Easysoft ODBC-PostgreSQL Driver User's Guide

This manual documents version 1.0.n of the Easysoft ODBC-PostgreSQL Driver.

Table of contents

Getting started

This section shows you how to install the Easysoft ODBC-PostgreSQL Driver and configure the ODBC data source that stores the connection details for your PostgreSQL database. You're then ready to work with PostgreSQL data in your application.

Installing the Easysoft ODBC-PostgreSQL Driver

Install the Easysoft ODBC-PostgreSQL Driver on the computer where the application you want to connect to PostgreSQL is running.

Installing on Linux or UNIX

The installation can be done by anyone with root access.

1. Download the Easysoft ODBC-PostgreSQL Driver distribution for your client application platform.

   If your client application is 64-bit, choose the 64-bit driver distribution from the **Platforms** list. If your client application is 32-bit, choose the 32-bit driver distribution from the **Platforms** list.
2. Copy the distribution to a temporary directory on the machine where the application you want to connect to PostgreSQL is installed.
3. Unpack the distribution and cd into the resultant directory.
4. As root, run:

   ```
   ./install
   ```
5. Follow the onscreen instructions to progress through the installation.

Further information

Preinstallation requirements

To install the Easysoft ODBC-PostgreSQL Driver you need:

- The Bourne shell in /bin/sh. If your Bourne shell is not located there, you may need to edit the first line of the installation script.
- Various commonly used commands such as:

  ```
  grep, awk, test, cut, ps, sed, cat, wc, uname, tr, find, echo, sum, head, tee, id
  ```

  If you do not have any of these commands, they can usually be obtained from the Free Software Foundation. As the tee command does not work correctly on some systems, the distribution

includes a tee replacement.

- Depending on the platform, you'll need up to 10 MB of temporary space for the installation files and up to 10 MB of free disk space for the installed programs. If you also install the unixODBC Driver Manager, these numbers increase by approximately 1.5 MB.
- For Easysoft licensing to work, you must do one of the following:
  - Install the Easysoft ODBC-PostgreSQL Driver in /usr/local/easysoft.
  - Install the Easysoft ODBC-PostgreSQL Driver elsewhere and symbolically link /usr/local/easysoft to wherever you chose to install the software.

    The installation will do this automatically for you so long as you run the installation as someone with permission to create /usr/local/easysoft.
  - Install the Easysoft ODBC-PostgreSQL Driver elsewhere and set the EASYSOFT_ROOT environment variable. For more information about setting the EASYSOFT_ROOT environment variable, refer to Post installation steps for non-root installations.
- An ODBC Driver Manager.

Easysoft ODBC-PostgreSQL Driver distributions include the unixODBC Driver Manager.

- You do not have to be the root user to install, but you will need permission to create a directory in the chosen installation path. Also, if you are not the root user, it may not be possible for the installation to:
  1. Register the Easysoft ODBC-PostgreSQL Driver with unixODBC.
  2. Create the example data source in the SYSTEM odbc.ini file.
  3. Update the dynamic linker entries (some platforms only).

If you are not root, these tasks will have to be done manually later.

We recommend that you install all components as the root user.

What you can install

This distribution contains:

- The Easysoft ODBC-PostgreSQL Driver.
- The unixODBC Driver Manager.

You need an ODBC Driver Manager to use the Easysoft ODBC-PostgreSQL Driver from your applications. The distribution therefore contains the unixODBC Driver Manager. Most (if not all) UNIX and Linux applications support the unixODBC Driver Manager. For example, Perl DBD::ODBC, PHP, Python, and so on.

You do not have to install the unixODBC Driver Manager included with this distribution. You can use an existing copy of unixODBC. For example, a version of unixODBC installed by another Easysoft product, a version obtained from your operating system vendor, or one that you built yourself. However, as Easysoft ensure that the unixODBC distributed with the Easysoft ODBC-PostgreSQL Driver has been tested with that driver, we recommend you use it.

If you choose to use an existing unixODBC Driver Manager, the installation script will attempt to locate it. The installation script looks for the ODBC Driver Manager in the standard places. If you have installed it in a non-standard location, the installation script prompts you for the location. The installation primarily needs unixODBC's odbcinst command to install drivers and data sources.

Where to install

This installation needs a location for the installed files. The default location is /usr/local.

At the start of the installation, you're prompted for an installation path. All files are installed in a subdirectory of your specified path called easysoft. For example, if you accept the default location

/usr/local, the product will be installed in /usr/local/easysoft and below.

If you choose a different installation path, the installation script tries to symbolically link /usr/local/easysoft to the easysoft subdirectory in your chosen location. This allows us to distribute binaries with built in dynamic linker run paths. If you are not root or the path /usr/local/easysoft already exists and is not a symbolic link, the installation will be unable to create the symbolic link. For information about how to correct this manually, refer to Post installation steps for non-root installations.

Note that you cannot license Easysoft products until either of the following is true:

- /usr/local/easysoft exists either as a symbolic link to your chosen installation path or as the installation path itself.
- You have set EASYSOFT_ROOT to *installation_path*/easysoft.

Changes made to your system

The installation script installs files in subdirectories of the path requested at the start of the installation. Depending on what is installed, a few changes may be made to your system:

1. If you choose to install the Easysoft ODBC-PostgreSQL Driver into unixODBC, unixODBC's odbcinst command will be run to add an entry to your odbcinst.ini file. You can locate this file with odbcinst -j. (odbcinst is in *installation_path*/easysoft/unixODBC/bin, if you are using the unixODBC included with this distribution.)
2. The installation script installs an example data source into unixODBC. This data source will be added to your SYSTEM odbc.ini file. You can locate your SYSTEM odbc.ini file by using odbcinst -j.
3. Dynamic linker. On operating systems where the dynamic linker has a file listing locations for shared objects (Linux and FreeBSD), the installation script will attempt to add paths under the path you provided at the start of the installation to the end of this list:
   - On Linux, this is usually the file /etc/ld.so.conf.
   - On FreeBSD this is usually the file /etc/defaults/rc.conf.

Installing alongside other existing Easysoft product installations

Each Easysoft distribution contains common files shared between Easysoft products. These shared objects are placed in *installation_path*/easysoft/lib. When you run the installation script, the dates and versions of these files are compared with the same files in the distribution. The files are only updated if the files being installed are newer or have a later version number.

You should ensure that nothing on your system is using Easysoft software before starting an installation. This is because on some platforms, files in use cannot be replaced. If a file cannot be updated, you get a warning during the installation. All warnings are written to a file called warnings in the directory you unpacked the distribution into.

If the installer detects you're upgrading a product, the installer will suggest you delete the product directory to avoid having problems with files in use. An alternative is to rename the specified directory.

If you are upgrading, you will need a new license from Easysoft to use the new driver.

Gathering information required during the installation

During the installation, you're prompted for various pieces of information. Before installing, you need to find out whether you have unixODBC already installed and where it is installed. The installation script searches standard places like /usr and /usr/local.

However, if you installed the Driver Manager in a non-standard place and you do not install the

included unixODBC, you will need to know the location.

Unpacking the distribution

The distribution for UNIX and Linux platforms is a tar file. To extract the installation files from the tar file, use:

```
tar -xvf odbc-postgres-1.0.0-linux-x86-64-ul64.tar
```

This creates a directory with the same name as the tar file (without the .tar postfix) containing further archives, checksum files, an installation script, and various other installation files.

Change into the directory created by unpacking the tar file to run the installation script. For example:

```
# cd odbc-postgres-1.0.0-linux-x86-64-ul64
```

License to use

The end-user license agreement (EULA) is in the file license.txt. Be sure to understand the terms of the agreement before continuing, as you're required to accept the license terms at the start of the installation.

Answering questions during the installation

Throughout the installation, you're prompted to answer some questions. In each case, the default choice displays in square brackets and you need only press Enter to accept the default. If there are alternative responses, these are shown in round brackets; to choose one of these, type the response and press Enter.

For example:

```
Do you want to continue? (y/n) [n]:
```

The possible answers to this question are y or n. The default answer when you type nothing and press Enter is n.

Running the installer

If you are considering running the installation as a non-root user, we suggest you review this carefully as you will have to get a root user to manually complete some parts of the installation afterwards. We recommend installing as the root user. (If you're concerned about the changes that will be made to your system, refer to Changes made to your system.)

To start the installation, run:

```
./install
```

You need to:

- Confirm your acceptance of the license agreement by typing "yes" or "no". For more information about the license agreement, refer to License to use.
- Supply the location where the software is to be installed.

We recommend accepting the default installation path.

For more information, refer to Where to install.

Locating or installing unixODBC

We strongly recommend you use the unixODBC Driver Manager because:

- The installation script is designed to work with unixODBC and can automatically add Easysoft ODBC-PostgreSQL Driver and data sources during the installation.
- Most applications and interfaces that support ODBC are compatible with unixODBC. The Easysoft ODBC-PostgreSQL Driver and any data sources that you add during the installation are automatically available to your applications and interfaces therefore.
- The unixODBC project is currently led by Easysoft developer Nick Gorham. This means that there is a great deal of experience at Easysoft of unixODBC in general and of supporting the Easysoft ODBC-PostgreSQL Driver running under unixODBC. It also means that if you find a problem in unixODBC, it's much easier for us to facilitate a fix.

The installation starts by searching for unixODBC. There are two possible outcomes here:

1. If the installation script finds unixODBC, the following message displays:

   ```
   Found unixODBC under path and it is version n.n.n
   ```

2. If the installation script can't find unixODBC in the standard places, you will be asked whether you have it installed.

If unixODBC is installed, you need to provide the unixODBC installation path. Usually, the path required is the directory above where odbcinst is installed. For example, if odbcinst is in /opt/unixODBC/bin/odbcinst, the required path is /opt/unixODBC.

If unixODBC is not installed, you should install the unixODBC included with this distribution.

If you already have unixODBC installed, you do not have to install the unixODBC included with the distribution, but you might consider doing so if your version is older than the one we provide.

The unixODBC in the Easysoft ODBC-PostgreSQL Driver distribution is not built with the default options in unixODBC's configure line.

Option	Description
--prefix=/etc	This means the default SYSTEM odbc.ini file where SYSTEM data sources are located is /etc/odbc.ini.
--enable-drivers=no	This means other ODBC drivers that come with unixODBC are not installed.
--enable-iconv=no	This means unixODBC does not look for libiconv. Warnings about not finding an iconv library were confusing our customers.
--enable-stats=no	Turns off unixODBC statistics, which use system semaphores to keep track of used handles. Many systems do not have sufficient semaphore resources to keep track of used handles.

Option	Description
--enable-readline=no	This turns off readline support in isql. We did this because it ties isql to the version of libreadline on the system we build on. We build on as old a version of the operating system as we can for forward compatibility. Many newer Linux systems no longer include the older readline libraries and so turning on readline support makes isql unusable on these systems.
--prefix=/usr/local/easysoft/unixODBC	This installs unixODBC into /usr/local/easysoft/unixODBC.

Installing the Easysoft ODBC driver

The Easysoft ODBC-PostgreSQL Driver installation script:

- Installs the driver.
- Registers the driver with the unixODBC Driver Manager.

  If the Easysoft ODBC-PostgreSQL Driver is already registered with unixODBC, a warning displays that lists the drivers unixODBC knows about. If you're installing the Easysoft ODBC-PostgreSQL Driver into a different directory than it was installed before, you need to edit your odbcinst.ini file after the installation and correct the Driver and Setup paths. unixODBC's odbcinst doesn't update these paths if a driver is already registered.
- Creates an example Easysoft ODBC-PostgreSQL Driver data source. If unixODBC is installed and you registered the Easysoft ODBC-PostgreSQL Driver with unixODBC, the installation script adds example data source to your odbc.ini file.

Licensing

The *installation_path*/easysoft/license/licshell program lets you obtain or list licenses.

Licenses are stored in *installation_path*/easysoft/license/licenses.

Important	After obtaining a license, you should make a backup copy of this file.

The installation script asks you if you want to request an Easysoft ODBC-PostgreSQL Driver license:

```
Would you like to request a Easysoft ODBC-PostgreSQL Driver license now (y/n) [y]:
```

You do not need to obtain a license during the installation, you can run licshell after the installation to obtain or view licenses.

If you answer y, the installation runs the licshell script.

To obtain a license automatically, you need to be connected to the Internet and allow outgoing connections to license.easysoft.com on port 8884. If you're not connected to the Internet or don't allow outgoing connections on port 8884, the License Client can create a license request file that you can email to us.

When you start the License Client, the following menu displays:

```
[0] exit
[1] view existing license
```

```
[n] obtain a license for the desired product.
```

To obtain a license, select one of the options from [2] onwards for the product you're installing. The License Client then runs a program that generates a key that's used to identify the product and operating system (we need this key to license you).

After you have chosen the product to license (Easysoft ODBC-PostgreSQL Driver), you need to supply:

- Your full name.
- Your company name.
- An email contact address. This must be the email address that you used when you registered on the Easysoft web site.
- A reference number (also referred to as an authorization code). When applying for a trial license, press Enter when prompted for a reference number. This field only applies to full (paid) licenses.

You're then asked to choose how you want to obtain the license.

The choices are:

- [1] Automatically by contacting the Easysoft License Daemon

  This requires a connection to the Internet and the ability to support an outgoing TCP/IP connection to license.easysoft.com on port 8884.

- [2] Write information to file

  The license request is output to license_request.txt.

- [3] Cancel this operation

If you choose to obtain the license automatically, the License Client tries to open a TCP/IP connection to license.easysoft.com on port 8884 and send the details you supplied along with your machine number. No other data is sent. The data sent is transmitted as plain text, so if you want to avoid the possibility of this information being intercepted by someone else on the Internet, you should choose [2] and send the the request to us. The License daemon returns the license key, prints it to the screen and make it available to the installation script in the file licenses.out.

If you choose option [2], the license request is written to the file license_request.txt. You should then exit the License Client by choosing option [0] and complete the installation. After you have sent the license request to us, we'll return a license key. Add this to the end of the file *installation_path*/easysoft/license/licenses.

Post installation steps for non-root installations

If you installed the Easysoft ODBC-PostgreSQL Driver as a non-root user (not recommended), there may be some additional steps you need to do manually:

1. If you attempt to install the Easysoft ODBC-PostgreSQL Driver under the unixODBC Driver Manager and you do not have write permission to unixODBC's odbcinst.ini file, the driver can't be added.

   You can manually install the driver under unixODBC by adding an entry to the odbcinst.ini file. Run odbcinst -j to find out the location of the DRIVERS file then append the lines from drv_template file to odbcinst.ini. (drv_template is in the directory where the Easysoft distribution was untarred to.)

2. No example data sources can be added into unixODBC if you do not have write permission to the SYSTEM odbc.ini file. Run odbcinst -j to find out the location of the SYSTEM DATA SOURCES file then add your data sources to this file.

3. On systems where the dynamic linker has a configuration file defining the locations where it looks for shared objects (Linux and FreeBSD), you need to add:

   ```
   installation_path/easysoft/lib
   installation_path/easysoft/unixODBC/lib
   ```

   The latter entry is only required if you installed the unixODBC included with this distribution. Sometimes, after changing the dynamic linker configuration file, you need to run a program to update the dynamic linker cache. (For example, /sbin/ldconfig on Linux.)

4. If you didn't install the Easysoft ODBC-PostgreSQL Driver in the default location, you need to do one of the following:
   - Link /usr/local/easysoft to the easysoft directory in your chosen installation path.

     For example, if you installed in /home/user, the installation creates /home/user/easysoft and you need to symbolically link /usr/local/easysoft to /home/user/easysoft:

     ```
     ln -s /home/user/easysoft /usr/local/easysoft
     ```

   - Set and export the EASYSOFT_ROOT environment variable to *installation_path*/easysoft.
5. If your system doesn't have a dynamic linker configuration file, you need to add the paths listed in step 3 to whatever environment path the dynamic linker uses to locate shared objects. You may want to add these paths to a system file run whenever someone logs. For example, /etc/profile.

   The environment variable depends on the dynamic linker. Refer to your ld or ld.so man page. It is usually:

   ```
   LD_LIBRARY_PATH, LIBPATH, LD_RUN_PATH, or SHLIB_PATH.
   ```

Uninstalling on Linux or UNIX

There is no automated way to remove the Easysoft ODBC-PostgreSQL Driver in this release. However, removal is quite simple. To do this:

1. Change directory to *installation_path*/easysoft and delete the product directory. *installation_path* is the Easysoft ODBC-PostgreSQL Driver installation directory, by default /usr/local.
2. If you had to add this path to the dynamic linker search paths (for example, /etc/ld.so.conf on Linux), remove it. You may have to run a linker command such as /sbin/ldconfig to get the dynamic linker to reread its configuration file. Usually, this step can only be done by the root user.
3. If you were using unixODBC, the Easysoft ODBC-PostgreSQL Driver entry needs to be removed from the odbcinst.ini file. To check whether the Easysoft ODBC-PostgreSQL Driver is configured under unixODBC, use odbcinst -q -d. If the command output contains [Easysoft ODBC-Postgres], uninstall the driver from unixODBC by using:

```
odbcinst -u -d -n Easysoft ODBC-Postgres
```

If a reduced usage count message is displayed, repeat this command until odbcinst reports that the driver has been removed.

1. If you created any Easysoft ODBC-PostgreSQL Driver data sources under unixODBC, you may want to delete these. To do this, first use odbcinst -j to locate USER and SYSTEM odbc.ini files. Then check those files for data sources that have the driver attribute set to Easysoft ODBC-Postgres.
2. Remove the install.info for the Easysoft ODBC-PostgreSQL Driver from the /usr/local/easysoft directory.

Installing on Windows

The Windows installation can be done by anyone with local administrator privileges.

1. Download the Easysoft ODBC-PostgreSQL Driver installer.
2. Follow the onscreen instructions to progress through the installation wizard.

Updating files that are in use

To avoid rebooting your computer, the Easysoft ODBC-PostgreSQL Driver installer prompts you when files that it needs to update are in use by another application or service. This frees the locked files and allows the installation to complete without a system restart. The installer uses the **Restart Manager** to locate the applications that are using files that need updating. These applications are displayed in the **Files in Use** dialog box. To avoid a system restart, choose **Automatically close applications and attempt to restart them after setup is complete**. The Easysoft ODBC-PostgreSQL Driver installer then uses **Restart Manager** to try to stop and restart each application or service in the list. If possible, **Restart Manager** restores applications to the same state that they were in before it shut them down.

Licensing

By default, the installer starts the Easysoft License Manager, because you can't use the Easysoft ODBC-PostgreSQL Driver until you have a license. If you choose not to run Easysoft License Manager as part of the installation process, run License Manager from the **Easysoft** group in the Windows **Start** menu when you're ready to license the Easysoft ODBC-PostgreSQL Driver. These types of license are available:

- A free time-limited trial license, which gives you free and unrestricted use of the product for a limited period (usually 14 days).
- A full license if you have purchased the product. On purchasing the product you are given an authorization code, which you use to obtain a license.

To license the Easysoft ODBC-PostgreSQL Driver:

1. In License Manager, enter your contact details.

   You **must** complete the **Name**, **E-Mail Address**, and **Company** fields.

   The e-mail address **must** be the same as the one used to register at the Easysoft web site. Otherwise, you won't be able to obtain a trial license.

2. Choose **Request License**.

   You're prompted to choose a license type.

3. Do one of the following:
   - For a trial license, choose **Time Limited Trial**, and then choose **Next**.
     -Or-
   - For a purchased license, choose **Non-expiring License**, and then choose **Next**.
4. Choose your product from the drop-down list when prompted, and then choose **Next**.
5. For a purchased license, enter your authorization code when prompted, and then choose **Next**.
6. Choose how to get your license when prompted.
7. Do one of the following:
   - Choose **On-line Request** if your machine is connected to the internet and can make outgoing connections to port 8884.
     With this method, License Manager automatically requests and then applies your license.
     -Or-
   - Choose **View Request**. Then open a web browser and go to https://www.easysoft.com/support/licensing/trial_license.html or https://www.easysoft.com/support/licensing/full_license.html, as appropriate. In the web page, enter your machine number (labelled **Number** in the license request). For purchased licenses, you also need to enter your authorization code (labelled **Ref** in the license request).
     We'll automatically email your license to the email address you supplied in License Manager.

     -Or-
   - Choose **Email Request** to email your license request to our licensing team.
     Once we've processed you request, we'll email your license to the email address you supplied in License Manager.
8. Close the License Manager windows and then choose **Finish**.

If you chose either **View Request** or **Email Request**, apply your license by double-clicking the email attachment when you get the license email from us. Alternatively, start License Manager from the **Easysoft** folder in the Windows **Start** menu. Then choose **Enter License** and paste the license in the space provided.

Once you've licensed the Easysoft ODBC-PostgreSQL Driver, the installation is complete.

Repairing the installation

The installer can repair a broken Easysoft ODBC-PostgreSQL Driver installation. For example, you can use the installer to restore missing Easysoft ODBC-PostgreSQL Driver files or registry keys. To do this:

1. In the Windows **Taskbar**, enter Add or remove programs in the Windows **Search** box.
2. Select Easysoft ODBC-PostgreSQL Driver in the list, and then choose **Repair**.

Uninstalling on Windows

This section explains how to remove the Easysoft ODBC-PostgreSQL Driver from your system.

Removing Easysoft ODBC-PostgreSQL Driver data sources

Easysoft ODBC-PostgreSQL Driver data sources are not removed when you uninstall the Easysoft ODBC-PostgreSQL Driver. You don't therefore need to recreate your Easysoft ODBC-PostgreSQL Driver data sources if you reinstall or upgrade. If you don't want to keep your Easysoft ODBC-PostgreSQL Driver data sources, use Microsoft **ODBC Data Source Administrator** to remove them, **before** uninstalling the Easysoft ODBC-PostgreSQL Driver:

1. In the Windows **Taskbar**, enter Run in the Windows **Search** box.
2. In the Windows **Run** dialog box, enter:

```
odbcad32.exe
```

3. Locate your data source in either the **User** or **System** tab.
4. Select the data source from the list, and then choose **Remove**.

   If the **Remove** button isn't available, close **ODBC Data Source Administrator**, and then, in the Windows **Run** dialog box, enter:

```
%windir%\syswow64\odbcad32.exe
```

   Repeat the previous two steps.

Removing the Easysoft ODBC-PostgreSQL Driver

1. In the Windows **Taskbar**, enter Add or remove programs in the Windows **Search** box.
2. Select Easysoft ODBC-PostgreSQL Driver in the list, and then choose **Uninstall**.

Note	Easysoft product licenses are stored in the Windows registry. When you uninstall, your licenses are not removed, so you do not need to relicense the product if you reinstall or upgrade.

Connecting to PostgreSQL

Applications that support ODBC interface with an ODBC Driver Manager, which is included with the operating system, and also the Easysoft ODBC driver distribution on some platforms. One of the jobs that the ODBC Driver Manager does is to manage ODBC data sources. A data source specifies which ODBC driver to load, which data store to connect to, and how to connect to it.

Before setting up a data source, you must have successfully installed the Easysoft ODBC-PostgreSQL Driver.

Connecting from Linux or UNIX

Creating an ODBC data source

There are two ways to create a data source to your PostgreSQL data:

- Create a SYSTEM data source, which is available to anyone who logs on to the computer where the Easysoft ODBC-PostgreSQL Driver is installed.

  – Or –

- Create a USER data source, which is only available to the user who is currently logged on to the computer where the Easysoft ODBC-PostgreSQL Driver is installed.

By default, the Easysoft ODBC-PostgreSQL Driver installation creates a sample SYSTEM data source named POSTGRES_SAMPLE. If you're using the unixODBC included in the Easysoft ODBC-PostgreSQL Driver distribution, the SYSTEM odbc.ini file is in /etc.

If you built unixODBC yourself, or installed it from some other source, SYSTEM data sources are stored in the path specified with the configure option --sysconfdir=*directory*. If sysconfdir was not specified when unixODBC was configured and built, it defaults to /usr/local/etc.

If you accepted the default choices when installing the PostgreSQL, USER data sources must be created and edited in $HOME/.odbc.ini.

Notes

- To display the directory where unixODBC stores SYSTEM and USER data sources, type odbcinst -j.
- By default, you must be logged in as root to edit a SYSTEM data source defined in /etc/odbc.ini.

You can either edit the sample data source or create new data sources.

Each section of the odbc.ini file starts with a data source name in square brackets [ ] followed by a number of attribute=value pairs.

The Driver attribute identifies the ODBC driver in the odbcinst.ini file to use for a data source. When the Easysoft ODBC-PostgreSQL Driver is installed into unixODBC, it places a Easysoft ODBC-Postgres entry into the odbcinst.ini file. You should always have Driver = Easysoft ODBC-Postgres in your Easysoft ODBC-PostgreSQL Driver data sources therefore.

To configure a Easysoft ODBC-PostgreSQL Driver data source, in your odbc.ini file, you need to specify:

- The hostname or IP address of the PostgreSQL server (Server).
- The database to connect to (Database).
- A user who has permission to access this database (User).
- The password for this user (Password).

For example:

```
[POSTGRES_SAMPLE]
Driver          = Easysoft ODBC-Postgres
Server          = localhost
Database        = Pagila
User            = postgres
Password        = p455w0rd
```

The Easysoft ODBC-PostgreSQL Driver must be able to find the following shared objects:

- libodbcinst.so

  By default, this is located in /usr/local/easysoft/unixODBC/lib/.

- libeslicshr.so

  By default, this is located in /usr/local/easysoft/lib/.

- libessupp.so By default, this is located in /usr/local/easysoft/lib/.

You may need to set and export LD_LIBRARY_PATH, SHLIB_PATH, or LIBPATH (depending on your operating system and run-time linker) to include the directories where libodbcinst.so, libeslicshr.so, and libessupp.so are located.

The isql query tool lets you test your Easysoft ODBC-PostgreSQL Driver data sources. To test the Easysoft ODBC-PostgreSQL Driver connection:

1. Change directory into /usr/local/easysoft/unixODBC/bin.
2. Enter ./isql -v *data_source*, where *data_source* is the name of the target data source.
3. At the prompt, enter an SQL query. For example:

   ```
   SQL> SELECT * FROM rental;
   ```

   –Or–

4. Enter help to return a list of tables:

   ```
   SQL> help
   ```

Connecting from Windows

Creating an ODBC data source

1. In the Windows **Taskbar Search** box, enter “Run”.
2. Do one of the following:
   - If your application is 64-bit, in the **Run** dialog box, enter:

     ```
     odbcad32.exe
     ```

     -Or-

   - If your application is 32-bit, in the **Run** dialog box, enter:

     ```
     %windir%\syswow64\odbcad32.exe
     ```

     Note If you’re not sure whether your application is 32-bit or 64-bit, start your

application, then in Windows **Task Manager** check whether your application's process name contains (32-bit). For example, the process name for the 32-bit version of Excel is Microsoft Excel (32-bit); the process name for the 64-bit version of Excel is Microsoft Excel. On older versions of Windows, 32-bit applications contain *32 in the process name rather than (32-bit).
For applications such as Oracle or SQL Server that run as a service, check the *Background processes* list rather than the **Apps** list in **Task Manager**.
If you're running a programming language from within a Windows command-line shell (for example, Command or PowerShell), in your shell, run the .exe file for the programming language. For example, run perl, php, python, or node. In **Task Manager**, expand the process list for **Windows Command Processor** or **Windows PowerShell**, as appropriate, and check whether the process for your programming language contains (32-bit).

3. Do one of the following:
   - To create a data source that only the user you're currently logged in as can access, choose the **User** tab.
     If your application is a Windows service (for example, SQL Server or Oracle) creating a user data source won't work, unless the service is running as the same user you're logged in as.
   - To create a data source that all users on this computer can access, choose the **System** tab.
4. Choose **Add**.
5. In the list of ODBC drivers, select Easysoft ODBC-PostgreSQL Driver, and then choose **Finish**.
6. Complete the Easysoft ODBC-PostgreSQL Driver configuration dialog box.
   To find out how to do this, refer to the Connection attributes section.
7. To test the connection to PostgreSQL, choose **Test**.
   Note that this doesn't test that the Easysoft ODBC-PostgreSQL Driver is licensed. If you haven't yet licensed the Easysoft ODBC-PostgreSQL Driver, this ODBC data source won't work with your application, even if the **Test** button succeeds.

Connection attributes

Setting on Linux and UNIX

Your Easysoft ODBC-PostgreSQL Driver data source in odbc.ini must contain these attributes:

- Server
- Database
- User
- Password

For example:

```
[POSTGRES_SAMPLE]
Driver          = Easysoft ODBC-Postgres
Server          = localhost
Database        = Pagila
User            = postgres
Password        = p455w0rd
```

These optional attributes may be set in odbc.ini.

- AnsiMode

- Character_Set_Client
- DisguiseLong
- DisguiseWLong
- LimitLong
- Encrypt
- LogFileName
- LoggingEnabled
- LogonTimeout
- TrustServerCertificate
- Port

For more information about these attributes, refer to the following table and the table in the next topic.

Name	Value

Character_Set_Client	Specifies the encoding on the Easysoft ODBC-PostgreSQL Driver machine.
	
	If set, the Easysoft ODBC-PostgreSQL Driver tries to convert to and from the specified encoding when retrieving and submitting character data. For example, if you have an application that is expecting EUC-JP encoded character data, you would need to set Character_Set_Client to specify the EUC-JP encoding:
	
	# Convert to EUC-JP when retrieving data from PostgreSQL.
	Character_Set_Client = EUC-JP
	
	Use Character_Set_Client if you experience data loss or corruption when working with character data and your application cannot convert data to the encoding scheme it expects.
	
	The Easysoft ODBC-PostgreSQL Driver uses a built-in version of iconv to do the conversion. For a list of available encodings for Character_Set_Client, run this command on the machine where the Easysoft ODBC-PostgreSQL Driver is installed:
	
	iconv -l
	
	Set Character_Set_Client to the encoding that corresponds with the LANG environment variable value on the client machine. For example, if LANG was set to en_US.UTF-8 on the client machine, you would set Character_Set_Client to UTF-8.

DisguiseLong	Whether the Easysoft ODBC-PostgreSQL Driver reports long PostgreSQL columns as a VARCHAR type. Don't include both the DisguiseLong attribute and the DisguiseWlong attribute in the same ODBC data source. Use in combination with the LimitLong attribute if you get error "ORA-00997: illegal use of LONG datatype" when using Oracle Heterogeneous Services (DG4ODBC) with PostgreSQL.

Name	Value
DisguiseWLong	Whether the Easysoft ODBC-PostgreSQL Driver reports long PostgreSQL columns as a WVARCHAR type. Do not include both the DisguiseLong attribute and the DisguiseWlong attribute in the same ODBC data source. Use in combination with the LimitLong attribute if you get error "ORA-00997: illegal use of LONG datatype" when using Oracle Heterogeneous Services (DG4ODBC) with PostgreSQL.
LimitLong	The maximum size in bytes that the Easysoft ODBC-PostgreSQL Driver reports for long PostgreSQL columns if you turn on either the DisguiseLong or DisguiseWlong attribute in your ODBC data source.

Setting on Windows

The Easysoft ODBC-PostgreSQL Driver data source configuration dialog box, accessible when you create or edit a Easysoft ODBC-PostgreSQL Driver data source in **ODBC Data Source Administrator** contains these fields:

Name	Value
DSN	The name of the data source. You'll need to specify this in your application. For example, your application may prompt you to choose this from a list of DSNs.
Description	Some applications display this to help users identify a particular data source.
Database	The name of the PostgreSQL database to connect to.
User Name	The PostgreSQL user name, if required.
Password	The password for User Name.
Server	The host name or IP address of the machine on which the PostgreSQL server is running.
Port	The TCP port number that the PostgreSQL server uses to listen for client connections. If you're connecting to a PostgreSQL server that's listening on port 5432, the Port setting can be omitted
SSL Encryption	Whether the Easysoft ODBC-Postgres requests an encrypted connection to PostgreSQL. Currently the Easysoft ODBC-PostgreSQL Driver only supports self-signed SSL certificates.
Trust Cert	Whether to bypass validation of the certificate used by the PostgreSQL server. This setting is only applicable if you turn on SSL encryption for the Easysoft ODBC-PostgreSQL Driver.
Logging	Whether to turn on Easysoft ODBC-PostgreSQL Driver logging. Normally, you'll only do this if so directed by the Easysoft support team.

Name	Value
Log File	The file name and path of the file you want the driver to write log information to. For example: C:\Windows\Temp\Easysoft.log If the file doesn't exist, the Easysoft ODBC-PostgreSQL Driver creates it.
Timeout	The number of seconds to wait for a TCP connection to the PostgreSQL machine to be established before returning to the application and generating a timeout error. The default value 0 means that no login timeout is applied by the Easysoft ODBC-PostgreSQL Driver.
Def. ANSI CP / Non UTF8 app	If your application does not support UTF-8 encoded data, turn on this option. When turned off (the default), the Easysoft ODBC-PostgreSQL Driver returns bound data as wide characters.
Character_Set_Client	If your application does not support UTF-8 encoded data, turn on this option. When turned off (the default), the Easysoft ODBC-PostgreSQL Driver returns bound data as wide characters.

DSN-less connections

Some applications allow you to make an ODBC connection without configuring a data source. To do this, you supply a connection string that contains the ODBC driver name and other driver-specific attribute-value pairs.

Here's an example Easysoft ODBC-PostgreSQL Driver connection string:

```
Driver={Easysoft Postgres ODBC
Driver};SERVER=localhost;UID=myuser;PWD=mypassword;DATABASE=Pagila;
```

For a list of the other attributes you can set in the connection string, refer to this section.

Logging

If you report an issue to us, we may ask you to turn on ODBC Driver Manager or Easysoft ODBC-PostgreSQL Driver logging, to help us diagnose the cause of the issue.

To turn on logging, refer to the following sections.

Note	If your application is a service (for example, Oracle or SQL Server), you may need to restart the service before enabling logging takes effect. To do this on Linux or UNIX, use service, systemctl, or a vendor-supplied script. To do this on Windows, use the Windows **Services** app.

ODBC Driver Manager logging on Linux or UNIX

For the unixODBC Driver Manager, add the following attributes to the [ODBC] section (create one if none exists) in odbcinst.ini.

```
Trace = Yes
TraceFile = /path/filename
```

For example:

```
[ODBC]
Trace = Yes
TraceFile = /tmp/sql.log
```

Ensure that the user who's running the application to log has write permission to TraceFile (and to the directory containing it), otherwise no logging information will be produced.

Easysoft ODBC-PostgreSQL Driver logging on Linux and UNIX

Driver manager trace files show all the ODBC calls an application makes, including their arguments and return values. Easysoft ODBC-PostgreSQL Driver logging is specific to the Easysoft driver and is of most use when making a support call.

To turn on Easysoft ODBC-PostgreSQL Driver logging, edit your ODBC data source in odbc.ini. For example:

```
[POSTGRES_SAMPLE]
.
.
Logging = Yes
LogFile = /tmp/easysoft-odbc-driver.log
```

The value shown in the example specifies a log file named /tmp/easysoft-odbc-driver.log. Ensure that the user who's running the application to log has write permission to the log file (and to the directory containing it), otherwise no logging information will be produced.

ODBC Driver Manager logging on Windows

1. In the Windows **Taskbar Search** box, enter "Run".
2. Do one of the following:
   - If your application is 64-bit, in the **Run** dialog box, enter:

```
odbcad32.exe
```

-Or-

- If your application is 32-bit, in the **Run** dialog box, enter:

```
%windir%\syswow64\odbcad32.exe
```

> **Note**
>
> If you're not sure whether your application is 32-bit or 64-bit, start your application, then in Windows **Task Manager** check whether your application's process name contains (32-bit). For example, the process name for the 32-bit version of Excel is Microsoft Excel (32-bit); the process name for the 64-bit version of Excel is Microsoft Excel. On older versions of Windows, 32-bit applications contain *32 in the process name rather than (32-bit).
> For applications such as Oracle or SQL Server that run as a service, check the *Background processes* list rather than the **Apps** list in **Task Manager**.
> If you're running a programming language from within a Windows command-line shell (for example, Command or PowerShell), in your shell, run the .exe file for the programming language. For example, run perl, php, python, or node. In **Task Manager**, expand the process list for **Windows Command Processor** or **Windows PowerShell**, as appropriate, and check whether the process for your programming language contains (32-bit).

3. Choose the **Tracing** tab.
4. Select **Machine-Wide tracing for all identities**.
5. Enter a log file name and path in the space provided. For example:

```
C:\Windows\Temp\SQL.log
```

6. Choose **Start Tracing Now**.

Note With SQL Server, you may get two Driver Manager log files, we need both. The first log file is in the folder that you specify in **ODBC Data Source Administrator**. The second file's location is defined by SQL Server. Two possible locations are the top-level folder (for example, C:\SQL.log) or the SQL Server temporary folder (for example, C:\Users\MSSQL$SQLEXPRESS\AppData\Local\Temp\SQL.log). If the Driver Manager log file isn't in these folders, search for it on the drive where SQL Server is installed.

Easysoft ODBC-PostgreSQL Driver logging on Windows

1. In the Windows **Taskbar Search** box, enter "Run".
2. Do one of the following:
   - If your application is 64-bit, in the **Run** dialog box, enter:

     ```
     odbcad32.exe
     ```

     -Or-
   - If your application is 32-bit, in the **Run** dialog box, enter:

     ```
     %windir%\syswow64\odbcad32.exe
     ```

     Note If you're not sure whether your application is 32-bit or 64-bit, start your application, then in Windows **Task Manager** check whether your application's process name contains (32-bit). For example, the process name for the 32-bit version of Excel is Microsoft Excel (32-bit); the process name for the 64-bit version of Excel is Microsoft Excel. On older versions of Windows, 32-bit applications contain *32 in the process name rather than (32-bit).
     For applications such as Oracle or SQL Server that run as a service, check the *Background processes* list rather than the **Apps** list in **Task Manager**.
     If you're running a programming language from within a Windows command-line shell (for example, Command or PowerShell), in your shell, run the .exe file for the programming language. For example, run perl, php, python, or node. In **Task Manager**, expand the process list for **Windows Command Processor** or **Windows PowerShell**, as appropriate, and check whether the process for your programming language contains (32-bit).
3. Do one of the following:
   - If you configured a system data source, choose the **System DSN** tab.
     -Or-
   - If you configured a system data source, choose the **System DSN** tab.
4. Choose your Easysoft ODBC-PostgreSQL Driver data source from the list, and then choose **Configure**.
5. In the Easysoft ODBC-PostgreSQL Driver data source configuration dialog box, turn on **Driver Logging**.
6. Enter a log file name and path in the space provided. For example:

   ```
   C:\Windows\Temp\Easysoft.log
   ```

Finding out what product version you have on Windows

If you have an issue with the Easysoft ODBC-PostgreSQL Driver, we may ask you to tell us what your product version is. To find this out:

1. In the Windows **Taskbar**, enter “Add or remove programs” in the Windows **Search** box.
2. Select Easysoft ODBC-PostgreSQL Driver in the list.

   The product version displays below.

Client applications

How to work with PostgreSQL data in some example applications and programming languages:

- Microsoft Access
- Microsoft Excel
- Microsoft Power BI
- SQL Server
- Oracle
- LibreOffice
- Go
- Node.js
- Perl
- PHP
- Python
- R

Microsoft Access

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as Access.
2. Configure an ODBC data source.
3. Choose one of the following ways to work with your PostgreSQL data in Access.

Linking a table

1. Open your Microsoft Access database.
2. Choose **External Data**.
3. In the **New Data Source** list, choose **From Other Sources > ODBC Database**.
4. In the **Get External Data** screen, choose **Link to the data source by creating a linked table**, and choose **OK**.
5. In the **Select Data Source** dialog box, choose the **Machine Data Source** tab.
6. Choose your Easysoft ODBC-PostgreSQL Driver ODBC data source from the **Machine Data Source** list, and then choose **OK**.
7. In the **Link Tables** dialog box, choose the tables that you want to link to, and then choose **OK**.

Importing a table

1. Open your Microsoft Access database.
2. Choose **External Data**.
3. In the **New Data Source** list, choose **From Other Sources > ODBC Database**.
4. In the **Get External Data** screen, choose **Import the source data into a new table in the current database**, and choose **OK**.
5. In the **Select Data Source** dialog box, choose the **Machine Data Source** tab.
6. Choose your Easysoft ODBC-PostgreSQL Driver ODBC data source from the **Machine Data Source** list, and then choose **OK**.
7. In the **Import Objects** dialog box, choose the tables you want to import, and then choose **OK**.

Microsoft Excel

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as Excel.
2. Configure an ODBC data source.
3. Choose one of the following ways to work with your PostgreSQL data in Excel.

Data Connection Wizard

1. Choose **Data > Get Data > From Other Sources > From ODBC**.
2. Choose your Easysoft ODBC-PostgreSQL Driver data source from the list, and then choose **OK**.
3. Enter the user name and password for your data store if applicable, otherwise, enter any text string to get past this stage. Choose **Next**.
4. Choose the table that contains the data you want to retrieve, and then choose **Load**.

Microsoft Query

1. Choose **Data > Get Data > From Other Sources > From Microsoft Query**.
2. In the **Choose Data Source** dialog box, choose your PostgreSQL data source from the list, and then choose **OK**.
3. In the **Query Wizard**, choose the columns that contain the data you want to retrieve, and then click **Next**.
4. If you want to return a subset of the data, use the **Filter Data** screen to filter the results of your query (this is the equivalent of a SQL WHERE clause), and then choose **Next**.
5. If you want to change the sort order of your data, use the **Sort Order** screen to sort the results of your query (this is the equivalent of a SQL ORDER BY clause), and then choose **Next**. Choose **Finish** to return your PostgreSQL data to Excel.

PowerPivot

1. On the **PowerPivot** tab, choose **Manage**.
2. In the **PowerPivot** window, choose **Get External Data > From Other Sources**.
3. In the **Connect to a Data Source** list, choose **Others (OLEDB/ODBC)**
4. In the **Specify a Connection** screen, enter a name for your connection in the space provided. Then choose **Build**.
5. In the **Data Link Properties** box, choose your Easysoft ODBC-PostgreSQL Driver data source from the list, and then choose **OK**.
6. Choose **Next**.
7. Choose how to import your PostgreSQL data and then choose **Finish**.
8. Choose **Close** to return the data to Excel.

Microsoft Power BI

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as Power BI Desktop.
2. Configure an ODBC data source.
3. In Power BI Desktop, choose **Get data from another source**.
4. In the **Get Data** dialog box, choose **ODBC**, and then choose **Connect**.
5. In the **From ODBC** dialog box, choose your PostgreSQL data source, and then choose **OK**.
6. Enter your database user name and password when prompted.

   If you make a mistake when entering the user name and password, cancel the connection process. Then in Power BI Desktop **Options and Settings**, edit the data source. Specify the correct user name or password in the data source credentials dialog box. Otherwise, Power BI Desktop will continue to use the cached incorrect credentials.

Note	If you do not normally need to enter a user name and password, enter some dummy strings in the spaces provided.

7. In the **Navigator** dialog box, choose the tables you want to analyse in Power BI Desktop, and then choose **Load**.

   Your PostgreSQL data is now available to use in Power BI visualisations.

Microsoft SQL Server

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as SQL Server.
2. Configure an ODBC data source.
3. In Microsoft SQL Server Management Studio, connect to the SQL Server instance you want to create the linked server against.

   You need to log on with an account that is a member of the SQL Server sysadmin fixed server role to create a linked server.
4. Right-click **Server Objects**. From the pop-up menu choose **New > Linked Server**.
5. In the **Linked server** box, enter "PostgreSQL".
6. From the **Provider** list, choose **Microsoft OLE DB Provider for ODBC drivers**.
7. In the **Data source** box, enter the name of your PostgreSQL data source, and then choose **OK**.

   SQL Server verifies the linked server by testing the connection.

   - If you get the error "Specified driver could not be loaded due to system error 126: The specified module could not be found," choose **Yes** when prompted whether to keep the linked server. You need to restart your SQL Server instance before you can use the linked server. If SQL Server was already running when you installed the Easysoft ODBC-PostgreSQL Driver, it will not have the latest version of the System Path environment variable. The Easysoft ODBC-PostgreSQL Driver Setup program adds entries for the driver to the System Path. Restarting the instance makes these changes available to SQL Server, allowing it to load the Easysoft ODBC-PostgreSQL Driver.
   - If you made a mistake when specifying the Easysoft ODBC-PostgreSQL Driver, you get the error "Data source name not found and no default driver specified." If you get this error, choose **No** when prompted whether to keep the linked server and edit the value in the **Data source** box.
8. You can query your Easysoft ODBC-PostgreSQL Driver data either by using a:
   - Four part table name in a distributed query.

     A four part table name has the format:

     ```
     server_name.[database_name].[schema_name].table_name
     ```

     For data stores where there is no database or schema, Easysoft ODBC drivers return a "dummy" value for both identifiers, because some ODBC applications expect there to be a database and a schema. To find out the identifier names, run:

     ```
     EXEC sp_tables_ex @table_server = 'PostgreSQL'
     ```

     If present, include these identifiers in your SQL statements. If not present, omit them. For example:

     ```
     SELECT * FROM [PostgreSQL]..DBO.Customers
     ```

     The capitalisation of the table name must be the same as it is in the result set returned by sp_tables_ex.
   - Pass-through query in an OPENQUERY function. For example:

     ```
     SELECT * FROM OPENQUERY([PostgreSQL], 'SELECT * FROM Customers')

     -- If you get an "RPC not enabled for this server" message, right-click your
     linked server and choose Properties.
     ```

```
-- In Server Options, set both RPC and RPC Out to `True`.
EXEC ('INSERT INTO Customers (Surname, GivenName, City, Phone, CompanyName)
VALUES (''Devlin'' , ''Michaels'' , ''Kingston'' , ''2015558966'' ,
''PowerGroup'')')
AT PostgreSQL

UPDATE OPENQUERY ([PostgreSQL], 'SELECT Surname FROM Customers WHERE
CompanyName = ''PowerGroup''') SET Surname='Jones'
DELETE OPENQUERY (PostgreSQL, 'SELECT Surname FROM Customers WHERE CompanyName
= ''PowerGroup''')
```

SQL Server sends pass-through queries as uninterpreted query strings to the PostgreSQL. This means that SQL Server does not apply any kind of logic to the query or try to estimate what that query will do.

Oracle

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as Oracle.
2. Configure an ODBC data source.
3. Follow the instructions for your Oracle platform.

Connecting PostgreSQL to Oracle on Windows

1. Create a DG4ODBC init file on your Oracle machine. To do this, change to the %ORACLE_HOME%\hs\admin directory. Create a copy of the file initdg4odbc.ora. Name the new file initPostgresql.ora.

Note	In these instructions, replace %ORACLE_HOME% with the location of your Oracle HOME directory. For example, C:\app\product\21c\homes\OraDB21Home1.

2. Ensure these parameters and values are present in your init file:

   ```
   HS_FDS_CONNECT_INFO = "PostgreSQL"
   ```

   Replace PostgreSQL with the name of your Easysoft ODBC-PostgreSQL Driver data source.

3. Comment out the line that enables DG4ODBC tracing. For example:

   ```
   #HS_FDS_TRACE_LEVEL = <trace_level>
   ```

4. Add an entry to %ORACLE_HOME%\network\admin\listener.ora that creates a SID_NAME for DG4ODBC. For example:

   ```
   SID_LIST_LISTENER =
    (SID_LIST =
      (SID_DESC=
        (SID_NAME=Postgresql)
        (ORACLE_HOME=%ORACLE_HOME%)
        (PROGRAM=dg4odbc)
      )
    )
   ```

5. Add a DG4ODBC entry to %ORACLE_HOME%\network\admin\tnsnames.ora that specifies the SID_NAME created in the previous step. For example:

   ```
   Postgresql =
     (DESCRIPTION =
       (ADDRESS = (PROTOCOL = TCP)(HOST = oracle_host)(PORT = 1521))
       (CONNECT_DATA =
         (SID = Postgresql)
       )
       (HS = OK)
     )
   ```

   Replace oracle_host with the host name of your Oracle machine.

6. Start (or restart) the Oracle Listener:

   ```
   cd %ORACLE_HOME%\bin
   lsnrctl stop
   ```

```
lsnrctl start
```

7. Connect to your Oracle database in SQL*Plus.
8. In SQL*Plus, create a database link for PostgreSQL. For example:

```
CREATE PUBLIC DATABASE LINK PostgresqlLink
    CONNECT TO "dbuser" IDENTIFIED BY "dbpassword"
    USING 'Postgresql';
```

   Replace dbuser and dbpassword with your backend user name and password, if applicable.

9. Try querying and updating your PostgreSQL data. For example:

```
SELECT "Surname" FROM "Customers"@PostgresqlLink;

DECLARE
  num_rows integer;
BEGIN
num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@PostgresqlLink
('INSERT INTO Customers (Surname, GivenName, City, Phone, CompanyName) VALUES
(''Devlin'', ''Michaels'', ''Kingston'', ''2015558966'', ''PowerGroup'')');
END;
/

DECLARE
  num_rows integer;
BEGIN
num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@PostgresqlLink
('UPDATE "Customers" SET "Surname" = ''Jones'' WHERE "CompanyName" =
''PowerGroup''');
END;
/

DECLARE
  num_rows integer;
BEGIN
num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@PostgresqlLink
('DELETE from "Customers" WHERE CompanyName = ''PowerGroup''');
END;
/
```

Notes

- If you have problems connecting to PostgreSQL from Oracle, enable DG4ODBC tracing and check the trace files written to the %ORACLE_HOME%\hs\trace directory. To enable DG4ODBC tracing, add the line HS_FDS_TRACE_LEVEL = DEBUG to initPostgreSQL.ora and then start or restart the Oracle listener. If the trace directory does not exist, create it.
- If you enable ODBC Driver Manager tracing, but do not get a log file in the location you specify, try looking in the top-level folder (for example, C:\SQL.log). Alternatively, in **ODBC Data Source Administrator**, change the trace file location to the Windows TEMP directory.

Connecting PostgreSQL to Oracle on Linux and UNIX

1. Create a DG4ODBC init file on your Oracle machine. To do this, change to the $ORACLE_HOME\hs\admin directory. Create a copy of the file initdg4odbc.ora. Name the new file initPostgresql.ora.

Note	In these instructions, replace $ORACLE_HOME with the location of your Oracle HOME directory. For example, /u01/app/oracle/product/21c/dbhome_1.

2. Ensure these parameters and values are present in your init file:

   ```
   HS_FDS_CONNECT_INFO = "PostgreSQL"
   ```

   Replace PostgreSQL with the name of your Easysoft ODBC-PostgreSQL Driver data source.

3. Comment out the line that enables DG4ODBC tracing. For example:

   ```
   #HS_FDS_TRACE_LEVEL = <trace_level>
   ```

4. Add an entry to $ORACLE_HOME/network/admin/listener.ora that creates a SID_NAME for DG4ODBC. For example:

   ```
   SID_LIST_LISTENER =
    (SID_LIST =
      (SID_DESC=
        (SID_NAME=Postgresql)
        (ORACLE_HOME=$ORACLE_HOME)
        (PROGRAM=dg4odbc)
        (ENVS=LD_LIBRARY_PATH = /usr/local/easysoft/unixODBC/lib:
               /usr/local/easysoft/lib)
      )
    )
   ```

   Replace oracle_home_directory with the value of $ORACLE_HOME. For example, /u01/app/oracle/product/21c/dbhome_1.

5. Add a DG4ODBC entry to $ORACLE_HOME/network/admin/tnsnames.ora that specifies the SID_NAME created in the previous step. For example:

   ```
   Postgresql =
     (DESCRIPTION =
       (ADDRESS = (PROTOCOL = TCP)(HOST = oracle_host)(PORT = 1521))
       (CONNECT_DATA =
         (SID = Postgresql)
       )
       (HS = OK)
     )
   ```

   Replace oracle_host with the host name of your Oracle machine.

6. Start (or restart) the Oracle Listener:

   ```
   cd $ORACLE_HOME/bin
   ./lsnrctl stop
   ./lsnrctl start
   ```

7. Connect to your Oracle database in SQL*Plus.

8. In SQL*Plus, create a database link for PostgreSQL. For example:

```
CREATE PUBLIC DATABASE LINK PostgresqlLink
    CONNECT TO "dbuser" IDENTIFIED BY "dbpassword"
    USING 'Postgresql';
```

   Replace dbuser and dbpassword with your backend user name and password, if applicable.

9. Try querying and updating your PostgreSQL data. For example:

```
SELECT "Surname" FROM "Customers"@PostgresqlLink;

DECLARE
  num_rows integer;
BEGIN
num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@PostgresqlLink
('INSERT INTO Customers (Surname, GivenName, City, Phone, CompanyName) VALUES
(''Devlin'', ''Michaels'', ''Kingston'', ''2015558966'', ''PowerGroup'')');
END;
/

DECLARE
  num_rows integer;
BEGIN
num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@PostgresqlLink
('UPDATE "Customers" SET "Surname" = ''Jones'' WHERE "CompanyName" =
''PowerGroup''');
END;
/

DECLARE
  num_rows integer;
BEGIN
num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@PostgresqlLink
('DELETE from "Customers" WHERE CompanyName = ''PowerGroup''');
END;
/
```

Notes

- If you have problems connecting to PostgreSQL from Oracle, enable DG4ODBC tracing and check the trace files written to the $ORACLE_HOME/hs/trace directory. To enable DG4ODBC tracing, add the line HS_FDS_TRACE_LEVEL = DEBUG to initPostgreSQL.ora and then start or restart the Oracle listener. If the trace directory does not exist, create it.

LibreOffice

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as LibreOffice.
2. Configure an ODBC data source.
3. Choose **File > New > Database**.
4. Choose **Connect to an existing database**.
5. Choose **ODBC** in the list, and then choose **Next**.
6. Choose **Browse**, double-click your data source, and then choose **Next**.
7. If your database requires a database user name, enter it in the **User name** box. If this user needs to supply a password choose the **Password required** check box.
8. Choose **Finish**.
9. Save the database when prompted.

   The database opens in a new Base window. From here you can access your data.

10. In the left pane of the database window, choose the **Tables** icon to display a hierarchy of tables. Enter the database password if prompted, and then choose **OK**.
11. To retrieve the data in a table, in the **Tables** pane, double-click a table.
12. Choose the **Queries** icon to create a query.

    Use any of the methods listed in the **Tasks** pane to create a query.

Go

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as Go.
2. Configure an ODBC data source.
3. Install the odbc package for Go:

```
go mod init test
go get github.com/alexbrainman/odbc
```

4. Create and then use Go to run this script, which retrieves some PostgreSQL data:

```
package main

import (
    _ "github.com/alexbrainman/odbc"
    "database/sql"
        "log"
)

func main() {
        // Replace the DSN value with the name of your ODBC data source.
        db, err := sql.Open("odbc",
                "DSN=PostgreSQL")
        if err != nil {
                log.Fatal(err)
        }

        var (
                name string
        )

        rows, err := db.Query("SELECT Surname FROM Customers")
        if err != nil {
                log.Fatal(err)
        }
        defer rows.Close()
        for rows.Next() {
                err := rows.Scan(&name)
                if err != nil {
                        log.Fatal(err)
                }
                log.Println(name)
        }
        err = rows.Err()
        if err != nil {
                log.Fatal(err)
        }

        defer db.Close()
}
```

Node.js

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as Node.js.
2. Configure an ODBC data source.
3. Install the odbc module for Node.js:

```
npm install odbc
```

4. Create and then use Node.js to run this script, which retrieves some PostgreSQL data:

```
const odbc = require('odbc');
// Replace PostgreSQL with the name of your Easysoft ODBC-PostgreSQL Driver
// data source.
const connection = odbc.connect('DSN=PostgreSQL', (error, connection) => {
    connection.query('SELECT Surname FROM Customers', (error, result) => {
        if (error) { console.error(error) }
        console.log(result);
    });
});
```

5. This script retrieves the tables and views in your Easysoft ODBC-PostgreSQL Driver data source:

```
const odbc = require('odbc');
const connection = odbc.connect('DSN=PostgreSQL', (error, connection) => {
    connection.tables(null, null, null, null, (error, result) => {
        if (error) { return; }
        const util = require('util');
        console.log(util.inspect(result, {maxArrayLength: null, depth:null}))
    });
});
```

6. This script retrieves the names of the columns in these tables and views:

```
const odbc = require('odbc');
const connection = odbc.connect('DSN=PostgreSQL', (error, connection) => {
    connection.columns(null, null, null, null, (error, result) => {
        if (error) { return; }
        const util = require('util');
        console.log(util.inspect(result, {maxArrayLength: null, depth:null}))
    });
});
```

7. These scripts insert, update, and then delete some PostgreSQL data:

```
const odbc = require("odbc");
const connection = odbc.connect("DSN=PostgreSQL", (error, connection) => {
    connection.query("INSERT INTO
    Customers (
        Surname,
        GivenName,
        City,
        Phone,
        CompanyName
```

```
    )
VALUES
    (
        'Devlin',
        'Michaels',
        'Kingston',
        '2015558966',
        'PowerGroup'
    )", (error, result) => {
        if (error) { console.error(error) }
        console.log(result);
    });
});

const odbc = require("odbc");
const connection = odbc.connect("DSN=PostgreSQL", (error, connection) => {
    connection.query("UPDATE Customers SET Surname = 'Jones' WHERE CompanyName =
'PowerGroup'", (error, result) => {
        if (error) { console.error(error) }
        console.log(result);
    });
});

const odbc = require("odbc");
const connection = odbc.connect("DSN=PostgreSQL", (error, connection) => {
    connection.query("DELETE FROM Customers WHERE CompanyName = 'PowerGroup'",
(error, result) => {
        if (error) { console.error(error) }
        console.log(result);
    });
});
```

Perl

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as Perl.
2. Configure an ODBC data source.
3. Check whether your Perl distribution supports ODBC:

```
perl -e 'use DBD::ODBC;'
```

4. Do one of the following:
   - If you get no output, your Perl distribution supports ODBC. Skip to the next step.
   - If you get:

     ```
     Can't locate DBD/ODBC.pm
     ```

     you need to install DBD::ODBC before you can use the Easysoft ODBC-PostgreSQL Driver to connect to PostgreSQL.
5. Create and then use Perl to run this script, which retrieves some PostgreSQL data:

```
use strict;
use DBI;
# Replace PostgreSQL with the name of your Easysoft ODBC-PostgreSQL Driver data
source.
my $dbh = DBI-> connect('dbi:ODBC:PostgreSQL');

my $sql = "SELECT Surname FROM Customers";

my $sth = $dbh->prepare($sql)
    or die "Can't prepare statement: $DBI::errstr";

$sth->execute();

my($Col);

# Fetch and display the result set values.
while(($Col) = $sth->fetchrow()){
   print("$Col\n");
}

$dbh->disconnect if ($dbh);
```

6. This script retrieves the tables and views in your Easysoft ODBC-PostgreSQL Driver data source:

```
use strict;
use DBI;
my $dbh = DBI-> connect('dbi:ODBC:PostgreSQL');

my $sth = $dbh->table_info()
    or die "Can't prepare statement: $DBI::errstr";

my @row;

while (@row = $sth->fetchrow_array) {
```

```
    print join(", ", @row), "\n";
}

$dbh->disconnect if ($dbh);
```

7. This script retrieves the names of the columns in these tables and views:

```
use strict;
use DBI;
my $dbh = DBI-> connect('dbi:ODBC:PostgreSQL');

my $sth = $dbh->column_info('','','','')
    or die "Can't prepare statement: $DBI::errstr";

my @row;
while (@row = $sth->fetchrow_array) {
    print join(", ", @row), "\n";
}

$dbh->disconnect if ($dbh);
```

8. These scripts insert, update, and then delete some PostgreSQL data:

```
use strict;
use DBI;
my $dbh = DBI-> connect('dbi:ODBC:PostgreSQL');

my $sth = $dbh->prepare(q/INSERT INTO Customers (Surname, GivenName, City, Phone,
CompanyName) VALUES (?, ?, ?, ?, ?)/)
    or die "Can't prepare statement: $DBI::errstr";

$sth->execute('Devlin', 'Michaels', 'Kingston', '2015558966', 'PowerGroup');

$dbh->disconnect if ($dbh);

use strict;
use DBI;
my $dbh = DBI-> connect('dbi:ODBC:PostgreSQL');

my $sth = $dbh->prepare('UPDATE Customers SET Surname = \'Jones\' WHERE
CompanyName = ?')
    or die "Can't prepare statement: $DBI::errstr";

$sth->execute('PowerGroup');

$dbh->disconnect if ($dbh);

use strict;
use DBI;
my $dbh = DBI-> connect('dbi:ODBC:PostgreSQL');

my $sth = $dbh->prepare('DELETE FROM Customers WHERE CompanyName = ?')
```

```
    or die "Can't prepare statement: $DBI::errstr";

$sth->execute('PowerGroup');

$dbh->disconnect if ($dbh);
```

Further information

- Perl DBI DBD::ODBC tutorial: Drivers, data sources, and connection

PHP

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as PHP.
2. Configure an ODBC data source.
3. Check whether your PHP distribution supports ODBC. In php.ini, make sure there is no comment character (;) before the extension_dir and extension=odbc settings (;extension_dir=*directory* becomes extension_dir=*directory* and ;extension=odbc becomes extension=odbc).
4. Create and then use PHP to run this script, which retrieves some PostgreSQL data:

```
<?php
  // Replace PostgreSQL with the name of your Easysoft ODBC-PostgreSQL Driver
data source.
  //  If your database requires a user name and password, supply them in the
odbc_connect_call.
  $con = odbc_connect("PostgreSQL", "", "");
  $stmt = odbc_exec($con, "SELECT * FROM Customers");
  // You may need to change the capitalisation of Surname to all upper case or
all lower case.
  while ($row = odbc_fetch_array($stmt)) {
      echo "Surname = ", $row["Surname"], "\n";
  }
  odbc_close($con);
?>
```

5. This script retrieves the tables and views in your Easysoft ODBC-PostgreSQL Driver data source:

```
<?php
$con = odbc_connect("PostgreSQL", "", "");
$tables = odbc_tables($con);
while (($row = odbc_fetch_array($tables))) {
    print_r($row);
}
odbc_close($con);
?>
```

6. This script retrieves the names of the columns in these tables and views:

```
<?php
$con = odbc_connect("PostgreSQL", "", "");
$columns = odbc_columns($con);
while (($row = odbc_fetch_array($columns))) {
    print_r($row);
}
odbc_close($con);
?>
```

7. These scripts insert, update, and then delete some PostgreSQL data:

```
<?php
  $cnx = odbc_connect("PostgreSQL", "", "");
  $stmt = odbc_prepare($cnx, "INSERT INTO Customers (Surname, GivenName, City,
Phone, CompanyName) VALUES (?, ?, ?, ?, ?)");
```

```
  $success = odbc_execute($stmt, array('Devlin', 'Michaels', 'Kingston',
'2015558966', 'PowerGroup'));
  odbc_close($cnx);
?>

<?php
  $cnx = odbc_connect("PostgreSQL", "", "");
  $stmt = odbc_prepare($cnx, "UPDATE Customers SET Surname = 'Jones' WHERE
CompanyName = ?");
  $success = odbc_execute($stmt, array('PowerGroup'));
  odbc_close($cnx);
?>

<?php
  $cnx = odbc_connect("PostgreSQL", "", "");
  $stmt = odbc_prepare($cnx, "DELETE FROM Customers WHERE CompanyName = ?");
  $success = odbc_execute($stmt, array('PowerGroup'));
  odbc_close($cnx);
?>
```

Further information

- Easysoft PHP tutorials and code samples

Python

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as Python.
2. Configure an ODBC data source.
3. Check whether your Python distribution supports ODBC.

```
pip list
```

   If you don't have pip installed:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python get-pip.py
```

4. Do one of the following:
   - If the output contains pyodbc, your Python distribution supports ODBC. Skip to the next step.
   - If the output does not contain pyodbc, use pip to install this module:

```
pip install pyodbc
```

5. Create and then use Python to run this script, which retrieves some PostgreSQL data:

```
import pyodbc

# Replace PostgreSQL with the name of your Easysoft ODBC-PostgreSQL Driver data
source.
cnxn = pyodbc.connect("DSN=PostgreSQL")
cursor = cnxn.cursor()
sql = "SELECT Surname FROM Customers"
cursor.execute(sql)
rows = cursor.fetchall()
# You may need to change the capitalisation of Surname to all upper case or all
lower case.
for row in rows:
     print(row.Surname)
exit()
```

6. This script retrieves the tables and views in your Easysoft ODBC-PostgreSQL Driver data source:

```
import pyodbc

# Replace PostgreSQL with the name of your Easysoft ODBC-PostgreSQL Driver data
source.
cnxn = pyodbc.connect("DSN=PostgreSQL")
cursor = cnxn.cursor()
cursor.tables()
rows = cursor.fetchall()
for row in rows:
     print(row.table_name)
exit()
```

7. This script retrieves the names of the columns in these tables and views:

```
import pyodbc
```

```
# Replace PostgreSQL with the name of your Easysoft ODBC-PostgreSQL Driver data
source.
cnxn = pyodbc.connect("DSN=PostgreSQL")
cursor = cnxn.cursor()
cursor.columns()
rows = cursor.fetchall()
for row in rows:
     print(row.table_name, row.column_name)
exit()
```

8. These scripts insert, update, and then delete some PostgreSQL data:

```
import pyodbc

cnxn = pyodbc.connect("DSN=PostgreSQL")
cursor = cnxn.cursor()
sql = "INSERT INTO Customers (Surname, GivenName, City, Phone, CompanyName)
VALUES (?, ?, ?, ?, ?)"
cursor.execute(sql, 'Devlin', 'Michaels', 'Kingston', '2015558966', 'PowerGroup')
cursor.commit()
exit()
```

```
import pyodbc

cnxn = pyodbc.connect("DSN=PostgreSQL")
cursor = cnxn.cursor()
sql = "UPDATE Customers SET Surname = 'Jones' WHERE CompanyName = ?"
cursor.execute(sql, 'PowerGroup')
cursor.commit()
exit()
```

```
import pyodbc

cnxn = pyodbc.connect("DSN=PostgreSQL")
cursor = cnxn.cursor()
sql = "DELETE FROM Customers WHERE CompanyName = ?"
cursor.execute(sql, 'PowerGroup')
cursor.commit()
exit()
```

Further information

- Easysoft Python tutorials and code samples

R

1. Install the Easysoft ODBC-PostgreSQL Driver on same computer as R.
2. Configure an ODBC data source.
3. In R Console, check whether your R distribution supports ODBC.

```
library("RODBC")
```

4. Do one of the following:
   - If you get no output, you have the ODBC library for R. Skip to the next step.
   - If you get an "there is no package" error, install the ODBC library for R:

```
install.packages("RODBC")
```

5. Create and then use R to run this script, which retrieves some PostgreSQL data:

```
library("RODBC")
# Replace PostgreSQL with the name of your Easysoft ODBC-PostgreSQL Driver data
source.
ch <- odbcConnect("PostgreSQL")
sqlQuery(ch, paste("SELECT Surname FROM Customers"))
odbcClose(ch)
quit()
```

6. This script retrieves the tables and views in your Easysoft ODBC-PostgreSQL Driver data source:

```
library("RODBC")
# Replace PostgreSQL with the name of your Easysoft ODBC-PostgreSQL Driver data
source.
ch <- odbcConnect("PostgreSQL")
sqlTables(ch)
odbcClose(ch)
quit()
```

7. This script retrieves the names of the columns in the specified table or view:

```
library("RODBC")
# Replace PostgreSQL with the name of your Easysoft ODBC-PostgreSQL Driver data
source.
ch <- odbcConnect("PostgreSQL")
# You may need to change the capitalisation of Customers to all upper case or all
lower case.
sqlColumns(ch, sqtable="Customers")
odbcClose(ch)
quit()
```

8. These scripts insert, update, and then delete some PostgreSQL data:

```
library("RODBC")
ch <- odbcConnect("PostgreSQL")
sqlQuery(ch, paste("INSERT INTO Customers (Surname, GivenName, City, Phone,
CompanyName) VALUES ('Devlin', 'Michaels', 'Kingston', '2015558966',
'PowerGroup')"))
```

```
odbcClose(ch)
quit()

library("RODBC")
ch <- odbcConnect("PostgreSQL")
sqlQuery(ch, paste("UPDATE Customers SET Surname = 'Jones' WHERE CompanyName =
'PowerGroup'"))
odbcClose(ch)
quit()

library("RODBC")
ch <- odbcConnect("PostgreSQL")
sqlQuery(ch, paste("DELETE FROM Customers WHERE CompanyName = 'PowerGroup'"))
odbcClose(ch)
quit()
```

About the Easysoft ODBC-PostgreSQL Driver

The Easysoft ODBC-PostgreSQL Driver provides real-time access to PostgreSQL data from any application that supports ODBC.

ODBC API and scalar functions

API functions

Use this table to find out what ODBC API functions the Easysoft ODBC-PostgreSQL Driver supports:

Function	Status
SQLAllocConnect	Supported
SQLAllocEnv	Supported
SQLAllocHandle	Supported
SQLAllocStmt	Supported
SQLBindCol	Supported
SQLBindParameter	Supported
SQLBrowseConnect	Not supported
SQLBulkOperations	Supported
SQLCancel	Supported
SQLCloseCursor	Supported
SQLColAttribute	Supported
SQLColAttributes	Supported
SQLColumnPrivileges	Supported
SQLColumns	Supported
SQLConnect	Supported
SQLCopyDesc	Supported
SQLDisconnect	Supported
SQLDriverConnect	Supported
SQLDrivers	Supported
SQLEndTran	Supported
SQLError	Supported
SQLExecDirect	Supported
SQLExecute	Supported
SQLExtendedFetch	Supported
SQLFetch	Supported
SQLFetchScroll	Supported
SQLForeignKeys	Supported
SQLFreeConnect	Supported
SQLFreeEnv	Supported
SQLFreeHandle	Supported

Function	Status
SQLFreeStmt	Supported
SQLGetConnectAtt	Supported
SQLGetConnectOption	Supported
SQLGetCursorName	Supported
SQLGetData	Supported
SQLGetDescField	Supported
SQLGetDescRec	Supported
SQLGetDiagField	Supported
SQLGetDiagRec	Supported
SQLGetEnvAttr	Supported
SQLGetFunctions	Supported
SQLGetInfo	Supported
SQLGetStmtAttr	Supported
SQLGetStmtOption	Supported
SQLGetTypeInfo	Supported
SQLMoreResults	Supported
SQLNativeSql	Supported
SQLNumParams	Supported
SQLNumResultCols	Supported
SQLParamData	Supported
SQLParamOptions	Supported
SQLPrepare	Supported
SQLPrimaryKeys	Supported
SQLProcedureColumns	Supported
SQLProcedures	Supported
SQLPutData	Supported
SQLRowCount	Supported
SQLSetConnectAttr	Supported
SQLSetConnectOption	Supported
SQLSetCursorName	Supported
SQLSetDescField	Supported
SQLSetDescRec	Supported
SQLSetEnvAttr	Supported
SQLSetParam	Supported
SQLSetPos	Supported

Function	Status
SQLSetScrollOptions	Supported
SQLSetStmtOption	Supported
SQLSetStmtAttr	Supported
SQLStatistics	Supported
SQLTablePrivileges	Supported
SQLTables	Supported
SQLTransact	Supported

Scalar functions

The Easysoft ODBC-PostgreSQL Driver supports a number of scalar functions:

- String functions
- Numeric functions
- Time, date, and interval functions
- Conversion functions

Use the SQL-92 format with scalar functions. For example:

```
SELECT CURRENT_DATE
```

String functions

The Easysoft ODBC-PostgreSQL Driver supports these string functions:

- ASCII(*string_exp*)
- BIT_LENGTH(*string_exp*)
- CHAR_LENGTH(*string_exp*)
- CHARACTER_LENGTH(*string_exp*)`
- CONCAT(*string_exp1*, *string_exp2*)
- LEFT(*string_exp*, *count*)
- LENGTH(*string_exp*)
- LTRIM(*string_exp*)
- OCTET_LENGTH(*string_exp*)
- POSITION(*char_exp IN char_exp*)
- REPEAT(*string_exp*, *count*)
- REPLACE(*string_exp1*, *string_exp2*, *string_exp3*)
- RIGHT(*string_exp*, *count*)
- RTRIM(*string_exp*)
- SUBSTRING(*string_exp*, *start*, *length*)
- TRIM(*string_exp*)`

Numeric functions

The Easysoft ODBC-PostgreSQL Driver supports these numeric functions:

- ABS(*numeric_exp*)
- ACOS(*float_exp*)
- ASIN(*float_exp*)
- ATAN(*float_exp*)

- CEILING(*numeric_exp*)
- COS(*float_exp*)
- COT(*float_exp*)
- DEGREES(*numeric_exp*)
- EXP(*float_exp*)
- FLOOR(*numeric_exp*)
- LOG(*float_exp*)
- LOG10(*float_exp*)
- PI()
- POWER(*numeric_exp, integer_exp*)
- RADIANS(*numeric_exp*)
- ROUND(*numeric_exp, integer_exp*)
- SIGN(*numeric_exp*)
- SIN(*float_exp*)
- SQRT(*float_exp*)
- TAN(*float_exp*)

Time, date, and interval functions

The Easysoft ODBC-PostgreSQL Driver supports these time, date, and interval functions:

- CURRENT_DATE()
- CURRENT_TIME[(*time-precision*)]
- CURRENT_TIMESTAMP[(*timestamp-precision*)]

Conversion functions

The Easysoft ODBC-PostgreSQL Driver supports supports the SQL-92 CAST function for conversion between compatible data types.

Data type mapping

The Easysoft ODBC-PostgreSQL Driver maps PostgreSQL data types to ODBC data types in this way:

PostgreSQL data type	ODBC data type
uuid	SQL_GUID
text	SQL_WLONGVARCHAR
varchar	SQL_WVARCHAR
char	SQL_WCHAR SQL_BIT SQL_CHAR
int2	SQL_TINYINT
int8	SQL_BIGINT
lo	SQL_LONGVARBINARY
bytea	SQL_VARBINARY
text	SQL_LONGVARCHAR
numeric	SQL_NUMERIC SQL_DECIMAL
int4	SQL_INTEGER
int2	SQL_SMALLINT
float8	SQL_FLOAT SQL_DOUBLE
float4	SQL_REAL
date	SQL_DATETIME
time	SQL_INTERVAL
timestamptz	SQL_TIMESTAMP
varchar	SQL_VARCHAR
varbinary	SQL_VARCHAR
date	SQL_TYPE_DATE
time	SQL_TYPE_TIME
timestamptz	SQL_TYPE_TIMESTAMP

Finding out more about data types on Windows

If you need more information about a data types, for example, the precision and scale, use Microsoft's ODBC Test to do this.

1. Download the version of ODBC Test that matches your application's architecture from:

   https://www.easysoft.com/ftp/pub/utils/windows/odbc-test/

2. Copy both files to a folder on the machine where Easysoft ODBC-PostgreSQL Driver is installed.
3. Double-click **odbcte32.exe**.
4. Select **Con > Full Connect**.

5. Choose your Easysoft ODBC-PostgreSQL Driver data source from the list.
6. Choose **Catalog > SQLGetTypeInfo**.
7. Either choose **SQL_ALL_TYPES=0 (1.0)** or a specific data type from the **DataType** list.
8. Choose **Results > Get Data All**.

SQL support

The Easysoft ODBC-PostgreSQL Driver supports these SQL statements, clauses, and operators:

- SELECT
- SELECT DISTINCT
- WHERE
- ORDER BY
- AND
- OR
- NOT
- INSERT INTO
- NULL
- UPDATE
- DELETE
- FIRST
- MIN
- MAX
- COUNT
- SUM
- AVG
- LIKE
- WILDCARDS
- IN
- BETWEEN
- ALIASES
- JOINS
- UNION
- GROUP BY
- HAVING
- EXISTS
- CASE

Example queries

- To fetch all records from a table, use the asterisk symbol (*) in your queries. For example:

```
SELECT * FROM Customers
```

- To only fetch records whose values are different, use DISTINCT. For example:

```
-- Which different sales regions are there?
SELECT DISTINCT Region AS Different_Regions FROM SalesOrders
-- How many different sales regions are there?
SELECT COUNT(DISTINCT Region) AS Different_Regions FROM SalesOrders
```

- To filter records, use WHERE. For example:

```
SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  Region = 'Eastern'

SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  Region = 'Eastern'
  OR Region = 'Western'
```

You can also supply a WHERE clause value as a parameter. For example, to do this in Python:

```
cursor.execute("SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  Region = ?", ['Eastern'])
```

- To fetch records that don't match the WHERE clause pattern use NOT. For example:

```
SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  NOT Region = 'Eastern'
```

- To sort the result set in either ascending or descending order, use ORDER BY. For example:

```
SELECT
  *
FROM
  SalesOrders
ORDER BY
  OrderDate ASC

SELECT
  *
FROM
  Contacts
ORDER BY
  (
    CASE
      WHEN Surname IS NULL THEN Title
      ELSE Surname
    END
  );
```

- To group a result set into summary rows, use GROUP BY. For example:

```
SELECT
    COUNT(Id) As "Number",
    ProductID
FROM
    SalesOrderItems
GROUP BY
    ProductID

SELECT
    COUNT(Id) As "Number",
    ProductID
FROM
    SalesOrderItems
GROUP BY
    ProductID
HAVING
  COUNT(Id) > 100;
```

- To do calculations based on result set vales, use the SQL aggregate functions MIN(), MAX(), COUNT(), SUM(), and AVG(). For example:

```
SELECT Max(Quantity) FROM SalesOrderItems
SELECT Sum(Quantity) FROM SalesOrderItems
```

- To convert between compatible data types, use CAST. For example:

```
SELECT CAST(Quantity AS Char(100))FROM  SalesOrderItems
```

- To fetch records that contain column values between a given range, use BETWEEN. For example:

```
SELECT ProductID FROM SalesOrderItems WHERE Quantity BETWEEN 10 AND 20
```

- To combine the result set of two or more SELECT statements, use UNION. For example:

```
SELECT City FROM Contacts
UNION
SELECT City FROM Customers
```

- To combine rows from two or more tables, use JOIN. For example:

```
SELECT SalesOrders.ID, Customers.Surname, SalesOrders.OrderDate
FROM SalesOrders
INNER JOIN Customers ON SalesOrders.CustomerID=Customers.ID;
```

- To fetch records that contain column values matching a search pattern, use LIKE. For example:

```
SELECT Surname, GivenName FROM Customers WHERE CompanyName LIKE 'R%'
SELECT Surname, GivenName FROM Customers WHERE CompanyName LIKE '_he'
```

- To search for columns without a value (NULL) or with a value (non NULL), use either IS NULL or IS NOT NULL. For example:

```
SELECT * FROM Customers WHERE CompanyName IS NULL
```

- To specify multiple values in a WHERE clause, you can use IN as an alternative to OR. For example:

```
SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  Region = 'Eastern'
  OR Region = 'Western'
  OR Region = 'Central'
```

  can be replaced with:

```
SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  Region IN  ('Eastern', 'Western', 'Central')
```

- To set the maximum number of records to return, use LIMIT. For example:

```
SELECT * FROM Customers LIMIT 10
```

- To test for the existence of records in a subquery, use EXISTS. For example:

```
SELECT
  Name
FROM
  Products
WHERE
  EXISTS (
    SELECT
     *
    FROM
      SalesOrderItems
    WHERE
     Products.ID = SalesOrderItems.ProductID
      AND Quantity < 20
  )
```

Example inserts, updates, and deletes

- To insert a PostgreSQL record, use INSERT INTO. For example:

```
INSERT INTO
    Customers (
        Surname,
        GivenName,
        City,
        Phone,
        CompanyName
    )
VALUES
    (
        'Devlin',
        'Michaels',
        'Kingston',
        '2015558966',
        'PowerGroup'
    )
```

- Here's a SQL Server linked server example:

```
EXEC ('INSERT INTO Customers (Surname, GivenName, City, Phone, CompanyName)
VALUES (''Devlin'' , ''Michaels'' , ''Kingston'' , ''2015558966'' ,
''PowerGroup'')')
```

- Here's an Oracle linked table example:

```
DECLARE
  num_rows integer;
BEGIN
num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@PostgresqlLink
('INSERT INTO Customers (Surname, GivenName, City, Phone, CompanyName) VALUES
(''Devlin'', ''Michaels'', ''Kingston'', ''2015558966'', ''PowerGroup'')');
END;
/
```

- The Easysoft ODBC-PostgreSQL Driver also supports parameterized inserts. Here's an example of doing this in Perl:

```
my $sth = $dbh->prepare(q/INSERT INTO Customers (Surname, GivenName, City, Phone,
CompanyName) VALUES (?, ?, ?, ?, ?)/)
    or die "Can't prepare statement: $DBI::errstr";

$sth->execute('Devlin', 'Michaels', 'Kingston', '2015558966', 'PowerGroup');
```

- To update a PostgreSQL record, use UPDATE. For example:

```
UPDATE Customers
SET
  Surname = 'Jones'
WHERE
  Account_Id = 'PowerGroup'
```

  The Easysoft ODBC-PostgreSQL Driver also supports parameterized updates. Here's an example of doing this in Perl:

```
my $sth = $dbh->prepare('UPDATE Customers SET Surname = \'Jones\' WHERE
CompanyName = ?')
    or die "Can't prepare statement: $DBI::errstr";

$sth->execute('PowerGroup');
```

- To delete a PostgreSQL record, use DELETE. For example:

```
-- Delete (mark inactive) a bank account
DELETE FROM Customers WHERE CompanyName = 'PowerGroup'
```

The Easysoft ODBC-PostgreSQL Driver also supports parameterized deletes. Here's an example of doing this in Python:

```
sql = "DELETE FROM Customers WHERE CompanyName = ?"
cursor.execute(sql, 'PowerGroup')
```

Index