices

Contents:

- Bug Reporting
- The pgAdmin Development Team
- Translation team
- The PostgreSQL Licence
- The MIT Kerberos Licence
- The OpenSSL Licence

pgAdmin3 LTS 1.23.0a documentation »

Bug Reporting

If you think you encountered a bug in pgAdmin III, please check the following:

- Is this behaviour expected? Have a look at the FAQs.
- Is this bug already reported? Check the known bugs.
- Are you running the latest version from the website? If not, try the newest stuff.

Ahh, so you've found a new one then?

Post your report to the support mailing list, and include the following informations:

- Platform you're running on
- Language
- Distribution you used (source tarball, or binary)
- Version you tested
- If possible and appropriate, include a backtrace from the crash.
 You may need to recompile wxWidgets and pgAdmin with the enable-debug option to do this.

Please do **not** send an email directly to one of the programmers; they might miss it, or even be the wrong person altogether.

Using the mailing list guarantees that your request isn't lost and will be answered in a timely fashion.

After a bug has been reported on the mailing list as being fixed, it might take one or two days until it has made it into the distribution tarballs, please be patient.

Thanks for giving pgAdmin3 a chance!

The pgAdmin Development Team

pgAdmin3 LTS 1.23.0a documentation » Appendices »

The pgAdmin Development Team

Version 2.3, 5 March 2009

1. Purpose

The pgAdmin Development Team exists for 2 reasons:

- 1. To manage and maintain the pgAdmin Project
- 2. To ensure that pgAdmin remains freely available to anyone that wants it.

2. Structure

The foundation consists of the Project Leader and Members.

- Project Leader: The Project Leader is Dave Page, founder and primary developer of the software. The Project Leader is also considered a member.
- Members: Members can be any contributor to the project who accepts an invitation from the existing Development Team.

3. Division of Responsibilities

Any major decision regarding the Project must be voted on by the Development Team. Any vote will be made via email through the relevant mailing list. Responses not received within 2 weeks will be discounted from the vote.

In the case of a hung vote, the Project Leader shall cast the deciding vote.

4. People

The pgAdmin Development Team includes the following people:

o Dave Page (Project Leader): Project founder & primary

developer.

- Hiroshi Saito: Produced a Japanese port of pgAdmin II, and countless patches for pgAdmin III.
- Raphaël Enrici: Maintains the Debian port, and helps with numerous other issues.
- Guillaume Lelarge: Manages the pgAdmin application and website translations, and has added a number of enhancements to pgAdmin.
- Erwin Brandstetter: Spends significant amounts of time testing new releases and helping us track down bugs and behavioural issues.
- Magnus Hagander: Has worked on numerous features and project infrastructure.

The following people no longer work on the project, but we remain grateful for their past contributions:

- Mark Yeatman: Works on controls and consults on UI design.
- Frank Lupo: Has made countless improvements to the pgAdmin II code from minor bug fixes to major enhancements.
- Jean-Michel Pouré: translations & web site management.
- Andreas Pflug: Has written vast amounts of pgAdmin III code and is responsible for much of its design.
- Adam H. Pendleton: Developed and maintains the build system, and helps with various porting issues.

Translation team

We are very grateful to those who commit time to translating pgAdmin III. Here is the list of translators, ordered by language code:

- af ZA (Afrikaans) by Petri Jooste
- bg_BG (Bulgarian) by Milen A. Radev
- ca_ES (Catalan) by Carme Cerda Torres
- de DE (German) by Harald Armin Massa
- es_ES (Spanish) by mailto:diego@adminsa.com">Diego A. Gil
- fi FI (Finnish) by Jori Juoto
- fr_FR (French) by Guillaume Lelarge and Raphael Enrici
- gl_ES (Galician) by Antonio Ricoy Riego
- it_IT (Italian) by Giuseppe Sacco
- ja JP (Japanese) by Hiroshi Saito
- ko KR (Korean) by Kang Hye Won
- Iv LV (Latvian) by Rudolfs Mazurs
- nl NL (Dutch) by Fred van Bemmel
- pl PL (Polish) by Slawomir Sudnik
- pt BR (Portuguese Brazilian) by Marcos Alves T. de Azevedo
- pt PT (Portuguese) by Helder M. Vieira
- ru RU (Russian) by Vladimir Kazimirov
- sr RS (Serbian) by Bojan Skaljac
- zh CN (Chinese simplified) by ChaoYi, Kuo
- zh_TW (Chinese traditional) by NSYSU MIS

There are other translations in progress, you can check the current translation status at our Translation status page.

The PostgreSQL Licence

Copyright (C) 2002 - 2016, The pgAdmin Development Team

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL THE PGADMIN DEVELOPMENT TEAM BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE PGADMIN DEVELOPMENT TEAM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE PGADMIN DEVELOPMENT TEAM SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE PGADMIN DEVELOPMENT TEAM HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

The MIT Kerberos Licence

pgAdmin PostgreSQL's libpq library which may be linked with MIT Kerberos Libraries on some distributions. The MIT Kerberos licence is included below:

Kerberos Copyright

This software is being provided to you, the LICENSEE, by the Massachusetts Institute of Technology (M.I.T.) under the following license. By obtaining, using and/or copying this software, you agree that you have read, understood, and will comply with these terms and conditions:

Permission to use, copy, modify and distribute this software and its documentation for any purpose and without fee or royalty is hereby granted, provided that you agree to comply with the following copyright notice and statements, including the disclaimer, and that the same appear on ALL copies of the software and documentation, including modifications that you make for internal use or for distribution:

Copyright 1992-2004 by the Massachusetts Institute of Technology. All rights reserved.

THIS SOFTWARE IS PROVIDED "AS IS", AND M.I.T. MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. By way of example, but not limitation, M.I.T. MAKES NO REPRESENTATIONS OR WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE LICENSED SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

The name of the Massachusetts Institute of Technology or M.I.T. may NOT be used in advertising or publicity pertaining to distribution of the software. Title to copyright in this software and any associated documentation shall at all times remain with M.I.T., and USER agrees to preserve same.

Project Athena, Athena, Athena MUSE, Discuss, Hesiod, Kerberos, Moira, OLC, X Window System, and Zephyr are trademarks of the Massachusetts Institute of Technology (MIT). No commercial use of these trademarks may be made without prior written permission of MIT.

pgAdmin3 LTS 1.23.0a documentation » Appendices »

previous | index

The OpenSSL Licence

pgAdmin uses code from the OpenSSL project to provide support for SSL encrypted connections. The OpenSSL licence is included below:

Copyright (c) 1998-2011 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
- The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.
- Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.
- Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

pgAdmin3 LTS 1.23.0a documentation » Appendices »

previous | index

Index

$\mathsf{B} \, | \, \mathsf{C} \, | \, \mathsf{D} \, | \, \mathsf{E} \, | \, \mathsf{G} \, | \, \mathsf{M} \, | \, \mathsf{O} \, | \, \mathsf{P} \, | \, \mathsf{Q} \, | \, \mathsf{R} \, | \, \mathsf{S} \, | \, \mathsf{V}$

В

Backup

Bug Reporting

C

Change Password
Command Line Parameters
Connect to server

Connection errors
Control Server
Creating paths and listens
Creating sets and subscriptions

D

Database Server Status

Default XSL Stylesheet

Ε

Edit Data

Execute DDL scripts with Slony-I Extended features

G

Grant Wizard

Graphical Query builder Guru Hints

M

Maintain a database object



Object search

P

pgAdmin Browser Options
pgAdmin Data Export
pgAdmin Database Designer
Options
pgAdmin Debugger
pgAdmin Main Window
pgAdmin Miscellaneous Options
pgAdmin Options

pgAdmin Query tool Options pgAdmin Server Status Options pgAgent Installation pgAgent Jobs pgAgent Schedules pgAgent Steps pgScript Scripting Language Reference Q

Query tool

Query Tool Macros

R

Report Tool

Restore

Sample XML Data Slony-I administration with pgAdmin III: installation Slony-I example Slony-I support Slony-I tasks Slony-I with pgAdmin III overview



View Data Options

pgAdmin3 LTS 1.23.0a documentation »

index