2021

PGNP Native OLE DB Provider for PostgreSQL, Greenplum, Redshift

Developer's Manual

This document contains description of various PGNP features, use cases and programming techniques. It is written for professional developers.

Konstantin Izmailov

2/12/2021

Table of Contents

1 Introduction

The PGNP Native OLE DB Provider exposes powerful low-level OLEDB interfaces to Windows applications connecting to PostgreSQL, Greenplum and Redshift databases. The provider can help you achieve performance and flexibility that are not available via either ODBC driver or .NET Provider:

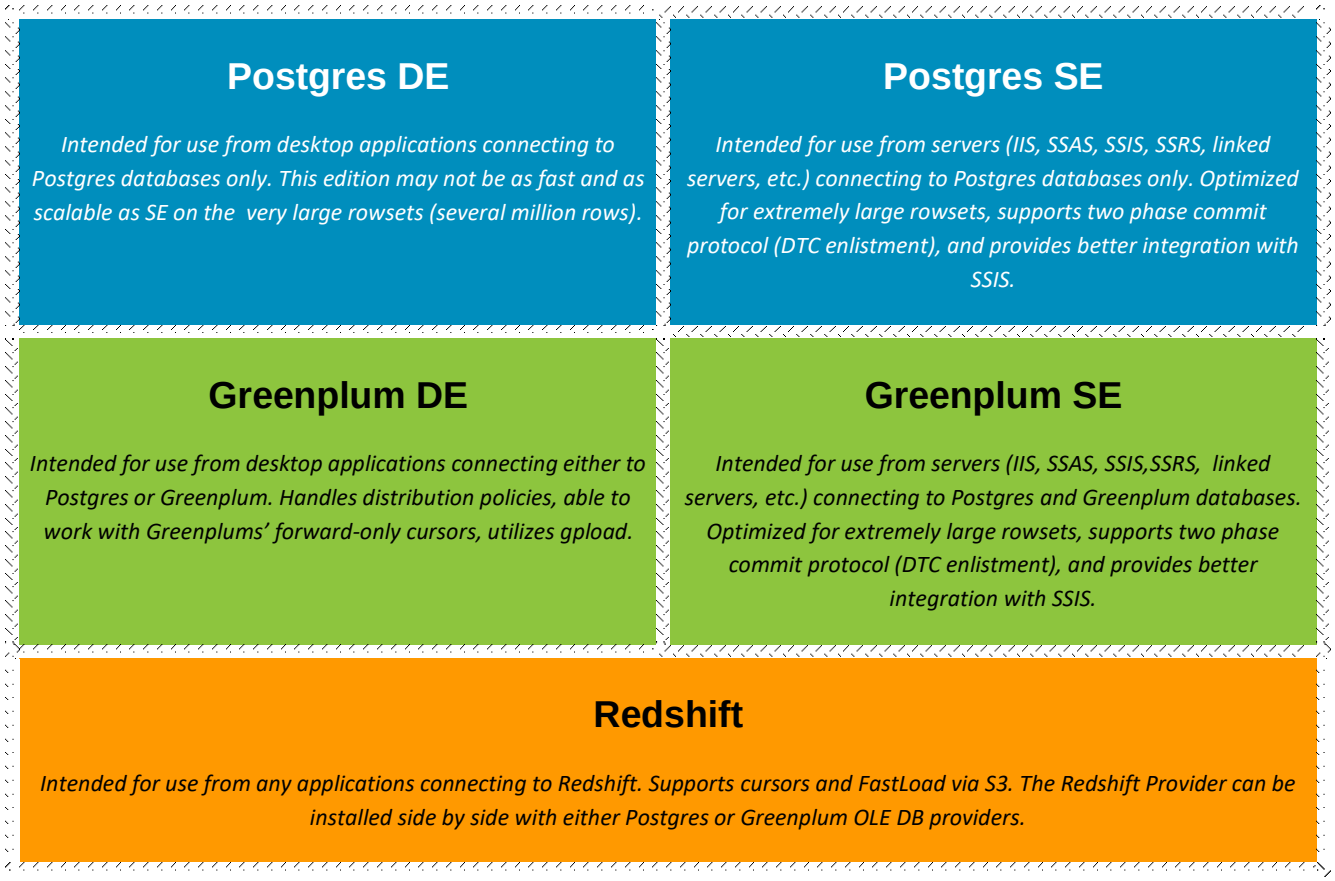
- Rich metadata (advanced schema and cursors)
- Databases transformation support
- Linked Servers
- Replication
- Database Reverse Engineering
- Bulk import

The Developer's Manual describes the Provider functionality and gives examples of the Provider usage. It is intended for use by software developers, system administrators and users of the OLE DB applications.

The supported operating systems are: Windows 2000 with MDAC 2.8 SP1, Windows XP, Windows Server 2003, Windows Server 2008, Windows Server 2012, Vista, Windows 7, Windows 8; both 32-bit and 64-bit.

The PGNP Provider works with the following versions of PostgreSQL database: PostgreSQL 8.0 and later, Greenplum 3.0 and later, EnterpriseDB Advanced Server 8.3 and later, Redshift. It may work on earlier versions of the corresponding databases but we have not tested those configurations, or Provider might not support them.

The Postgres and Greenplum Providers are available in two editions: Desktop Edition (DE) and Server Edition (SE). Following table summarizes differences:



Note: Server Edition (SE) has all the features of Desktop Edition (DE), and adds more features as shown in the table above.

2 Installation and Product Activation

To install the PGNP OLEDB Provider launch the installation module. The Module name may vary depending on the provider variant/edition:

Module name	Description
PGNP-Postgres-DE-1.4.3200.exe	OLEDB Provider for Postgres, Desktop Edition
PGNP-Postgres-SE-1.4.3200.exe	OLEDB Provider for Postgres, Server Edition
PGNP-Greenplum-DE-1.4.3200.exe	OLEDB Provider for Greenplum, Desktop Edition
PGNP-Greenplum-SE-1.4.3200.exe	OLEDB Provider for Greenplum, Server Edition
PGNP-Redshift-SE-1.4.0.3200.exe	OLEDB Provider for Redshift

Note: Evaluation module name includes word “trial”, e.g.: PGNP-Postgres-DE-Trial-1.4.3020.exe.

The latest releases have .msi extension, both 64-bit and 32-bit, e.g.:

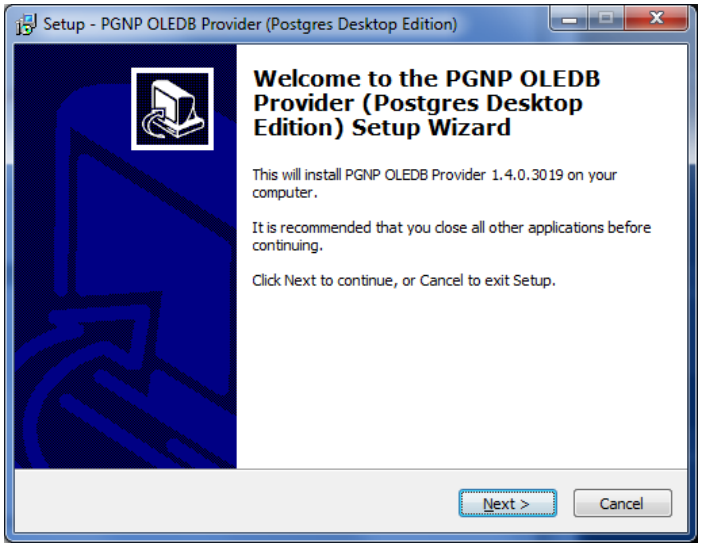
PGNP-Postgres-SE-1.4.0.3456-x64.msi

PGNP-Postgres-SE-1.4.0.3456-x32.msi

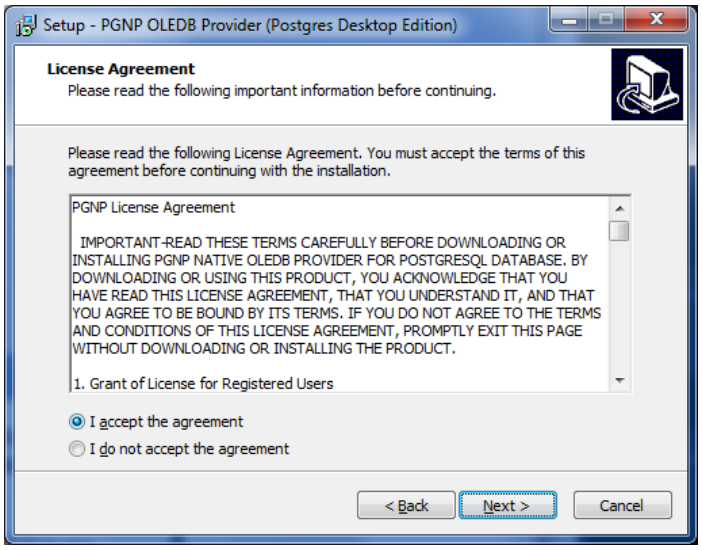
2.1.1 UI based installation

When installation module is launched without using command line parameter /SILENT, the UI-based is used.

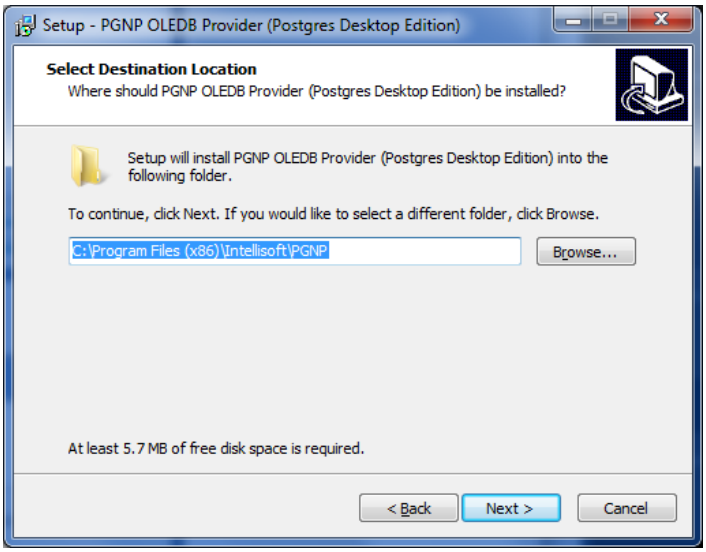
First page of the installation application is shown below. Click Next through the wizard pages.



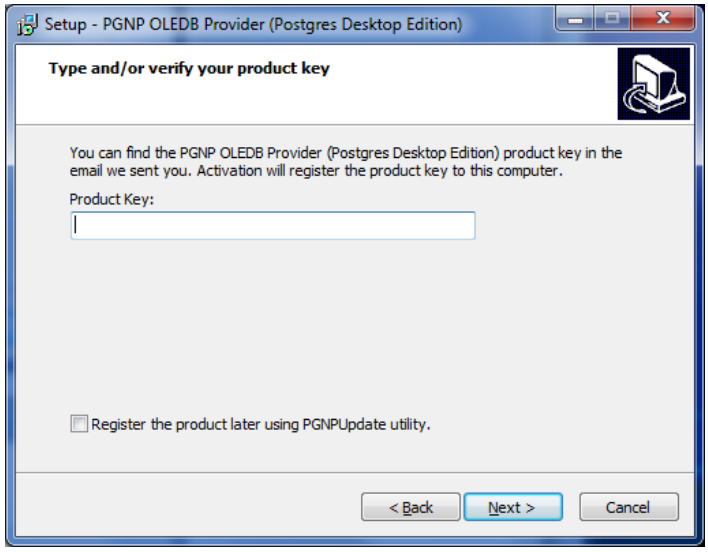
Please read the License Agreement carefully. Click Next.



Specify the installation folder. By default, the Postgres and Greenplum providers are installed into C:\Program Files (x86)\Intellisoft\PGNP folder, and the Redshift – into C:\Program Files (x86)\Intellisoft\RSNP folder.

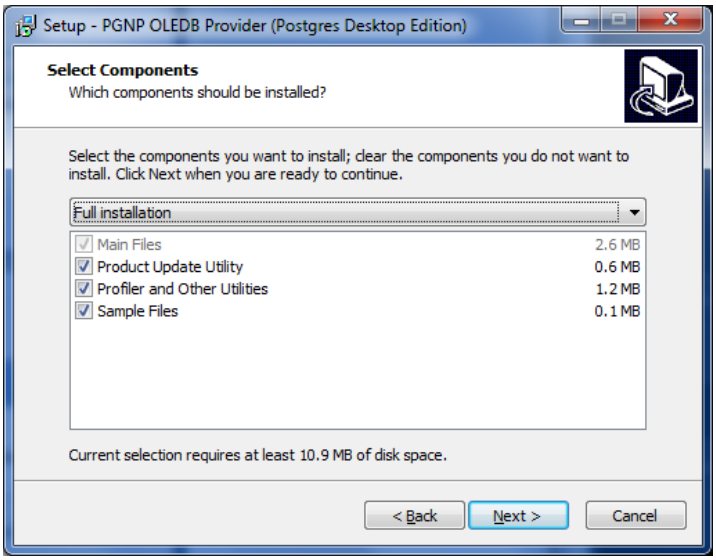


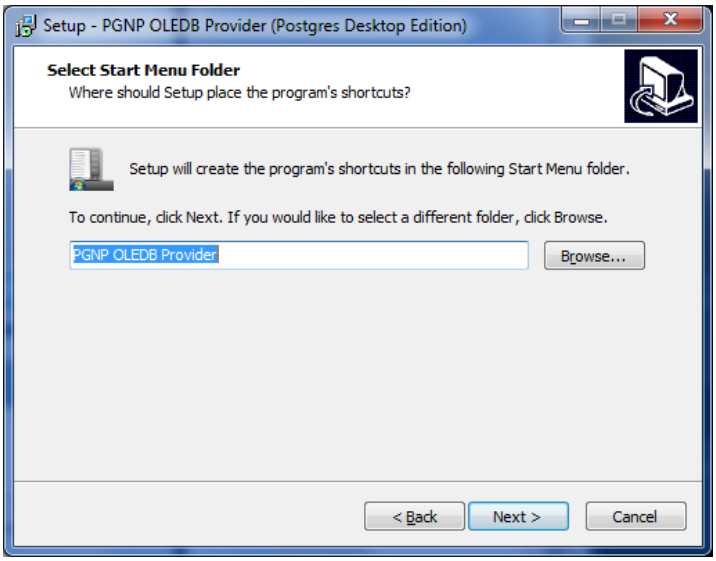
Enter the Product Key (License Key). The installation module will automatically activate the product on the computer. This step requires an Internet connection. If Internet connection is not available, or you prefer to activate the product later, select the check box on the bottom of the page (Register the product later using PGNPUpdate utility). Please refer to PGNP Update utility guide below, specifically to section 5.4.2 “Working in Activation Mode”.



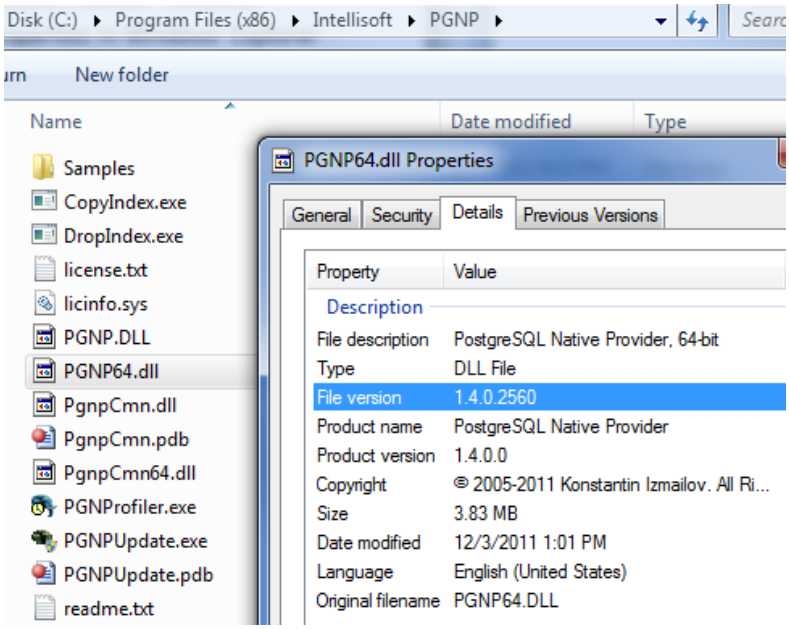
Note: If the provider was previously activated on the computer the installation will not require entering the product key. In that case the above page is not displayed, and installation proceeds to the next step.

Click Next in following dialogs.





Note. To determine the installed version of the PGNP provider run appwiz.cpl from a command line (or go to Start->Control Panel->Programs and Features/Add or Remove Programs). Line starting with "PGNP OLEDB Provider" contains version and build number, e.g. "1.4.0.3200". Another method is to view PGNP.DLL or PGNP64.DLL Properties in Windows Explorer:



The provider version can also be determined programmatically via call to pgnp_getlicenceinfo.

2.1.2 Unattended/silent installation

The product can be installed in an unattended/silent mode when no UI is shown and no user interaction is needed. Use command line parameters /SILENT and /activate <key>, where <key> is the license key.

Example: **PGNP-Postgres-SE-1.4.0.3364.exe /silent /activate A1726A2B156D4DA2829AF34335C7DF23**

There is also a special word "later" that can be used in place of the <key>. It allows unattended installation without the product activation. The product can be activated later using PGNPUpdate utility.

Example: **PGNP-Postgres-SE-1.4.0.3364.exe /silent /activate later**

Note: similarly, product can be uninstalled in an unattended mode using /SILENT parameter, for example: **"C:\Program Files (x86)\Intellisoft\PGNP\unins000.exe" /silent**

Note: if an error occurs during installation, a message box could be shown. This might require user interaction.

MSI module can be installed using following command:

msiexec.exe /i PGNP-Postgres-SE-1.4.0.3420-x64.msi /quiet PRODUCT_KEY=A1726A2B156D4DA2829AF34335C7DF23

2.1.3 Activation on a computer behind a firewall (in DMZ)

When a computer does not have access to the Internet for online product activation, the PGNPUpdate utility can be used for offline activation. The utility is installed with the non-trial version of the product.

The steps for the offline activation:

1. Launch the PGNPUpdate utility and generate an activation request.
2. Copy the activation request text to a flash drive, or use other method to bring the text to the computer that can send e-mail.
3. E-mail the request to the licensing@pgoledb.com. The mail box is automated, so the response will be sent within a few minutes.
4. When the response is received, launch the PGNPUpdate utility again, and enter the response in order to complete the activation.

For details about the PGNPUpdate utility, please refer to section 5.4.2.

To check that the product is properly activated, please launch the PGNPUpdate or the PGNProfiler utility. The utilities should display valid registration information (PGNProfiler will display the info in the About dialog).

3 Provider's Features

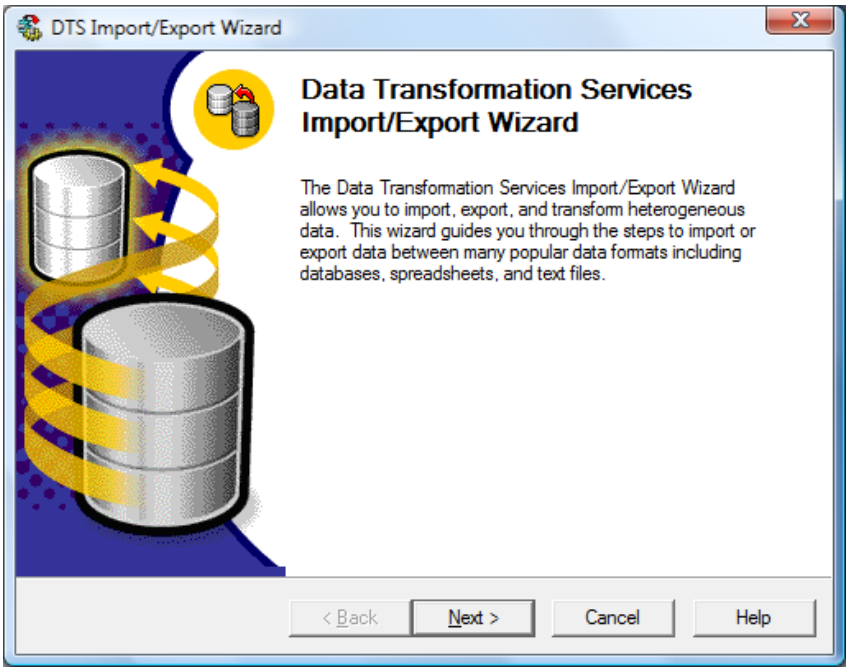
This chapter contains information about various features and use cases of the Provider.

3.1 Database Transformation

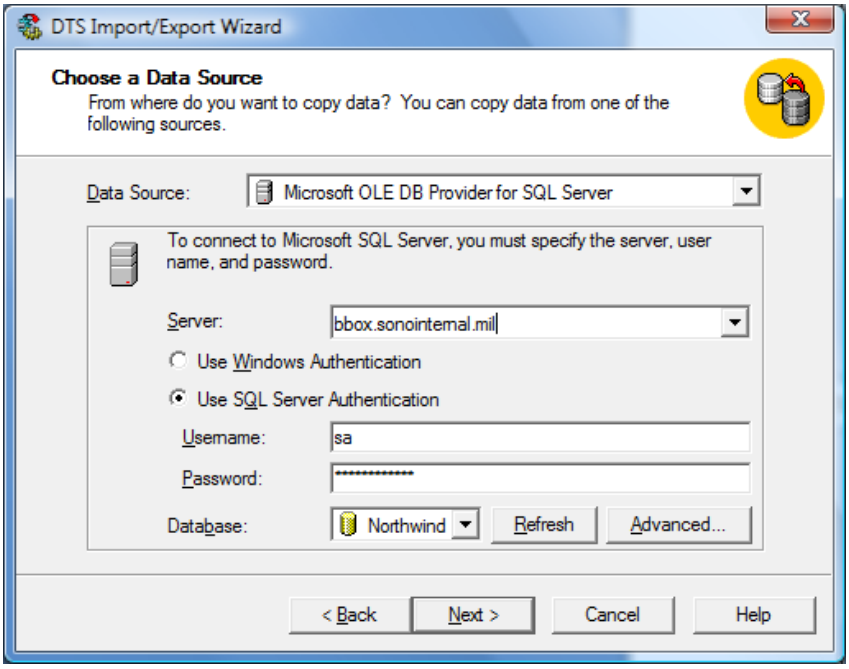
MS DTS/SSIS tools allow copying databases between various OLE DB sources. PGNP Provider can be used to transform a non-PostgreSQL database into a PostgreSQL database and vice versa.

3.1.1 Transforming MS SQL 2000 database into a PostgreSQL database using DTSWizard

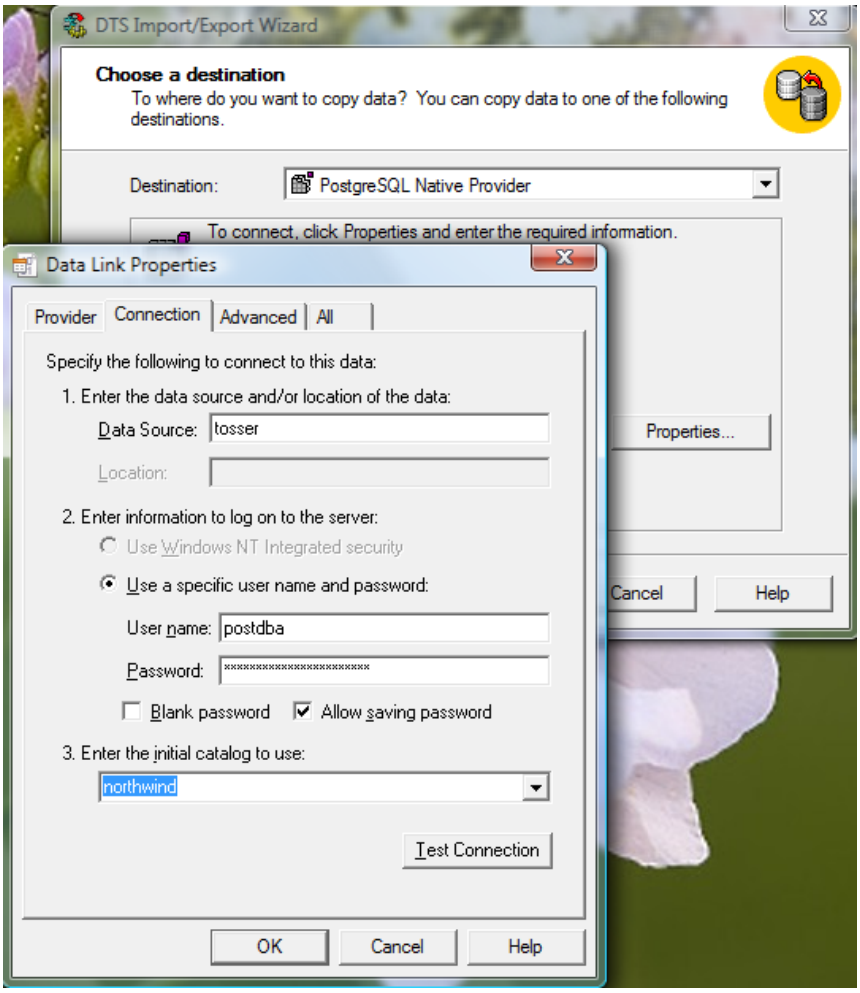
1. Make sure MS SQL Server 2000 is running and source database available.
2. Create an empty destination database in PostgreSQL.
3. Launch DTSWizard.exe ("C:\Program Files\Microsoft SQL Server\80\Tools\Binn\dtswiz.exe").



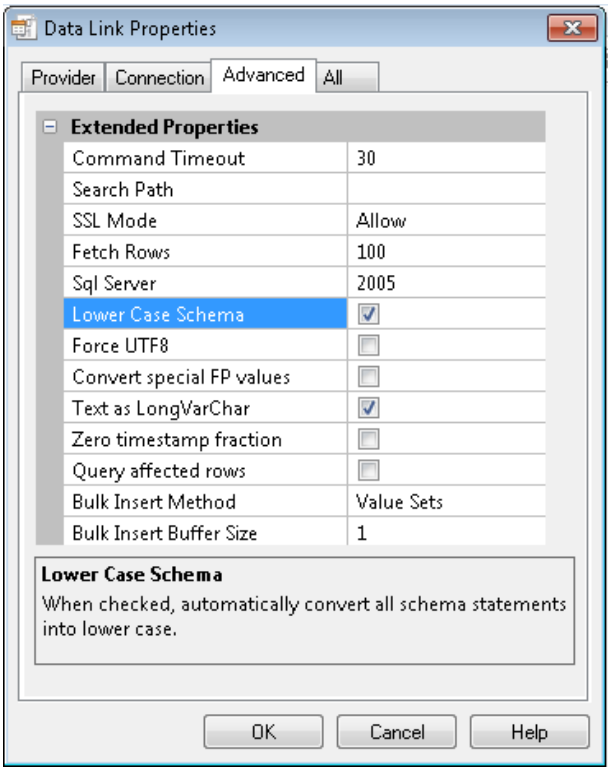
4. Select source database. Click Next.



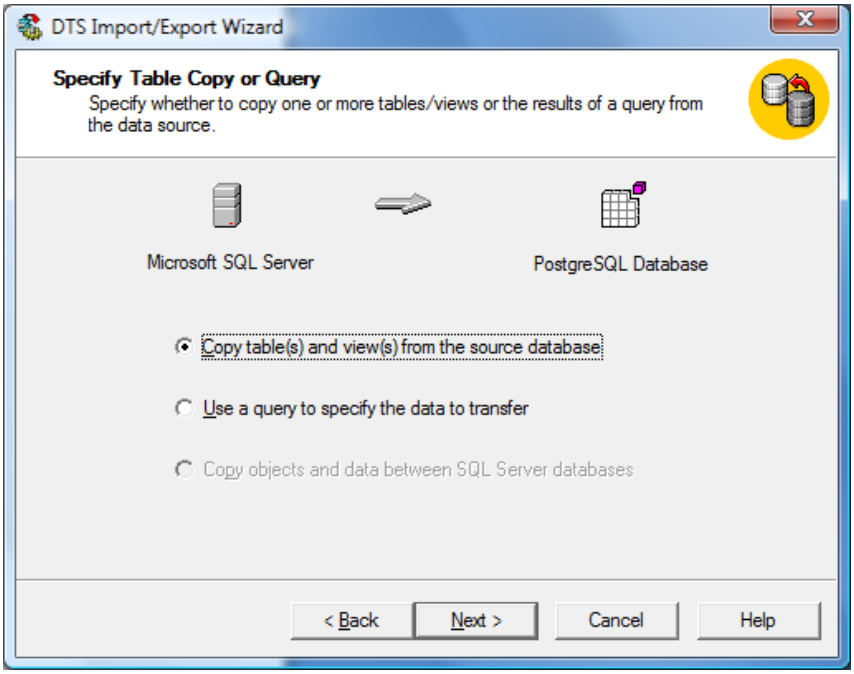
5. Select “PostgreSQL Native Provider” from the “Destination” combo-list, click Properties button and specify connection parameters of the destination database. Click Next.



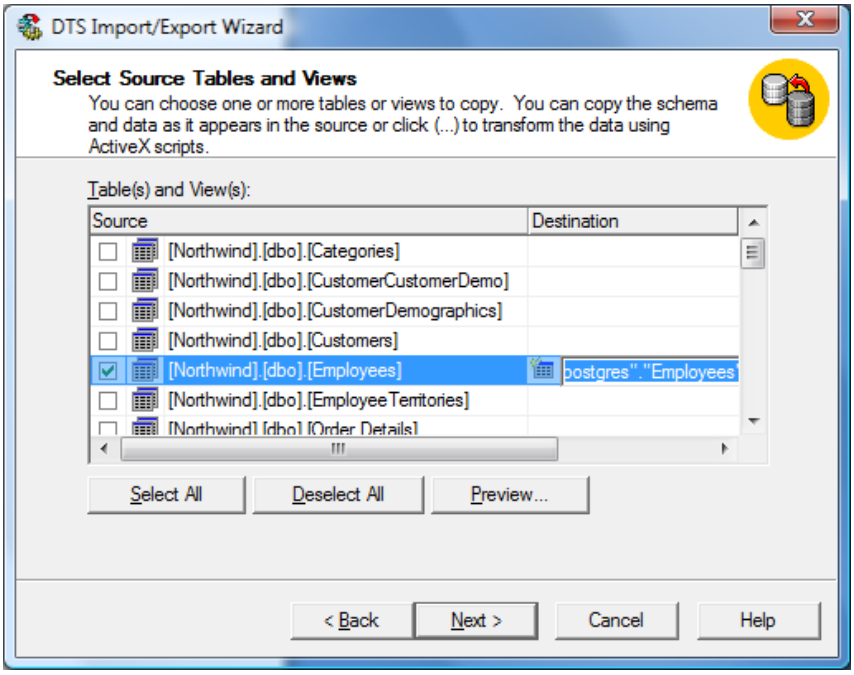
6. Select “Lower Case Schema” to convert tables and columns names into lower case (recommended):



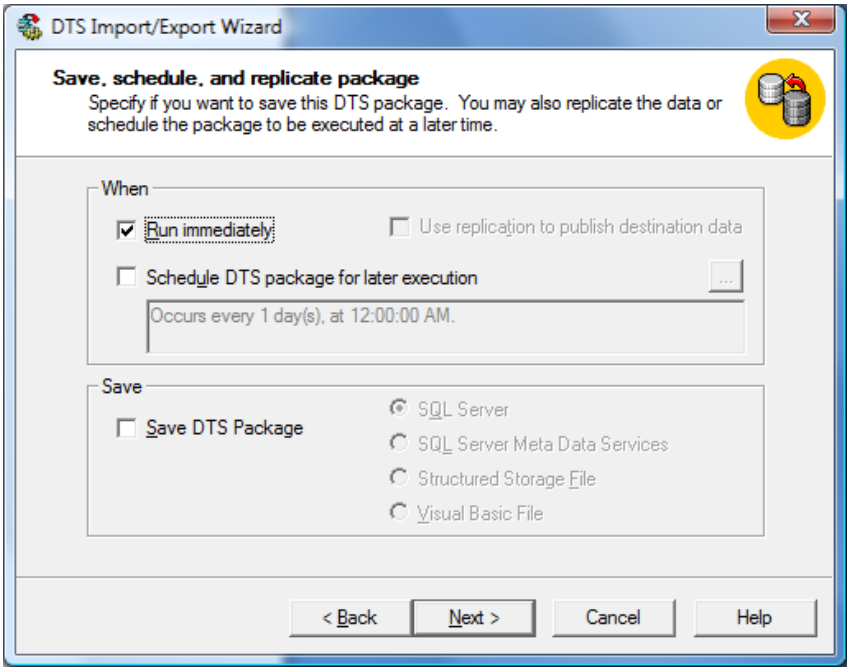
7. Click Next button on the “Specify Table Copy or Query” page.



8. Select tables and views you want to be copied to destination database. Click Next.



9. Choose “Run Immediately” and click Next. Click Finish.



10. Inspect result of conversion.

To convert indexes use CreateIndex utility. See more information in Utilities chapter of the Manual.

3.1.2 Transforming MS SQL 2005, 2008, 2012 database into a PostgreSQL database using DTSWizard

The database transformation procedure between SQL Server 2005, 2008, 2012 and Postgres, Greenplum, Redshift is similar to the one described above. Launch SQL Server Import and Export Wizard, and specify OLEDB source and destination.

3.2 Transactions

The PGNP provider supports standard OLE DB transactions (Local transactions), COM+ Distributed transactions (starting version 1.3.0), PostgreSQL nested transactions via direct OLEDB calls or SQL commands, and “savepoints”.

3.2.1 Local Transactions

Transactions are a convenient mechanism to achieve “all or nothing” effect when multiple changes are made in a database. Transactions can be used via ADO, ADO.NET or OLE DB calls to Connection object.

Example 1. Using transactions in Delphi (ADO).

Example 2. Using transactions in C++ (OLE DB).

Example 3. Using transactions in C# (ADO.NET).

```
OleDbConnection con1 = new OleDbConnection(sb.ToString());
con1.Open();

// begin main transaction
OleDbTransaction trans = con1.BeginTransaction();

// insert a new record
OleDbCommand cmd_insert = new OleDbCommand("insert into contact(fname,lname) values (:fname,:lname)", con1,
trans);
cmd_insert.Parameters.AddWithValue(":fname", "Joe");
cmd_insert.Parameters.AddWithValue(":lname", "Blah");
Debug.Assert(1 == cmd_insert.ExecuteNonQuery());

// begin nested transaction
```

```
OleDbTransaction nested_trans1 = trans.Begin();

// begin nested transaction inside the nested one
OleDbTransaction nested_trans2 = nested_trans1.Begin();

// remove recently inserted record in the nested transaction level 2
OleDbCommand cmd_delete = new OleDbCommand("delete from contact where lname=?", con1, nested_trans2);
cmd_delete.Parameters.AddWithValue("?", "Blah");
Debug.Assert(1 == cmd_delete.ExecuteNonQuery());

// rollback
nested_trans2.Rollback();

// check that the record was not removed from the nested transaction level 2
OleDbCommand cmd_check = new OleDbCommand("select count(*) from contact where lname=?", con1, nested_trans);
cmd_check.Parameters.AddWithValue("?", "Blah");

Debug.Assert(1 == Convert.ToInt32(cmd_check.ExecuteScalar()));

// remove the record
cmd_delete.Transaction = nested_trans;
Debug.Assert(1 == cmd_delete.ExecuteNonQuery());

// commit the changes
nested_trans.Commit();

// check that the record was actually removed
cmd_check.Transaction = trans;
Debug.Assert(0 == Convert.ToInt32(cmd_check.ExecuteScalar()));

trans.Commit();
con1.Close();
```

Starting version 1.3.0 PGNP supports nested transactions. Note, since ADO does not support nested transactions, you can use them from either OLE DB or ADO.NET (see C# Sample 26). To enable nested transactions specify NESTED_TRANS=ON in Extended Properties parameter of the connection string.

PGNP provider triggers transactions related events on the Connection object: BeginTransComplete, CommitTransComplete, RollbackTransComlete.

Example 4. Using transaction events in Delphi.

```
TADOConnection fConn;
...
// "begin transaction" callback
procedure PGNPBeginTransComplete(
              Connection: TADOConnection;
              TransactionLevel: Integer;
              const Error: Error;
              var EventStatus: TEventStatus);
...
// specify isolation level
fConn.IsolatioLevel := ilReadCommitted;

// subscribe for transaction event(s)
fConn.OnBeginTransComplete := PGNPBeginTransComplete;

// start transaction
fConn.BeginTrans;         // note: PGNPBeginTransComplete callback is called from provider
...
// some useful work
...
if Res then
   fConn.Commit           // note: OnCommitComplete callback is called
else
   fConn.Rollback;        // note: OnRollback callback is called
```

3.2.2 Distributed Transactions

Distributed transactions allow coordinating database transactions among multiple databases or transactional systems. PGNP Provider supports transactions enlistment in Distributed Transactions Coordinator (DTC) and handles events from the DTC. Note: distributed transactions support added to PGNP 1.3.0 and later.

One of the scenarios that require distributed transactions is when you need to pull data from two dependent constantly changing tables in SSIS, so that when it's imported on MSSQL side the data is in consistent state. You can configure the Integration Services project with a Control Flow property "Transaction Option" equal to "Require".

The following C# example demonstrates distributed transactions use:

```
// .Net 2.0 and later supports distributed transactions via TransactionsScope object.
// The object uses distributed transaction if there is more than one connection open in its context,
// otherwise a local transaction is used.
using System;
using System.Data.OleDb;
using System.Diagnostics;
using System.Text;
using System.Transactions;

namespace DistribTrans
{
    class Program
    {
        static void Main(string[] args)
        {
            StringBuilder sb = new StringBuilder();
            sb.Append("Provider=PGNP.1;");
            sb.Append("Persist Security Info=True;");
            sb.Append("Data Source=localhost;");      // PostgreSQL server;
            sb.Append("Initial Catalog=postgres;");   // Database name
            sb.Append("User ID=postgres;");           // User name
            sb.Append("Password=12345;");             // User password
            sb.Append("Extended Properties=\"PORT=5432;\"");

            // TransactionScrope automatically links local transactions to the distributed one.
            // Here are two local transactions and one distributed.
            using (TransactionScope scope = new TransactionScope())
            {
                OleDbConnection con1 = new OleDbConnection(sb.ToString());
                con1.Open();   // A local transaction is started automatically

                // Insert first record
                OleDbCommand cmd_insert = new OleDbCommand("insert into country (country,currency) values
(:country,:currency)", con1);
                cmd_insert.Parameters.AddWithValue("country", "Russia");
                cmd_insert.Parameters.AddWithValue("currency", "Rouble");
                Debug.Assert(1 == cmd_insert.ExecuteNonQuery());

                OleDbConnection con2 = new OleDbConnection(sb.ToString());
                con2.Open();   // A local transaction is started automatically

                // Insert second record
                cmd_insert.Connection = con2;
                cmd_insert.Parameters["country"].Value = "Latvia";
                cmd_insert.Parameters["currency"].Value = "Lat";
                Debug.Assert(1 == cmd_insert.ExecuteNonQuery());

                // Commit distributed transaction.
                // Commit() will be called for all local transactions.
                scope.Complete();
            }
        }
    }
}
```

3.2.3 PostgreSQL Nested Transactions

ADO does not support nested transactions. However, PostgreSQL nested transactions can be used by calling OLEDB interfaces or executing START TRANSACTION, SAVEPOINT and other SQL commands. To enable nested transaction support in PGNP provider add NESTED_TRANS=ON parameter to Extended Properties of a connection string. Note: nested transactions support added to PGNP 1.3.0 and later.

The PGNP provider recognizes transaction related SQL commands and tracks internal transaction state automatically.

The following C# example demonstrates nested transactions and isolation levels use:

```
//
// Transaction isolation level defines visibility of changes among different parallel transactions.
//
// The isolation level can be set with System.Data.IsolationLevel parameter in call to
// OleDbConnection.BeginTransaction().
//
using System;
using System.Collections.Generic;
using System.Data.OleDb;
using System.Text;
using System.Diagnostics;

namespace NestedTrans
{
    class Program
    {
        static void Main(string[] args)
        {
            StringBuilder sb = new StringBuilder();
            sb.Append("Provider=PGNP.1;");
            sb.Append("Persist Security Info=True;");
            sb.Append("Data Source=localhost;");      // PostgreSQL server;
            sb.Append("Initial Catalog=postgres;");   // Database name
            sb.Append("User ID=postgres;");           // User name
            sb.Append("Password=12345;");             // User password
            sb.Append("Extended Properties=\"NESTED_TRANS=ON;\""); // Enable nested transactions

            OleDbConnection con1 = new OleDbConnection(sb.ToString());
            con1.Open();

            // Start main transaction
            OleDbTransaction trans = con1.BeginTransaction();

            // Add new record
            OleDbCommand cmd_insert =
                new OleDbCommand("insert into country (country,currency) values (:country,:currency)", con1,
trans);
            cmd_insert.Parameters.AddWithValue(":country", "Russia");
            cmd_insert.Parameters.AddWithValue(":currency", "Ruble");
            Debug.Assert(1 == cmd_insert.ExecuteNonQuery());

            // Start first nested transaction
            OleDbTransaction internal_transaction = trans.Begin();

            // Start second nested transaction
            OleDbTransaction internal_transaction2 = internal_transaction.Begin();

            // Delete the record in second nested transaction
            OleDbCommand cmd_delete =
                new OleDbCommand("delete from country where country=?", con1, internal_transaction2);
            cmd_delete.Parameters.AddWithValue("?", "Russia");
            Debug.Assert(1 == cmd_delete.ExecuteNonQuery());

            // Rollback second nested transaction
            internal_transaction2.Rollback();

            // Check that the record was actually not removed
            OleDbCommand cmd_check =
                new OleDbCommand("select count(*) from country where country=?", con1, internal_transaction);
            cmd_check.Parameters.AddWithValue("?", "Russia");
            Debug.Assert(1 == Convert.ToInt32(cmd_check.ExecuteScalar()));
```

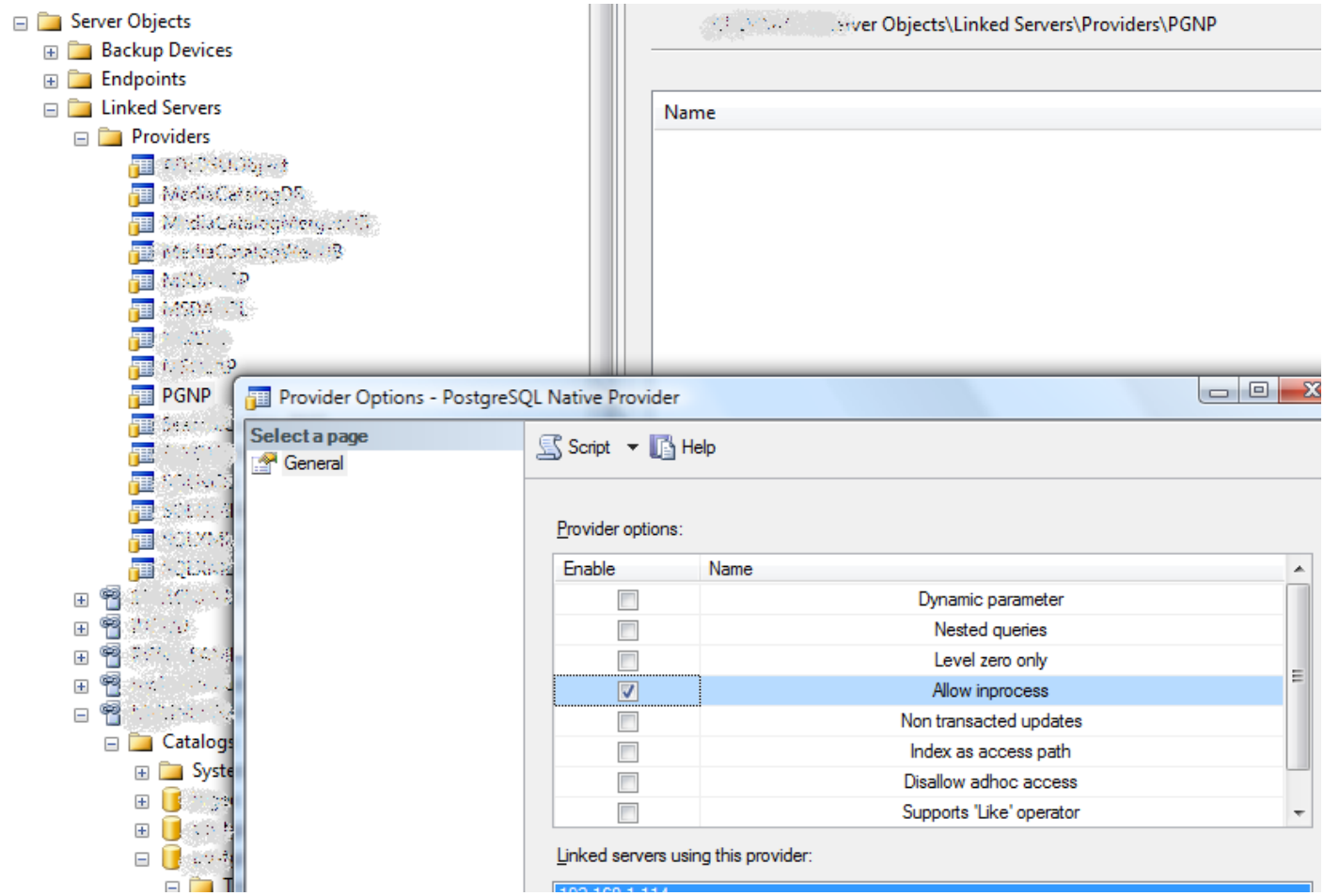
```
            // Delete the record in first nested transaction
            cmd_delete.Transaction = internal_transaction;
            Debug.Assert(1 == cmd_delete.ExecuteNonQuery());

            // Commit changes
            internal_transaction.Commit();

            // Check in main transaction that the record was removed
            cmd_check.Transaction = trans;
            Debug.Assert(0 == Convert.ToInt32(cmd_check.ExecuteScalar()));

            trans.Commit();
            con1.Close();
        }
    }
}
```

3.3 Linked Servers

"Linked Server" is a feature of MS SQL Server that allows access to non-SQL Server databases through the SQL Server. Note, only SQL Server Developer and Enterprise support Linked servers. SQL Server Express edition has several limitations that do not allow configuring or using Linked Servers.

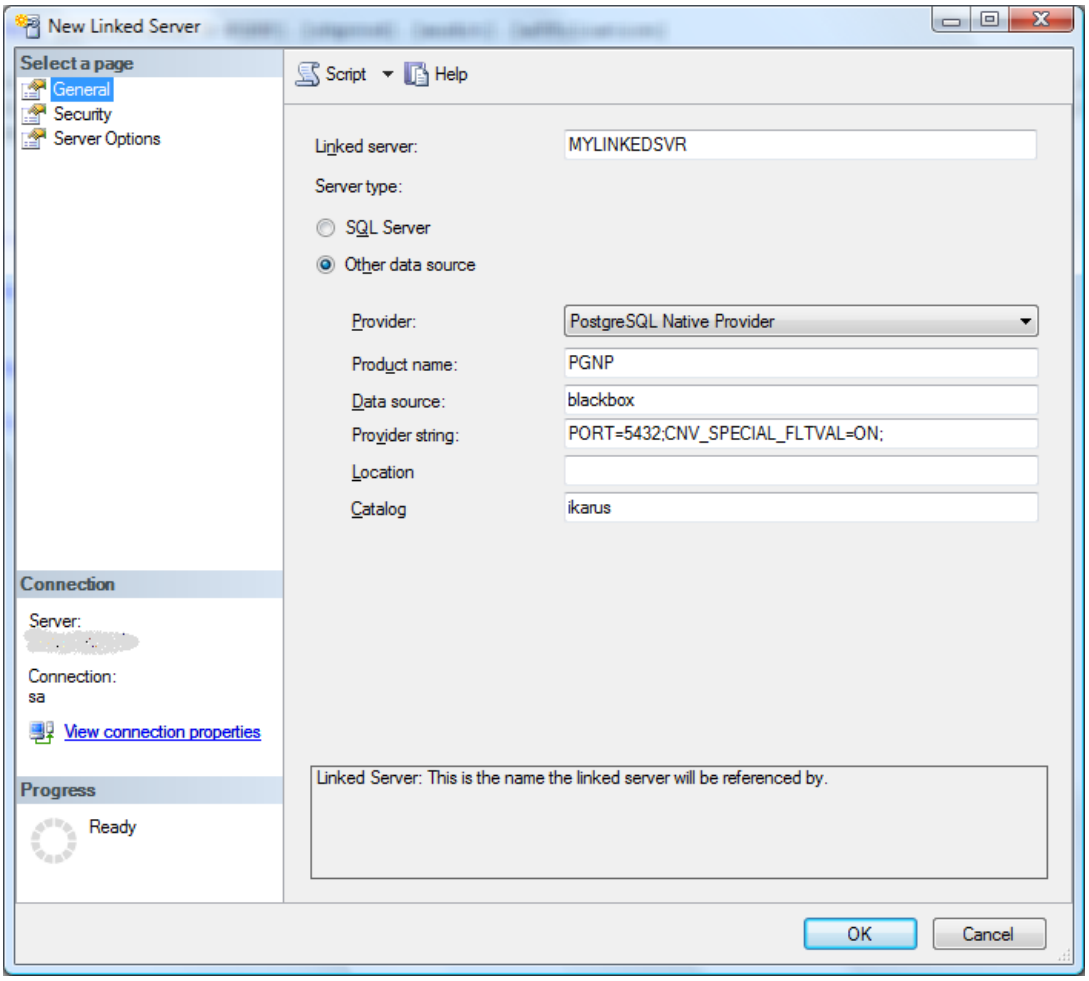
3.3.1 Create Linked Server using SQL Server Wizard

Before creating or using Linked Servers make sure that "Allow inprocess" option is checked. Right click on Linked Servers/Provider/PGNP and select Properties menu item.



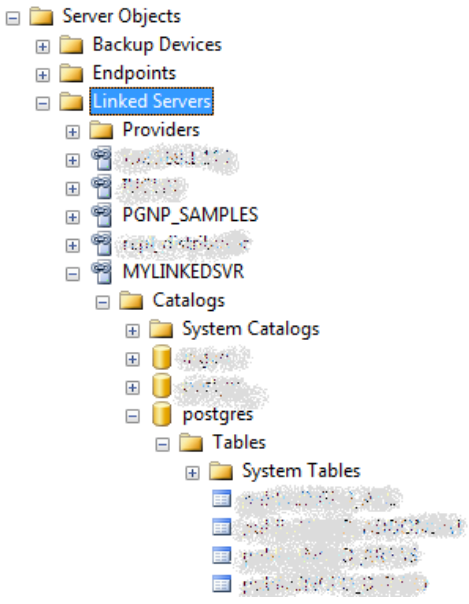
To create a linked server, launch Enterprise Manager 2000 or Management Studio 2005/2008/2012. Right click on Security/Linked Servers or Server Objects/Linked Servers and select "New Linked Server…" menu item. Enter the Linked

server name and choose “PostgreSQL Native Provider” in “Other Data Sources”/Provider combo box. Enter your product name, Postgres server, Extended Properties of the connection string and database name as shown:



In “Security” tab select “Be made using this security context” radio button and enter credentials to access Postgres database. Click OK. The Linked Server is created.

To test the linked server, expand Linked Servers/MYLINKEDSVR/Catalogs/postgres:



You can query or change data using SQL commands like:

```
SELECT * FROM "MYLINKEDSVR"."postgres"."public"."contacts"

SELECT * FROM "MYLINKEDSVR"..."contacts"

SELECT * FROM openquery("MYLINKEDSVR", 'SELECT * FROM "contacts"')

INSERT * INTO "MYLINKEDSVR"..."contacts" VALUES('John', 'Smith', 1988)

SELECT 1 FROM openquery("MYLINKEDSVR", 'UPDATE contacts SET dob=2002 WHERE dob=02')
```

Note, "openquery" syntax allows using of PostgreSQL-specific statements. Refer to MSDN documentation for more details. Please remember using double quotes if your PostgreSQL schema is in mixed case.

3.3.2 Create Linked Server using SQL Server Stored Procedures

Ensure that PGNP provider is configured to run as "In Process DLL":

```
EXEC master.dbo.sp_MSset_oledb_prop N'PGNP.1', N'AllowInProcess', 1
EXEC master.dbo.sp_MSset_oledb_prop N'PGNP.1', N'DynamicParameters', 1
```

Execute following SQL statements to create a linked server with name 'PGNP_SAMPLES':

```
declare @LINKED_SERVER_NAME varchar(max);
declare @PRODUCT_NAME varchar(max);
declare @PGPROVIDER varchar(max);
declare @DATA_SOURCE varchar(max);
declare @CN_STR varchar(max);
declare @SAMPLE_CATALOG varchar(max)

-- postgres database info
set @LINKED_SERVER_NAME = N'PGNP_SAMPLES';
set @PRODUCT_NAME = N'PGNP';
set @PGPROVIDER = N'PGNP';
set @DATA_SOURCE = N'localhost';
set @CN_STR = 'PORT=5432;CNV_SPECIAL_FLTVAL=ON;';
set @SAMPLE_CATALOG = N'linkedtest';

/* DROP LINKED SERVER */
IF EXISTS (SELECT srv.name FROM sys.servers srv WHERE srv.server_id != 0 AND srv.name =
@LINKED_SERVER_NAME)
EXEC master.dbo.sp_dropserver @server=@LINKED_SERVER_NAME, @droplogins='droplogins'

/* CREATE LINKED SERVER */
EXEC master.dbo.sp_addlinkedserver
@server = @LINKED_SERVER_NAME,
@srvproduct = @PRODUCT_NAME,
@provider = @PGPROVIDER,
@datasrc = @DATA_SOURCE,
@provstr = @CN_STR,
@catalog = @SAMPLE_CATALOG

/* set up Extended properties of the Linked Server */
EXEC master.dbo.sp_addlinkedsrvlogin
@rmtsrvname=@LINKED_SERVER_NAME,@useself=N'False',@locallogin=NULL,@rmtuser=N'postgres',@rmtpassword='12
345'

EXEC master.dbo.sp_serveroption @server=@LINKED_SERVER_NAME, @optname='data access', @optvalue='true'
EXEC master.dbo.sp_serveroption @server=@LINKED_SERVER_NAME, @optname='rpc', @optvalue='true'
EXEC master.dbo.sp_serveroption @server=@LINKED_SERVER_NAME, @optname='rpc out', @optvalue='true'
EXEC master.dbo.sp_serveroption @server=@LINKED_SERVER_NAME, @optname='use remote collation',
@optvalue='true'
```

3.3.3 Viewing and changing Linked Server RPC status

Some SQL commands sent to a linked server require RPC configuration. For example, following command:

```
EXEC ('CREATE TABLE contact(Id INT NOT NULL, LNAME VARCHAR)') AT PGNP_SAMPLES
```

May return error: Server 'PGNP_SAMPLES' is not configured for RPC.

Linked server configuration can be viewed with following command:

```
exec sp_helpserver 'PGNP_SAMPLES'
```

To configure the linked server execute following commands:

```
exec sp_serveroption @server='PGNP_SAMPLES', @optname='rpc', @optvalue='true'
exec sp_serveroption @server='PGNP_SAMPLES', @optname='rpc out', @optvalue='true'
```

3.3.4 Running Linked Server in a separate process (out-of-proc)

Some IT departments have policies that do not allow running a third party component within the MSSQL Server process. The advantage is that possible memory leaks and crashes in the PGNP provider won't affect the MSSQL Server process, since PGNP provider runs in a separate surrogate process (dllhost.exe).
Here are steps to configure the OLEDB provider “out-of-proc”:
1. Register a surrogate process for PGNP provider to make it visible in DCOMCNFG utility. For this either run OleView.exe utility (part of Visual Studio or SDK), or apply registry script (see below).
In OleView expand "All Objects" in the treeview on the left, and select "PostgreSQL Native Provider" object. Go to "Implementation" tab and check "Use Surrogate Process" option. In the edit box below specify path to dllhost.exe, for example: C:\WINDOWS\system32\dllhost.exe. Then select "Registry" tab, and you will see that DllSurrogate named value was added under AppId key.
Following scripts can be executed instead of running OleView.

For 64bit PGNP provider (Postgres/Greenplum):

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\AppID\{3170DFF1-4803-42a0-A1B3-D14656857070}]
"DllSurrogate"="C:\\Windows\\SysWow64\\dllhost.exe"
[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{3170DFF1-4803-42a0-A1B3-D14656857071}\InprocServer32]
@="C:\\Program Files (x86)\\Intellisoft\\PGNP\\PGNP64.dll"
"ThreadingModel"="Both"
```

For 32bit PGNP provider on 64bit Windows (Postgres/Greenplum):

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Classes\AppID\{3170DFF1-4803-42a0-A1B3-D14656857070}]
"DllSurrogate"="C:\\Windows\\SysWow64\\dllhost.exe"
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Classes\CLSID\{3170DFF1-4803-42a0-A1B3-D14656857071}\InprocServer32]
@="C:\\Program Files (x86)\\Intellisoft\\PGNP\\PGNP.dll"
"ThreadingModel"="Both"
```

For 32bit PGNP provider on 32bit Windows (Postgres/Greenplum):

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\AppID\{3170DFF1-4803-42a0-A1B3-D14656857070}]
"DllSurrogate"="C:\\Windows\\System32\\dllhost.exe"
[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{3170DFF1-4803-42a0-A1B3-D14656857071}\InprocServer32]
@="C:\\Program Files\\Intellisoft\\PGNP\\PGNP.dll"
"ThreadingModel"="Both"
```

For 64bit PGNP provider (Redshift):

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\AppID\{FFCCB099-E3D0-4FFC-87F9-AC3FAAF91270}]
"DllSurrogate"="C:\\Windows\\SysWow64\\dllhost.exe"
[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{FFCCB099-E3D0-4FFC-87F9-AC3FAAF91271}\InprocServer32]
@="C:\\Program Files (x86)\\Intellisoft\\RSNP\\PGNP64.dll"
"ThreadingModel"="Both"
```

For 32bit PGNP provider on 64bit Windows (Redshift):

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Classes\AppID\{FFCCB099-E3D0-4FFC-87F9-AC3FAAF91270}]
"DllSurrogate"="C:\\Windows\\SysWow64\\dllhost.exe"
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Classes\CLSID\{FFCCB099-E3D0-4FFC-87F9-AC3FAAF91271}\InprocServer32]
@="C:\\Program Files (x86)\\PGNP\\PGNP.dll"
"ThreadingModel"="Both"
```

For 32bit PGNP provider on 32bit Windows (Redshift):

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\AppID\{FFCCB099-E3D0-4FFC-87F9-AC3FAAF91270}]
"DllSurrogate"="C:\\Windows\\System32\\dllhost.exe"
[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{FFCCB099-E3D0-4FFC-87F9-AC3FAAF91271}\InprocServer32]
@="C:\\Program Files\\Intellisoft\\RSNP\\PGNP.dll"
"ThreadingModel"="Both"
```

2. Run DCOMCNFG.exe utility. Expand "Component Services", "Computers", "My Computer" and select "DCOM Config" folder. In the right pane find application named "{3170DFF1-4803-42a0-A1B3-D14656857070}" (or in Redshift case "{FFCCB099-E3D0-4FFC-87F9-AC3FAAF91270}"), right-click and select Properties. Configure Security and Identity in the corresponding tabs. Default values could work in many cases. Close the utility.

3. In SQL Server Management Studio uncheck "allow inprocess" option in PGNP provider properties. The changes are immediate and usually do not require restart of SQL Server or the computer. Alternatively the following stored procedure can be executed:

```
EXEC master.dbo.sp_MSset_oledb_prop N'PGNP.1', N'AllowInProcess', 0
```

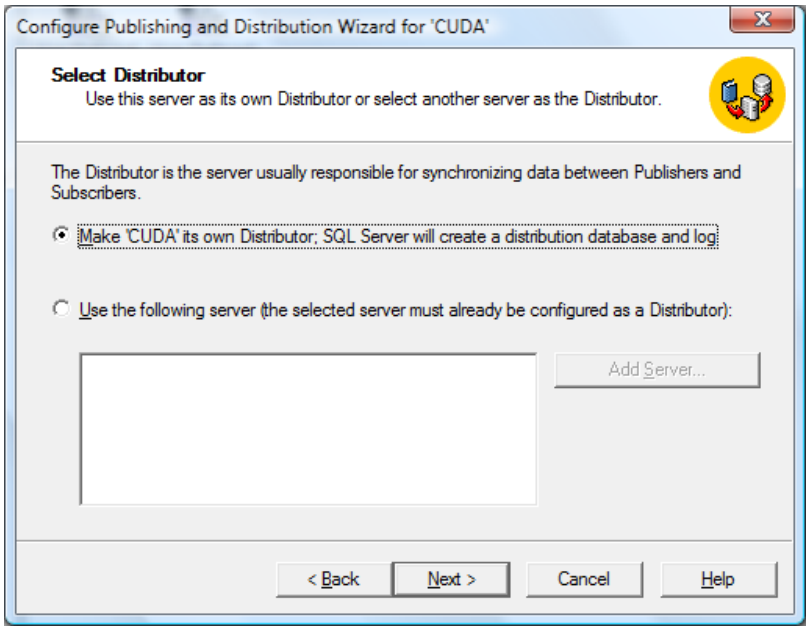
Perform a query for the Linked Server. If issue occurs see Events Viewer for errors information (e.g. run eventvwr.msc from command prompt). If a test query succeeded you can use Process Explorer (http://www.sysinternals.com) to find which surrogate process hosts the PGNP provider DLL.

3.4 Replication with SQL Server 2000

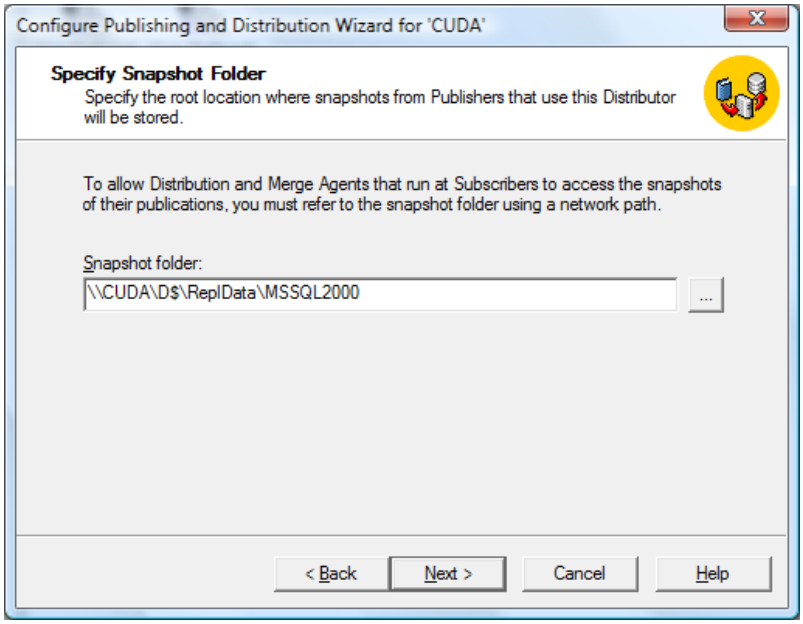
To set up replication use a login account that is a member of SQL Server's Process Administrators or higher authority server role, e.g. “sa” user.

3.4.1 Configuring Publisher, Subscribers and Distributor

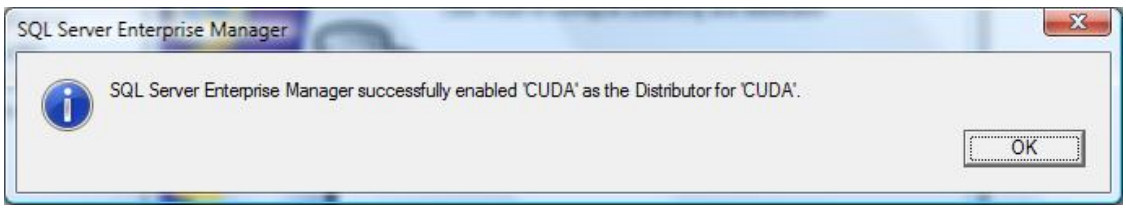
Start “SQL Server Enterprise Manager” and click “*Configure Publishing, Subscribers and Distribution…*” menu item under main menu Tools->Replication. Make your server a local distributor (CUDA is a computer name):



Specify location for snapshots:

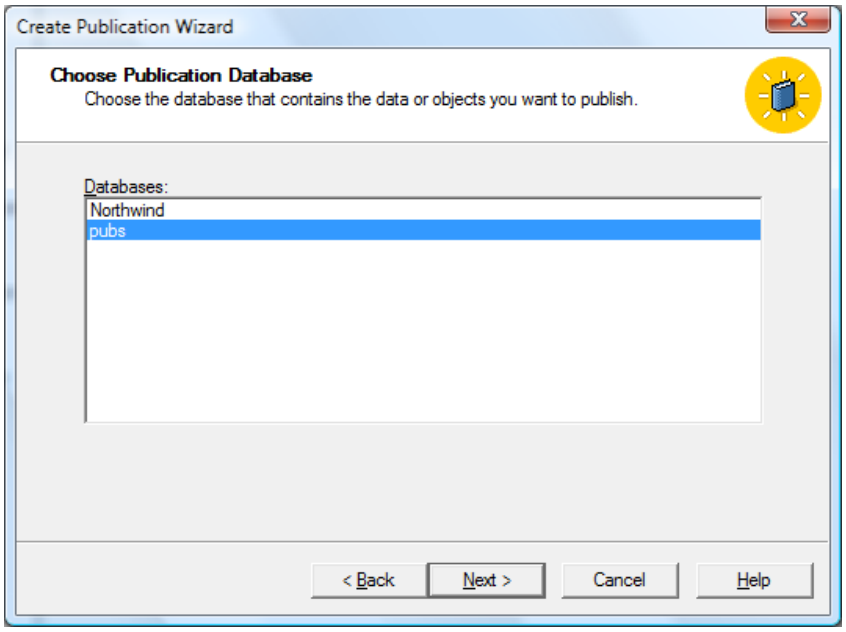


Accept default settings on the next page for simplicity (or choose custom setting). The server is configured as Distributor:

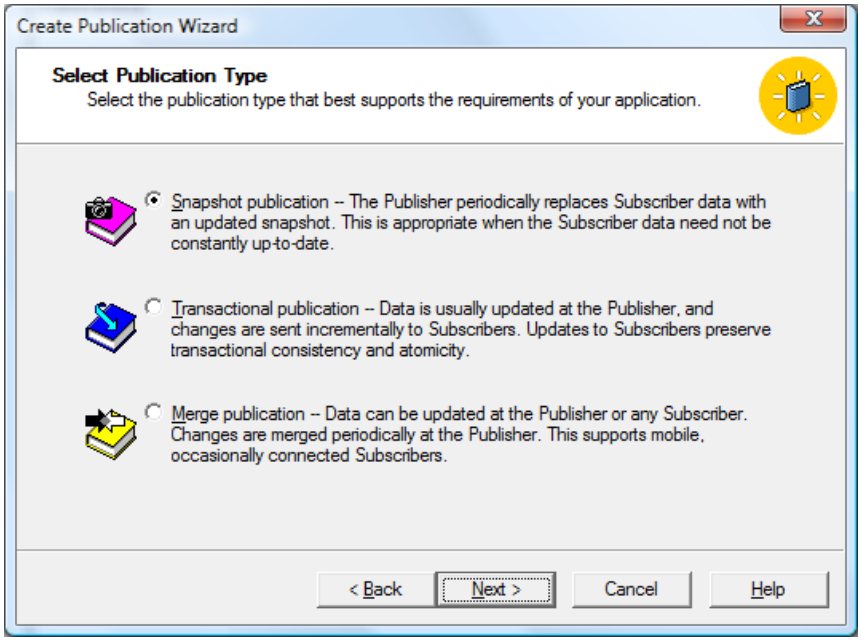


3.4.2 Creating publication

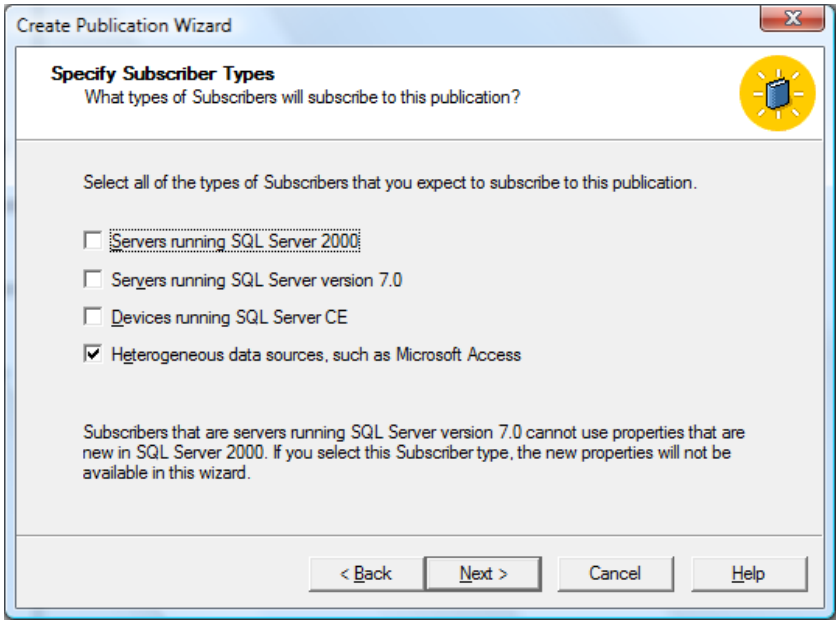
Right click on *"pubs"* database, select menu item "New->Publication…" and choose "pubs":



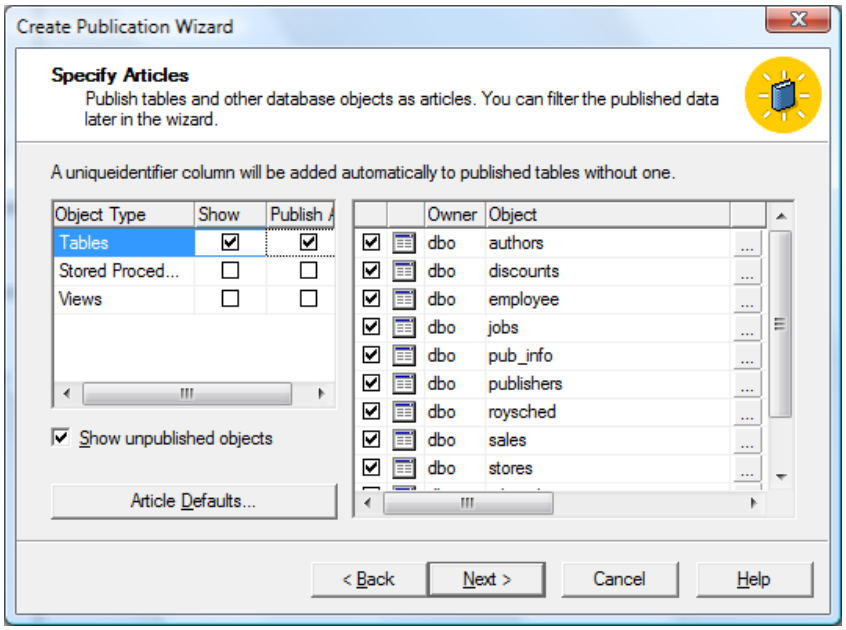
Choose “Snapshot” or “Transactional” publication type (“Merge” is not supported yet”):



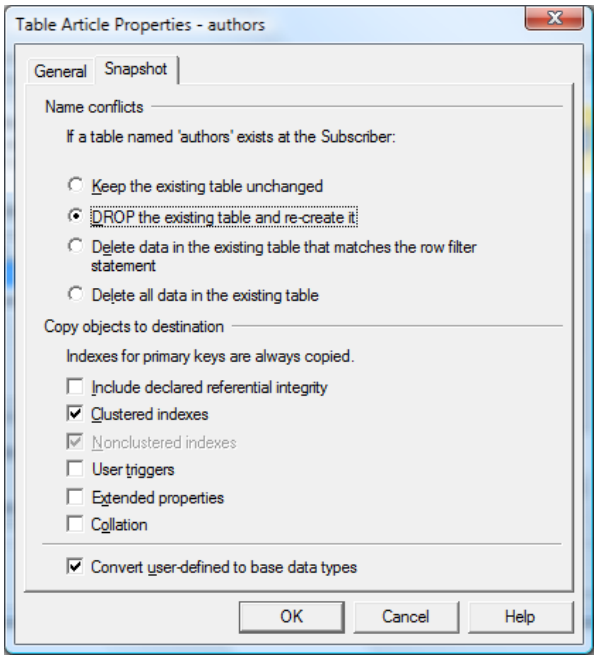
Choose “Heterogeneous data sources”:



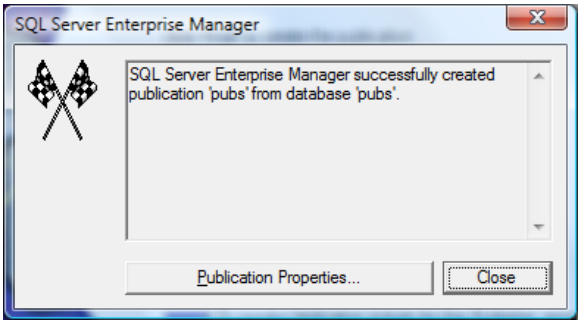
Choose all tables for replication:



Click on the ellipses for every table and review properties (optional):

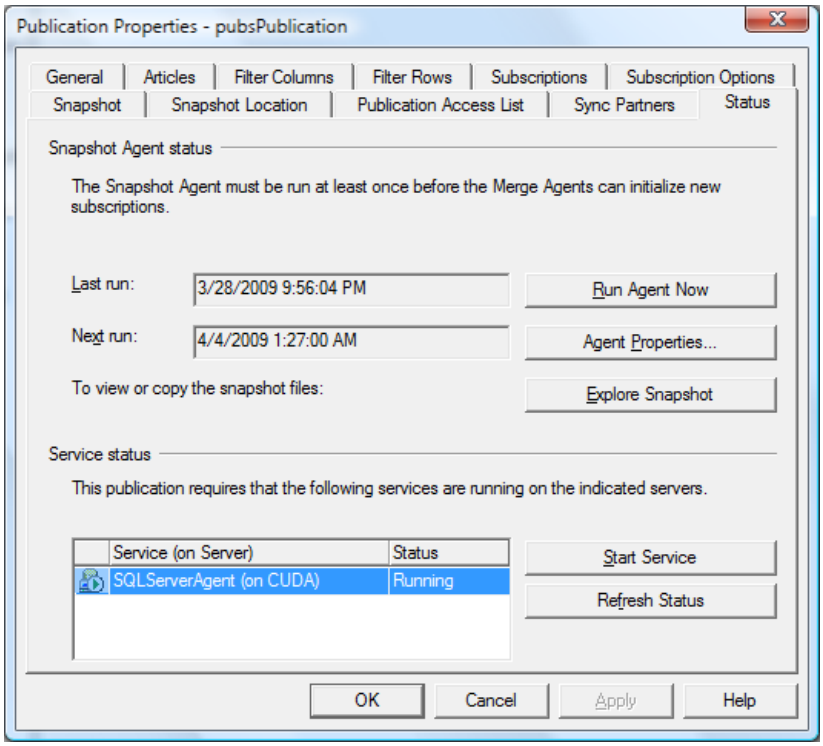


Click Next button “Create Publication Wizard” dialog until:



3.4.3 Create Snapshot

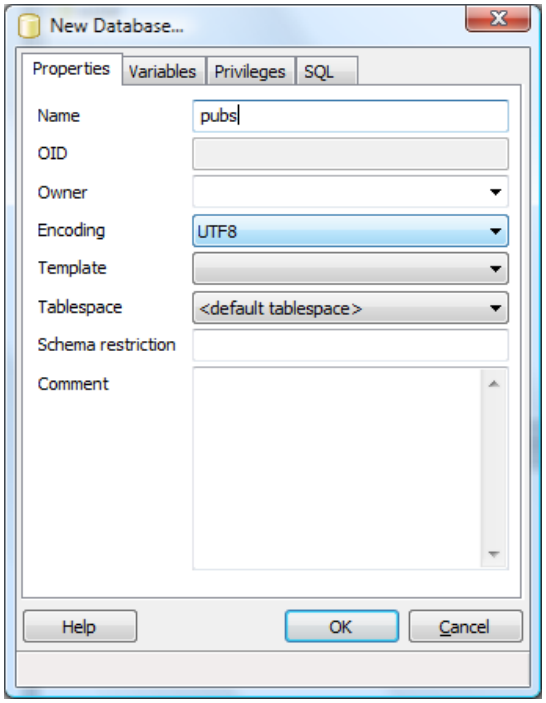
Right click on <*Server*>\Replication\Publications\pubsPublication:pubs and select Properties menu item. Then select Status tab:



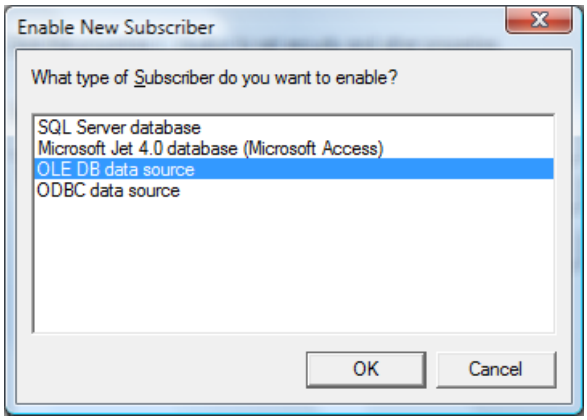
Click “Run Agent Now” button to create the snap shot files.

3.4.4 Adding Subscribers

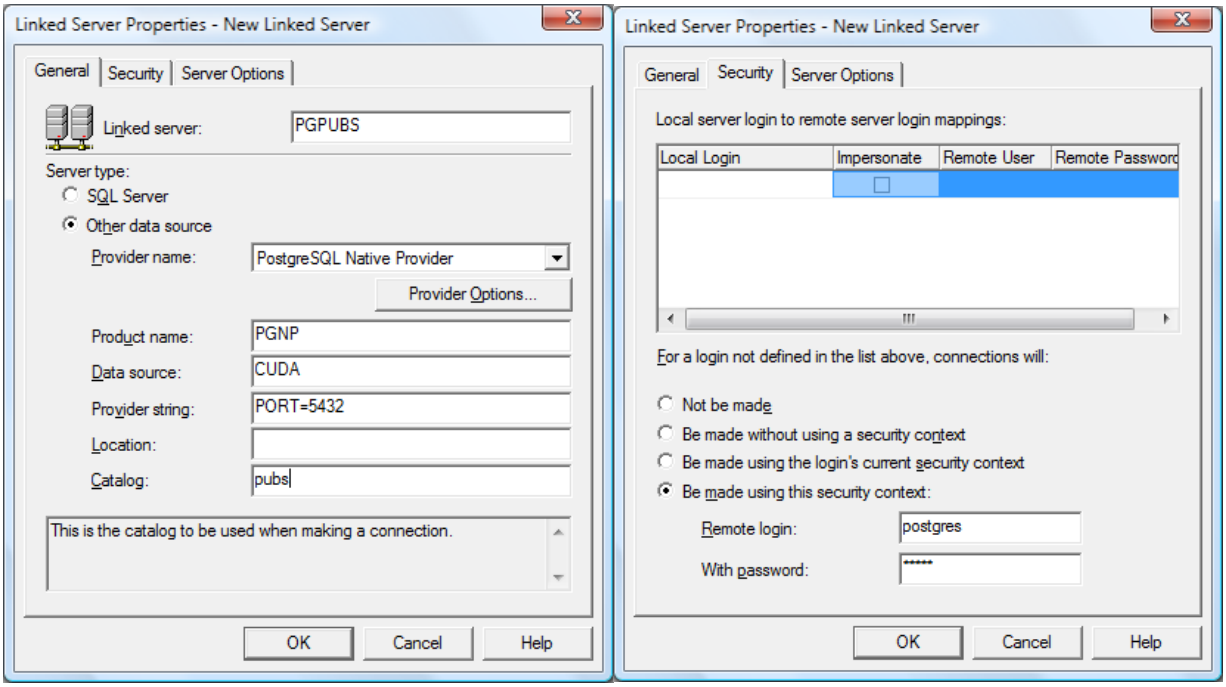
Create a new PostgreSQL Database:



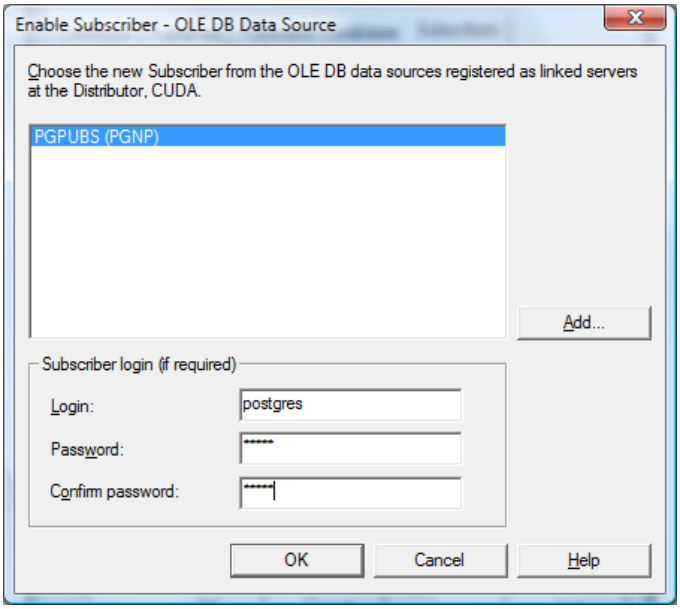
Click “*Configure Publishing, Subscribers and Distribution…*” menu item under main menu Tools->Replication and select Subscribers tab. Click “New…” button and select “OLEDB data source”:



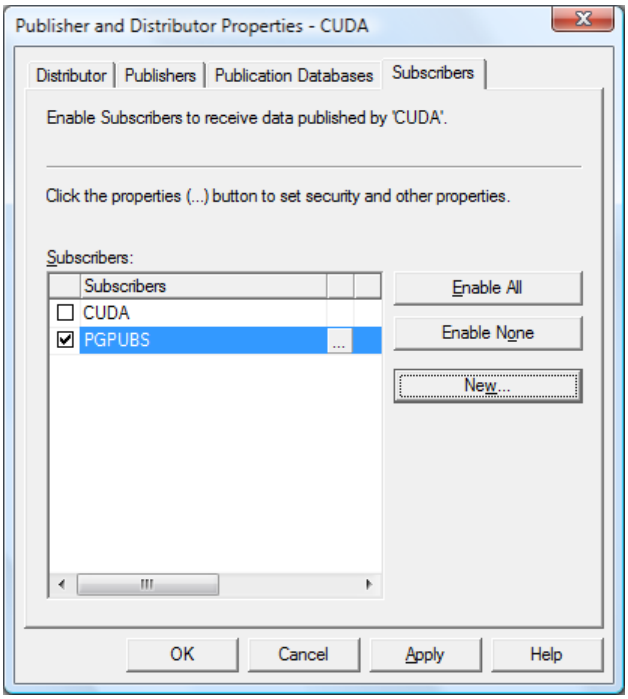
Then choose the subscriber (for instructions on how to create linked server see Chapter “Linked Servers” in this manual). Here are screenshots of the Linked Server configuration:



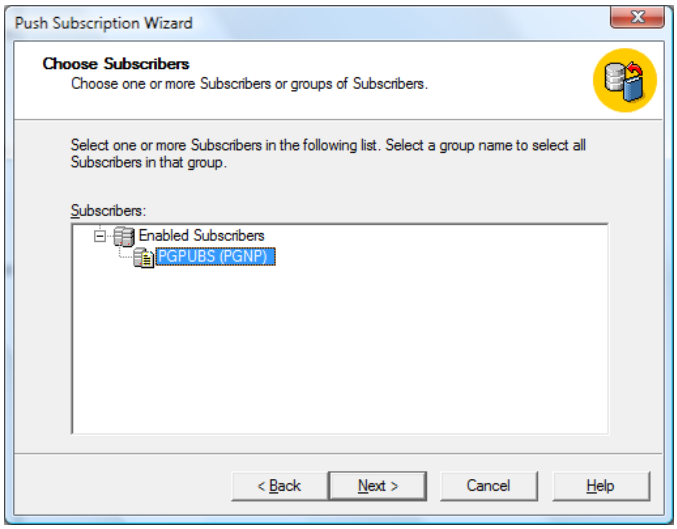
This is dialog for adding a subscriber:

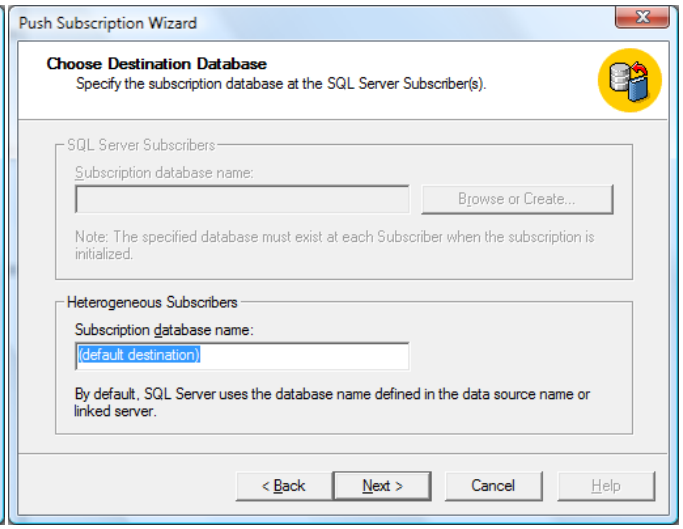


Click “OK” button in Publisher and Distributor Properties dialog:

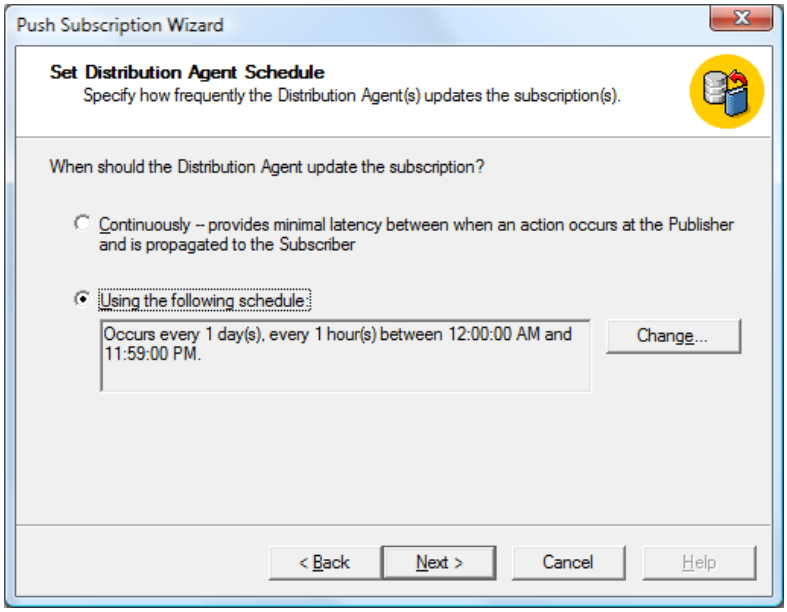


Right click on *<Server>/Replication/Publications/pubsPublication:pubs* and click “Push New Subscription” menu item to run the New Subscription wizard. Choose the newly created subscriber and the default destination for the Heterogeneous Subscribers:

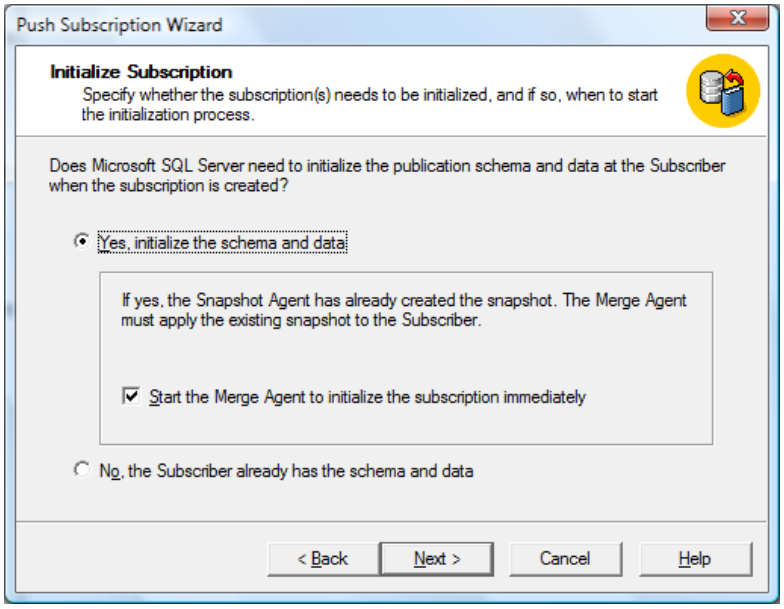




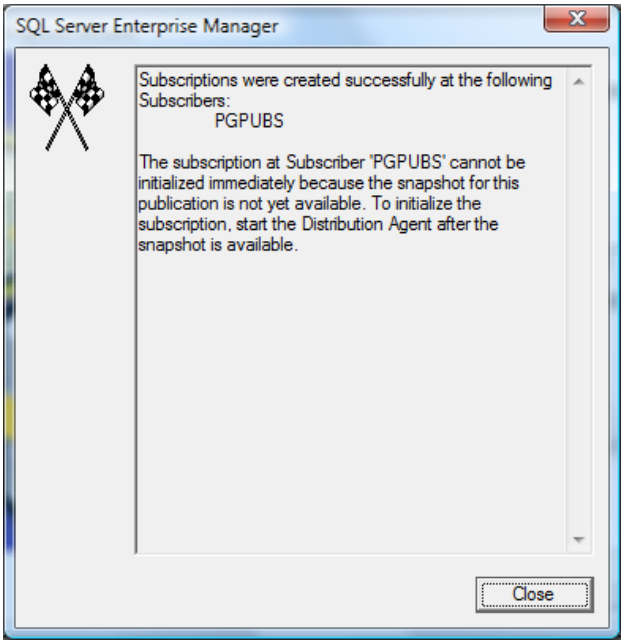
Then set Distribution Agent schedule:



Choose “Yes, initialize the schema and data” option and “Start … immediately” check box:

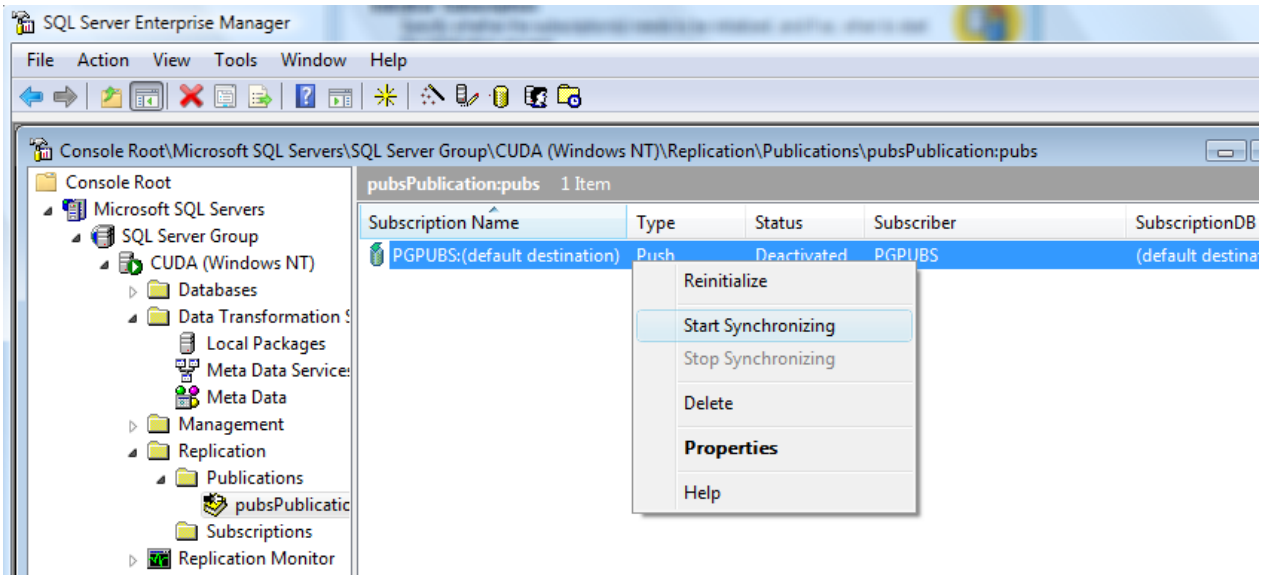


Click Next and Finish:



3.4.5 Synchronize

Right click on the *PGPUBS:(default destination)* item and select “Start Synchronization” menu item:


SQL Server Enterprise Manager
File Action View Tools Window Help
Console Root\Microsoft SQL Servers\SQL Server Group\CUDA (Windows NT)\Replication\Publications\pubsPublication:pubs
Console Root
Microsoft SQL Servers
SQL Server Group
CUDA (Windows NT)
Databases
Data Transformation S
Local Packages
Meta Data Service
Meta Data
Management
Replication
Publications
pubsPublicatic
Subscriptions
Replication Monitor
pubsPublication:pubs 1 Item
Subscription Name
Type
Status
Subscriber
SubscriptionDB
PGPUBS:(default destination)
Push
Deactivated
PGPUBS
(default destina
Reinitialize
Start Synchronizing
Stop Synchronizing
Delete
Properties
Help

3.5 Replication with SQL Server 2005/2008/2012

MS SQL Servers 2005/2008/2012, unlike SQL Server 2000, do not have the user friendly graphical interface for replication configuration. However, the replication can be configured via SQL stored procedures in SSMS.

Preliminary steps for Distributor:

```
EXEC sp_configure 'show advanced options', 1
GO
RECONFIGURE
GO
EXEC sp_configure 'Agent XPs', 1
GO
RECONFIGURE
GO
```

To enable SQL Agent configuration grant full access to Registry Key ***HKLM\System\CurrentControlSet\Services\SQLAgent$SQL2016*** to the user that runs ***sqlservr.exe*** process, e.g. ***MSSQL$SQL2016***.

3.5.1 Configure SQL Server as Distributor

In the following script replace

1) <server-name> with your SQL Server name, e.g. CUDA\INST5 – computer name and optional SQL instance name;

2) <path> with a folder path where replication files can be created, e.g. E:\MSSQL\ReplData – create the path manually before executing the script;

3) <user> with your SQL server user, e.g. sa;

4) <password> with the SQL user's password, e.g. sapwd.

```
use master
GO

exec sp_adddistributor @distributor = N'<server-name>', @password = N'<password>'
GO

exec sp_adddistributiondb @database = N'distribution', @data_folder = N'<path>', @log_folder = N'<path>',
 @log_file_size = 2, @min_distretention = 0, @max_distretention = 72, @history_retention = 48, @security_mode =
1
GO
```

3.5.2 Configure the publisher to use a specified distribution database

```
use [distribution]
if (not exists (select * from sysobjects where name = 'UIProperties' and type = 'U '))
        create table UIProperties(id int)
if (exists (select * from ::fn_listextendedproperty('SnapshotFolder', 'user', 'dbo', 'table', 'UIProperties',
null, null)))
        EXEC sp_updateextendedproperty N'SnapshotFolder', N'<path>', 'user', dbo, 'table', 'UIProperties'
else
        EXEC sp_addextendedproperty N'SnapshotFolder', N'<path>', 'user', dbo, 'table', 'UIProperties'
GO

exec sp_adddistpublisher @publisher = N'<server-name>', @distribution_db = N'distribution', @security_mode = 0,
  @login = N'<user>', @password = N'<password>', @working_directory = N'<path>', @trusted = N'false',
  @thirdparty_flag = 0, @publisher_type = N'MSSQLSERVER'
GO
```

3.5.3 Create Linked Server

This is an optional step, in case you want to test connectivity between SQL Server and Postgres. Create an empty PostgreSQL database using psql or PGAdmin and a Linked Server in SSMS (refer to section 2.3.2 for details).

Please double check that “Allow inprocess” box is checked in Server Objects\Linked Servers\Providers\PGNP\Properties dialog (access it via treeview pane on the left in the SSMS).

3.5.4 Create the snapshot publication

In the following script replace

1) <sql-database> with your SQL database name;

2) <SQLHOST> with SQL Server host name (user friendly name), e.g. CUDA_INST5;

```
use [<sql-database>]
exec sp_replicationdboption @dbname = N'<sql-database>', @optname = N'publish', @value = N'true'
GO

exec sp_addpublication @publication = N'pgnpsnap1',
   @description = N'Snapshot publication of database ''distribution'' from Publisher ''<SQLHOST>''.',
   @sync_method = N'native', @retention = 0, @allow_push = N'true', @allow_pull = N'true', @allow_anonymous =
N'true',
   @enabled_for_internet = N'false', @snapshot_in_defaultfolder = N'true', @compress_snapshot = N'false',
@ftp_port = 21,
   @ftp_login = N'anonymous', @allow_subscription_copy = N'false', @add_to_active_directory = N'false',
   @repl_freq = N'snapshot', @status = N'active', @independent_agent = N'true', @immediate_sync = N'true',
   @allow_sync_tran = N'false', @autogen_sync_procs = N'false', @allow_queued_tran = N'false', @allow_dts =
N'false',
   @replicate_ddl = 1
GO

exec sp_addpublication_snapshot @publication = N'pgnpsnap1',
@frequency_type = 1, @frequency_interval = 0, @frequency_relative_interval = 0, @frequency_recurrence_factor =
0, @frequency_subday = 0, @frequency_subday_interval = 0, @active_start_time_of_day = 0, @active_end_time_of_day
= 235959, @active_start_date = 0, @active_end_date = 0, @job_login = null, @job_password = null,
@publisher_security_mode = 1
GO
```

For each table <table-name> execute following command:

```
exec sp_addarticle @publication = N'pgnpsnap1', @article = N'<table-name>',
 @source_owner = N'dbo', @source_object = N'<table-name>', @type = N'logbased',
 @description = null, @creation_script = null, @pre_creation_cmd = N'none', @schema_option = 0x000000000803509D,
@identityrangemanagementoption = N'manual', @destination_table = N'<table-name>', @destination_owner = N'dbo',
 @vertical_partition = N'false'
GO
```

Note: *the* `@pre_creation_cmd` *parameter can be set to* `N'drop'` *if table already exists, and needs to be dropped.*

Ensure that non-SQL Server Subscribers are supported:

```
exec sp_changepublication N'pgnpsnap1', N'enabled_for_het_sub', N'true',
@force_invalidate_snapshot = 1, @force_reinit_subscription = 1
```

3.5.5 Create the snapshot subscription

This final step adds a new scheduled agent job. Please replace

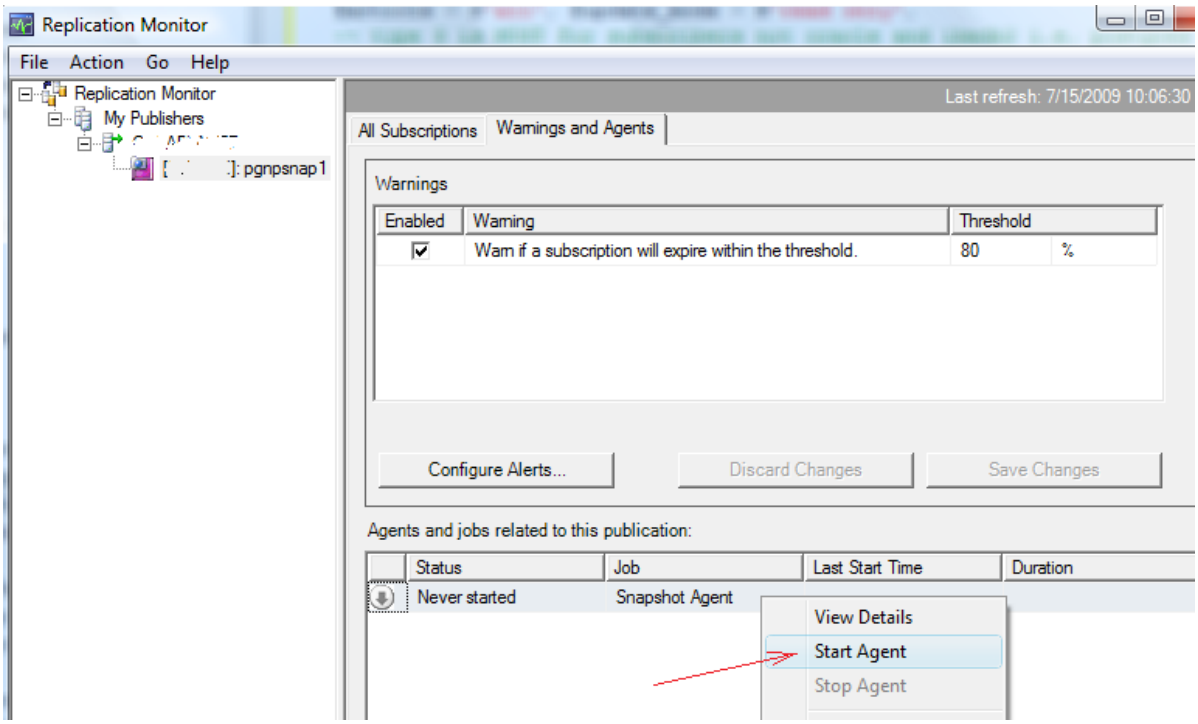
1) <postgres-user> with PostgreSQL user name, e.g. ‘postgres’;
2) <postgres-pwd> with PostgreSQL user’s password, e.g. 12345;
3) <postgres-host> with computer name running PostgreSQL Server;
4) <extended-properties> with any PGNP extended properties, e.g. ‘LOWERCASESCHEMA=OFF;SEARCH_PATH=vinci;PORT=5432;’, for more details see Connection String;
5) <postgres-database> with PostgreSQL database name created on “Create Linked Server” step above.

```
exec sp_addsubscription @publication = N'pgnpsnap1', @subscriber = N'<postgres-host>', @destination_db =
N'<postgres-database>',
  @subscription_type = N'Push',
  @sync_type = N'automatic',
  @article = N'all', @update_mode = N'read only',
  -- type 3 is MUST for subscribers not oracle and ibmdb2 i.e. postgres
```

```
  @subscriber_type = 3

exec sp_changesubscription @publication = N'pgnpsnap1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_login', @value=N'<postgres-user>'

exec sp_changesubscription @publication = N'pgnpsnap1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_password', @value=N'<postgres-pwd>'

exec sp_changesubscription @publication = N'pgnpsnap1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_location', @value=N''

exec sp_changesubscription @publication = N'pgnpsnap1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_datasource', @value=N'<postgres-host>'

exec sp_changesubscription @publication = N'pgnpsnap1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_provider', @value=N'PGNP'

exec sp_changesubscription @publication = N'pgnpsnap1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_providerstring',
  @value=N'<extended-properties>'

exec sp_changesubscription @publication = N'pgnpsnap1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_catalog', @value=N'<postgres-database>'

exec sp_addpushsubscription_agent @publication = N'pgnpsnap1',
  @subscriber = N'<postgres-host>', @subscriber_db = N'<postgres-database>',
  @job_login = null, @job_password = null, @subscriber_security_mode = 0,
  @subscriber_provider = N'PGNP',
  @subscriber_datasrc = N'<postgres-host>',
  @subscriber_location =N'',
  @subscriber_provider_string = N'<extended-properties>',
  @subscriber_catalog = N'<postgres-database>',
  @subscriber_login = N'<postgres-user>', @subscriber_password =N'<postgres-pwd>',
  @frequency_type = 64,
  @frequency_interval = 0, @frequency_relative_interval = 0,
  @frequency_recurrence_factor = 0, @frequency_subday = 0,
  @frequency_subday_interval = 0, @active_start_time_of_day = 0,
  @active_end_time_of_day = 235959, @active_start_date = 20090715,
  @active_end_date = 99991231, @enabled_for_syncmgr = N'False',
  @dts_package_location = N'Distributor'
GO
```

For more information read http://msdn.microsoft.com/en-us/library/bb510544.aspx.

To create first snapshot and start replication, open Replication Monitor (right-click on Replication\Launch Replication Monitor), then select pgnpsnap1 publisher on the left, right-click Snapshot Agent line in “Warnings and Agents” tab and click “Start Agent”:



3.5.6 Deleting subscription and publication

Following commands can be used for deleting previously created subscription and/or publication:

```
exec sp_dropsubscription @publication = N'pgnpsnap1', @article = N'all', @subscriber = N'<postgres-host>';

exec sp_droppublication @publication = N'pgnpsnap1'
```

The steps above described snapshot replication configuration. To configure transactional replication perform following steps instead of steps 2.5.4 and 2.5.5.

3.5.7 Create publication for transactional replication

In the following script replace

1) <sql-database> with your SQL database name;

2) <SQLHOST> with SQL Server host name (user friendly name), e.g. CUDA_INST5;

```
use [<sql-database>]
exec sp_replicationdboption @dbname = N'<sql-database>', @optname = N'publish', @value = N'true'
GO

exec sp_addpublication @publication = N'pgnptrans1',
   @description = N'Transactional publication of database ''distribution'' from Publisher ''<SQLHOST>''.',
   @sync_method = N'native', @retention = 0, @allow_push = N'true', @allow_pull = N'true', @allow_anonymous =
N'true',
   @enabled_for_internet = N'false', @snapshot_in_defaultfolder = N'true', @compress_snapshot = N'false',
@ftp_port = 21,
   @ftp_login = N'anonymous', @allow_subscription_copy = N'false', @add_to_active_directory = N'false',
   @repl_freq = N'continuous', @status = N'active', @independent_agent = N'true', @immediate_sync = N'true',
   @allow_sync_tran = N'false', @autogen_sync_procs = N'false', @allow_queued_tran = N'false', @allow_dts =
N'false',
   @replicate_ddl = 1, @allow_initialize_from_backup = N'false', @enabled_for_p2p = N'false',
@enabled_for_het_sub = N'false'
GO

exec sp_addpublication_snapshot @publication = N'pgnptrans1',
@frequency_type = 1, @frequency_interval = 0, @frequency_relative_interval = 0, @frequency_recurrence_factor =
0, @frequency_subday = 0, @frequency_subday_interval = 0, @active_start_time_of_day = 0, @active_end_time_of_day
= 235959, @active_start_date = 0, @active_end_date = 0, @job_login = null, @job_password = null,
@publisher_security_mode = 1
GO
```

For each table <table-name> execute following command:

```
exec sp_addarticle @publication = N'pgnptrans1', @article = N'<table-name>',
 @source_owner = N'dbo', @source_object = N'<table-name>', @type = N'logbased',
 @description = null, @creation_script = null, @pre_creation_cmd = N'none', @schema_option = 0x000000000803509D,
 @identityrangemanagementoption = N'manual', @destination_table = N'<table-name>', @destination_owner = N'dbo',
 @vertical_partition = N'false'
GO
```

Ensure that non-SQL Server Subscribers are supported:

```
exec sp_changepublication N'pgnptrans1', N'enabled_for_het_sub', N'true',
  @force_invalidate_snapshot = 1, @force_reinit_subscription = 1
```

3.5.8 Create subscription for transactional replication

This final step adds a new scheduled agent job. Please replace

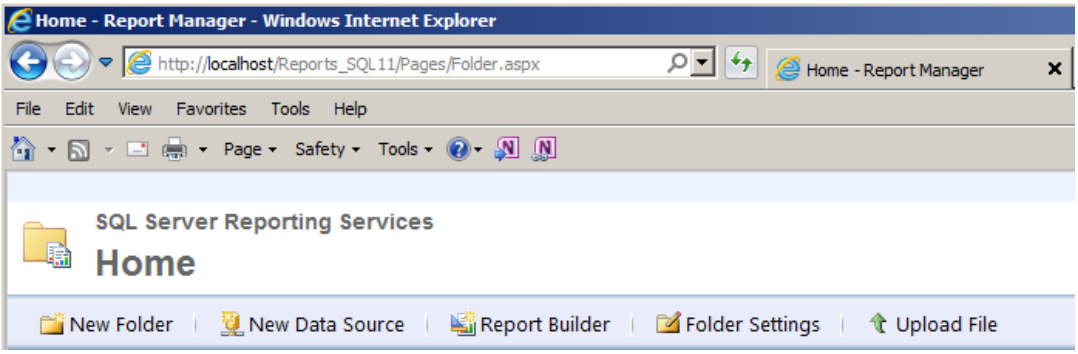
1) <postgres-user> with PostgreSQL user name, e.g. 'postgres';
2) <postgres-pwd> with PostgreSQL user's password, e.g. 12345;
3) <postgres-host> with computer name running PostgreSQL Server;
4) <extended-properties> with any PGNP extended properties, e.g. the following could be used in most cases 'BULK_METHOD=PIPECOPY;BULK_INSERT=1000;COMMAND_TIMEOUT=900;', for more details see Connection String;
5) <postgres-database> with PostgreSQL database name created on "Create Linked Server" step above.

```
exec sp_addsubscription @publication = N'pgnptrans1', @subscriber = N'<postgres-host>', @destination_db =
N'<postgres-database>',
  @subscription_type = N'Push',
  @sync_type = N'none',
  @article = N'all', @update_mode = N'read only',
  @subscriber_type = 3

exec sp_changesubscription @publication = N'pgnptrans1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_login', @value=N'<postgres-user>'

exec sp_changesubscription @publication = N'pgnptrans1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_password', @value=N'<postgres-pwd>'

exec sp_changesubscription @publication = N'pgnptrans1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_location', @value=N''

exec sp_changesubscription @publication = N'pgnptrans1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_datasource', @value=N'<postgres-host>'

exec sp_changesubscription @publication = N'pgnptrans1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_provider', @value=N'PGNP'

exec sp_changesubscription @publication = N'pgnptrans1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_providerstring',
  @value=N'<extended-properties>'

exec sp_changesubscription @publication = N'pgnptrans1', @subscriber = N'<postgres-host>',
  @destination_db = N'<postgres-database>', @article = N'all',
  @property=N'subscriber_catalog', @value=N'<postgres-database>'

exec sp_addpushsubscription_agent @publication = N'pgnptrans1',
  @subscriber = N'<postgres-host>', @subscriber_db = N'<postgres-database>',
  @job_login = null, @job_password = null, @subscriber_security_mode = 0,
  @subscriber_provider = N'PGNP',
  @subscriber_datasrc = N'<postgres-host>',
  @subscriber_location =N'',
  @subscriber_provider_string = N'<extended-properties>',
  @subscriber_catalog = N'<postgres-database>',
  @subscriber_login = N'<postgres-user>', @subscriber_password =N'<postgres-pwd>',
  @frequency_type = 64,
  @frequency_interval = 0, @frequency_relative_interval = 0,
```

```
  @frequency_recurrence_factor = 0, @frequency_subday = 0,
  @frequency_subday_interval = 0, @active_start_time_of_day = 0,
  @active_end_time_of_day = 235959, @active_start_date = 20090715,
  @active_end_date = 99991231, @enabled_for_syncmgr = N'False',
  @dts_package_location = N'Distributor'
GO
```

3.6 Generating reports in SQL Server Reporting Services

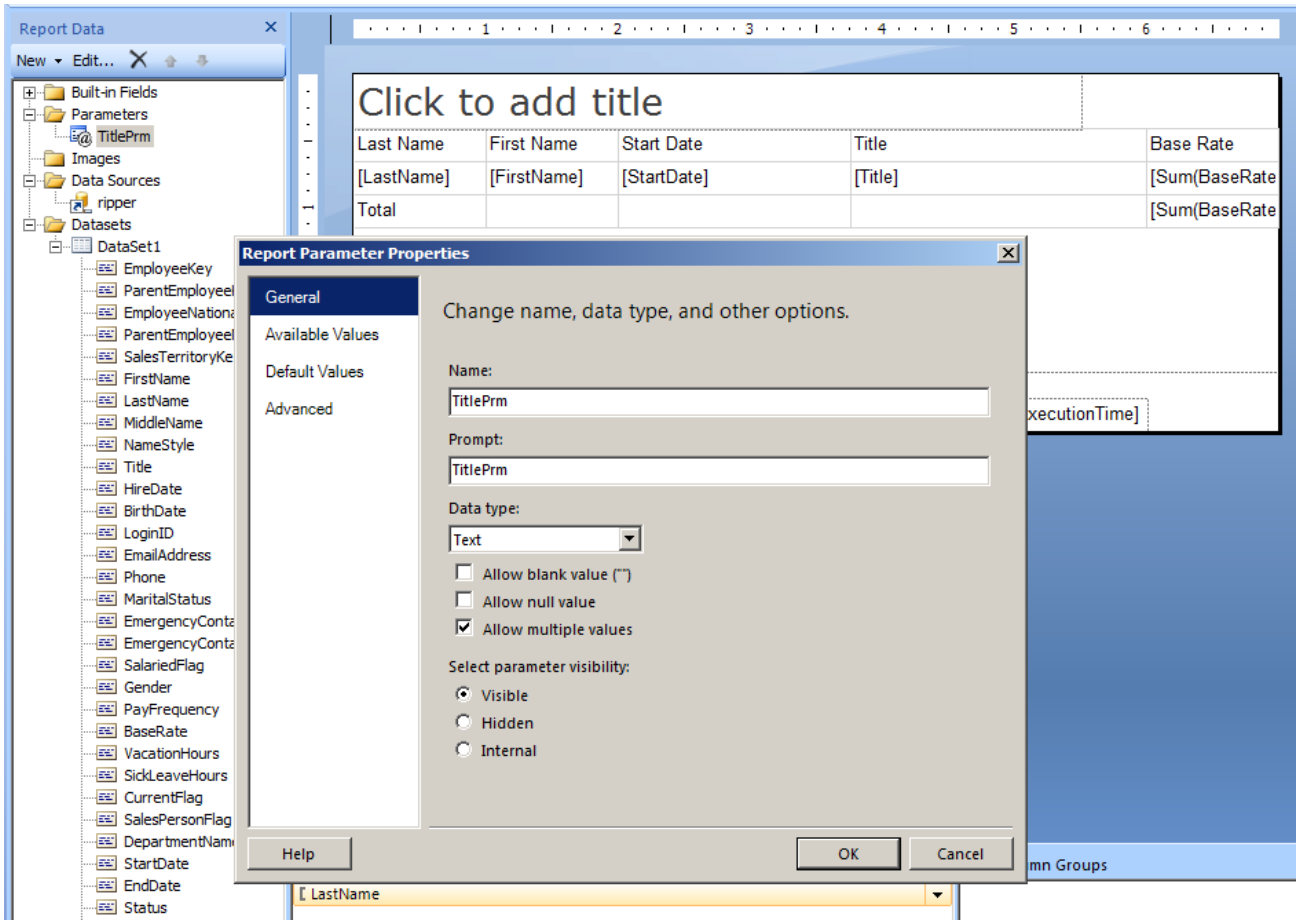
Navigate to the Report Manager in the Internet Explorer using a URL, e.g. http://localhost/Reports_SQL11 (where SQL11 is the name of the SLQ Server database instance; it could be different).



Launch the Report Builder, and create "New Report" using "Table or Matrix Wizard". Create new OLE Data Source, and specify connection parameters to the Postgres database. For the Dataset use following command:

="select * from [DimEmployee] where [Title] in ('" & JOIN(Parameters!TitlePrm.Value, "','") & "') "

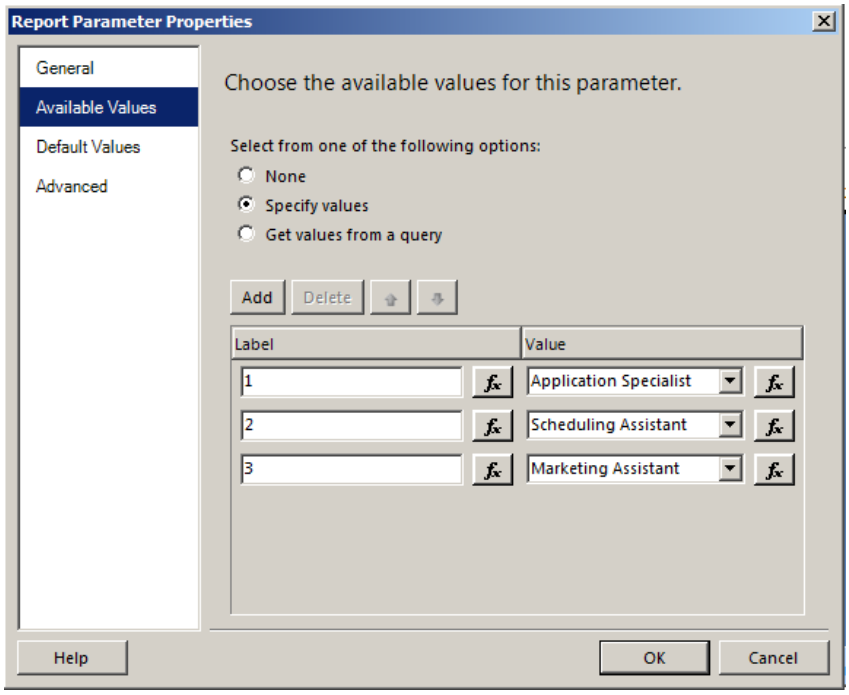
Choose columns to populate in the report. Then create the report parameter as shown below:



Note: The OLE DB standard says that multiple parameters can be used only with commands that do not return rowsets. This is the reason why we must use IN clause and the JOIN function to pass multiple parameters. Otherwise SSRS will

display the error: “Cannot add multi value query parameter 'prm' for dataset 'DataSet1' because it is not supported by the data extension. (rsErrorAddingMultiValueQueryParameter)”.

On the Available Values page specify some hard coded values:



Run the report with all parameter values:

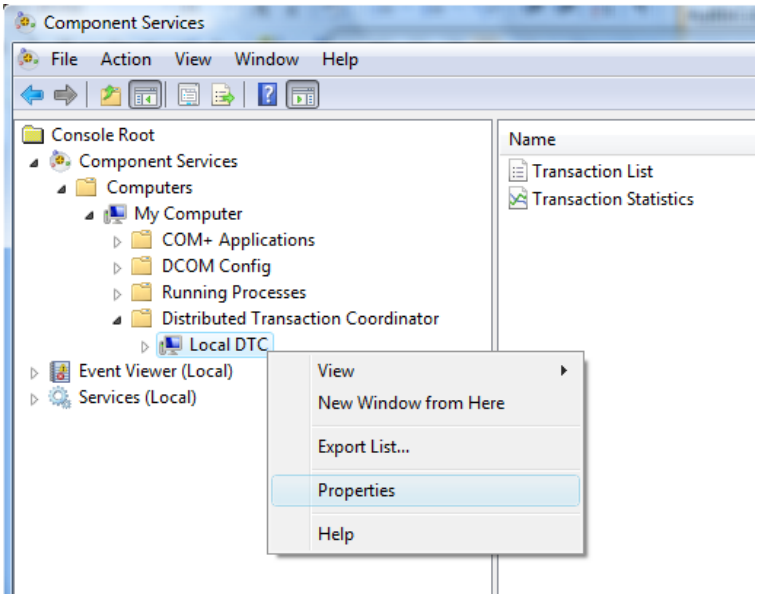
Last Name	First Name	Start Date	Title	Base Rate
Bacon	Dan	2/12/2003 12:00:00 AM	Application Specialist	27.4038
Benshoof	Wanida	2/7/2005 12:00:00 AM	Marketing Assistant	13.4615
Berg	Karen	3/20/2003 12:00:00 AM	Application Specialist	27.4038
Brewer	Alan	3/17/2003 12:00:00 AM	Scheduling Assistant	16.0000
Brown	Kevin	2/26/2001 12:00:00 AM	Marketing Assistant	13.4615
Bueno	Janaina	1/24/2003 12:00:00 AM	Application Specialist	27.4038
Dempsey	Mary	3/17/2005 12:00:00 AM	Marketing Assistant	13.4615
LaMee	Brian	4/4/2003 12:00:00 AM	Scheduling Assistant	16.0000
Meyyappan	Ramesh	3/7/2003 12:00:00 AM	Application Specialist	27.4038
Uddin	Sairaj	2/27/2003 12:00:00 AM	Scheduling Assistant	16.0000
Vong	William	2/8/2003 12:00:00 AM	Scheduling Assistant	32.0000
Total				229.9997

3.7 Two phase commit protocol (2PC)

PGNP Provider implements support for 2PC. You will need to configure and run Distributed Transaction Coordinator.

3.7.1 Configuring DTC

From a command prompt execute following command: **dcomcnfg**. Expand *Component Services* node and right-click on *Local DTC* node, select properties as shown below:



Navigate to *Security* tab and make sure that "Enable XA Transactions" checkbox is selected. "Network DTC Access", "Allow Inbound" and "Allow Outbound" must be selected as well. Please read MSDN articles for more details on configuring DTC.

Click OK in the *Local DTC Properties* dialog and restart DTC service (see the next paragraph).

3.7.2 Starting DTC Service

DTC can be started either from Services snap-in (**services.msc** command) or using commands "**net stop msdtc**" and "**net start msdtc**". If you planning to use 2PC regularly then consider configuring DTC Service for automatic startup in the Services snap-in.

3.7.3 Enabling prepared transactions in PostgreSQL

Some later versions of PostgreSQL have prepared transactions disabled by default. To enable the prepared transactions, edit *postgresql.conf* file as described below. Open the *postgresql.conf* file in editor and find line with `max_prepared_transactions` parameter (if missing, new line can be added). Uncomment the line by removing '#' symbol in front and set the parameter equal to maximum allowed number of connections or more, e.g.

```
max_prepared_transactions = 100;     # zero disables the feature
```

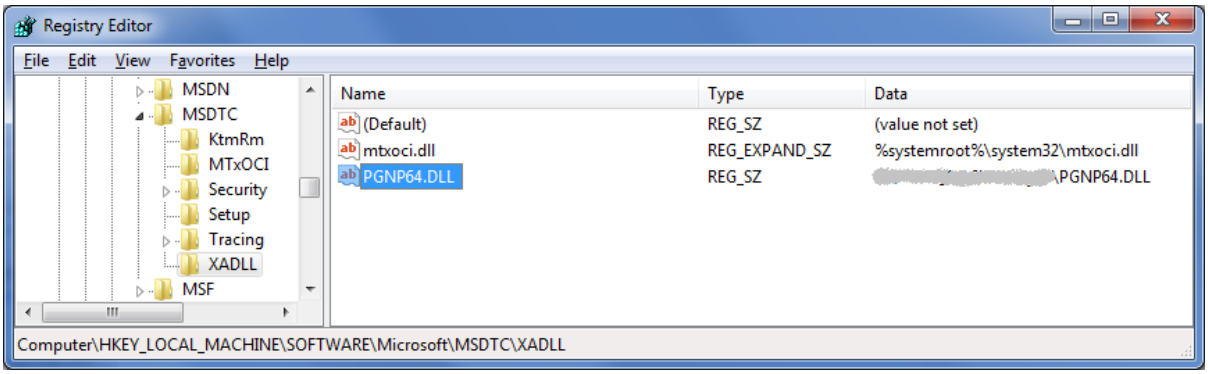
Restart PostgreSQL Server.

3.7.4 Troubleshooting issues with 2PC

Often issues with 2PC are caused by misconfiguration, or failure in registration. Sometimes reinstalling provider, and repeating the configuration steps from this chapter can resolve the issues. Running the PGNP Profiler may provide additional error information that might help in troubleshooting.

If the following error is returned: "MSDTC XARMCreate error", please check if the following registry keys exist:

HKLM\Software\Wow6432Node\Microsoft\MSDTC\XADLL\PGNP.DLL=<path>
HKLM\Software\Microsoft\MSDTC\XADLL\PGNP64.DLL=<path>



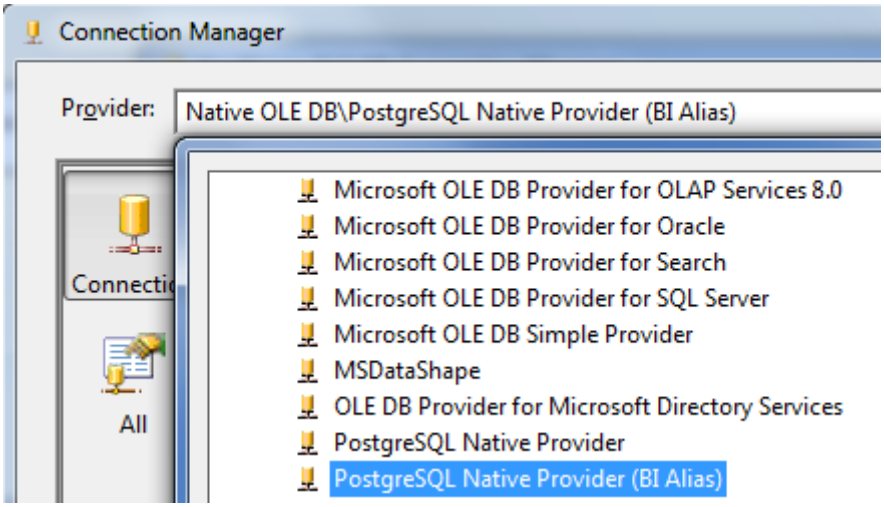
The key is created automatically when the provider is enlisting in a distributed transaction for the first time. The provider may not be able to create the key depending on the parent process security context. In this case, either manually create the keys, or set proper security permissions on the registry folder, and restart the distributed transaction. For example, grant "Network Service" account permissions for full access to HKLM\Software\Microsoft\MSDTC\XADLL, if DTC runs under the "Network Service" account.

3.8 FastLoad feature

The original implementation of the PGNP provider allowed using "BULK_METHOD=COPY;BULK_INSERT=1000;" for fast data import into the Postgres database. Starting with 1.4.0.3076 a new method is available based on implementation of IRowsetFastLoad interface (Server Edition only). Using the FastLoad is a recommended way of loading data into the Postgres database. This chapter describes an example of table loading from SQL Server into Postgres using FastLoad.

3.8.1 Configuring OLE DB connection in BIDS

Launch Business Intelligence Development Studio (BIDS), and then create a new "Integration Services Project". Click the right mouse button in "Connection Managers" tab, and select "New OLE DB Connection…" menu item. Click "New…" button, and then select "PostgreSQL Native Provider (BI Alias)" in the "Connection Manager" dialog. Note that "(BI Alias)" must be selected for the FastLoad feature to work.

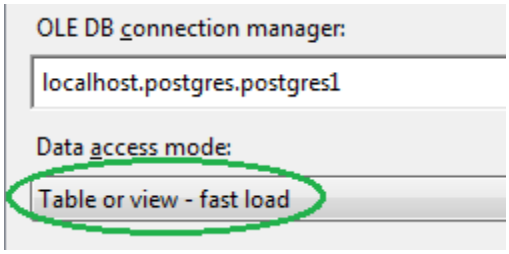


Regular PGNP Provider name used in a connection string is “PGNP”. But the “BI Alias” name is “SQLOLEDBPGNP”. This is needed because of undocumented Microsoft implementation which enables various features only to providers with names started with “SQLOLEDB”.

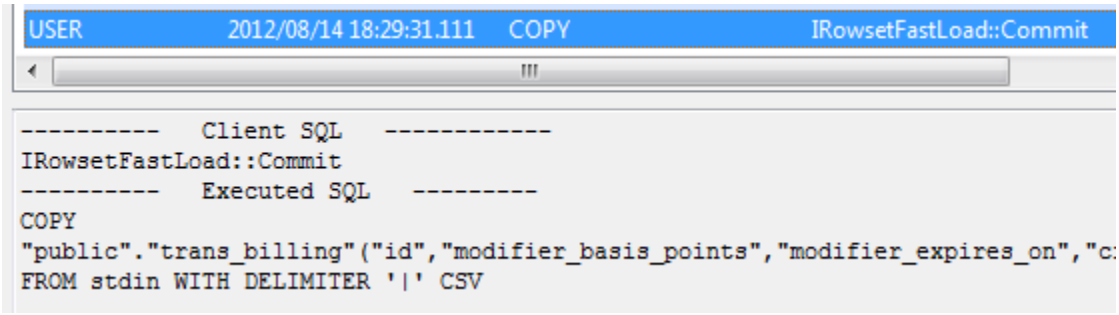
Also create a connection to a SQL Server database with source data (using “SQL Server Native Client”).

3.8.2 Configuring Source and Destination

Add “Data Flow Task”, and configure OLE DB Source by choosing a table in SQL Server database. Then add OLE DB Destination, connect the green arrow from the Source to the Destination, and edit the Destination. In the “OLE DB Destination Editor” choose Postgres connection, and the “Data access mode” as shown:



Proceed by configuring column mapping. Then execute the package, and check the result of the rows copying. Note, that PGNP Profiler will show something like the following:



3.9 The Query Optimizer

The Query Optimizer is a built-in mechanism to transform SQL queries in the PGNP Provider. It can be used to troubleshoot performance issues in large databases without a need to change SQL in an application. The Optimizer is closely integrated with PGNP Profiler (see 5.3 in Appendix A. Utilities). User can define rules for replacing SQL queries "on the fly". The rules can use Exact Match, or Template Match modes for optimizing the queries.

In Exact Match mode, an application query is replaced with the "optimized" query only when the application query is the same as one specified in a rule.

In Template Match mode, an application query is replaced with the "optimized" query when the application query matches a query in a rule so that any numerical constants or literals ignored.

To enable Optimizer add **OPTIMIZER=ON** to the Extended Properties of the Connection String.

The Optimizer uses a special table to store configuration. It can be created manually by executing the statements shown below, or let the PGNP Profiler automatically create it (recommended).

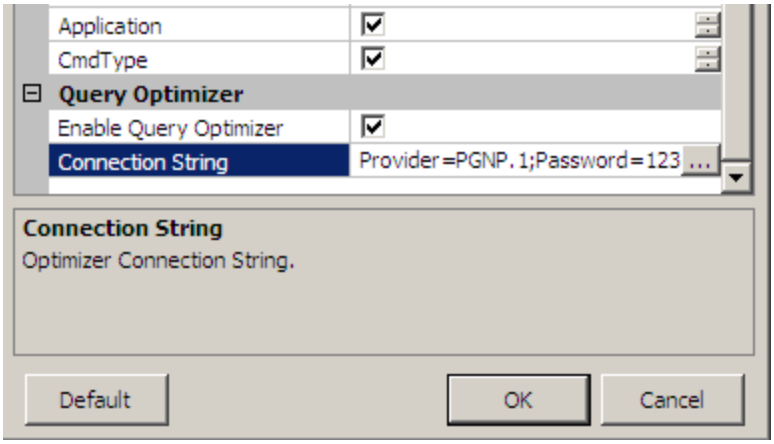
Commands for enabling Optimizer in Postgres and Greenplum:

```
CREATE TABLE pgnp_optimizer
(
  optimizer_id serial PRIMARY KEY,
  hashtype int2,
  enabled character(1),
  original text,
  hash int4,
  final text,
  category varchar(32),
  modified timestamp
);

UPDATE PGNP_OPTIMIZER SET ENABLED='Y' WHERE HASH=0;
INSERT INTO PGNP_OPTIMIZER(HASHTYPE, ENABLED, HASH, CATEGORY, MODIFIED) SELECT 0, 'Y', 0,
'System', now() WHERE NOT EXISTS (SELECT 1 FROM PGNP_OPTIMIZER WHERE HASH=0);
```

Commands for enabling Optimizer in Redshift:

```
CREATE TABLE PGNP_OPTIMIZER
(
  OPTIMIZER_ID int IDENTITY(1,1) PRIMARY KEY,
  HASHTYPE int2 NOT NULL,
  ENABLED char,
  ORIGINAL text,
  HASH int4 NOT NULL,
  FINAL text,
  CATEGORY varchar(32),
  MODIFIED timestamp
);

DELETE FROM PGNP_OPTIMIZER WHERE HASH=0;
INSERT INTO PGNP_OPTIMIZER(HASHTYPE, ENABLED, HASH, CATEGORY, MODIFIED) VALUES (0, 'N', 0,
'System', '2014-01-01 00:00:00');
```

In PGNP Profiler select menu "View"->"Options…", and click ellipsis in the Query Optimizer's Connection string:

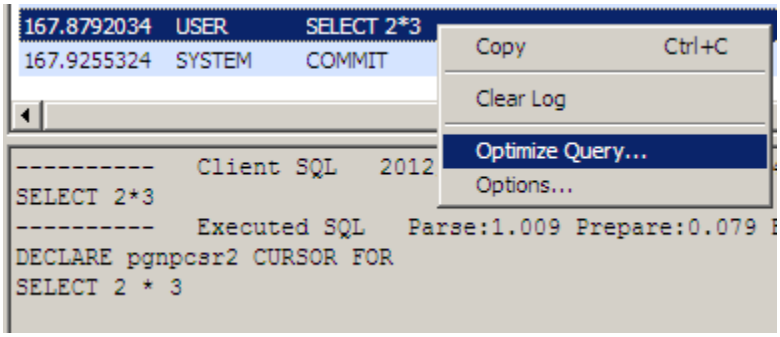


In the Data Link Properties dialog, configure the connection to a database where queries optimization should be performed. Click OK in the Options dialog, and the pgnp_optimizer table will be created automatically. To disable Query Optimizer uncheck the “Enable” option in the dialog.

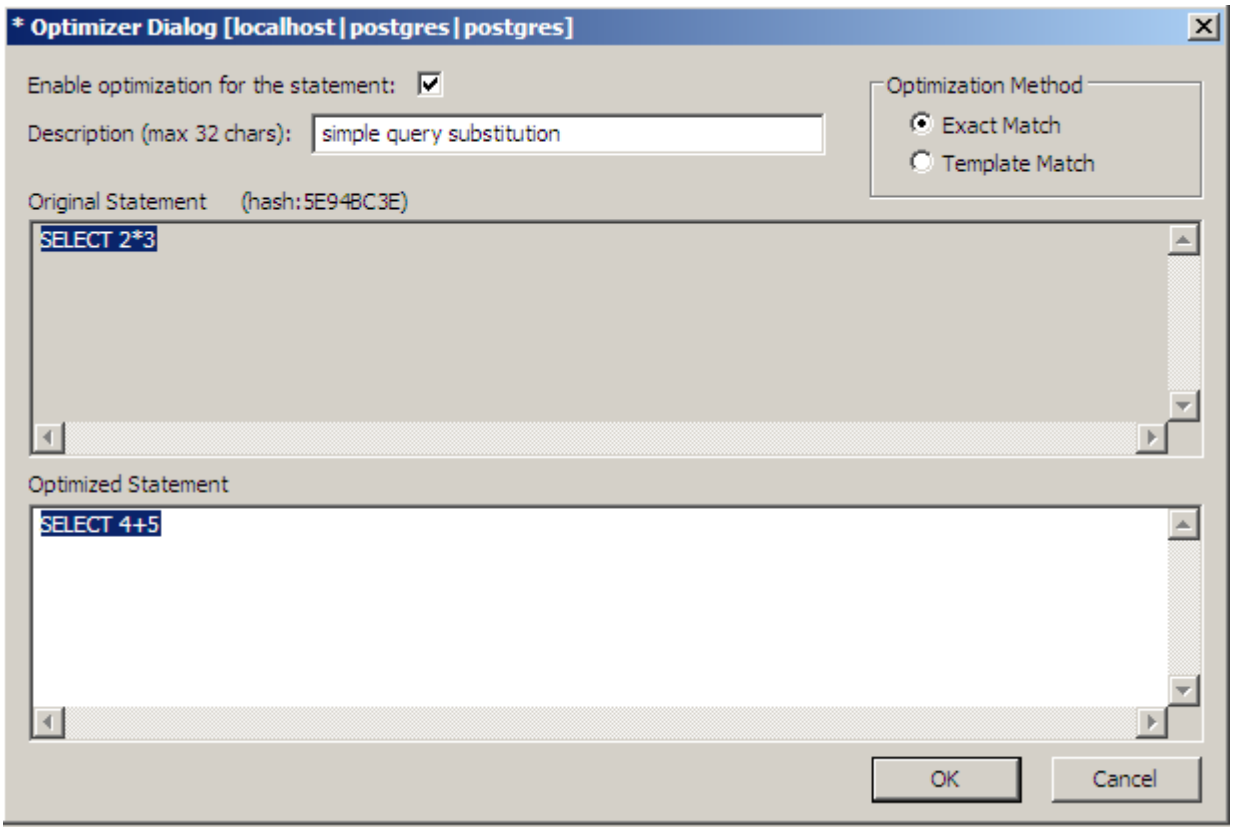
This chapter describes examples of using the Query Optimizer.

3.9.1 Simple query substitution

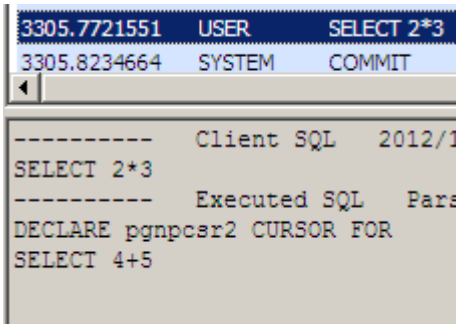
Execute the following query “SELECT 2*3”, locate corresponding trace entry in the PGNP Profiler, click right mouse button on the entry, and select “Optimize Query…” menu item:



In the Optimizer dialog replace the original statement with “SELECT 4+5”, keep “Exact Match” method, and optionally enter any description as shown below:



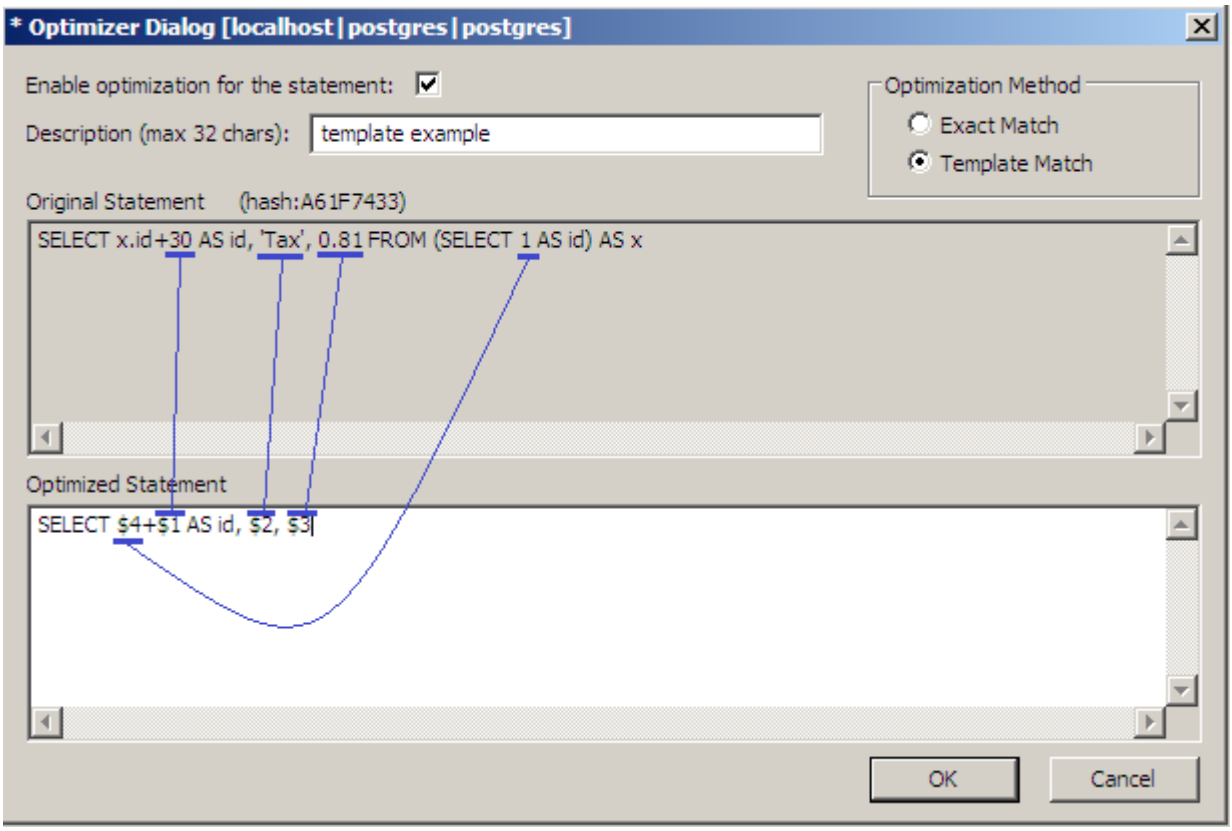
Click OK, execute the original statement again (SELECT 2*3), and locate new trace entry in the PGNP Profiler. Notice, that result is now equal to 9, and the Profiler shows how the statement was substituted:



3.9.2 Template based substitution

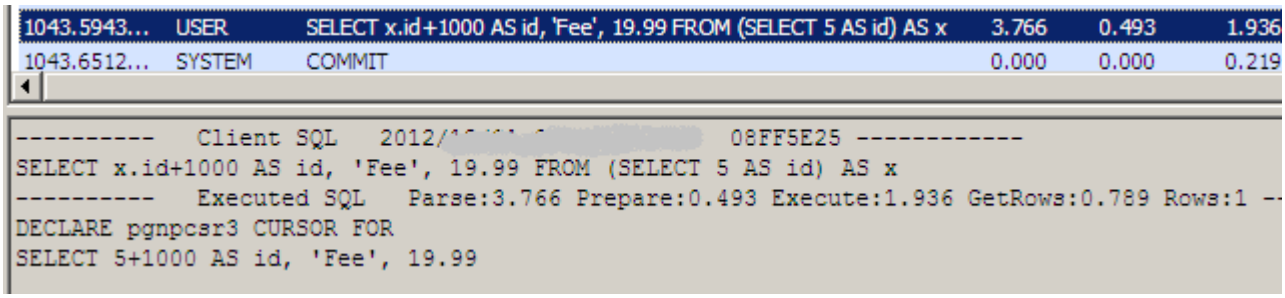
Execute the following query “SELECT id+30 AS id, 'Tax', 0.81 FROM (SELECT 1 AS id) AS x”, locate corresponding trace entry in the PGNP Profiler, right click mouse on the entry, and select “Optimize Query…” menu item.

In the Optimizer dialog replace original statement with “SELECT $4+$1 AS id, $2, $3”, set “Template Match” method, and optionally enter some description as shown below:



In the Template Match mode integer, floating point and string constants from original query will be passed "into parameters" of the template.

Execute original statement, or a statement with different constants, and locate new trace entry in the PGNP Profiler. The Profiler shows how the statement was substituted:

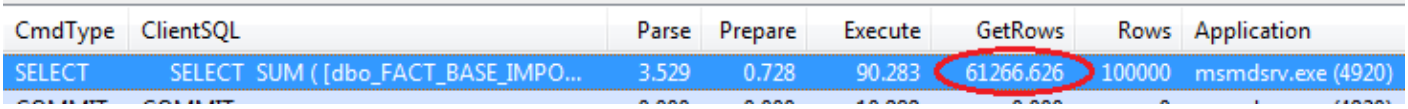


As shown on the picture above, a query with different constants was executed: "SELECT x.id+1000 AS id, 'Fee', 19.99 FROM (SELECT 5 AS id) AS x".

3.9.3 Exact Match scenario: optimizing ROLAP cube

When developing and running a SQL Server Analysis Services (SSAS) project we do not have direct control over the generated statements. However, some of the statements may not be using all power of a Postgres/Greenplum database. The performance issues can be addressed by optimizing (substituting) the statements in the OLEDB Provider.

Run the ROLAP cube and locate slowest query in the PGNP Profiler's trace:



Copy the Client SQL from the “details view”. Here is a simplified representation of the slow query:

```
SELECT SUM(...), ...
FROM(
    SELECT ...
    FROM FACT
    LEFT OUTER JOIN CONCEPT ...) AS dbo_FACT,
  CONCEPT AS dbo_CONCEPT
WHERE
  (dbo_FACT.pk=dbo_CONCEPT.pk)
  AND
  (dbo_FACT.key1=?)
GROUP BY ...
```

Create materialized view as shown below. Notice, the parameterized conditions (marked with yellow background) were removed from the query.

```
CREATE TABLE view_FACT AS
SELECT SUM(...), ...
FROM(
    SELECT ...
    FROM FACT
    LEFT OUTER JOIN CONCEPT ...) AS dbo_FACT,
  CONCEPT AS dbo_CONCEPT
WHERE
  (dbo_FACT.pk=dbo_CONCEPT.pk)
GROUP BY ...
DISTRIBUTED BY (key1)
```

Right click mouse on the slow query, and select “Optimize Query…” menu item.

In the Optimizer dialog replace Optimized Statement with:

```
SELECT * FROM view_FACT
WHERE (key1=$1)
```

Keep the “Exact Match” method; click OK in the Optimizer dialog. Then run the cube again, locate the corresponding trace entry in the Profiler, and see that execution is now significantly faster. In our test with 400 million rows fact table, acceleration was more than 100.

3.9.4 Optimizing metadata retrieval

It is possible to optimize not only client queries but the provider generated metadata retrieval queries as well. This could be used to accelerate metadata retrieval in very complex databases containing hundreds of schemas with thousands tables, where pg_catalog size exceeds 10GB.

Run application, and locate slow schema query in PGNP Profiler’s trace:

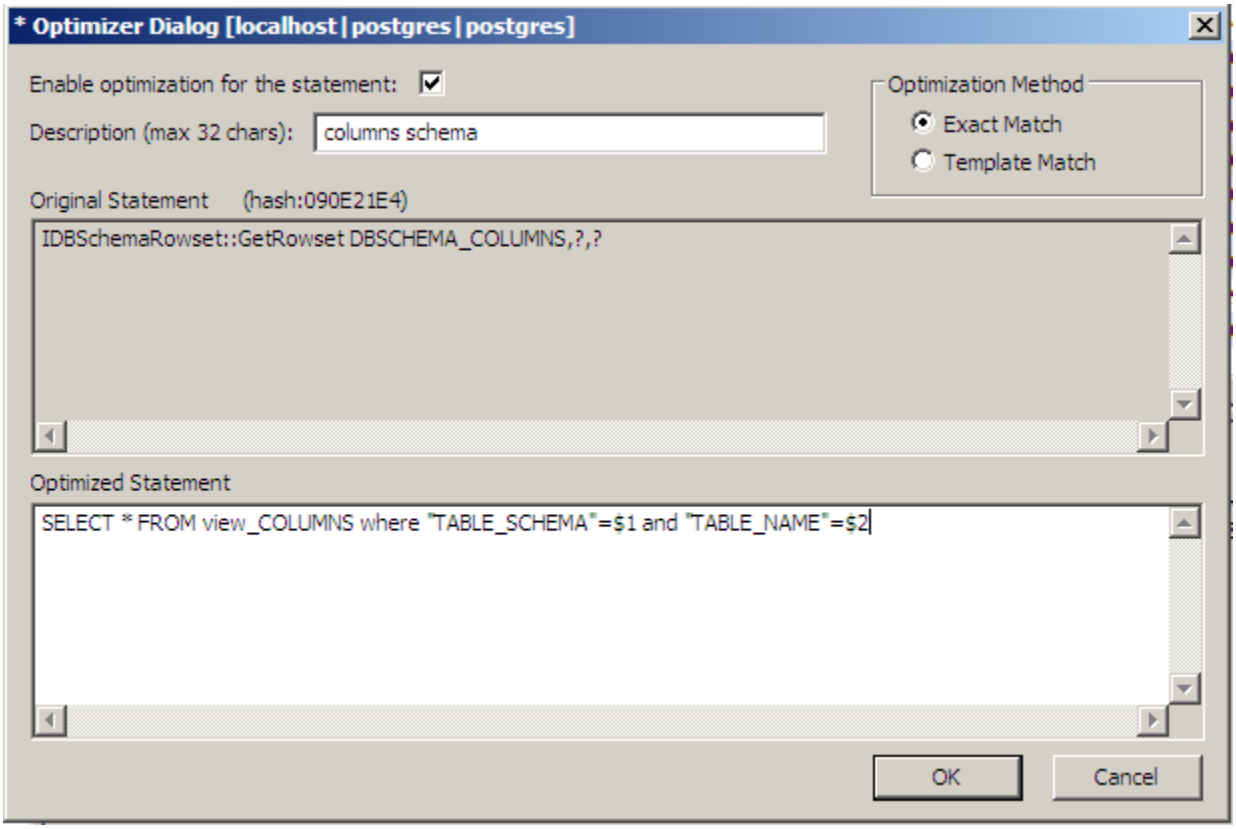
SQLType	ClientSQL	Parse	Prepare	Execute
SYS_SCHEMA	IDBSchemaRowset::GetRowset(DBSCHEMA_COLUMNS)	0.000	0.000	18467.978

Copy “Executed SQL” from the Details view, and create a materialized view as shown below:

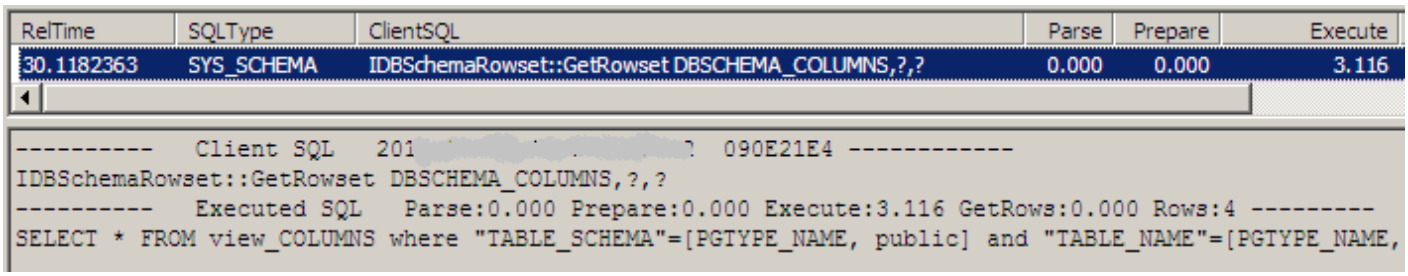
```
CREATE TABLE view_COLUMNS AS
select * from (select T.schemaname as "TABLE_SCHEMA", T.tablename as "TABLE_NAME",
A.attname as "COLUMN_NAME", A.attnum as "ORDINAL_POSITION", A.atthasdef as
"COLUMN_HASDEFAULT", A.atttypid as "DATA_TYPE", TY.typname as "TYPNAME", A.attnotnull as
"NOT_NULL", A.attlen as "FIELD_LENGTH", A.atttypmod as "FIELD_MOD", D.adsrc as
```

```
"COLUMN_DEFAULT" from pg_attribute A inner join pg_class C on (A.attrelid=C.oid) inner
join pg_tables T on (C.relname=T.tablename) inner join pg_namespace NS on
(NS.oid=C.relnamespace and NS.nspname=T.schemaname) inner join pg_type TY on
(TY.oid=A.atttypid) left outer join pg_attrdef D on (D.adrelid=C.oid and
D.adnum=A.attnum) where A.attnum>0 and A.attisdropped='f'  union all select T.schemaname
as "TABLE_SCHEMA", T.viewname as "TABLE_NAME", A.attname as "COLUMN_NAME", A.attnum as
"ORDINAL_POSITION", A.atthasdef as "COLUMN_HASDEFAULT", A.atttypid as "DATA_TYPE",
TY.typname as "TYPNAME", A.attnotnull as "NOT_NULL", A.attlen as "FIELD_LENGTH",
A.atttypmod as "FIELD_MOD", D.adsrc as "COLUMN_DEFAULT" from pg_attribute A inner join
pg_class C on (A.attrelid=C.oid) inner join pg_views T on (C.relname=T.viewname) inner
join pg_namespace NS on (NS.oid=C.relnamespace and NS.nspname=T.schemaname) inner join
pg_type TY on (TY.oid=A.atttypid) left outer join pg_attrdef D on (D.adrelid=C.oid and
D.adnum=A.attnum) where A.attnum>0 and A.attisdropped='f') s where "TABLE_SCHEMA"=[…] and
"TABLE_NAME"=[…] order by "TABLE_SCHEMA", "TABLE_NAME", "ORDINAL_POSITION"
```

Note: conditions marked with red color should be removed. Right click mouse on the entry, select “Optimize Query…” menu item, and replace statement as shown below:



Re-run the application, and locate the schema query. Here is result of the statement substitution:



3.10 "Hinting" statements

The Microsoft SQL Server provides both short-characters ("text") and wide-characters ("ntext") column types, while both Postgres and Greenplum – only short-character ("text"). Microsoft left it to a higher level application to recognize the type of a column, and pass either single-character or wide-character streams in ISequentialStream. However, the stream itself does not specify what types of characters are in it. By historical reason, the PGNP provider handles stream data as short-characters. This may result in issues when only first character is copied, etc.

To address the issue of characters streams handling, the PGNP OLE DB provider memorizes table creation statements, and uses the knowledge for interpreting ISequentialStream. Below we describe two real-life scenarios.

3.10.1 Copying table from SQL Server to Postgres in DTSWizard

Let us consider example of copying a table with "text", "varchar", "ntext" and "nvarchar" columns from SQL Server to Postgres. Create source table in SQL Server using scripts below:

```
CREATE TABLE TestHint1
(
id            int IDENTITY(1,1) NOT NULL PRIMARY KEY,
col1   varchar(100),
col2   text,
col3   nvarchar(100),
col4   ntext,
)
GO

INSERT INTO TestHint1(col1,col2,col3,col4)
VALUES ('val1_0','val2_0','val3_0','val4_0'), ('val1_1','val2_1','val3_1','val4_1')
GO
```

Then launch DTSWizard, for the source choose SQL Native Client, and select TestHint1 table, for destination choose a Postgres/Greenplum database, and optionally configure Copy parameters. Click button "Edit Mappings…" and the "Edit SQL…". Leave the SQL text as is:

```
CREATE TABLE "TestHint1" (
"id" int NOT NULL,
"col1" varchar(100),
"col2" text,
"col3" nvarchar(100),
"col4" ntext
)
```

Note, we left to the PGNP OLEDB provider handling "nvarchar" and "ntext" columns types. The OLEDB Provider will replace the column types with "varchar" and "text", but it will remember that they were intended to have wide-characters.

When the test package executed, the PGNP Profiler shows following:

```
----------   Executed SQL   Parse:0.168 Prepare:0.004 Execute:0.018 GetRows:0.000 Rows:-1 ------
---
CREATE TABLE "TestHint1"
("id" int NOT NULL,
"col1" varchar(100),
"col2" text,
"col3" varchar(200),
"col4" text)
```

Check result, and see that all values were copied correctly:

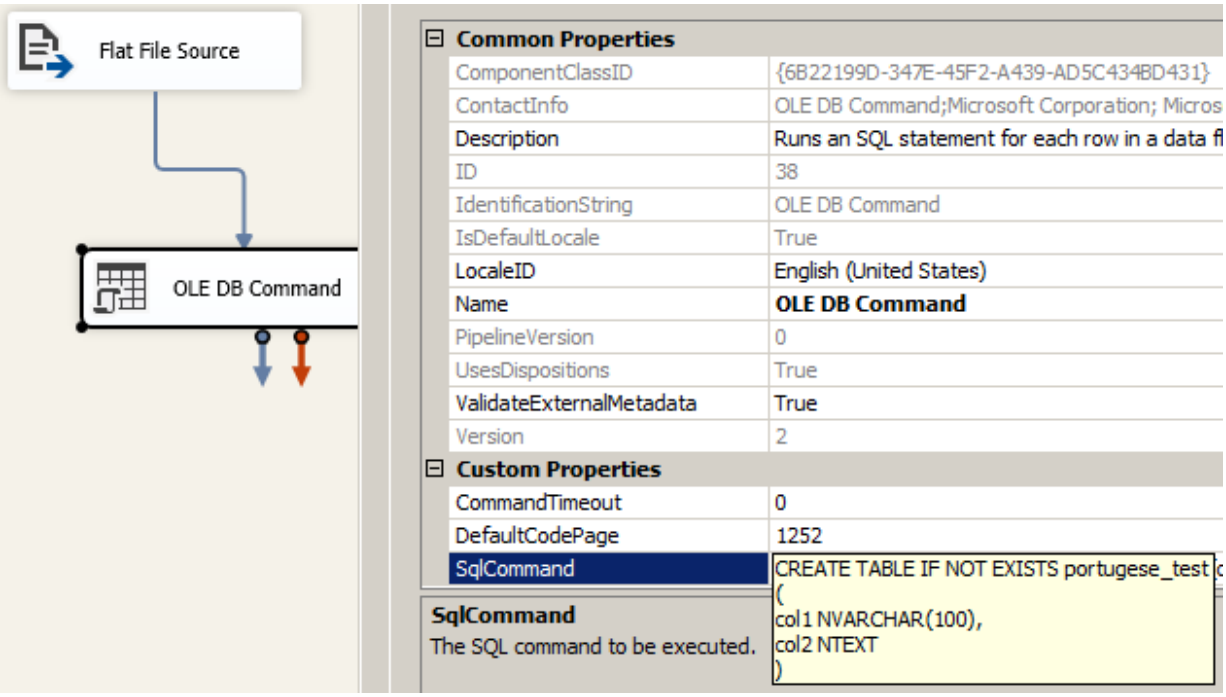
id integer	col1 character varying(100)	col2 text	col3 character varying(200)	col4 text
1	val1_0	val2_0	val3_0	val4_0
2	val1_1	val2_1	val3_1	val4_1

3.10.2 Tweaking Data Flow in SSIS Package

Let us consider another example: copying flat file with Unicode text to Postgres in SSIS. Create UTF-16 text file containing any text, for example – with Portugese special characters:

ãáàâçéê€,íõóôúü«»
ÃÁÀÂÇÉÊ€,ÍÕÓÔÚÜ

In the SSIS package add new Flat File Connection, point it to the text file, uncheck “Column names in the first data row”. Add Flat File Source to the Data Flow, then add OLE DB Command as shown below:

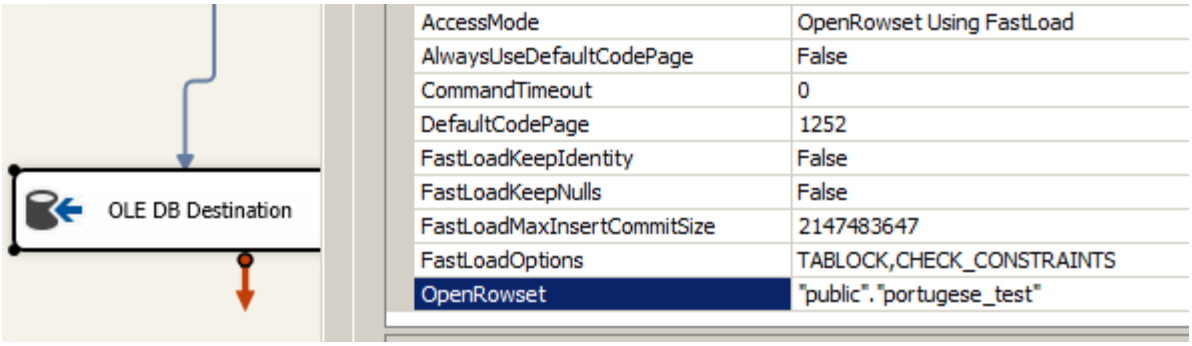


The OLE DB Command is used for hinting the PGNP Provider that columns will receive wide-characters from SSIS. The clause IF NOT EXISTS can also be used with Greenplum DB. The PGNP Provider for Greenplum will remove the clause, but if error “table already exists” is returned, it will convert it into success code.

Create destination table in PGAdminIII:

```
CREATE TABLE IF NOT EXISTS portugese_test
(
col1 VARCHAR(100),
col2 TEXT
)
```

Add OLE DB destination (with optional FastLoad), select destination table portugese_test, and map columns:



Execute package and check the resulting data.

Note, in the PGNP Profiler trace the “hinting statement”:

```
CREATE TABLE IF NOT EXISTS portugese_test
(
col1 NVARCHAR(100),
col2 NTEXT
)
```

Will be executed as:

```
----------  Executed SQL  Parse:0.000 Prepare:0.000 Execute:0.184 GetRows:0.000 Rows:0 ---------
PREPARE "ddb10300a206c586b4fb5bb074951761" AS CREATE TABLE portugese_test
(col1 varchar(200),
col2 text)
----------  Executed SQL  Parse:0.000 Prepare:0.000 Execute:0.378 GetRows:0.000 Rows:0 ---------
EXECUTE "ddb10300a206c586b4fb5bb074951761"
```

3.10.3 Using comments to change Extended Properties parameters per statement

Some Extended Properties parameters can be changed per SQL statement by a comment at the beginning of a statement.

Example 1. Cursor could be disabled by inserting following (highlighted) comment line:

```
--PGNP:CURSOR=OFF;
SELECT * FROM TABLE1
```

Example 2. Cursor could be enabled and minimum fetch rows set to a new value:

```
--PGNP:CURSOR=ON;MIN_FETCH_ROWS=5000;
SELECT * FROM TABLE2
```

Here are properties that can be changed per statement in builds 1.4.0.3606 and later: CURSOR, MIN_FETCH_ROWS, COMMAND_TIMEOUT, NUM_PRECISION, NUM_SCALE, ZERO_TS_FRACTION.

4 Programming with the Provider

4.1 Connection String

PGNP connection string consists of a list of `name=value` pairs separated by semicolon, e.g.:

```
Provider=PGNP.1;User ID=postdba;Password=montreal;
Initial Catalog=aloha;Data Source=bbox;
Extended Properties="PORT=5432;COMMAND_TIMEOUT=900;";
```

4.1.1 Main String parameters

Parameter	Type	Version	Required	Default	Description
Provider	String	All	Yes		Provider name. Always equals to PGNP.1
User ID	String	All	Yes		Database user name
Password	String	All	Yes		Database password
Initial Catalog	String	All	Yes		Database name*
Data Source	String	All	Yes		PostgreSQL server host name or IP address
Connect Timeout	Integer	1.3.0	No	15	Connect timeout in seconds. Zero is used for indefinite timeout.
Extended Properties	String	All	No		Extended parameters list in double quotes. See details below.
Application Name	String	1.4.0.3420	No		Passes application name to the underlying DBMS.

* Note: A special reserved value "$NO_CATALOG" can be used for the Initial Catalog. It allows creating OLEDB session without connecting to a database. Only two stored procedures are working in the NO_CATALOG mode: pgnp_getlicenseinfo and pgnp_checklicense.

4.1.2 Extended Properties

Parameter	Type	Version	Default	Description
Port	Integer	All	5432	PostgreSQL server port.
CURSOR	Boolean	1.4.0.3170	ON	When set to ON the provider uses cursors to query and change data. Main advantage of using cursors is lower memory consumption on client side. However, since cursors work only inside transaction (that can be automatically created by the provider), it may result in undesirable side-effects. Other supported values: OFF and SINGLEROW. When OFF entire rowset is loaded into memory. The SINGLEROW allows processing of unlimited size rowsets (for ETL and Analytics), but does not support scroll-back and fetch-back modes.
LowerCaseSchema	Boolean	All	OFF	If ON, automatically convert all schema into lower case, e.g. Create table "MiXed" () will be transformed into Create table "mixed" () This parameter is used mostly during Database Transformation of a case insensitive database, such as MS SQL, into PostgreSQL.
SYNTAX	String	1.3.0	Pass-Through	Reserved. Specifies which SQL syntax the Provider should expect as input: Pass-Through – for PostgreSQL compatible syntax; T-SQL – MS SQL Server 2005 compatible syntax; or Oracle – Oracle 10g2 compatible syntax. The Provider converts input statements into PostgreSQL

				compatible statements.
SEARCH_PATH	String	1.2.8.1107 and later	$user,public	"Specifies the order in which schemas are searched when an object (table, data type, function, etc.) is referenced by a simple name with no schema component." See description of search_path parameter in http://www.postgresql.org/docs/8.3/interactive/runtime-config-client.html for more details. Example of when the use of this parameters can be useful is a DTS package.
FORCECP	String or Integer	1.4.0.3264		The code page identifier is used for the conversion of Postgres data into Windows Unicode (UTF-16). One of the predefined strings (see below), or and integer can be used. By default, the provider uses "client_encoding" parameter of the Postgres database. In some case using this parameter can fix conversion issues. For example, when SQL_ASCII database stores characters with values larger than 127, the parameter can be set as WIN1252 for correct conversion. List of supported CP names: BIG5, EUC_CN, EUC_JIS_2004, EUC_JP, EUC_KR, EUC_TW, GB18030, GBK, ISO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, JOHAB, KOI8, LATIN1, LATIN2, LATIN3, LATIN4, LATIN5, LATIN6, LATIN7, LATIN8, LATIN9, LATIN10, SHIFT_JIS_2004, SJIS, SQL_ASCII, UHC, UTF8, WIN1250, WIN1251, WIN, WIN1252, WIN1253, WIN1254, WIN1255, WIN1256, WIN1257, WIN1258, TCVN, WIN866, ALT, WIN874. An integer value can be used for the CP, e.g. 1252 instead of WIN1252.
CNV_SPECIAL_FLTVAL	Boolean	1.3.0	OFF	When OFF – no conversion of special floating point values (INFINITY, NaN, etc.) is performed when reading Real and Float fields from database. When ON – the following conversion is performed: * 'Infinity' is converted into MAX float 3.402823466e+38F or double 1.7976931348623158e+308; * '-Infinity' is converted into negative max float or double; * NaN is converted into zero. We recommend turning this option ON when working with Linked servers.
NESTED_TRANS	Boolean	---	OFF	When OFF – nested transactions are not allowed. When ON – allow nested transactions. This was implemented by request of some customers in 1.2.8 and currently disabled in all versions. In 1.3.x nested transactions are allowed by default.
SSL	String	1.3.0	Prefer	SSL parameters. Allowed values: Require, Prefer, Disable, or Allow.
SSLCERT	String	1.4.0.3612	%APPDATA% \postgresql \postgresql.crt	File name of the client SSL certificate.
SSLKEY	String	1.4.0.3612	%APPDATA% \postgresql \postgresql.key	Secret key used for the client certificate.

SSLROOTCERT	String	1.4.0.3612	%APPDATA% \postgresql \root.crt	File containing SSL certificate authority (CA) certificate(s).
SSLCRL	String	1.4.0.3612	%APPDATA% \postgresql \root.crl	File name of the SSL certificate revocation list.
BULK_INSERT	Integer	1.2.8.1110	1	Specifies the number of rows for bulk insert (e.g. during DTS/SSIS data import). Valid range is 1..1000000. See BULK_METHOD below for details on how this parameter is used. Normally the larger the number the less roundtrips performed for data insertion.
BULK_METHOD	String	1.3.0	Values	Bulk insert method: COPY, PIPECOPY, GPLOAD and VALUES. For more details read a paragraph below.
COMMAND_TIMEOUT	Integer	1.3.0	30	Number of seconds for command timeout. If a command execution takes more than the specified duration, then the command is automatically cancelled and error is returned. Please specify 0 (zero) for the infinite wait.
TEXT_AS_LONGVARCHAR	Boolean	1.3.0	ON	If this parameter is set to OFF then Provider returns Text BLOBs as Strings. This parameter can be used when migrating PostgreSQL ODBC applications to PGNP.
ACCEPT_CERT	String	1.3.0	<none>	MD5 hexadecimal sum of the PostgreSQL certificate. This parameter can be used to avoid server certificate validation dialog and basically tells Provider to accept the specific certificate temporarily for the current session.
ZERO_TS_FRACTION	Boolean	1.3.0	OFF	When the parameter is set to ON, the Provider zeroes timestamp fraction in rowset fields. This addresses Datetime overflow issue when transforming databases in SSIS.
USE_TIME2	Boolean	1.4.0.3330	ON	When the parameter is set to OFF, the provider returns time and timetz fields types as legacy ADO time (DBTYPE_DBTIME). Otherwise, the provider handles the fields as time with microseconds precision (DBTYPE_DBTIME2).
SQLSERVER	Integer	1.3.0	2005	Setting this parameter to 2005, 2008 or 2010 exposes provider's type system differently to support database transformations between PostgreSQL and corresponding versions of SQL Server. For example, when set to 2005 PostgreSQL types **date** and **time** are handled as SQL Server's **datetime**; when 2008 – support for new SQL Server types **datetime2**, **time2**, **interval, datetimeoffset** is added.
MIN_FETCH_ROWS	Integer	1.3.0	100	Specifies how many rows should be fetched in PostgreSQL Server Side cursor. Valid range is 1..1000000. If an invalid value is specified, then default value is used. It is recommended to use value of 2000 or larger for tables with millions of rows (to reduce number of round-trips for fetching the data).
HTTP_PORT	Integer	Greenplum only	8081	The HTTP port on which gpfdist.exe will serve files.
NIC_ADDR	String	Greenplum only	<none>	List of IP addresses if the computer has multiple NICs, e.g. NIC_ADDR=10.20.30.40,10.20.30.41,10.20.30.42

DISTINCT_VALUES	String	Greenplum only	DISTINCT	When non-default value is used (GROUPBY) the provider adds to queries with the DISTINCT clause the following clauses: GROUP BY 1,2. This parameter may be used for performance.
FORCE_SORTED_LOV	String	Greenplum only	NO	When non-default value is used (YES) the provider adds to queries with the DISTINCT clause the following clauses: ORDER BY 1,2. Works only with DISTINCT_VALUES=GROUPBY. This parameter may be used for performance.
EVENTS	String	1.4.0.3132	ALL	When default value is used (ALL), OLEDB Provider subscribes to all events, including metadata and optimizer hints change events. To disable subscribing to any events use NONE. Other values are reserved for future use. Provider triggers events each time metadata or optimizer hints change occurs despite of the parameter.
FAILOVER	String	1.4.0.3594		Comma-separated list of failover servers. Each entry in the list can have colon separated host name or IP address, port, number of retries, and delay in milliseconds. Example: 10.11.12.9:5432:1:500, hostalt2:5444.
AWS_ACCESS_KEY	String	Redshift		Amazon S3 Access Key.
AWS_SECRET_KEY	String	Redshift		Amazon S3 Secret Key.
AWS_BUCKET	String	Redshift		Amazon S3 Bucket Name.
AWS_S3_HEADER	String	Redshift 1.4.0.3456		Amazon S3 PUT header parameters used for FastLoad and BulkInsert. For example, add “AWS_S3_HEADER=x-amz-server-side-encryption:AES256;” to allow SSE-S3 at-rest data encryption. Multiple parameters can be specified separated by comma, and they will be passed on separate lines in the header.
AWS_PROXY_ADDRESS	String	Redshift		Proxy name to access Amazon S3.
AWS_PROXY_USER	String	Redshift		Proxy user to access Amazon S3.
AWS_PROXY_PASSWORD	String	Redshift		Proxy password to access Amazon S3.
OPTIMIZER	Boolean	1.4.0	OFF	Turns the PGNP Optimizer ON or OFF.
REFRESH_META	Integer	1.4.0.3460	0	Specifies number of minutes between automatic metadata cache refresh. Zero (default) disables the auto-refresh. The refresh could be useful for scenarios when database schema changes and no notification is sent (especially for the Redshift). The provider refreshes cache when next access to the cache occurs. PGNProfiler displays the refresh as “pgnp_rtx (internal)” command in the trace.
SET <name>	String	1.4.0.3430		Used for passing SET command(s) that will be executed by the OLEDB provider after connection is open Greenplum example: ***set statement_mem='1999MB';set optimizer=on;*** (enables GP Optimizer)
STREAM	String	1.4.0.3440	SHORT	Specifies how the OLEDB provider should handle ISequentialStream. Default is SHORT, which means that ISequentialStream will contain short-characters (8-bit). When set to WIDE, the provider handles the stream as containing wide-characters (16-bit Unicode).
SQL_PROLOG	String	1.4.0.3456		Path to file with SQL commands that should be executed immediately after connection is open. The prolog file can used for initial configuration of the connection.

TIMEZONE	String	1.4.0.3456	Specifies target time zone in which the TIMESTAMP WITH TIMEZONE (timestamptz) values should be converted. By default, the OLEDB provider converts timestamptz values into the local time zone on the PC. For example, "TIMEZONE=Singapore" will display values in Singapore time. For details refer to Appendix D.

4.1.3 Parameter BULK_METHOD

When set to 'VALUES' (default) then BULK_INSERT specifies number of value sets in the SQL command:

```
INSERT INTO table (col1, col2, … colN) VALUES
   ($1, $2, … $N),
   ($N+1, $N+2, … $2N),
    …
   ($mN+1, $mN+2, … $mN+N)
```

When set to 'COPY' then BULK_INSERT specifies number of rows buffered for COPY command.

When set to 'PIPECOPY', then multiple INSERT statements are combined into COPY command. This is special mode designed for fast transactional replication.

When set to 'GPLOAD' then a Greenplum's loader utility is used (gpload/gpfdist). 'GPLOAD' parameter is used only by the Greenplum edition of the PGNP provider.

BULK_METHOD is an obsolete way of fast importing data into database. We recommend using FastLoad for fast data import. However, PIPECOPY may be only an option when fast transactional replication is needed.

4.1.4 Parameters for Redshift

The FastLoad and BulkLoad algorithms in Redshift OLEDB provider are using the COPY command and Amazon S3. The connection string parameters (AWS_*) must be populated for the FastLoad to work properly. Proxy usage is optional. Temporary security tokens are currently not supported. Please refer Amazon online documentation on the description of the parameters: http://docs.aws.amazon.com/redshift/latest/dg/t_loading-tables-from-s3.html.

4.1.5 Deprecated and not supported parameters

Following connection string parameters were removed:

Parameter	Last supported	Replaced with
FORCEUTF8	1.4.0.3254	FORCECP

4.2 Data type mapping between PostgreSQL and OLE DB

The PGNP OLE DB Provider supports the following data type mappings between PostgreSQL data types and OLE DB data types.

Provider Data Type	PostgreSQL Data Type	OLE DB Data Type(s)	OLE DB Type Name(s)
int2 smallint	int2 smallint	DBTYPE_I2	"DBTYPE_I2"

int4 int integer	int4 int integer	DBTYPE_I4	“DBTYPE_I4”
float4 real float(p), p<25	float4 real float(p), p<25	DBTYPE_R4	“DBTYPE_R4”
float8 double precision float(p), p>=25	float8 double precision float(p), p>=25	DBTYPE_R8	“DBTYPE_R8”
money	money	DBTYPE_CY	“DBTYPE_CY”
bool boolean bit	bool boolean	DBTYPE_BOOL	“DBTYPE_BOOL”
tinyint byte	int2	DBTYPE_UI1	“DBTYPE_UI1”
int8 bigint	int8 bigint	DBTYPE_I8	“DBTYPE_I8”
uuid	uuid	DBTYPE_GUID	“DBTYPE_GUID”
binary varbinary image	bytea	DBTYPE_BYTES	"DBTYPE_BINARY" "DBTYPE_VARBINARY" "DBTYPE_LONGVARBINARY" and DBCOLUMNFLAGS_ISLONG or DBPARAMFLAGS_ISLONG
char	char	DBTYPE_STR	“DBTYPE_CHAR”
varchar	varchar	DBTYPE_STR	“DBTYPE_VARCHAR”
text	text	DBTYPE_STR and DBCOLUMNFLAGS_ISLONG or DBPARAMFLAGS_ISLONG	“DBTYPE_CHAR” “DBTYPE_VARCHAR” “DBTYPE_LONGVARCHAR” and DBCOLUMNFLAGS_ISLONG or DBPARAMFLAGS_ISLONG
ntext	text	DBTYPE_WSTR and DBCOLUMNFLAGS_ISLONG or DBPARAMFLAGS_ISLONG	“DBTYPE_CHAR” “DBTYPE_VARCHAR” “DBTYPE_LONGVARCHAR” and DBCOLUMNFLAGS_ISLONG or DBPARAMFLAGS_ISLONG
nvarchar	varchar	DBTYPE_WSTR	“DBTYPE_VARCHAR”
nchar	char	DBTYPE_WSTR	“DBTYPE_CHAR”
numeric decimal	numeric decimal	DBTYPE_NUMERIC	“DBTYPE_NUMERIC”
timestamp	timestamp	DBTYPE_DBTIMESTAMP	“DBTYPE_DBTIMESTAMP”
xml	xml	DBTYPE_XML	“DBTYPE_XML”

“Provider Data Type” column specifies data types that PGNP provider exposes to a user. These types are recognized in CREATE and ALTER statements and substituted “on-the-fly” with corresponding PostgreSQL Data Type. These types are used by MS Data Transformation Services Wizard database conversion.

4.3 Internal Stored Procedures

The purpose of the provider’s internal stored procedures is to supply user application with additional services. Current implementation includes procedures: to obtain the provider license information, to refresh metadata cache, check license, publish comments into PGNP log, etc. In future more stored procedures are going to be written and even support for custom third-party procedures can be added. Almost all stored procedures work even when no connection is open to database (i.e. Initial Catalog=$NO_CATALOG).

4.3.1 Get License Information

The procedure pgnp_getlicenseinfo() can be used to obtain information about current provider owner and license type. It can be called as any PostgreSQL procedure (see C# sample 16):

```
pgnp_getlicenseinfo()
```

The resulting recordset has the following columns:

Ordinal	Column name	Type	Description
1	LICENSEDTO	String	Shows user name or company who purchased the product
2	TYPE	String	License type, i.e. Single User License or a License Pack
3	PURCHASEDATE	Date	Date of the product purchase
4	VERSION	String	Version of the product, e.g. 1.2.8
5	UPDATEDATE	Date	If this is an update of the product, then shows date, otherwise is NULL
6	SPECIAL	String	Unique identifier that uniquely identifies the product build

4.3.2 Refresh Metadata Cache

The procedure pgnp_refreshmetadata(<reserved>, <name>) can be used to request refreshing of internal metadata cache in all provider instances connected to the same database from any computer. This could be useful when database schema has changed (a table or a procedure were altered). The procedure has two input string parameters. First parameter is reserved for the future and should always be specified as empty string. Second parameter can contain a table or a procedure name. Here are examples of valid calls:

```
pgnp_refreshmetadata(‘’, ‘employee’)        -- refresh info about employee table in the cache

pgnp_refreshmetadata(‘’, ‘myproc1’)         -- refresh info about myproc1 procedure in the cache
```

The procedure does not return any data.

4.3.3 Check license

The procedure pngp_checklicense() is similar to pgnp_getlicenseinfo(), and can be used to perform simple check if the license is valid (for diagnostics purposes). Both procedures can operate in a special mode when there is no connection to database made. Initial Catalog parameter of the connection string can be set to the reserved value “$NO_CATALOG” to open a session without connecting to a database. Pgnp_checklicense() returns a row with two columns: integer error code, and description. Zero in first column means success, and description contains either Success, or Error description.

4.3.4 Publish comment into PGNP Profiler log

The procedure pgnp_comment can be used by applications to insert comments into PGNP Profiler's log. Also, PGNP Provider can be configured to populate the internal execution trace into the Profiler's log. The procedure accepts two parameters: integer "level", and string "message". Level specifies how important is the comment (1-most important, 2-medium, 3-least important). PGNP Profiler can be configured to listen to "no comments", or comments with levels 1, 2 or 3 (all comments). The procedure returns not result, and there is no way to know if it succeeded. If PGNP Profiler is not running, or configured to listen for "SQL Statements Only", the procedure does nothing. Here is example of calling the function via linked server in SSMS:

```
EXEC ('pgnp_comment(1, ''test message 123'')') AT PGSVR
```

5 Appendix A. Utilities

5.1 CreateIndex

Purpose of the CopyIndex utility is copying of indexes from source DB to a destination DB. Common use case: it can be used after DTS/SSIS wizard or program package copied data between databases to append information about indexes.

USAGE:

```
   CopyIndex.exe  -s <.udl File> -d <.udl File>  -a <string> [-i <string>] [-t <string>] [--] [-v] [-h]

   -s <.udl File>,  --srcudl <.udl File>
     (required)  (value required)  A source database .udl file.

   -d <.udl File>,  --dstudl <.udl File>
     (required)  (value required)  A destination database .udl file.

  -a <string>,  --schema <string>
     (required)  (value required)  Schema restriction.

   -i <string>,  --index <string>
     (value required)  Optional index name restriction.

   -t <string>,  --table <string>
     (value required)  Optional table name restriction.

   --,  --ignore_rest
     Ignores the rest of the labeled arguments following this flag.

   -v,  --version

   -h,  --help
     Displays usage information and exits.
```

Example 1. Copy all indexes from MSDB (MSSQL, dbo) to PGDB (Postgres, dbo).

```
  CopyIndex.exe -s MSDB.udl -d PGDB.udl -a dbo
```

Example 2. Copy all indexes related to Employees table. Employees table should exist in source and destination DBs.

```
  CopyIndex.exe -s MSDB.udl -d PGDB.udl -a dbo -t Employees
```

Example 3. Copy PK_Employees indexe in Employees table. Employees table should exist in source and destination DBs.

```
  CopyIndex.exe -s MSDB.udl -d PGDB.udl -a dbo -t Employees -i PK_Employees
```

See CreateIndex.cpp source code below:

```
// CopyIndex.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include "IndexDesc.h"
#include "VersionInfo.h"
#include <tclap/CmdLine.h>
using namespace TCLAP;

typedef map<wstring, CIndexDesc> MapIndexes;
```

```
typedef pair<wstring, CIndexDesc> PairIndexes;

void DisplayErrorInfo()
{
	CDBErrorInfo errors;

	ULONG errorCount;
	HRESULT hr = errors.GetErrorRecords(&errorCount);
	if (FAILED(hr))
	{
		cerr << "** ERROR: hr = " << hex << hr << endl;
		return;
	}

	for (ULONG i = 0; i < errorCount; i++)
	{
		_bstr_t description;
		errors.GetAllErrorInfo(i, GetSystemDefaultLCID(), description.GetAddress());

		cerr << "** " << description << endl;
	}
}

#define SET_ORDINAL(Rowset, ColName, Ordinal)                       \
	/*const*/ DBORDINAL Ordinal;                                    \
	if (!Rowset.GetOrdinal(ColName, &Ordinal)) {        \
		_ftprintf(stderr, _T("** ERROR: Column %s ordinal not found in INDEXES schema\n"), ColName);  \
		return E_FAIL;            \
	}


int _tmain(int argc, _TCHAR* argv[])
{
	HRESULT hr = CoInitialize(NULL);
	if (FAILED(hr))
	{
		cerr << "** ERROR: CoInitialize returned " << hex << hr << endl;
		return hr;
	}

	TCHAR filename[MAX_PATH];
	GetModuleFileName(NULL, filename, MAX_PATH);

	VersionInfo verInfo(filename);

	_bstr_t srcUdl, dstUdl, bstrSchema, bstrIndex, bstrTable;

	try
	{
		CmdLine cmdline(verInfo.FileDescription(), ' ', verInfo.productVersion_str());

		ValueArg<_tstring> tableArg(_T("t"), _T("table"), _T("Optional table name restriction."), false, _T(""),
_T("string"), cmdline);
		ValueArg<_tstring> indexArg(_T("i"), _T("index"), _T("Optional index name restriction."), false, _T(""),
_T("string"), cmdline);
		ValueArg<_tstring> schemaArg(_T("a"), _T("schema"), _T("Schema restriction."), true, _T(""), _T("string"),
cmdline);
		ValueArg<_tstring> dudlFileArg(_T("d"), _T("dstudl"), _T("A destination database .udl file."), true, _T(""),
_T(".udl File"), cmdline);
		ValueArg<_tstring> sudlFileArg(_T("s"), _T("srcudl"), _T("A source database .udl file."), true, _T(""),
_T(".udl File"), cmdline);

		cmdline.parse(argc, argv);

		srcUdl = sudlFileArg.getValue().c_str();
		dstUdl = dudlFileArg.getValue().c_str();
		bstrSchema = schemaArg.getValue().c_str();
		bstrIndex = indexArg.getValue().c_str();
		bstrTable = tableArg.getValue().c_str();
	}
	catch (ArgException &e)
	{
		cerr << endl << "** ERROR: " << "Invalid Arguments: " << endl << e.error().c_str() << endl;
		return 1;
	}

	CDataSource                        srcDataSource, dstDataSource;
	CSession                 srcSession, dstSession;

	// STEP 1. Load DBSCHEMA_INDEXES from source Provider.
	cout << "Loading DBSCHEMA_INDEXES from source database..." << endl;

	hr = srcDataSource.OpenFromFileName(srcUdl);                //OpenFromInitializationString(srcConnString);
	if (FAILED(hr))
	{
		DisplayErrorInfo();
		return hr;
	}
```

```
        hr = srcSession.Open(srcDataSource);
        if (FAILED(hr))
        {
                DisplayErrorInfo();
                return hr;
        }

        CComPtr<IDBSchemaRowset> spSchemaRowset;
        hr = srcSession.m_spOpenRowset->QueryInterface(__uuidof(IDBSchemaRowset), (void**)&spSchemaRowset);
        if (FAILED(hr))
        {
                cerr << "** ERROR: QueryInterface(IDBSchemaRowset) returned " << hex << hr << endl;
                return hr;
        }

        VARIANT vNoRestriction;
        vNoRestriction.vt = VT_EMPTY;

        VARIANT vTableSchemaRestriction;
        vTableSchemaRestriction.vt = VT_EMPTY;
        if (bstrSchema.length() > 0)
        {
                vTableSchemaRestriction.vt = VT_BSTR;
                vTableSchemaRestriction.bstrVal = SysAllocString(bstrSchema.GetBSTR());
        }

        VARIANT vIndexNameRestriction;
        vIndexNameRestriction.vt = VT_EMPTY;
        if (bstrIndex.length() > 0)
        {
                vIndexNameRestriction.vt = VT_BSTR;
                vIndexNameRestriction.bstrVal = SysAllocString(bstrIndex.GetBSTR());
        }

        VARIANT vTableNameRestriction;
        vTableNameRestriction.vt = VT_EMPTY;
        if (bstrTable.length() > 0)
        {
                vTableNameRestriction.vt = VT_BSTR;
                vTableNameRestriction.bstrVal = SysAllocString(bstrTable.GetBSTR());
        }

        VARIANT rgRestrictions[] = { vNoRestriction, vTableSchemaRestriction, vIndexNameRestriction, vNoRestriction,
vTableNameRestriction };

        CAccessorRowset<CDynamicAccessor, CBulkRowset> pRS;
        hr = spSchemaRowset->GetRowset(NULL, DBSCHEMA_INDEXES, sizeof(rgRestrictions)/sizeof(rgRestrictions[0]),
rgRestrictions, IID_IRowset, 0, NULL, (IUnknown**)&pRS.m_spRowset);
        if (FAILED(hr))
        {
                DisplayErrorInfo();
                return hr;
        }

        hr = pRS.Bind();

        hr = pRS.MoveFirst();

        SET_ORDINAL(pRS, _T("TABLE_SCHEMA"), iTableSchema);
        SET_ORDINAL(pRS, _T("TABLE_NAME"), iTableName);
        SET_ORDINAL(pRS, _T("INDEX_NAME"), iIndexName);
        SET_ORDINAL(pRS, _T("PRIMARY_KEY"), iPrimaryKey);
        SET_ORDINAL(pRS, _T("UNIQUE"), iUnique);
        SET_ORDINAL(pRS, _T("CLUSTERED"), iClustered);
        SET_ORDINAL(pRS, _T("ORDINAL_POSITION"), iOrderedPosition);
        SET_ORDINAL(pRS, _T("COLUMN_NAME"), iColumnName);
        SET_ORDINAL(pRS, _T("INTEGRATED"), iIntegrated);

        map<wstring, CIndexDesc> mapIndexes;       // map index name into index description

        while (S_OK == hr)
        {
                LPCTSTR tableSchema = (LPCTSTR)pRS.GetValue(iTableSchema);
                LPCTSTR tableName = (LPCTSTR)pRS.GetValue(iTableName);
                LPCTSTR indexName = (LPCTSTR)pRS.GetValue(iIndexName);
                bool primaryKey = *(bool*)pRS.GetValue(iPrimaryKey);
                bool unique = *(bool*)pRS.GetValue(iUnique);
                bool clustered = *(bool*)pRS.GetValue(iClustered);
                LPCTSTR columnName = (LPCTSTR)pRS.GetValue(iColumnName);

                MapIndexes::iterator it = mapIndexes.find(indexName);
                CIndexDesc& index = (it != mapIndexes.end()) ? it->second : mapIndexes.insert(PairIndexes(indexName,
CIndexDesc(tableName, primaryKey, unique, clustered))).first->second;

                index.AddColumn(columnName);

                hr = pRS.MoveNext();
        }
```

```
        pRS.Close();

        cout << "  Indexes loaded count: " << mapIndexes.size() << endl;


        // STEP 2. Access IIndexDefinition in destination Provider.
        cout << "Querying IIndexDefinition from destination Provider..." << endl;

        hr = dstDataSource.OpenFromFileName(dstUdl);                //OpenFromInitializationString(dstConnString);
        if (FAILED(hr))
        {
                DisplayErrorInfo();
                return hr;
        }

        hr = dstSession.Open(dstDataSource);
        if (FAILED(hr))
        {
                DisplayErrorInfo();
                return hr;
        }

        CComPtr<IIndexDefinition> spIndexDefinition;
        hr = dstSession.m_spOpenRowset->QueryInterface(__uuidof(IIndexDefinition), (void**)&spIndexDefinition);
        if (FAILED(hr))
        {
                cerr << "** ERROR: QueryInterface(IIndexDefinition) failed hr=" << hr << endl;
                return hr;
        }

        // STEP 3. Integrity check.

        // STEP 4. Create indexes.
        cout << "Creating indexes..." << endl;
        bool bIgnoreAllErrors = false;
        int nSuccess = 0;

        for (MapIndexes::iterator it = mapIndexes.begin(); it != mapIndexes.end(); it++)
        {
                CIndexDesc& index = it->second;
                vector<wstring>& lColumns = index.GetColumns();

                DBID TableName;
                DBID IndexName;

                TableName.eKind          = DBKIND_NAME;
                TableName.uName.pwszName = (LPOLESTR) index.GetTableName();

                IndexName.eKind          = DBKIND_NAME;
                IndexName.uName.pwszName = (LPOLESTR) it->first.c_str();

                DBPROP              indexdbprop[4];
                DBPROPSET           indexdbpropset[1];
                DBINDEXCOLUMNDESC*  rgIndexColumnDescs = (DBINDEXCOLUMNDESC*)_alloca(lColumns.size() *
sizeof(DBINDEXCOLUMNDESC));
                DBID*               dbidColumns = (DBID*)_alloca(lColumns.size() * sizeof(DBID));

                // Enforce index properties.
                indexdbprop[0].dwPropertyID   = DBPROP_INDEX_NULLS;
                indexdbprop[0].dwOptions      = DBPROPOPTIONS_REQUIRED;
                indexdbprop[0].vValue.vt      = VT_I4;
                indexdbprop[0].vValue.lVal    = DBPROPVAL_IN_DISALLOWNULL;
                indexdbprop[0].colid          = DB_NULLID;

                indexdbprop[1].dwPropertyID   = DBPROP_INDEX_PRIMARYKEY;
                indexdbprop[1].dwOptions      = DBPROPOPTIONS_REQUIRED;
                indexdbprop[1].vValue.vt      = VT_BOOL;
                indexdbprop[1].vValue.lVal    = index.GetPrimary() ? VARIANT_TRUE : VARIANT_FALSE;
                indexdbprop[1].colid          = DB_NULLID;

                indexdbprop[2].dwPropertyID   = DBPROP_INDEX_CLUSTERED;
                indexdbprop[2].dwOptions      = DBPROPOPTIONS_REQUIRED;
                indexdbprop[2].vValue.vt      = VT_BOOL;
                indexdbprop[2].vValue.lVal    = index.GetClustered() ? VARIANT_TRUE : VARIANT_FALSE;
                indexdbprop[2].colid          = DB_NULLID;

                indexdbprop[3].dwPropertyID   = DBPROP_INDEX_UNIQUE;
                indexdbprop[3].dwOptions      = DBPROPOPTIONS_REQUIRED;
                indexdbprop[3].vValue.vt      = VT_BOOL;
                indexdbprop[3].vValue.lVal    = index.GetUnique() ? VARIANT_TRUE : VARIANT_FALSE;
                indexdbprop[3].colid          = DB_NULLID;

                // Initialize the property set.
                indexdbpropset[0].guidPropertySet = DBPROPSET_INDEX;
                indexdbpropset[0].rgProperties    = indexdbprop;
                indexdbpropset[0].cProperties     = sizeof(indexdbprop)/sizeof(indexdbprop[0]);

                // Set up DBINDEXCOLUMNDESC structures to define the columns in the
                // index and the ordering for each column within that index.
```

```
                for (unsigned i=0; i < lColumns.size(); i++)
                {
                        rgIndexColumnDescs[i].eIndexColOrder = DBINDEX_COL_ORDER_ASC;
                        rgIndexColumnDescs[i].pColumnID      = &dbidColumns[i];

                        // Specify the column names for the composite index on
                        // LastName and FirstName.
                        dbidColumns[i].eKind = DBKIND_NAME;
                        dbidColumns[i].uName.pwszName = (LPOLESTR) lColumns[i].c_str();
                }

                wcout << L"  Table: " << TableName.uName.pwszName << L"\t  Index: " << IndexName.uName.pwszName << endl;

                // Create a two-column composite index named full_name_index over the
                // LastName and FirstName columns in the Employees table.
                hr = spIndexDefinition->CreateIndex(&TableName, &IndexName, (DBORDINAL)lColumns.size(),
                        rgIndexColumnDescs, sizeof(indexdbpropset)/sizeof(indexdbpropset[0]), indexdbpropset, NULL);

                if (FAILED(hr))
                {
                        DisplayErrorInfo();
                        if (!bIgnoreAllErrors)
                        {
                                cerr << "Terminate the indexes creation process? Enter 'Y' to terminate, " << endl
                                        << "   'N' to continue, 'I' to ignore all further errors [Y]: ";

                                string sInput;
                                std::getline(cin, sInput, '\n');
                                if (sInput.length() == 0)
                                        break;

                                bIgnoreAllErrors = sInput[0] == 'I' || sInput[0] == 'i';
                                if (!bIgnoreAllErrors && !(sInput[0] == 'N' || sInput[0] == 'n'))
                                        break;
                        }
                }
                else
                        nSuccess++;
        }

        CoUninitialize();

        cout << "Successfully created " << nSuccess << " indexes. Not being able to create: " << mapIndexes.size() - nSuccess
<< "." << endl;

        return 0;
}
```

5.2 DropIndex

Purpose of the DropIndex utility is deleting indexes in DB. Common use case: it can be used to undo changes made by CopyIndex utility.

USAGE:

```
DropIndex.exe  [-c] -u <.udl File> -a <string> [-i] <string>]
               [-t <string>] [--] [-v] [-h]
```

Where:

```
-c,  --confirmation
  Ask confirmation before dropping an index. Recommended.

-u <.udl File>,  --udl <.udl File>
  (required)  (value required)  A database .udl file.

-a <string>,  --schema <string>
  (required)  (value required)  Schema restriction.

-i <string>,  --index <string>
  (value required)  Optional index name restriction.

-t <string>,  --table <string>
  (value required)  Optional table name restriction.

--,  --ignore_rest
  Ignores the rest of the labeled arguments following this flag.

-v,  --version
  Displays version information and exits.

-h,  --help
  Displays usage information and exits.
```

Example 1. Delete all indexes in a PGDB (Postgres, dbo).

```
DropIndex.exe -u PGDB.udl -a dbo
```

Example 2. Delete PK_Employee index in Employees table.

```
DropIndex.exe -u PGDB.udl -a dbo -t Employees -i PK_Employee
```

See DropIndex.cpp source code below.

```
// DropIndex.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include "VersionInfo.h"
#include <tclap/CmdLine.h>
using namespace TCLAP;

void DisplayErrorInfo()
{
        CDBErrorInfo errors;

        ULONG errorCount;
        HRESULT hr = errors.GetErrorRecords(&errorCount);
        if (FAILED(hr))
        {
                cerr << "** ERROR: hr = " << hex << hr << endl;
                return;
```

```
        }

        for (ULONG i = 0; i < errorCount; i++)
        {
                _bstr_t description;
                errors.GetAllErrorInfo(i, GetSystemDefaultLCID(), description.GetAddress());

                cerr << "** " << description << endl;
        }
}

#define SET_ORDINAL(Rowset, ColName, Ordinal)                          \
        /*const*/ DBORDINAL Ordinal;                                   \
        if (!Rowset.GetOrdinal(ColName, &Ordinal)) {        \
                _ftprintf(stderr, _T("** ERROR: Column %s ordinal not found in INDEXES schema\n"), ColName);  \
                return E_FAIL;            \
        }

struct CIndexStruc
{
        CIndexStruc(wstring i_sTable, wstring i_sIndex) :
                sTable(i_sTable), sIndex(i_sIndex)
        {}
        wstring sTable;
        wstring sIndex;
};

int _tmain(int argc, _TCHAR* argv[])
{
        HRESULT hr = CoInitialize(NULL);
        if (FAILED(hr))
        {
                cerr << "** ERROR: CoInitialize returned " << hex << hr << endl;
                return hr;
        }

        TCHAR filename[MAX_PATH];
        GetModuleFileName(NULL, filename, MAX_PATH);

        VersionInfo verInfo(filename);

        _bstr_t aUdl, bstrSchema, bstrIndex, bstrTable;

        try
        {
                CmdLine cmdline(verInfo.FileDescription(), ' ', verInfo.productVersion_str());

                ValueArg<_tstring> tableArg(_T("t"), _T("table"), _T("Optional table name restriction."), false, _T(""),
_T("string"), cmdline);
                ValueArg<_tstring> indexArg(_T("i"), _T("index"), _T("Optional index name restriction."), false, _T(""),
_T("string"), cmdline);
                ValueArg<_tstring> schemaArg(_T("a"), _T("schema"), _T("Schema restriction."), true, _T(""), _T("string"),
cmdline);
                ValueArg<_tstring> udlFileArg(_T("u"), _T("udl"), _T("A database .udl file."), true, _T(""), _T(".udl File"),
cmdline);
                SwitchArg sInteractive(_T("c"), _T("confirmation"), _T("Ask confirmation before dropping an index.
Recommended."), false, cmdline);

                cmdline.parse(argc, argv);

                aUdl = udlFileArg.getValue().c_str();
                bstrSchema = schemaArg.getValue().c_str();
                bstrIndex = indexArg.getValue().c_str();
                bstrTable = tableArg.getValue().c_str();
        }
        catch (ArgException &e)
        {
                cerr << endl << "** ERROR: " << "Invalid Arguments: " << endl << e.error().c_str() << endl;
                return 1;
        }

        CDataSource                     dataSource;
        CSession                session;

        // STEP 1. Load DBSCHEMA_INDEXES from source Provider.
        cout << "Loading DBSCHEMA_INDEXES from database..." << endl;

        hr = dataSource.OpenFromFileName(aUdl);
        if (FAILED(hr))
        {
                DisplayErrorInfo();
                return hr;
        }

        hr = session.Open(dataSource);
        if (FAILED(hr))
        {
                DisplayErrorInfo();
                return hr;
```

```
	}

	CComPtr<IDBSchemaRowset> spSchemaRowset;
	hr = session.m_spOpenRowset->QueryInterface(__uuidof(IDBSchemaRowset), (void**)&spSchemaRowset);
	if (FAILED(hr))
	{
		cerr << "** ERROR: QueryInterface(IDBSchemaRowset) returned " << hex << hr << endl;
		return hr;
	}

	VARIANT vNoRestriction;
	vNoRestriction.vt = VT_EMPTY;

	VARIANT vTableSchemaRestriction;
	vTableSchemaRestriction.vt = VT_EMPTY;
	if (bstrSchema.length() > 0)
	{
		vTableSchemaRestriction.vt = VT_BSTR;
		vTableSchemaRestriction.bstrVal = SysAllocString(bstrSchema.GetBSTR());
	}

	VARIANT vIndexNameRestriction;
	vIndexNameRestriction.vt = VT_EMPTY;
	if (bstrIndex.length() > 0)
	{
		vIndexNameRestriction.vt = VT_BSTR;
		vIndexNameRestriction.bstrVal = SysAllocString(bstrIndex.GetBSTR());
	}

	VARIANT vTableNameRestriction;
	vTableNameRestriction.vt = VT_EMPTY;
	if (bstrTable.length() > 0)
	{
		vTableNameRestriction.vt = VT_BSTR;
		vTableNameRestriction.bstrVal = SysAllocString(bstrTable.GetBSTR());
	}

	VARIANT rgRestrictions[] = { vNoRestriction, vTableSchemaRestriction, vIndexNameRestriction, vNoRestriction,
vTableNameRestriction };

	CAccessorRowset<CDynamicAccessor, CBulkRowset> pRS;
	hr = spSchemaRowset->GetRowset(NULL, DBSCHEMA_INDEXES, sizeof(rgRestrictions)/sizeof(rgRestrictions[0]),
rgRestrictions, IID_IRowset, 0, NULL, (IUnknown**)&pRS.m_spRowset);
	if (FAILED(hr))
	{
		DisplayErrorInfo();
		return hr;
	}

	hr = pRS.Bind();

	hr = pRS.MoveFirst();

	SET_ORDINAL(pRS, _T("TABLE_NAME"), iTableName);
	SET_ORDINAL(pRS, _T("INDEX_NAME"), iIndexName);
	SET_ORDINAL(pRS, _T("ORDINAL_POSITION"), iOrderPosition);

	list<CIndexStruc> lstIndexes;     // map index name into index description

	while (S_OK == hr)
	{
		LPCTSTR tableName = (LPCTSTR)pRS.GetValue(iTableName);
		LPCTSTR indexName = (LPCTSTR)pRS.GetValue(iIndexName);
		int orderPos = *(int*)pRS.GetValue(iOrderPosition);

		lstIndexes.push_back(CIndexStruc(tableName, indexName));

		hr = pRS.MoveNext();
	}

	pRS.Close();

	cout << "  Indexes found: " << lstIndexes.size() << endl;


	// STEP 2. Access IIndexDefinition in destination Provider.
	cout << "Querying IIndexDefinition from target Provider..." << endl;

	CComPtr<IIndexDefinition> spIndexDefinition;
	hr = session.m_spOpenRowset->QueryInterface(__uuidof(IIndexDefinition), (void**)&spIndexDefinition);
	if (FAILED(hr))
	{
		cerr << "** ERROR: QueryInterface(IIndexDefinition) failed hr=" << hr << endl;
		return hr;
	}

	// STEP 3. Drop indexes.
	cout << "Dropping indexes..." << endl;
	bool bIgnoreAllErrors = false;
```

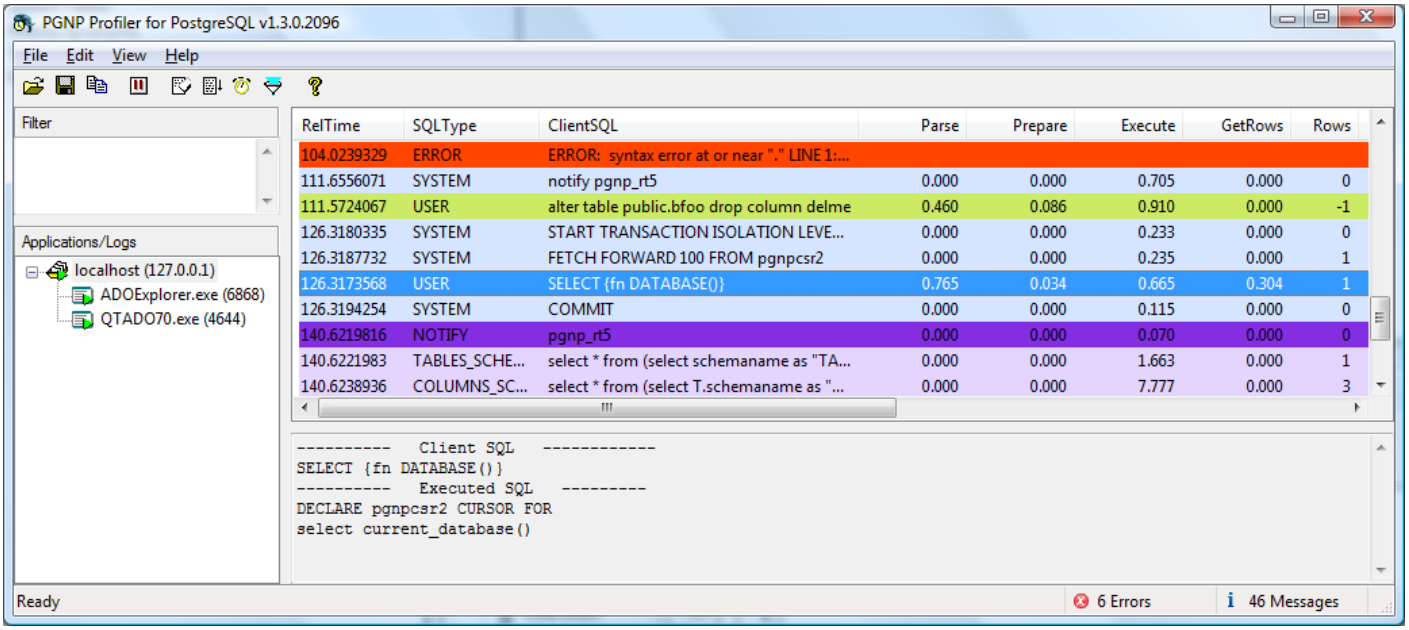
```
	int nSuccess = 0;

	for (list<CIndexStruc>::iterator it = lstIndexes.begin(); it != lstIndexes.end(); it++)
	{
		DBID TableName;
		DBID IndexName;

		TableName.eKind          = DBKIND_NAME;
		TableName.uName.pwszName = (LPOLESTR) it->sTable.c_str();

		IndexName.eKind          = DBKIND_NAME;
		IndexName.uName.pwszName = (LPOLESTR) it->sIndex.c_str();

		wcout << L"  Table: " << TableName.uName.pwszName << L"\t  Index: " << IndexName.uName.pwszName << endl;

		// Create a two-column composite index named full_name_index over the
		// LastName and FirstName columns in the Employees table.
		hr = spIndexDefinition->DropIndex(&TableName, &IndexName);

		if (FAILED(hr))
		{
			DisplayErrorInfo();
			if (!bIgnoreAllErrors)
			{
				cerr << "Terminate the indexes deletion process? Enter 'Y' to terminate, " << endl
					<< "   'N' to continue, 'I' to ignore all further errors [Y]: ";

				string sInput;
				std::getline(cin, sInput, '\n');
				if (sInput.length() == 0)
					break;

				bIgnoreAllErrors = sInput[0] == 'I' || sInput[0] == 'i';
				if (!bIgnoreAllErrors && !(sInput[0] == 'N' || sInput[0] == 'n'))
					break;
			}
		}
		else
			nSuccess++;
	}

	CoUninitialize();

	cout << "Successfully dropped " << nSuccess << " indexes. Not being able to drop: " << lstIndexes.size() - nSuccess <<
"." << endl;

	return 0;
}
```

5.3 PGNP Profiler (1.3.x and later)

The profiler utility allows collecting SQL statements with performance counters and internal execution trace from the PGNP provider. Applications can also publish events into the Profiler's log using pgnp_comment stored procedure. The SQL trace and counters information can be filtered, stored to an external file, and loaded later. Any application or service (32- and 64-bit) that uses PGNP provider can be profiled.

5.3.1 User interface explained

There are four main panels in the profiler window:



Applications/Logs panel contains list of computer hosts ( ), applications ( ) and log files ( ).

Filter panel allows user to enter text criteria for filtering out the SQL trace.

Messages panel display columns with statement time, SQL text, profiler counters, etc. Columns contain following information:

"AbsTime" and "RelTime" is first column that displays either absolute or relative time. User can click button on the toolbar to switch to AbsTime or button – to switch to RelTime;

"SQLType" – displays SQL type, i.e.:

USER	statements sent by the user application to PGNP provider
SYSTEM	provider generated statements
NOTIFY	schema change notifications sent to or received from Postgres
ERROR	error condition in the provider
xxx_SCHEMA	statement generated by the provider to support corresponding OLEDB schema

"ClientSQL" – displays either SQL from user application or generated by the provider;

"Parse", "Prepare", "Execute", "GetRows" – time in milliseconds spent by provider to parse, prepare, execute statement and to read rows from database server;

"Rows" – number of rows read or affected during execution of the SQL;

"Database", "User" – database and user name for the connection;

"PID" – process ID of the application that issued the message;

"SessId" – logical OLEDB session ID, it can be used to distinguish between connections made from the same application;

"CmdId" – logical OLEDB command ID, reserved for future, can be used to distinguish different commands in the same session;

"CursorMode" – displays "Forward Only" for ADO client cursor or "Can scroll backwards" – for ADO server cursor;

"Application" – application name and PID, i.e. origin of the message;

"CmdType" – statement type, e.g. SELECT, UPDATE, INSERT, DELETE, CREATE xxx, ALTER, SET, SHOW, DROP xxx, PROCEDURE, INTERNALPROC, START, COMMIT, ROLLBACK, NOTIFY, COPY, etc.

Details panel displays detailed SQL text for the selected in **Messages panel** messages. For a selected message it can display one of or both "Client SQL" and "Executed SQL". "Client SQL" displays the statement text sent by a user application. "Executed SQL" displays statement sent by PGNP provider to database. "Client SQL" and "Executed SQL" can be different. Latter displays parameter values, e.g.:

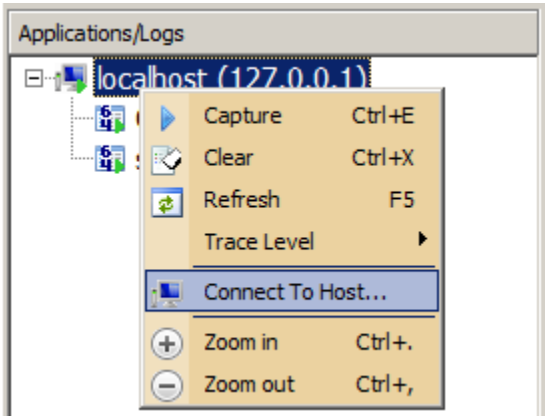
```
----------   Client SQL   ------------
UPDATE tbl SET modified=?, article=? WHERE id=? AND created=?
----------   Executed SQL   ---------
update tbl set modified=[DBTYPE_DBTIMESTAMP,20091016
16:00:00.000],article=[DBTYPE_WSTR,"Nexus12"] where id=[DBTYPE_I4,2000] and
created=[DBTYPE_DBTIMESTAMP,20091009 16:00:00.000]
```

5.3.2 Main actions in the profiler

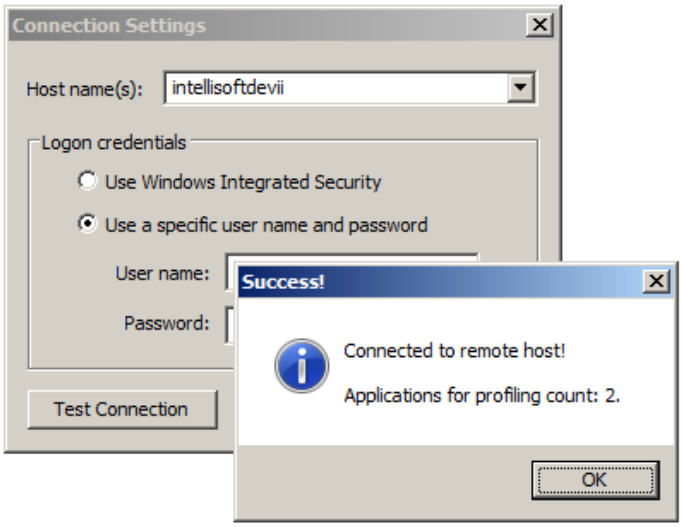
Action	How-to
Start profiling an application	Select an application in "Application/Logs" panel and click button in the toolbar.
Start profiling all applications on a host computer	Select a host in "Application/Logs" panel and click button in the toolbar.
Stop/pause profiling an application	Select an application in "Application/Logs" panel and click button in the toolbar.
Stop/pause profiling all applications on a host computer	Select a host in "Application/Logs" panel and click button in the toolbar.
Remove application from the list	An application can be removed from the list only if it terminated, i.e. has following icon: . Click Clear button ( ) in the toolbar.
Start or stop filtering	Press on filter button in the toolbar ( ).
Switch from absolute to relative time and back	Click on button-indicator (to switch to relative time) or on (to switch to absolute time).

5.3.3 Collecting trace from remote computers

Starting with build 1.4.0.3400, the PGNP Profiler supports receiving event traces from remote computers (not only from localhost). In order to connect to a remote system, right-click mouse in the Applications panel:

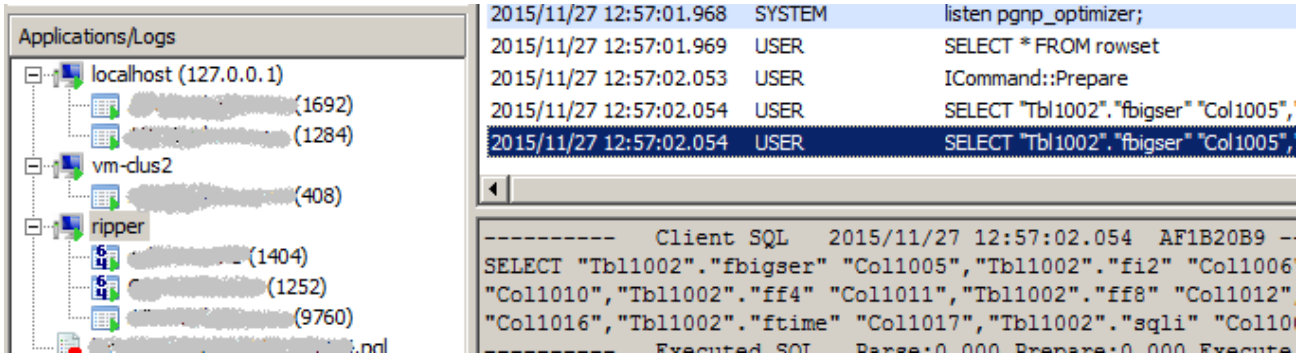


In the Connection Settings dialog enter a remote computer name or an IP address, and (optionally) user name and password. Then click Test Connection button. If connection test succeeds, the message will also display the number of applications available for profiling:



The Host name(s) combo-box keeps history of connections, and can be used to quickly choose one of the previous connections.

Click OK button to open connection to the remote host. Then click Capture button in the toolbar on top in order to start collecting trace. The PGNP Profiler can look like the following:



The list of applications on remote hosts will refresh automatically every 10 seconds, similarly to the list on the local host. The closed applications will be marked with the “red bar” sign, and the new applications will be appended to the list. The

list can be refreshed manually by right-clicking mouse on the remote host item, and selecting Refresh menu item, or by pressing F5.

Note: The PGNP Profiler copies itself to and runs on remote host as Windows service in order to obtain list of applications available. When the Profiler closes, it performs cleanup. However, if physical connection broke, or some other irregular termination occurred, manual cleanup might be needed on the remote host computer. Use the following VB script for the cleanup:

```
' The script stops and deletes "PGNProfiler" services
Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\.\root\cimv2")
Set colListOfServices = objWMIService.ExecQuery ("Select * from Win32_Service ")

Set shell = CreateObject("WScript.Shell")
Set fso = WScript.CreateObject("Scripting.FileSystemObject")

For Each objService in colListOfServices
        If Left(objService.name,11) = "PGNProfiler" then
                If vbYes = Msgbox("Delete " & objService.name & "?", vbYesNo, "Delete Service") Then
                        shell.Run "sc stop " & objService.name, 0, True
                        shell.Exec "sc delete " & objService.name
                        filename = shell.ExpandEnvironmentStrings("%WinDir%") & "\" & objService.name & ".exe"
                        fso.DeleteFile(filename), DeleteReadOnly
                End if
        End if
Next

Set shell = Nothing
Set fso = Nothing

wscript.echo "Done!"
```

5.3.4 Filtering messages in the trace

Filter pane accepts numeric, string and Boolean expressions on variables of a message. The variables names are the same as column names in “Messages panel”, i.e. clientsql, execute, etc. (see the columns list and descriptions above).

There are several preconfigured filters available by right mouse click in the Filter panel:

“ClientSQL Like…” – display SQL messages according to the regular expression, e.g.

```
clientsql ilike 'select(.)+'
```

“Show Errors” – display error messages only, i.e.

```
SQLType = ERROR
```

“Hide System” – hide any “system”, i.e. the provider generated messages, i.e.

```
SQLType <> SYSTEM
```

“ExecuteTime > 1ms” – display statements with execution time over 1 millisecond, i.e.

```
execute > 1.0
```

“Schema Alterations” – display only schema alteration messages such as DROP TABLE tbl, i.e.

```
(ClientSQL ilike 'Alter(.)+' || ClientSQL ilike 'Create(.)+' || ClientSQL ilike 'Drop(.)+')
&& SQLType != ERROR
```

“Stored Procedures and Notifications” – display only stored procedures calls and notifications sent and received, i.e

```
CMDType == PROCEDURE || CMDType == INTERNALPROC || CMDType == NOTIFY
```

Following operators can be used in the expressions:

Operator	Description

+	Add variables or numeric constants
-	Subtract variables or numeric constants, or negate a value
*	Multiply variables or numeric constants
/	Divide variables or numeric constants
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
!=	Not equal
<>	Not equal
=	Equal
==	Equal
!	Logical NOT
(…)	Sub-expressions grouping
&&	Logical AND
||	Logical OR
like	Case sensitive regular expression LIKE
ilike	Case insensitive regular expression LIKE
not like	Case sensitive regular expression NOT LIKE
not ilike	Case insensitive regular expression NOT LIKE

Like/ILike operators are based on **boost** regular expressions evaluation engine. For more details read article: http://www.boost.org/doc/libs/1_40_0/libs/regex/doc/html/boost_regex/syntax/perl_syntax.html ("Perl Regular Expression Syntax").

Following constants can be used in expressions when comparing a variable to a constant:

Variable	Constant
SQLType	USER SYSTEM NOTIFY ERROR COLUMNS_SCHEMA INDEXES_SCHEMA TABLES_SCHEMA CATALOGS_SCHEMA FOREIGN_KEYS_SCHEMA PRIMARY_KEYS_SCHEMA PROCEDURE_COLUMNS_SCHEMA PROCEDURE_PARAMETERS_SCHEMA PROCEDURES_SCHEMA
CMDType	SELECT INSERT

	UPDATE DELETE CREATE DATABASE CREATE TABLE CREATE VIEW CREATE INDEX CREATE FUNCTION ALTER SET SHOW DROP DATABASE DROP TABLE DROP VIEW DROP INDEX DROP FUNCTION PROCEDURE INTERNALPROC START TRANSACTION COMMIT ROLLBACK NOTIFY COPY
CursorMode	Forward only Can scroll backwards

5.3.5 Format of PGL file

Size	Value	Description
12	"PGNProfiler\1"	File header, same for all files. Single byte after 'PGNProfiler' is format version number.
4	Integer N	Total number of messages.
4	Integer E	Number of errors. This number is only for convenience of displaying PGL file in viewer. It is for avoiding calculation of errors count.
N*4		Array of 4-byte integers representing messages and errors sizes. Messages and errors in format described in separate tables below. The messages and errors are always 4-byte aligned by zeroes appended to the end. The array of sizes contains sizes including the alignment.
>4		Logger ID to Name value pairs. Logger ID is 4-byte integer used in messages and error to resolve into Logger Name (usually Process name). The list can have one or more pairs. Logger Name ends with '\0'.

5.3.5.1 Message format

Size	Value	Description
4		Logger ID

1	0 (TRC_NONE)	Trace type
	1 (TRC_CLIENTSQL)	
	2 (TRC_SYSTEMSQL)	
	3 (TRC_NOTIFIES)	
	4 (TRC_ERROR)	
	5 (TRC_SCHEMA_COLUMNS)	
	6 (TRC_SCHEMA_INDEXES)	
	7 (TRC_SCHEMA_TABLES)	
	8 (TRC_SCHEMA_CATALOGS)	
	9 (TRC_SCHEMA_FOREIGN_KEYS)	
	10 (TRC_SCHEMA_PRIMARY_KEYS)	
	11 (TRC_SCHEMA_PROCEDURE_COLUMNS)	
	12 (TRC_SCHEMA_PROCEDURE_PARAMETERS)	
	13 (TRC_SCHEMA_PROCEDURES)	
	14 (TRC_SYS_SCHEMA)	
	15 (TRC_USER_SCHEMA)	
	16 (TRC_COMMENT)	
1	0 (QT_NONE)	Command type
	1 (QT_SELECT)	
	2 (QT_INSERT)	
	3 (QT_UPDATE)	
	4 (QT_DELETE)	
	5 (QT_CREATE_DATABASE)	
	6 (QT_CREATE_TABLE)	
	7 (QT_CREATE_VIEW)	
	8 (QT_CREATE_INDEX)	
	9 (QT_CREATE_FUNCTION)	

	10 (QT_ALTER)	
	11 (QT_SET)	
	12 (QT_SHOW)	
	13 (QT_DROP_DATABASE)	
	14 (QT_DROP_TABLE)	
	15 (QT_DROP_VIEW)	
	16 (QT_DROP_INDEX)	
	17 (QT_DROP_FUNCTION)	
	18 (QT_PROCEDURE)	
	19 (QT_INTERNAL_PROC)	
	20 (QT_START_TRANS)	
	21 (QT_COMMIT)	
	22 (QT_ROLLBACK)	
	23 (QT_NOTIFY)	
	24 (QT_COPY)	
1	0 (FORWARD_ONLY)	Cursor type
	1 (CANSCROLLBACKWARDS)	
	2 (VIRTUAL)	
1		License chip (deprecated)
8	FILETIME	Timestamp when the message was received
8	FILETIME	Parse duration in microseconds
8	FILETIME	Prepare duration in microseconds
8	FILETIME	Execution duration in microseconds
8	FILETIME	Data get operation duration
4		Number of rows affected
2		Session ID
2		Command ID

4+len	Client SQL length and text
4+len	Executed Statement length and text
4+len	Database name length and text
4+len	User name length and text

5.3.5.2 Error format

Size	Value	Description
4		Logger ID
1	4 (TRC_ERROR)	Trace type
1	– Same as in Message –	Command type
1	– Same as in Message –	Cursor type
1		License chip (deprecated)
8	FILETIME	Timestamp when the message was received
2		Session ID
2		Command ID
4+len		Database name length and text
4+len		User name length and text
4+len		Error length and text

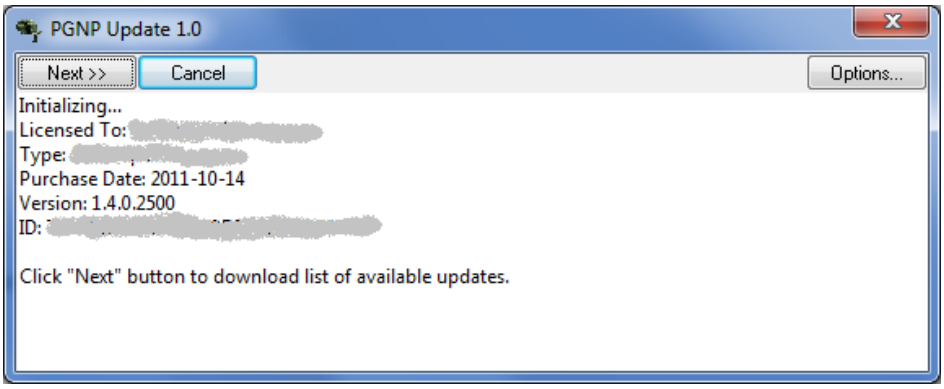
5.4 PGNPUpdate (1.4.x and later)

The purpose of ***PGNPUpdate*** application is help users to automate the product activation and updates. It may function in either one of the two modes that are automatically determined on the application startup.

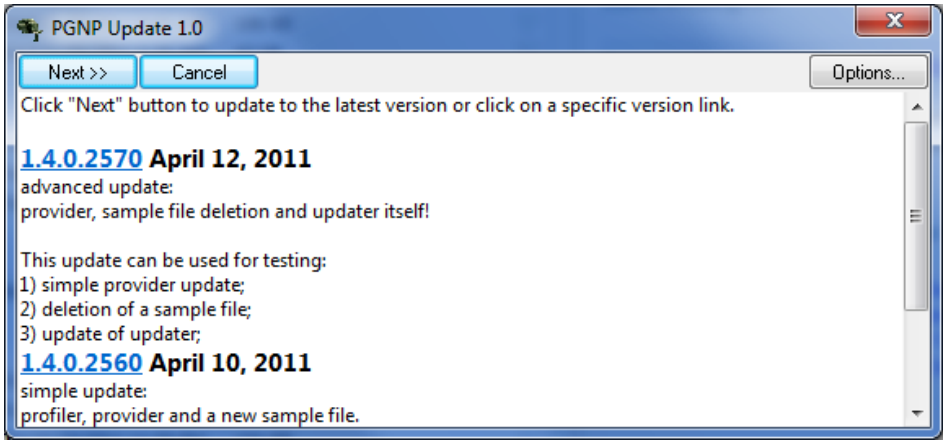
PGNPUpdate launches in Normal mode when the product is activated and is fully functional. It launches in Activation mode when the product requires activation.

5.4.1 Working in Normal mode

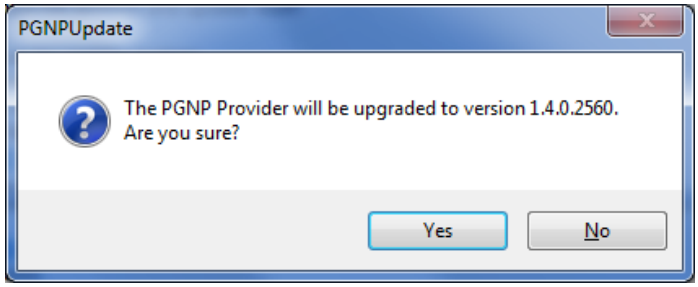
The application displays registration information on startup as shown below:



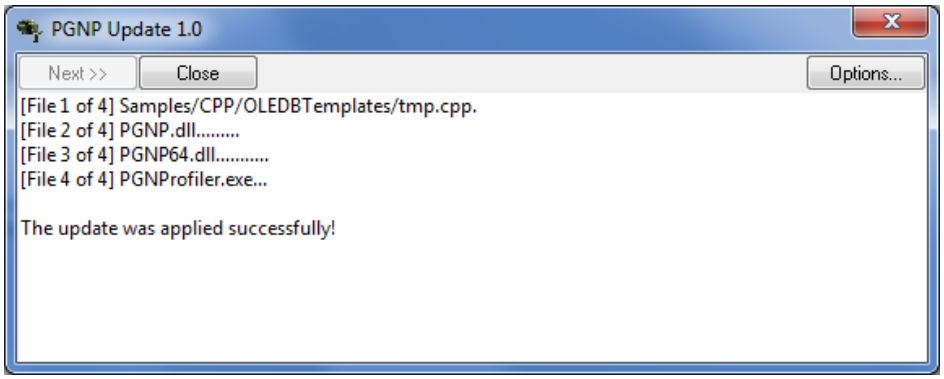
Click Next to get the list of available updates:



The updates list is sorted by the publication date so that newest updates are listed first. User can install a specific update by clicking on the link, e.g. 1.4.0.2560, or can install the latest available update by just clicking on the Next button. Click "Yes" in the confirmation box:

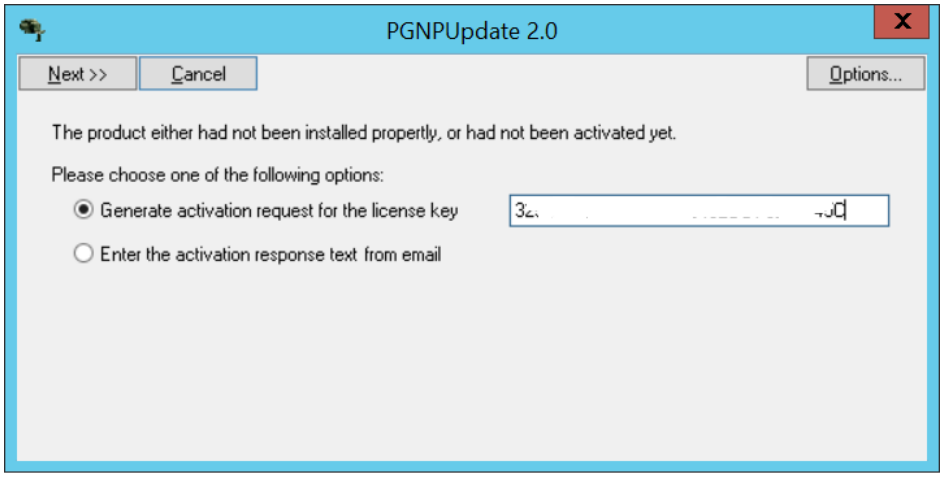


Then the update progress and the completion result are shown:

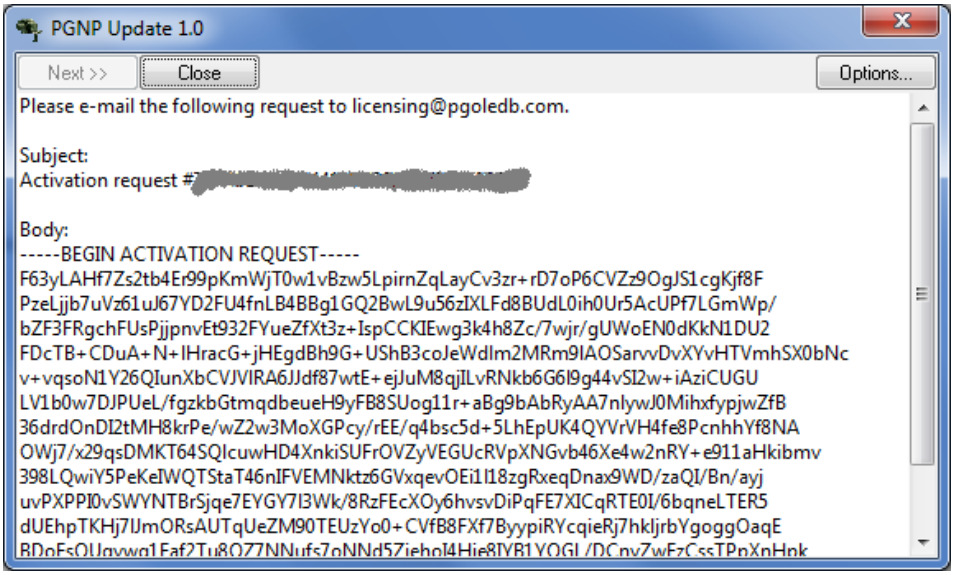


5.4.2 Working in Activation mode

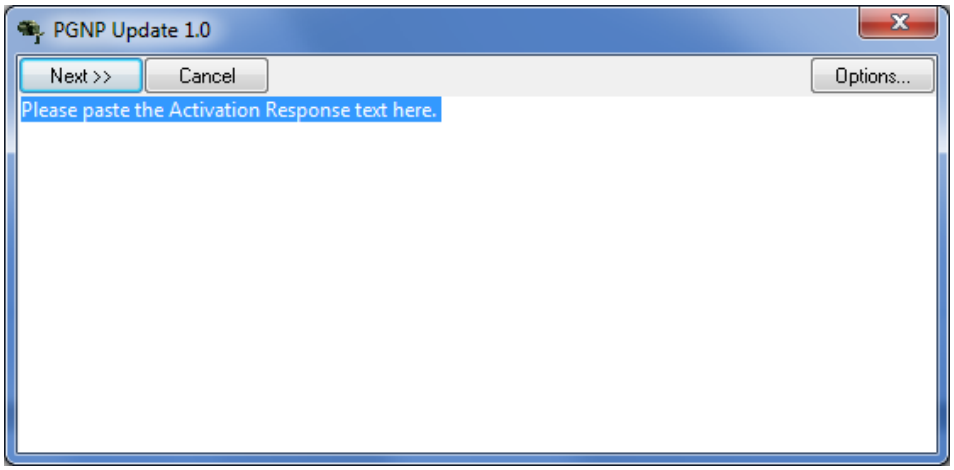
The application allows the user to manually activate the product in situations when Internet access is restricted from the computer. This assumes that user can copy text files inside LAN and send or receive e-mails from the same or the other computer. On startup, it offers user either to generate an activation request, or to activate the product by previously received activation response:

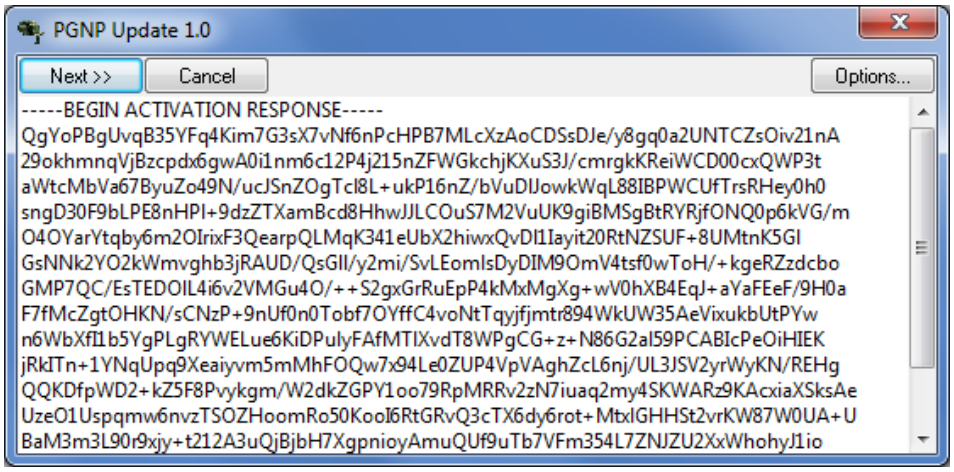


After user entered Product Key and clicked Next button, the Activation Request screen will be shown:

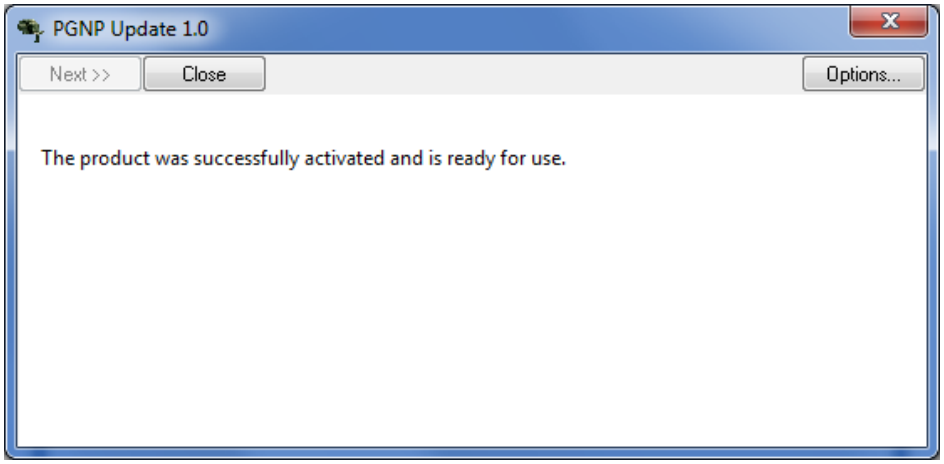


E-mail the request to the specified address, and restart the application in order to finish the manual activation process. After received the Activation Response in an e-mail, enter it on the following page:





Next page displays result of the activation:



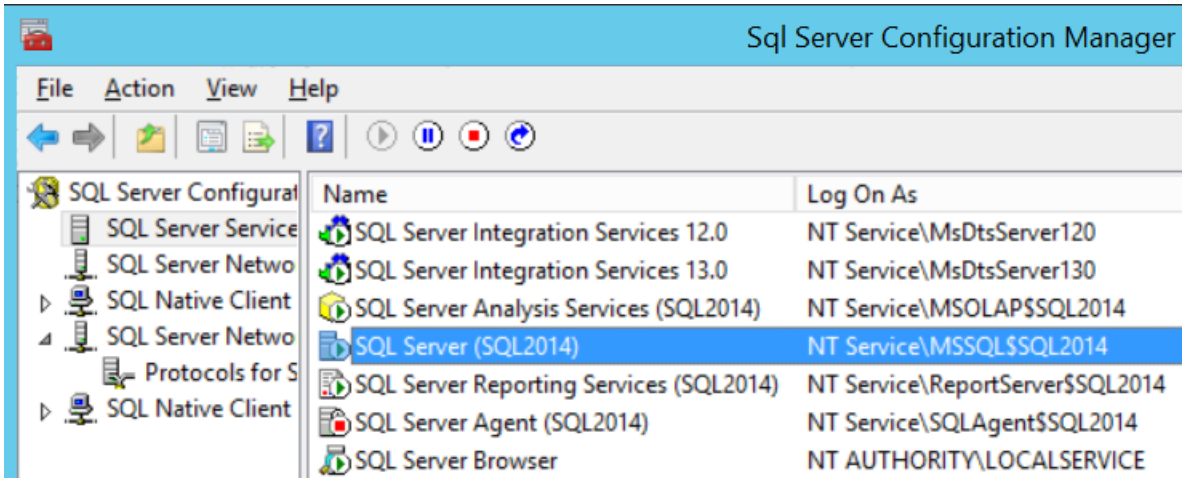
5.4.3 Granting permissions to OLE DB provider for special users

On some systems the SQL Server, SSAS, etc. can be configured to run under special account with restricted permissions to access registry and file system. Symptoms of the issue could be “The provider 'PGNP.1' is not registered. The following

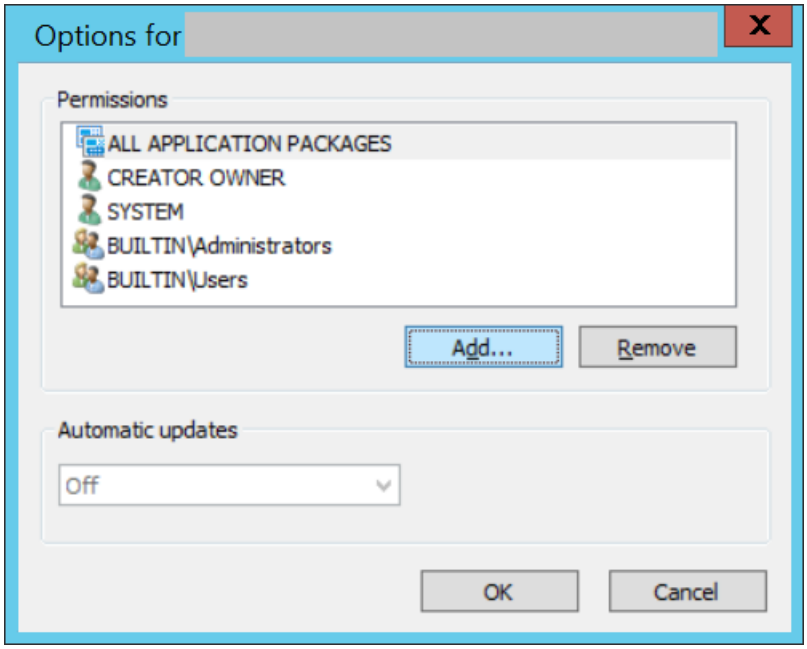
system error occurred: Class not registered", or some other error when opening connection. In order to enable a restricted user access to the PGNP provider, perform the following steps.

Note: it is highly recommended to backup registry before performing the following steps. Even though the changes should affect only PGNP provider COM registry, it is still a good practice to make backups in order to be able to restore system to previous state.

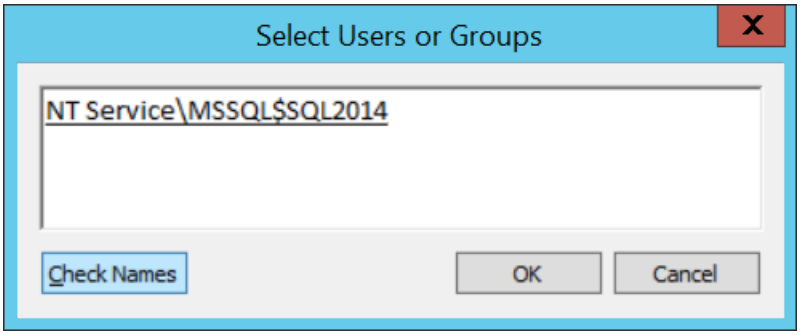
Step 1. Locate name of the restricted user running the SQL Server/SSAS/etc. either in Process Explorer or in Sql Server Configuration Manager. In figure below, it is "NT Service\MSSQL$SQL2014":



Step 2. Launch PGNPUpdate utility, and open Options dialog. Click "Add…" button in the Permissions pane:



Step 3. In the Select Users or Groups dialog, enter the user name (or multiple names separated by semicolon), and click Check Names button. If user name is recognized by the Check, it will be underlined as shown below:



Click OK, OK in order to save changes.

Then check if SQL Server/SSAS/etc. can access the PGNP provider after granting the access to the user.

5.4.4 Using from command line

When command line parameter(s) specified, the application starts in console mode.

USAGE:

C:\Progran Files (x86)\Intellisoft\PGNP\PGNPUpdate.exe [-c] [-a] [-u <version>] [-g <license key>] [-r] [--] [-v] [-h]

Where:

-c, --check
Check for available updates.

-a, --all
Apply all available updates.

-u <version>, --update <version>
(value required) Update to the target version, e.g. 1.4.0.3456.

-g <license key>, --generate <license key>
(value required) Generate activation request if product is not activated yet. The request text will be printed to stdout.

-r, --response
Enter the activation response from stdin.

--, --ignore_rest
Ignores the rest of the labeled arguments following this flag.

-v, --version
Displays version information and exits.

-h, --help
Displays usage information and exits.

Example 1. Generate Activation Request.

C:\> PGNPUpdate.exe -g 7XXXXXXXXXXXXXXXXXXXXX1 > request123.txt

The Activation Request will be saved into the request123.txt file.

Example 2. Enter the Activation Response.

C:\> PGNPUpdate.exe -r < response456.txt

The activation response will be processed and activation result will be displayed.

Example 3. Update product to the specified version.

C:\> PGNPUpdate.exe -u 1.4.0.3538

The product will be updated to the version 1.4.0.3538. Only forward updates are currently supported.

Error codes returned by PGNPUpdate application in console mode:

Error code	Description
1	Could not determine PGNP Provider installation location
2	Only one instance of PGNPUpdate can run at a time
3	Target file not found
4	Cannot map source file
5	Could not read PGNP Provider license info
6	PGNP Provider already activated
7	Invalid command line arguments
8	Invalid activation response
10	Main window creation failed
11	Unable to spawn kernel transaction.
12	Failed to get temporary path.
13	Failed to create temporary folder.
14	Failed to download file.
15	Could not apply update since some files were locked.
16	Internal Error: Could not open a source file.
17	Internal Error: Could not read a source file.
18	No files were downloaded. Possibly there are no newer files to download.
20	Failed to access stdin handle
21	Failed to open stdin
22	Failed to access stdout handle
23	Failed to open stdout
25	Failed to create console

6 Appendix B. Handling ISequentialStream in the OLEDB provider

Source data	STREAM parameter	Hint* (CREATE TABLE)	Action (destination UTF8)
char	SHORT	-	Copy**
wchar	SHORT	-	**Copy (error!)**
char	SHORT	text	Copy
wchar	SHORT	ntext	WideCharToMbyte
char	WIDE	-	**WideCharToMbyte (error!)**
wchar	WIDE	-	WideCharToMbyte
char	WIDE	text	Copy
wchar	WIDE	ntext	WideCharToMbyte

** Hint is an optional CREATE STATEMENT that the OLEDB provider uses for handling ISequentialStream. Hint overrides the provider behavior defined by the STREAM connection string parameter.*

*** "Copy" means single-characters copying of data from the stream.*

7 Appendix C. Samples

The samples use schema defined in samples_schema.sql. Make sure to create the schema by executing the sql commands before launching samples. The schema can be removed by executing commands from samples_clear.sql.

Samples can be compiled with Visual Studio 2005 or later, and Delphi 7.

7.1 C# Samples

No/Name	Description
01_GetRecords.cs	This sample demonstrates how programmatically connect to PostgreSQL database and get records from pgnp_samples.contact table.
02_SingleValue.cs	This sample demonstrates getting single value from database. It reads number of contacts with last name starting with 'S'.
03_InsertUpdateDelete.cs	This sample demonstrates creation of a temporary table, and records insertion, updates and deletion.
04_CommandParameters.cs	This sample demonstrates passing positional and named parameters to a command.
05_StoredProcedures.cs	This sample demonstrates how to call stored procedures returning a single result and multiple results.
06_TwoPhaseCommit.cs	This sample demonstrates various scenarios in Two Phase Commit protocol.
10_Arrays.cs	This sample demonstrates how to query multi-dimensional arrays in .NET

11_NestedTrans.cs	This sample demonstrates how to work with nested transactions in .NET
12_DistribTrans.cs	This sample demonstrates how to work with distributed transactions in .NET (TransactionsScope object).
13_SavePoints.cs	This sample demonstrates how to control transactions via PostgreSQL SAVE POINTs.

7.2 C++ Samples

Name	Description
ADOEvents	This example demonstrates howto subscribe for ADO connection and recordset events.
OLEDBTemplates	This sample demonstrates howto select records from a table and insert data using OLEDB templates; and -- how to call PostgreSQL function with one parameter and get the returned value using OLEDB templates; and -- how to call PostgreSQL function returning TABLE (recordset).

7.3 Delphi 7 Samples

Name	Description
TestGrid	This sample demonstrates howto populate grid with records from database.

8 Appendix D. Time zones conversion

See below the content of tznames.csv file included in the provider installation. Any of the names can be passed in TIMEZONE parameter of the connection string. The OLE DB provider will match the parameter value to a name in the first column, and will use the corresponding time zone information from Windows registry for the ***timestamptz*** values conversion. The time zones names can be added or changed in the file. The OLE DB provider will load the file when connection is open to DB.

Window name	Display name	Postgres name	Alternative	Abbrev
Afghanistan Standard Time	(UTC+04:30) Kabul	Asia/Kabul	Afghanistan	AFT
Alaskan Standard Time	(UTC-09:00) Alaska	America/Anchorage	Alaska	AKDT
		America/Juneau		
		America/Nome		
		America/Yakutat		
Aleutian Standard Time	(UTC-10:00) Aleutian Islands	Aleutian Islands	Aleutian Islands	HAST
Altai Standard Time	(UTC+07:00) Barnaul, Gorno-Altaysk	Asia/Barnaul	Barnaul	OMSK
Arab Standard Time	(UTC+03:00) Kuwait, Riyadh	Asia/Kuwait	Kuwait	AST
Arabian Standard Time	(UTC+04:00) Abu Dhabi, Muscat	Asia/Muscat	Muscat	GST
Arabic Standard Time	(UTC+03:00) Baghdad	Asia/Baghdad	Baghdad	AST
Argentina Standard Time	(UTC-03:00) City of Buenos Aires	America/Buenos_Aires	Argentina	ART
		America/Cordoba		
		America/Jujuy		
		America/Mendoza		
		America/Rosario		
Astrakhan Standard Time	(UTC+04:00) Astrakhan, Ulyanovsk	Europe/Samara	Samara	SAMT
Atlantic Standard Time	(UTC-04:00) Atlantic Time (Canada)	America/Virgin	Port_of_Spain	ADT
		Canada/Atlantic	Halifax	
AUS Central Standard Time	(UTC+09:30) Darwin	Australia/NSW	Sydney	ACST
		Australia/North	Darwin	
Aus Central W. Standard Time	(UTC+08:45) Eucla	Australia/Eucla	Eucla	ACWST

AUS Eastern Standard Time	(UTC+10:00) Canberra, Melbourne, Sydney	Australia/ACT	Sydney	AEST
		Australia/Canberra		
		Australia/LHI	Lord_Howe	AHST
		Australia/Victoria	Melbourne	AEST
Azerbaijan Standard Time	(UTC+04:00) Baku	Asia/Baku	Baku	AZST
Azores Standard Time	(UTC-01:00) Azores	Atlantic/Azores	Azores	AZOST
Bahia Standard Time	(UTC-03:00) Salvador	America/Bahia	Bahia	BRT
Bangladesh Standard Time	(UTC+06:00) Dhaka	Asia/Dacca	Dhaka	BDT
		Asia/Thimbu	Thimphu	BTT
Belarus Standard Time	(UTC+03:00) Minsk	Europe/Minsk	Minsk	MSK
Bougainville Standard Time	(UTC+11:00) Bougainville Island	Pacific/Bougainville	Bougainville	BST
Canada Central Standard Time	(UTC-06:00) Saskatchewan	Canada/East-Saskatchewan	Regina	CST
		Canada/Saskatchewan	Saskatchewan	
Cape Verde Standard Time	(UTC-01:00) Cabo Verde Is.	Atlantic/Cape_Verde	Cape_Verde	CVT
Caucasus Standard Time	(UTC+04:00) Yerevan	Asia/Yerevan	Yerevan	AMT
Cen. Australia Standard Time	(UTC+09:30) Adelaide	Australia/South	Adelaide	ACST
		Australia/Yancowinna	Broken_Hill	
Central America Standard Time	(UTC-06:00) Central America	America/Belize	Belize	CST
Central Asia Standard Time	(UTC+06:00) Astana	Asia/Astana	Tselinigrad	ALMT
Central Brazilian Standard Time	(UTC-04:00) Cuiaba	America/Cuiaba	America/Cuiaba	AMT
Central Europe Standard Time	(UTC+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague	Europe/Belgrade	Poland	CEST
Central European Standard Time	(UTC+01:00) Sarajevo, Skopje, Warsaw, Zagreb	Europe/Warsaw		
Central Pacific Standard Time	(UTC+11:00) Solomon Is., New Caledonia	Pacific/Pohnpei	Ponape	PONT
Central Standard Time	(UTC-06:00) Central Time (US & Canada)	America/Indiana/Knox	America/Knox_IN	CDT
		Canada/Central	Winnipeg	
		US/Central	Chicago	
		US/Indiana-Starke	Indiana/Knox	
Central Standard Time (Mexico)	(UTC-06:00) Guadalajara, Mexico City, Monterrey	Mexico/General	Mexico_City	CDT
Chatham Islands Standard Time	(UTC+12:45) Chatham Islands	Chatham	Chatham	CHADT
China Standard Time	(UTC+08:00) Beijing, Chongqing, Hong Kong, Urumqi	Asia/Shanghai	Chongqing	CST
		Asia/Shanghai	Chungking	
		Asia/Shanghai	Harbin	
		Asia/Kashgar	Urumqi	

		Asia/Macao	Macau	
		Asia/Hongkong	HongKong	
		PRC	Asia/Shanghai	
Cuba Standard Time	(UTC-05:00) Havana	Cuba	Havana	CDT
Dateline Standard Time	(UTC-12:00) International Date Line West	Dateline	Dateline	IDL
E. Africa Standard Time	(UTC+03:00) Nairobi	Africa/Asmera	Asmara	EAT
E. Australia Standard Time	(UTC+10:00) Brisbane	Australia/Queensland	Brisbane	AEST
E. Europe Standard Time	(UTC+02:00) Chisinau	Europe/Paris	Paris	CEST
E. South America Standard Time	(UTC-03:00) Brasilia	Brazil/East	Sao_Paulo	BRT
Easter Island Standard Time	(UTC-06:00) Easter Island	EasterIsland	Easter_Island	CST
Eastern Standard Time	(UTC-05:00) Eastern Time (US & Canada)	America/Coral_Harbour	America/Atikokan	EST
		America/Fort_Wayne	Indiana/Indianapolis	
		America/Indianapolis	Indianapolis	
		America/Louisville	Louisville	
		Canada/Eastern	Toronto	
		US/Eastern	New_York	
		US/Michigan	America/Detroit	
Eastern Standard Time (Mexico)	(UTC-05:00) Chetumal	America/Cancun	Chetumal	EST
Egypt Standard Time	(UTC+02:00) Cairo	Africa/Cairo	Egypt	EEST
Ekaterinburg Standard Time	(UTC+05:00) Ekaterinburg	Asia/Yekaterinburg	Ekaterinburg	YEKT
Fiji Standard Time	(UTC+12:00) Fiji	Pacific/Fiji	Fiji	FJST
FLE Standard Time	(UTC+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius	Europe/Helsinki	Helsinki	FLET
Georgian Standard Time	(UTC+04:00) Tbilisi	Asia/Tbilisi	Tbilisi	GET
GMT Standard Time	(UTC+00:00) Dublin, Edinburgh, Lisbon, London	Etc/GMT	London	GMT
Greenland Standard Time	(UTC-03:00) Greenland	America/Godthab	Greenland	WGT
Greenwich Standard Time	(UTC+00:00) Monrovia, Reykjavik	Atlantic/Reykjavik	Reykjavik	GMT
GTB Standard Time	(UTC+02:00) Athens, Bucharest	Europe/Athens	Athens	EEST
Haiti Standard Time	(UTC-05:00) Haiti	America/Haiti	Haiti	EST
Hawaiian Standard Time	(UTC-10:00) Hawaii	Pacific/Honolulu	Hawaii	HST
India Standard Time	(UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi	India/Chennai	India	IST
Iran Standard Time	(UTC+03:30) Tehran	Asia/Tehran	Iran	IRT
Israel Standard Time	(UTC+02:00) Jerusalem	Asia/Jerusalem	Israel	IST
Jordan Standard Time	(UTC+02:00) Amman	Asia/Amman	Jordan	EEST
Kaliningrad Standard Time	(UTC+02:00) Kaliningrad	Europe/Kaliningrad	Kaliningrad	CET
Kamchatka Standard Time	(UTC+12:00) Petropavlovsk-Kamchatsky - Old	Asia/Kamchatka	Kamchatka	PET
Korea Standard Time	(UTC+09:00) Seoul	Asia/Seoul	Korea	KDT

Libya Standard Time	(UTC+02:00) Tripoli	Africa/Tripoli	Libya	EET
Line Islands Standard Time	(UTC+14:00) Kiritimati Island	Pacific/Kiritimati	Kiribati	LINT
Lord Howe Standard Time	(UTC+10:30) Lord Howe Island	Australia/Lord_Howe	Australia	LHDT
Magadan Standard Time	(UTC+11:00) Magadan	Asia/Magadan	Magadan	MAGT
Marquesas Standard Time	(UTC-09:30) Marquesas Islands	Pacific/Marquesas	Marquesas	MART
Mauritius Standard Time	(UTC+04:00) Port Louis	Indian/Mauritius	Mauritius	MUT
Middle East Standard Time	(UTC+02:00) Beirut	Asia/Beirut	Lebanon	EEST
Montevideo Standard Time	(UTC-03:00) Montevideo	America/Montevideo	Uruguay	UYT
Morocco Standard Time	(UTC+00:00) Casablanca	Africa/Casablanca	Morocco	WET
Mountain Standard Time	(UTC-07:00) Mountain Time (US & Canada)	America/Denver	Colorado	MST
Mountain Standard Time (Mexico)	(UTC-07:00) Chihuahua, La Paz, Mazatlan	America/Chihuahua	Chihuahua	MDT
Myanmar Standard Time	(UTC+06:30) Yangon (Rangoon)	Asia/Rangoon	Myanmar	MMT
N. Central Asia Standard Time	(UTC+06:00) Novosibirsk	Asia/Novosibirsk	Novosibirsk	NOVT
Namibia Standard Time	(UTC+01:00) Windhoek	Africa/Windhoek	Namibia	WAT
Nepal Standard Time	(UTC+05:45) Kathmandu	Asia/Katmandu	Nepal	NPT
New Zealand Standard Time	(UTC+12:00) Auckland, Wellington	Pacific/Auckland	New Zealand	NZST
Newfoundland Standard Time	(UTC-03:30) Newfoundland	America/St_Johns	Newfoundland	NDT
Norfolk Standard Time	(UTC+11:00) Norfolk Island	Pacific/Norfolk	Norfolk	NFT
North Asia East Standard Time	(UTC+08:00) Irkutsk	Asia/Irkutsk	Irkutsk	IRKT
North Asia Standard Time	(UTC+07:00) Krasnoyarsk	Asia/Krasnoyarsk	Krasnoyarsk	KRAT
North Korea Standard Time	(UTC+08:30) Pyongyang	Asia/Pyongyang	North Korea	KST
Pacific SA Standard Time	(UTC-04:00) Santiago	America/Santiago	Chile	CLT
Pacific Standard Time	(UTC-08:00) Pacific Time (US & Canada)	America/Los_Angeles	Los_Angeles	PST
Pacific Standard Time (Mexico)	(UTC-08:00) Baja California	America/Tijuana	Tijuana	PST
Pakistan Standard Time	(UTC+05:00) Islamabad, Karachi	Asia/Islamabad	Pakistan	PKT
Paraguay Standard Time	(UTC-04:00) Asuncion	America/Asuncion	Paraguay	PYST
Romance Standard Time	(UTC+01:00) Brussels, Copenhagen, Madrid, Paris	Europe/Brussels	Belgium	CEST
Russia Time Zone 10	(UTC+11:00) Chokurdakh	Asia/Srednekolymsk	Srednekolymsk	SRET
Russia Time Zone 11	(UTC+12:00) Anadyr, Petropavlovsk-Kamchatsky	Asia/Anadyr	Anadyr	ANAT
Russia Time Zone 3	(UTC+04:00) Izhevsk, Samara	Europe/Izhevsk	Samara	SAMT
Russian Standard Time	(UTC+03:00) Moscow, St. Petersburg, Volgograd	Europe/Moscow	Moscow	MSK
SA Eastern Standard Time	(UTC-03:00) Cayenne, Fortaleza	America/Cayenne	Cayenne	GFT
SA Pacific Standard Time	(UTC-05:00) Bogota, Lima, Quito, Rio Branco	America/Bogota	Peru	PET

SA Western Standard Time	(UTC-04:00) Georgetown, La Paz, Manaus, San Juan	America/La Paz	Bolivia	BOT
Saint Pierre Standard Time	(UTC-03:00) Saint Pierre and Miquelon	America/Saint Pierre	Saint Pierre	PMST
Sakhalin Standard Time	(UTC+11:00) Sakhalin	Asia/Sakhalin	Sakhalin	SAKT
Samoa Standard Time	(UTC+13:00) Samoa	America/Samoa	Samoa	SST
SE Asia Standard Time	(UTC+07:00) Bangkok, Hanoi, Jakarta	Asia/Bangkok	Bangkok	ICT
Singapore Standard Time	(UTC+08:00) Kuala Lumpur, Singapore	Asia/Singapore	Singapore	SGT
South Africa Standard Time	(UTC+02:00) Harare, Pretoria	Africa/Harare	Harare	SAST
Sri Lanka Standard Time	(UTC+05:30) Sri Jayawardenepura	Asia/Colombo	Sri_Lanka	SLST
Syria Standard Time	(UTC+02:00) Damascus	Asia/Damascus	Syria	EEST
Taipei Standard Time	(UTC+08:00) Taipei	Asia/Taipei	Taiwan	CST
Tasmania Standard Time	(UTC+10:00) Hobart	Hobart	Tasmania	AEST
Tocantins Standard Time	(UTC-03:00) Araguaina	America/Tocantins	Tocantins	BRT
Tokyo Standard Time	(UTC+09:00) Osaka, Sapporo, Tokyo	Asia/Tokyo	Japan	JST
Tomsk Standard Time	(UTC+07:00) Tomsk	Asia/Tomsk	Tomsk	ALMT
Tonga Standard Time	(UTC+13:00) Nuku'alofa	Pacific/Tongatapu	Tongatapu	TOT
Transbaikal Standard Time	(UTC+09:00) Chita	Asia/Atamanovka	Chita	TST
Turkey Standard Time	(UTC+02:00) Istanbul	Europe/Istanbul	Turkey	EEST
Turks And Caicos Standard Time	(UTC-04:00) Turks and Caicos	America/Turks_Caicos	Cockburn_Town	AST
Ulaanbaatar Standard Time	(UTC+08:00) Ulaanbaatar	Asia/Ulaanbaatar	Mongolia	ULAT
US Eastern Standard Time	(UTC-05:00) Indiana (East)	America/Indiana	Indianapolis	EDT
US Mountain Standard Time	(UTC-07:00) Arizona	America/Phoenix	Arizona	MST
UTC	(UTC) Coordinated Universal Time			UTC
UTC+12	(UTC+12:00) Coordinated Universal Time+12			UTCp12
UTC-02	(UTC-02:00) Coordinated Universal Time-02			UTCm2
UTC-08	(UTC-08:00) Coordinated Universal Time-08			UTCm8
UTC-09	(UTC-09:00) Coordinated Universal Time-09			UTCm9
UTC-11	(UTC-11:00) Coordinated Universal Time-11			UTCm11
Venezuela Standard Time	(UTC-04:00) Caracas	America/Caracas	Venezuela	VET
Vladivostok Standard Time	(UTC+10:00) Vladivostok	Asia/Vladivostok	Vladivostok	VLAT
W. Australia Standard Time	(UTC+08:00) Perth	Australia/Perth	Perth	AWST
W. Central Africa Standard Time	(UTC+01:00) West Central Africa	Africa/Kinshasa	Kinshasa	WAT
W. Europe Standard Time	(UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna	Europe/Amsterdam	Holland	CEST
W. Mongolia Standard Time	(UTC+07:00) Hovd	Asia/Hovd	Hovd	HOVT

West Asia Standard Time	(UTC+05:00) Ashgabat, Tashkent	Asia/Ashgabat	Turkmenistan	TMT
West Bank Standard Time	(UTC+02:00) Gaza, Hebron	Asia/Gaza	Palestina	EEST
West Pacific Standard Time	(UTC+10:00) Guam, Port Moresby	Asia/Guam	Guam	CHST
Yakutsk Standard Time	(UTC+09:00) Yakutsk	Asia/Yakutsk	Yakutsk	YAKT

Note: time zones names in the tznames.csv file cannot contain double quote and semicolon characters. Also, lines with no name in first column, or all empty columns except the first are ignored. PGNProfiler shows messages regarding the tznames.csv loading at Comment Level 2.

The time zones can be selected in the Data Link Properties dialog: