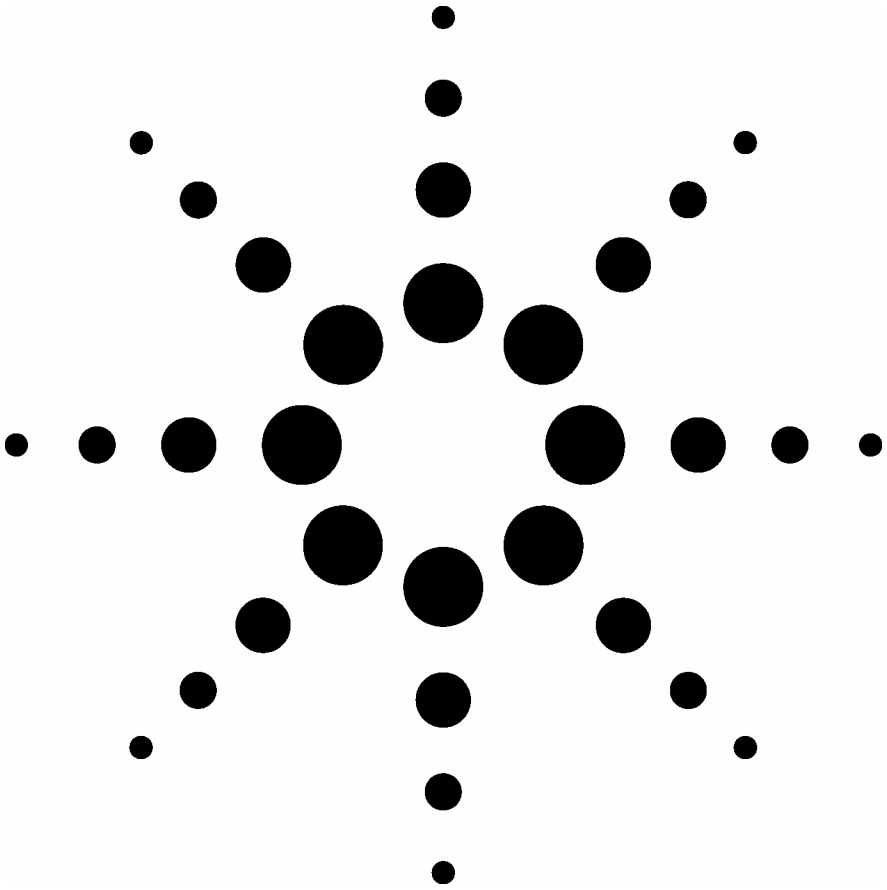
Accessing a PostgreSQL Database from VEE

White Paper



Dong Jia
Yukiko Sugiyama
Agilent Technologies

Agilent Technologies

Introduction

Many of today's test environments demand more than spreadsheets, text file archives, or flat files to compile, manage, and safeguard the volumes of data they generate. Databases often provide an elegant, but sometimes costly or inconvenient, alternative. There is, however, an inexpensive and straightforward way to incorporate databases into a VEE-based test environment.

This paper shows you how to quickly download, configure, and use a set of freely available database tools in order to store data generated by VEE Pro and to retrieve that data for display within VEE. We will not discuss the merits of using databases or the thinking behind effective database design for T&M applications. The intention here is to give the reader the basics required to get started.

The software that we will use in this discussion is PostgreSQL, Npgsql, and VEE Pro. You can download the first two, as well as a 30-day fully functional copy of the VEE software, for free. You will also use the .NET Framework, but this is included as part of the VEE Pro installation for the evaluation and licensed versions.

This paper assumes that the reader has experience with VEE. More specifically, it assumes that the reader is moderately familiar with VEE's "dot" notation: *myLibrary.myFunction( )*. Also, although formal knowledge of .NET is not required, a basic understanding of using .NET in VEE is helpful for understanding the content herein.

The PostgreSQL Database and Npgsql Data Provider

PostgreSQL is an open-source, object-relational database management system. It supports much of the SQL standard as well as a broad range of advanced functionality, such as complex queries and foreign keys. PostgreSQL also is extensible; users can add data types, operators, index methods, etc. For more information on PostgreSQL, visit www.postgresql.org.

Npgsql is a .NET Data Provider for Postgresql Database Server. Npgsql allows .NET client applications, such as VEE Pro programs, to use the PostgreSQL server to send data to the database and to retrieve it. For more information about Npgsql, visit pgfoundry.org/projects/npgsql.

Requirements

PostgreSQL 8.1.1 (or higher)
Npgsql 0.7.1, Npgsql 0.7.1FixedAssemblyVesion
VEE Pro 7.5 (or higher)
postgresql_example.vee

Download and Installation

Download PostgreSQL 8.1.1 (for free) from the following site:
http://wwwmaster.postgresql.org/download/mirrors-ftp?file=binary%2Fv8.1.1%2Fwin32%2Fpostgresql-8.1.1-1.zip

Unzip the archive and run *postgresql-8.1.msi* to start the installation. Follow the prompts as instructed.

The PostgreSQL 8.1.1 installation includes Npgsql files, but you will need to update one of the dlls. Therefore, download *Npgsql 0.7.1FixedAssemblyVesion* (for free) from:
http://pgfoundry.org/frs/download.php/522/Npgsql0.7.1FixedAssemblyVersion.zip

There is one file in the zip file: *npgsql.dll* (dated 4 Sept 2005). Use this file to replace npgsql.dll, which resides in: *<PostgreSQL application directory>\8.1\Npgsql\bin\ms1.1.* (The default directory is: *C:\Program Files\PostgreSQL\8.1\Npgsql\MS1.1.)*

To download a fully functional 30-day evaluation version of VEE Pro 7.0 (or later), go to www.agilent.com/find/adnevalvee. You also will need IO Libraries Suite 14.0 (or later), available at: http://adn.tm.agilent.com/index.cgi?CONTENT_ID=26.

And finally, you can download the VEE Pro sample program (*Accessing a PostgreSQL database*), explained in this document, at http://adn.tm.agilent.com/index.cgi?CONTENT_ID=2970.

Setting up a simple PostgreSQL database

Once you have installed the PostgreSQL software, you can set up the database that you will access from postgresql_example.vee.

1. Go to Start > Programs > PostgreSQL8.1 and launch *pgAdmin III*. You will see the following screen (Figure 1):

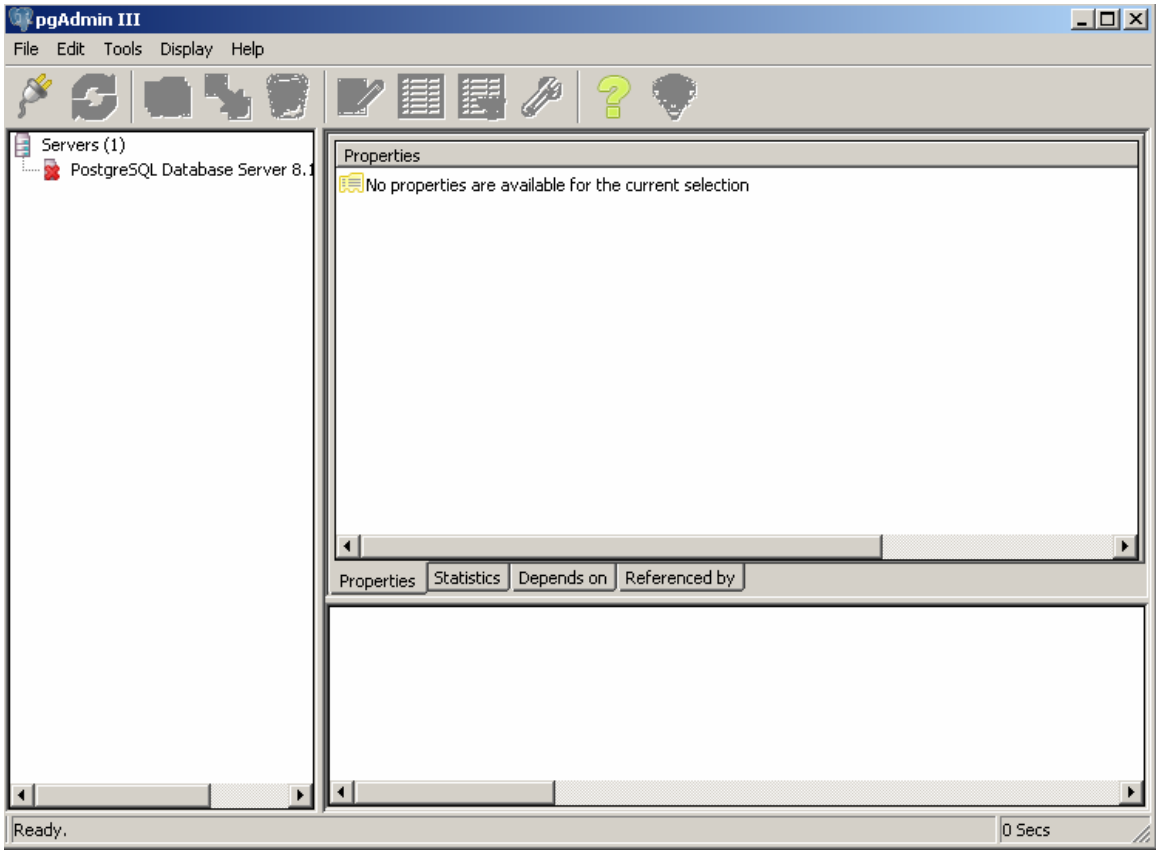


Figure 1

2. Right click on “PostgreSQL Database Server 8.1…” and select *Connect*.

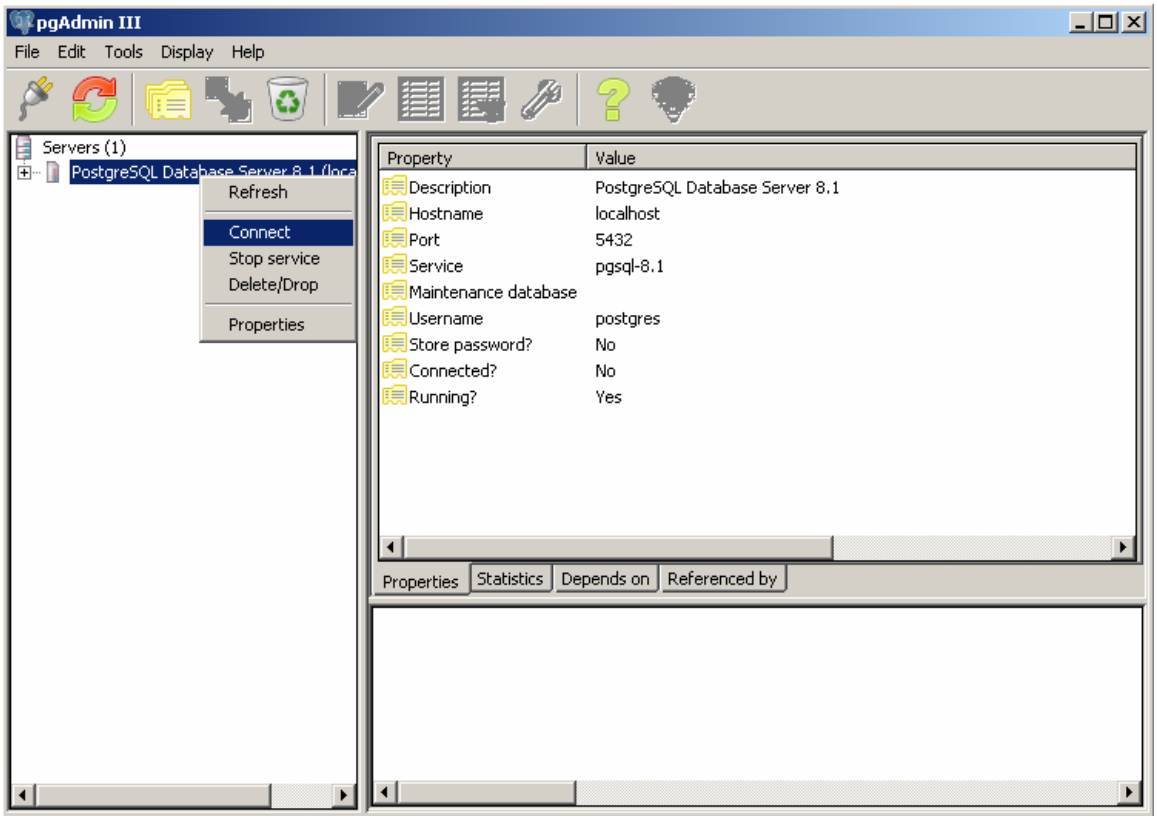


Figure 2

3. During the installation, you will be asked whether you want to have an account and password automatically generated. You may leave the account name blank, but do not let pgAgmin III create a random password for you. (It will be cryptic.)

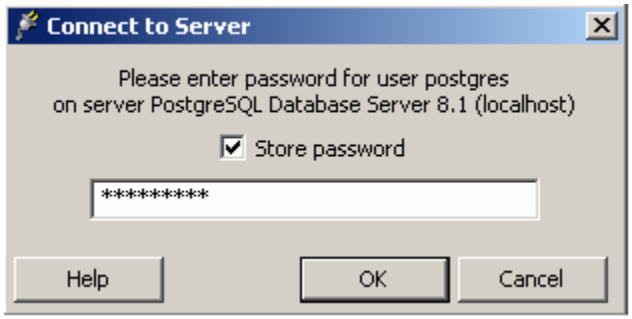


Figure 3

4. A default “postgres” database structure will be setup for you. You now have to create the table, as shown in Figure 4.

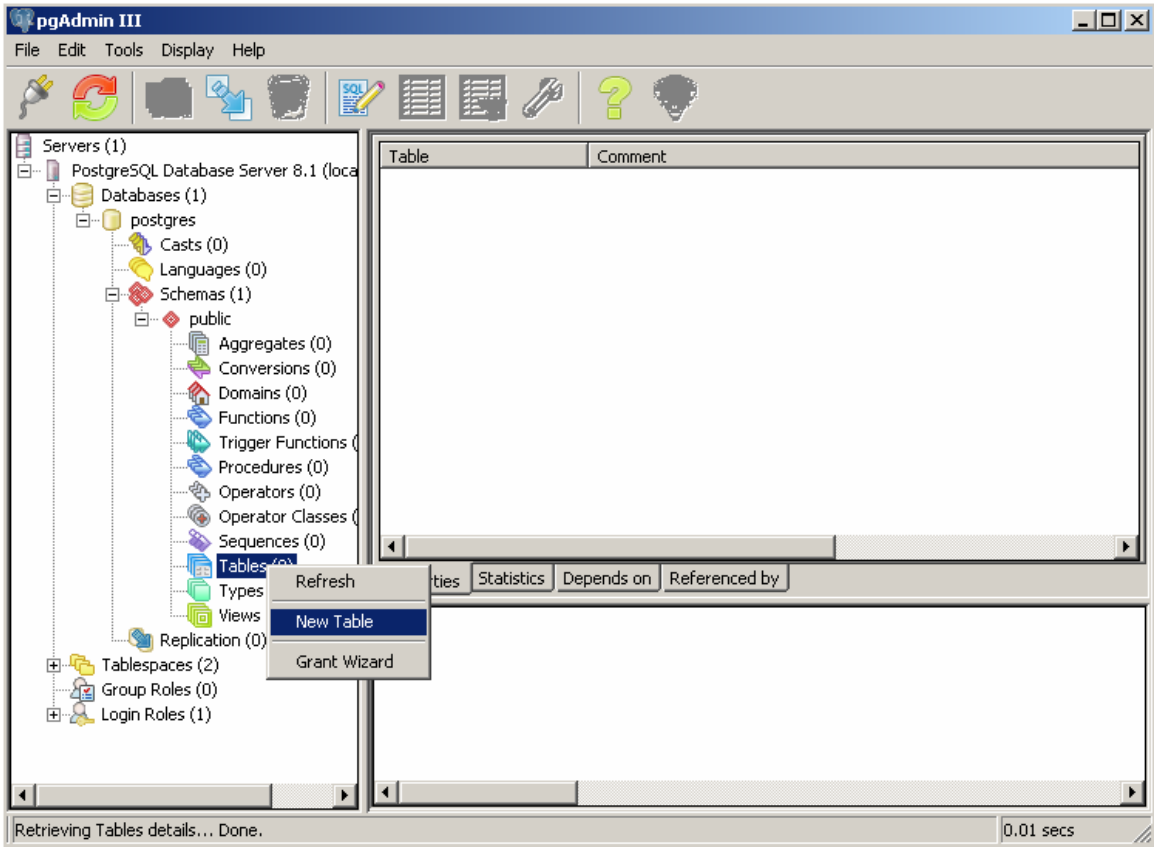


Figure 4

5. Select the Name, Owner, and Tablespace as shown in the screenshot here:

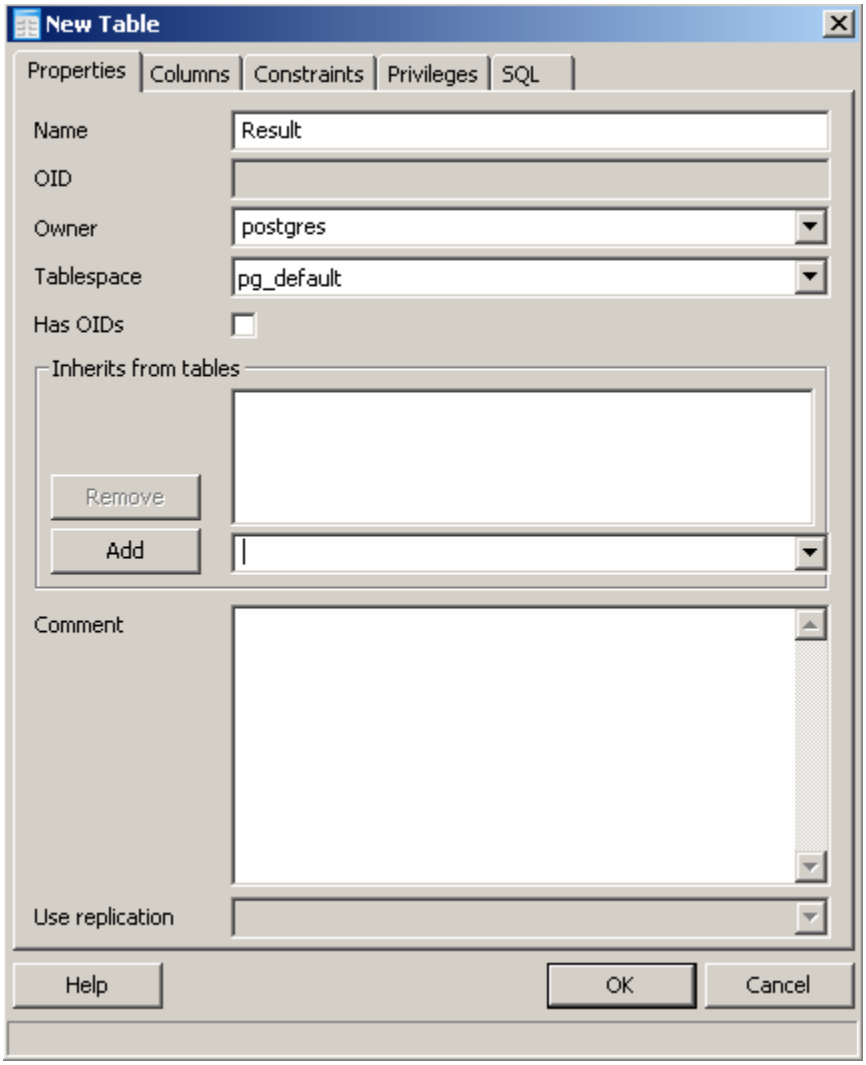


Figure 5

6. On the “Columns” tab, create 5 columns, adding one at a time. (If you make errors while creating the columns, it is much easier to edit them by returning to the “Explorer” view, shown in Figure 4.)

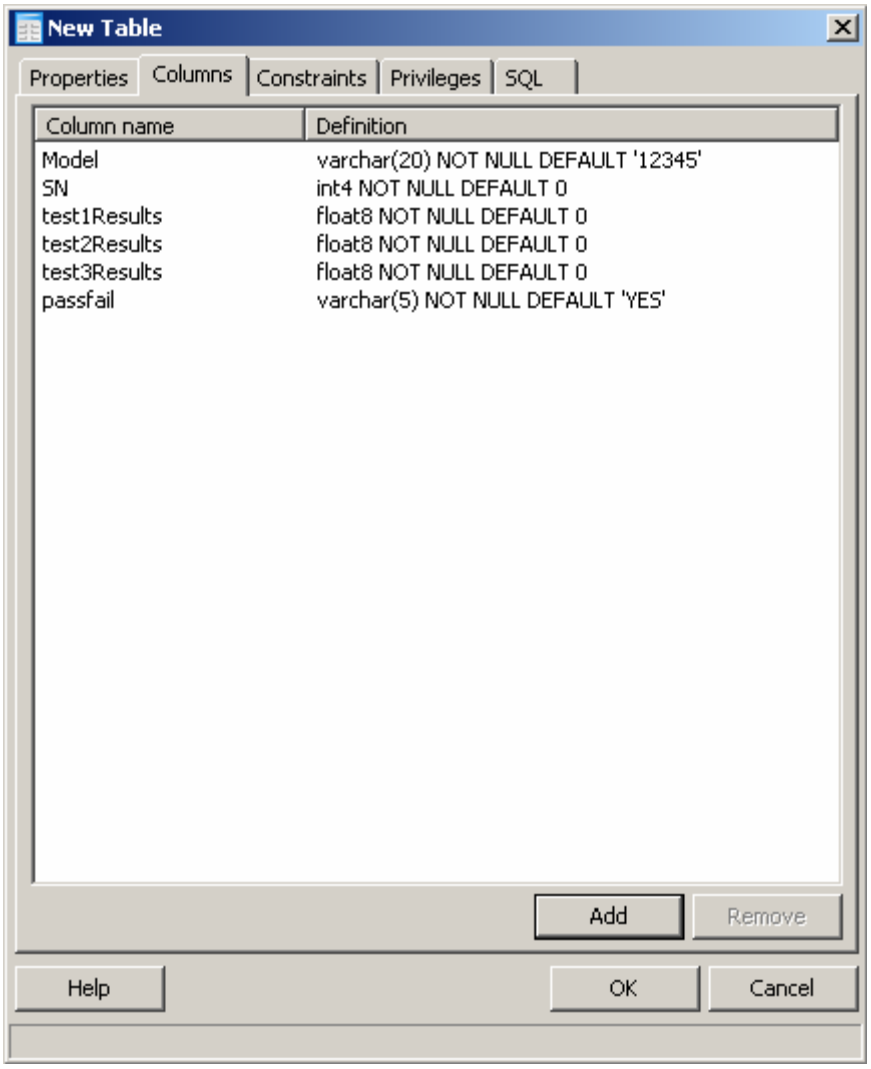


Figure 6

7. Before using the VEE Pro example program (in the following sections), set the password below to *postgres* to make sure it matches the sample program value.

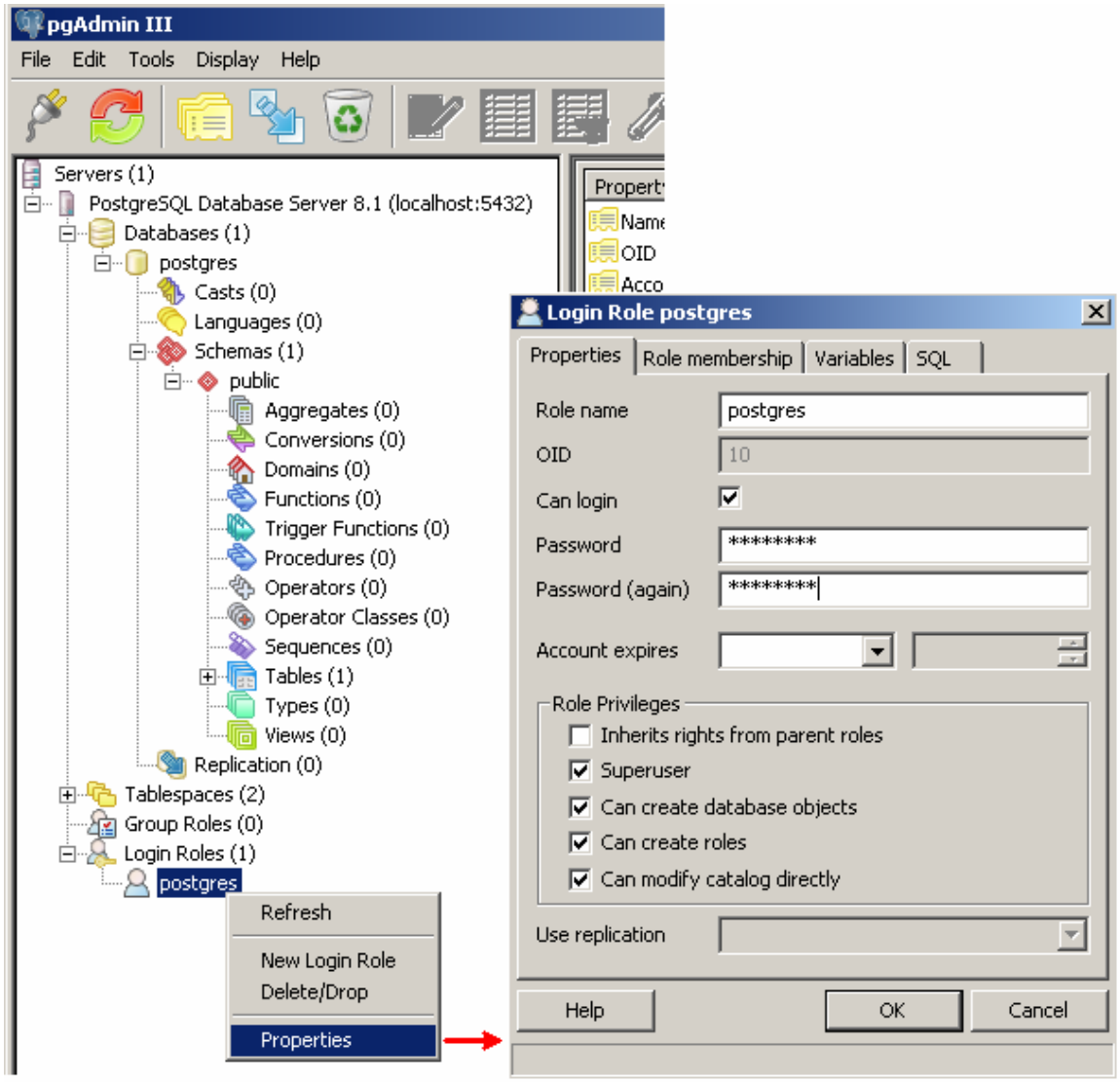


Figure 7

Once you complete step 7, your database is ready to be reference from and used in VEE Pro.

Referencing the Npgsql .NET Data Provider

We will use the Npgsql Data Provider to talk to the database. Because VEE is not "aware" of the Npgsql Data Provider yet, we first need to create a reference to its dll.

Go to VEE's Device menu and select ".NET Assembly References".

Click the "Browse…" button, go to <PostgreSQL application directory>\8.1\Npgsql\MS1.1.

Select *Mono.Security.dll* and *Npgsql.dll* one at a time. Your .NET Assembly References dialog box now should look like Figure 8. If S*ystem.Data* is not already selected, select it now.

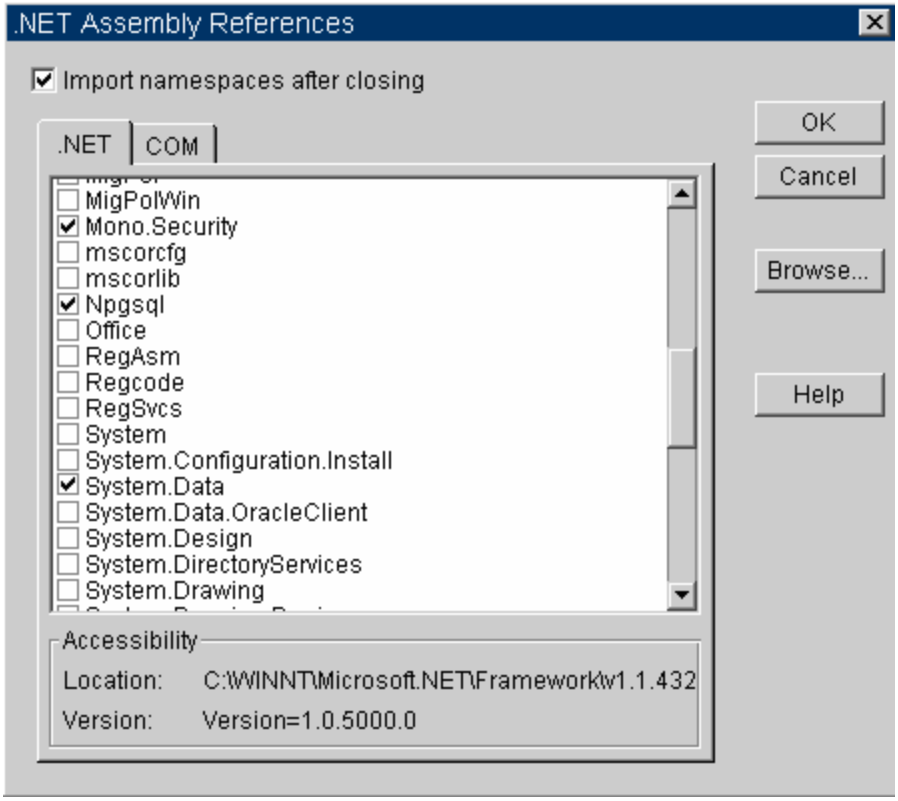


Figure 8

When you hit “OK”, you will be taken to the dialog box that looks like Figure 9. Check the boxes checked below, including System.Data.

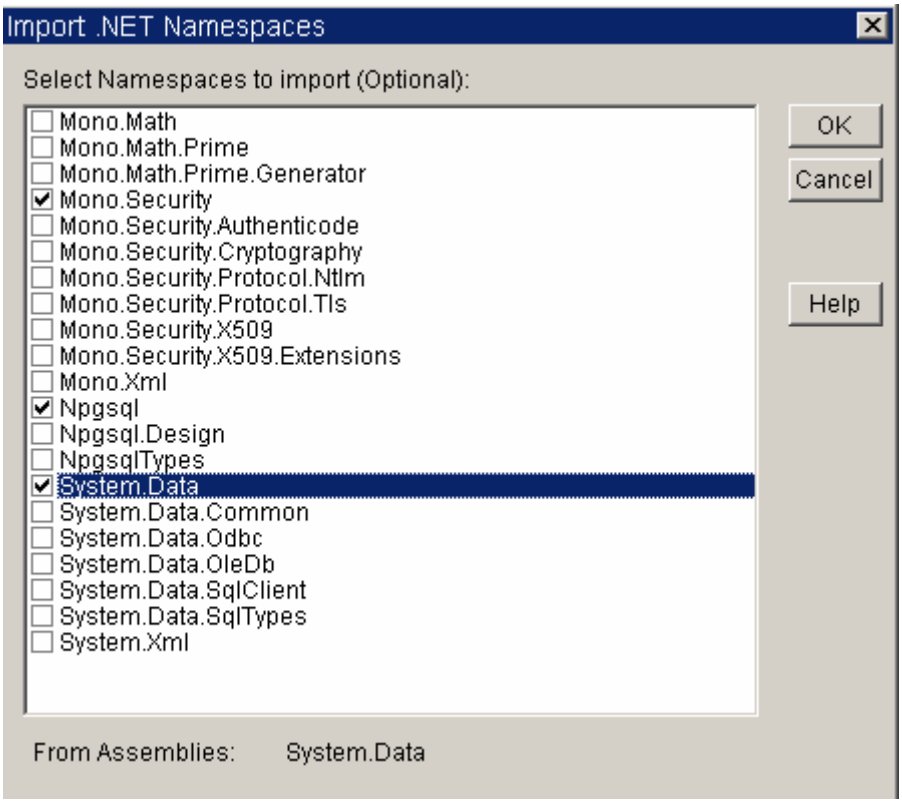


Figure 9

The postgresql_example.vee Program

We suggest starting with the accompanying postgresql_example.vee program (Figure 10) in order to familiarize yourself with the database use methodology.

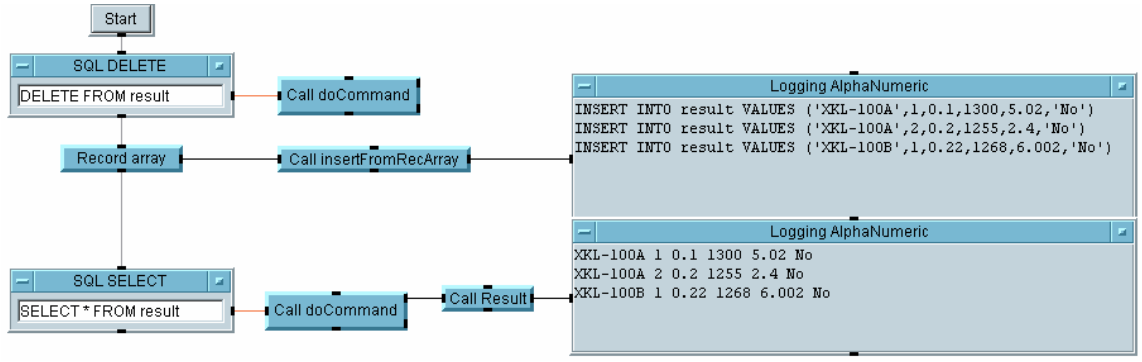


Figure 10

The program performs the following high-level steps:

1) Clears the records in the 'result' database (with the first **Call doCommand** object)
2) Creates a data record array in VEE (with the **Record array** object)
3) Deconstructs the data record array, builds SQL statements, and executes them to add data to the database (in **Call insertFromRecArray**)
4) Queries the database to extract its information (with the second **Call doCommand** object)
5) Analyzes and deconstructs the input data then formats it for display (**Call Result**)

The **Call doCommand** UserFunction is particularly important. Let's take a closer look.

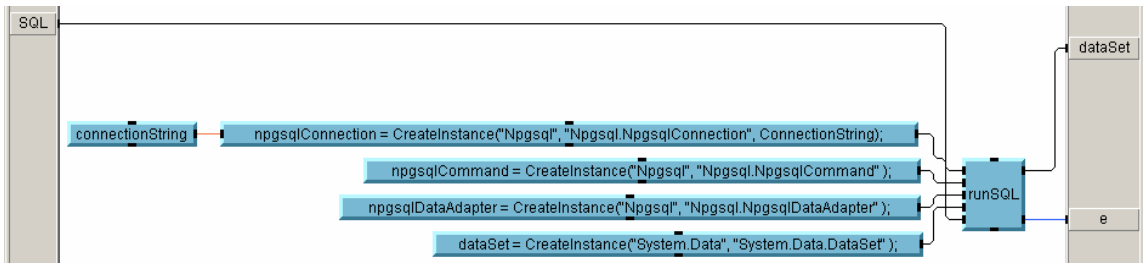


Figure 11

The **Call doCommand** UserFunction includes several steps. The first 4 are as follows:

1) Create a Connection – This accepts several parameters from **connectionString**, which indicates the host, password, etc., to create a connection to a specific data source

   You can find more information about this and the following steps at:
   `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconadonetproviders.asp`

   See "Mechanics" for information about how to add the **npgsqlConnection** object (etc) to your program.

2) Create a Command – Creates and executes a command against a data source
3) Create a DataAdapter – Populates a DataSet and resolves updates with the data source
4) Create a DataSet – Creates an instance of the DataSet object. The DataSet provides a consistent relational programming model regardless of the data source. It represents the complete set of data, including related tables and relationships between tables.

   The information on the DataSet was excerpted from MSDN:
   `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpcontheadonetdataset.asp`

After creating instances of the classes mentioned above, the UserFunction executes **runSQL**, a VEE Formula object with the following calls (also shown below):

conn.open(); → Connects to the data source identified by connectionString
command.Connection = conn; → Specifies the connection on which the command will be executed
command.CommandText = SQL; → Specifies the command string
adapter.SelectCommand = command; → Assigns the command to the DataAdapter
adapter.Fill(ds); → Executes the SQL command and puts the result into DataSet

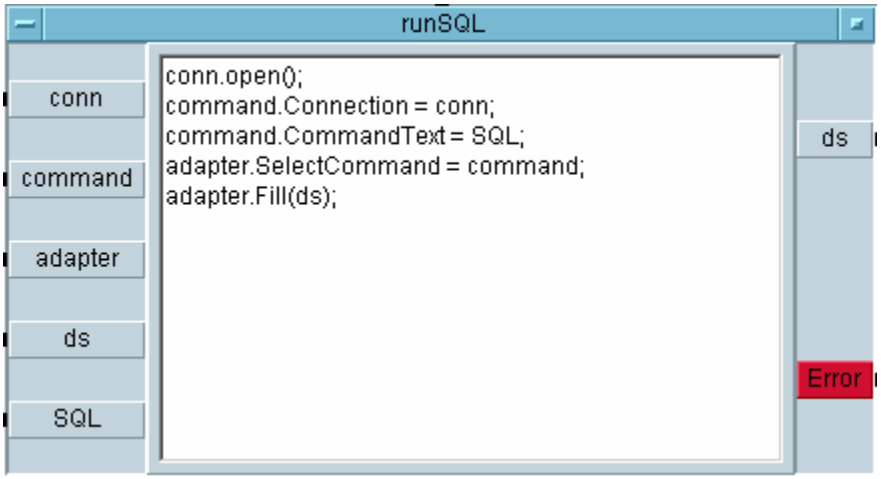


Figure 12

Now run the program to make sure everything is working properly. Then try to add new records using the **Record Array** object. This will not change the format of the data written or the data fields read, but this will confirm that you can change the data set.

When you are fairly familiar with the program's inner workings, try to modify the write/query portions. As another check of your understanding, go back to the database and add a field. Then change your VEE program's write/query portions appropriately.

Mechanics

It's important to understand how the program was constructed so that you can extend it for your own purposes or create programs from scratch. Although there are a lot of programming elements used in the sample program, we focus here on a few of the most important: the CreateInstance objects, like those of the doCommand User Function.

In order to create a connection to the database (data source), we need to instantiate the connection object. If you know the syntax of the call, you can enter it directly in a VEE Formula object. We used the direct entry approach in the runSQL Formula object above. Often – particularly when you are just getting started with using a particular object model or library – you will need to browse for the right function (or member). In VEE, we do that through the Function & Object Browser (Device > Function & Object Browser), shown below.

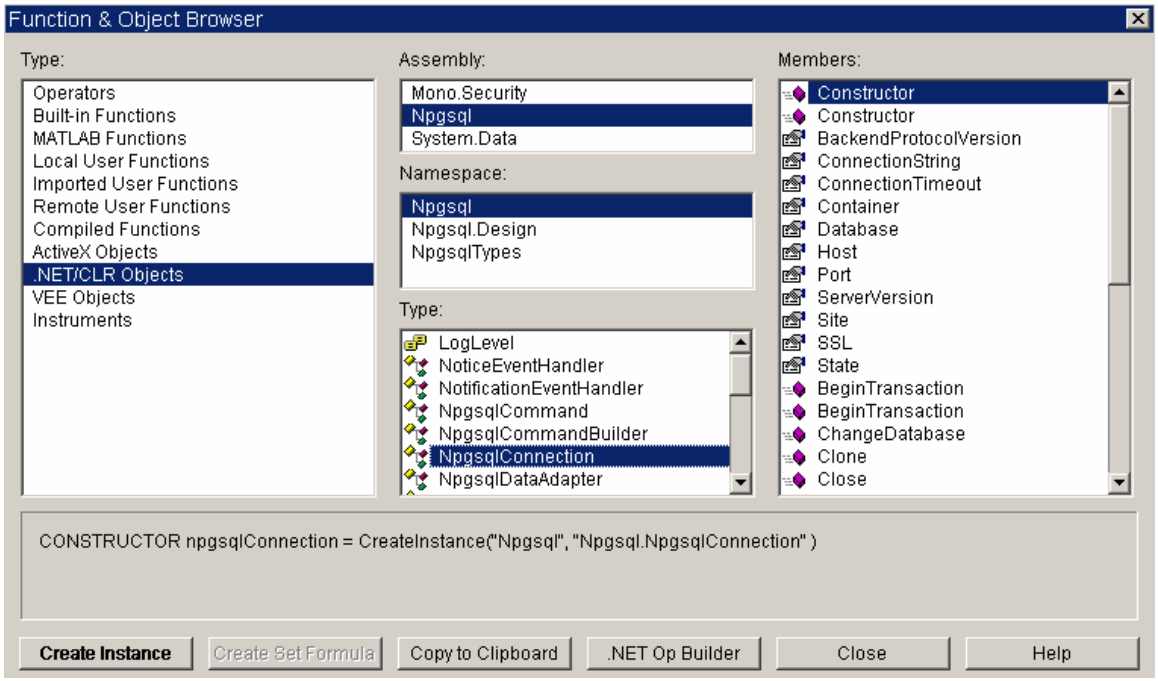


Figure 13

Here you can see that we've selected the .NET/CLR Objects as the Type. (This is not the same as a .NET Type. Think of the 'Type' here as the category.) We then chose the Npgsql Assembly (essentially a supercharged .dll) and Npgsql Namespace (a grouping of .NET Types). The .NET Type that we need here is NpgsqlConnection, and we use a Constructor to create the instance.

Additional Information

To facilitate programming with .NET, VEE 7.5 introduced the .NET Operation Builder. You can create a functionally equivalent call to the CreateInstance constructor by using the .NET Operation Builder. Go to Device > .NET Operation Builder. Once you've selected it, you'll see a list of .NET assembly references. Then you can select the appropriate .NET Type.

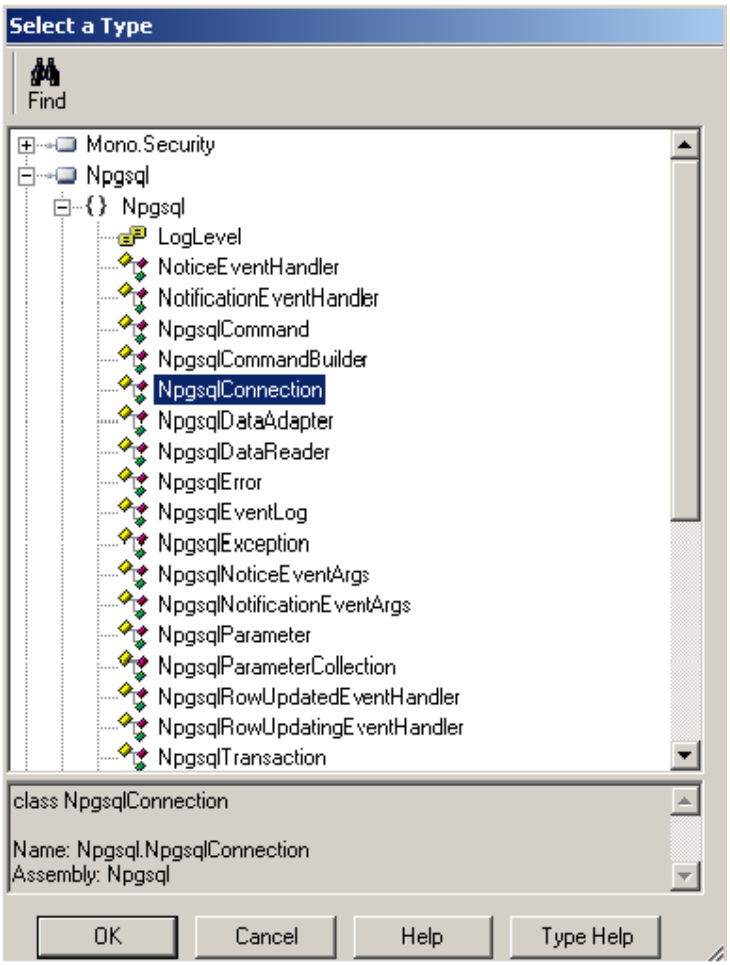


Figure 14

VEE then creates a transaction boxed linked to that Type. You can add transactions as shown in Figure 15. (Note: We've clicked the "Create Only" button in order to show the constructors.)

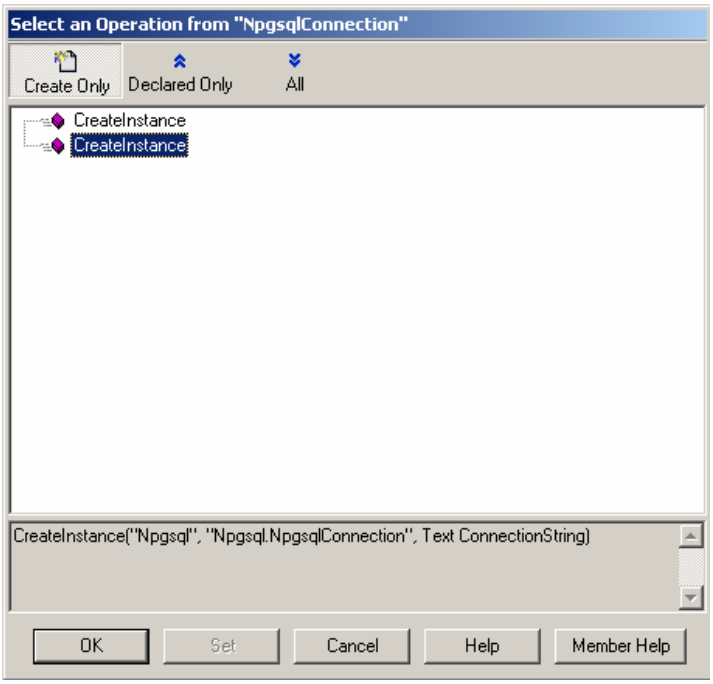


Figure 15

Whether you use the Function & Object Browser or the .NET Operation Builder, sometimes you will see multiple Members with the same name. This is due to a member's supporting optional parameters. VEE lists the member in its base-case (i.e. without optional parameters) and in each case with optional parameters.

If you need additional information on using the Function & Object Browser or the .NET Operation Builder, refer to VEE's online help.

Summary

Databases can be vital to test systems, but they often are expensive or require complicated or expensive programming interfaces. The tools introduced in this paper do not eliminate the inherent challenges of designing and administering databases, but they do provide an inexpensive, standard, and relatively intuitive means for using databases with VEE programs.

www.agilent.com

Agilent Technologies' Test and Measurement Support, Services, and Assistance
Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

Our Promise
Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you receive your new Agilent equipment, we can help verify that it works properly and help with initial product operation.

Your Advantage
Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and on-site education and training, as well as design, system integration, project management, and other professional engineering services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

Agilent Direct

www.agilent.com/find/agilentdirect
Quickly choose and use your test equipment solutions with confidence.

For more information on Agilent Technologies' products, applications or services, please contact your local Agilent office. The complete list is available at:

www.agilent.com/find/contactus

Phone or Fax
United States:
(tel) 800 829 4444
(fax) 800 829 4433

Canada:
(tel) 877 894 4414
(fax) 800 746 4866

China:
(tel) 800 810 0189
(fax) 800 820 2816

Europe:
(tel) 31 20 547 2111

Japan:
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Korea:
(tel) (080) 769 0800
(fax) (080) 769 0900

Latin America:
(tel) (305) 269 7500

Taiwan:
(tel) 0800 047 866
(fax) 0800 286 331

Other Asia Pacific Countries:
(tel) (65) 6375 8100
(fax) (65) 6755 0042
Email: tm_ap@agilent.com

Printed in USA, March 6, 2006
5989-4913EN

Agilent Technologies