VISIGENIC INTERBASE DRIVER HELP

Welcome to the online help for the Visigenic ODBC InterBase Driver. You can find information by selecting a topic area, or by choosing a driver to view a list of related topics. Go to the Dialog Boxes section at the bottom of the screen to resolve interface questions.

Look here to find out information about:

- ODBC DriverSet Basics
- Setting Up a Driver
- Adding and Modifying Data Sources
- Connecting to a Data Source
- Connection String Attributes
- ODBC Conformance Levels
- Mapping Data Types
- Scalar Functions with ODBC

Dialog Boxes

ODBC InterBase Setup Dialog
ODBC InterBase Connect Dialog

For Help on Help, Press F1

SCALAR FUNCTIONS

SETTING UP THE INTERBASE DRIVER

Look below to discover setup and system requirement information for the InterBase driver.

Although the InterBase driver was installed during the installation procedure, you must confirm that your system has the appropriate software installed.

For this product to work properly you must have the InterBase Server installed and running.

CONNECTION STRING ATTRIBUTES FOR THE INTERBASE DRIVER

Look below to learn the connection string attributes for the InterBase driver.

Some applications may require a connection string that specifies data source connection information instead of using a dialog box to obtain this information. The connection string is made up of a number of attributes that specify how a driver connects to a data source. An attribute identifies a specific piece of information that the driver needs to know before it can make the appropriate data source connection. Each driver may have a different set of attributes but the connection string format is always the same. A connection string has the following format: (Click where the hand appears for more information)

"DSN = data-source-name ■
[;HostName = host-name]
[;Database = database-name]
[;NetworkProt='TCP/IP'|'Novell SPX'|'NetBEUI']
[;NoHoldCursors=0|1]
[;UsesOwners=0|1] [;UsesUDFs=0|1]
[;UID = value] [;PWD = value]"
■ = required

You must specify the data-source-name. However, all other attributes are optional. If you do not specify an attribute, that attribute defaults to the one that is specified in the data source specifications section (for the data source specified in the connection string) of your ODBC.INI file.

For example, a connection string that connects to the Windows stores data source would have the following attributes in the connection string:

"DSN=Stores;Database=stores"

In this example, the database requested is stores.

In addition, a connection string that connects to the stores data source using the mickey server, could have the following attributes in the connection string:

"DSN=stores;HostName=mickey;Service=sqlexec; Database=stores;NetworkProt=TCP/IP;
Driver=C:\APPS\INTERBASE;NoHoldCursors=0;TrueCharPrecision=0;
TxnDMLOnly=0;UID=kerri;PWD=secret"

In this example, the database requested is stores and the user ID is kerri.

NOTE: if a HostName is used it must be accompanied by a Network Protocol (NetworkProt) in order to work properly. Otherwise it will assume the server is local.

DATA SOURCE NAME

This name is listed in the ODBC Data Sources section, and has its own data source specifications section in the ODBC.INI file.

NETWORKPROT

The specification for the network protocol used on remote connection. Must be either 'TCP/IP', 'Novell SPX', or 'NetBEUI'. All other entries (i.e. '<local>', blank, etc. . .) are treated as local.

DATABASE NAME

The name of the INTERBASE database that contains the tables you want to access.

HOSTNAME NAME

The name of the machine that you want to access.

NOHOLDCURSORS

Set this option to 1 to disable hold cursors. Setting this option to 1 can improve performance. Set this option to 0 if you need cursors enabled to maintain your cursor positioning within a result set between connections.

USESOWNERS

Specifies whether the driver allows the usage of owners in the catalog functions. This should normally be turned off in order to be compliant with programs that try to create SQL using the owner.tablename format. This format is not supported by InterBase, and turning the ownername information on will confuse certain applications.

USESUDFS

If this options is 1 then the GetInfo calls will report that all SQL functions (i.e. Numeric Functions, String Functions, Time/Date Functions) are available for use. This options does not impliment the functions in the DBMS, it only forces the driver to report that they are available. To make the functions work properly they must be created by the user and added on to the DBMS as UDFs (User Defined Functions).

UID VALUE

The host user ID.

PWD VALUE

The user password for the host.

ODBC CONFORMANCE LEVELS FOR THE INTERBASE DRIVER

Look below to discover the ODBC conformance levels supported by the InterBase driver. See ODBC Conformance Levels for general information on the ODBC API and SQL conformance levels.

Supported API Functions
The InterBase driver supports all Core and Level 1 API functions, and several options for Level 1 functions. The driver also supports several Level 2 functions.

SQL Conformance Level
The InterBase driver supports the Minimum SQL grammar and Core SQL grammar. The InterBase driver supports the following ODBC extensions to SQL:

- DATE, TIME, and TIMESTAMP data
- Outer joins
- UDF (User Defined Functions) Support
- The following string functions:
  CONCAT LENGTH
- The USER system function.

Click HERE for more information on the numeric, date, string, and system functions.

OPTIONS FOR LEVEL 1 API FUNCTIONS WITH THE INTERBASE DRIVER

The driver supports the following options for the SQLGetConnectOption() and SQLSetConnectOption() Level 1 functions:

- SQL_AUTOCOMMIT
- SQL_QUIET_MODE
- SQL_TRANSLATE_DLL
- SQL_TRANSLATE_OPTION
- SQL_TXN_ISOLATION

The driver supports the following options for the SQLSetStmtOption() and SQLGetStmtOption() Level 1 functions:

- SQL_MAX_ROWS
- SQL_NOSCAN

SUPPORTED LEVEL 2 FUNCTIONS FOR THE INTERBASE DRIVER

SQLBrowseConnect()	SQLNumParams()
SQLColumnPrivileges()	SQLPrimaryKeys()
SQLDataSources()*	SQLProcedureColumns()
SQLDrivers()*	SQLProcedures()
SQLExtendedFetch()*	SQLSetPos()*
SQLForeignKeys()	SQLSetScrollOptions()*
SQLNativeSql()	SQLTablePrivileges()

** Supported through the Cursor Library or Driver Manager*

USER DEFINED FUNCTIONS

User Defined Functions are a feature of InterBase. They allow the DBMS users to create and use any function they desire for the database. InterBase by itself does not support many of the Numeric,,String,, and Time/Date Functions supported by the SQL Standard. However through the utilization of UDFs these functions can be implemented and added on as features for usage in the InterBase DBMS.

MAPPING DATA TYPES FOR THE INTERBASE DRIVER

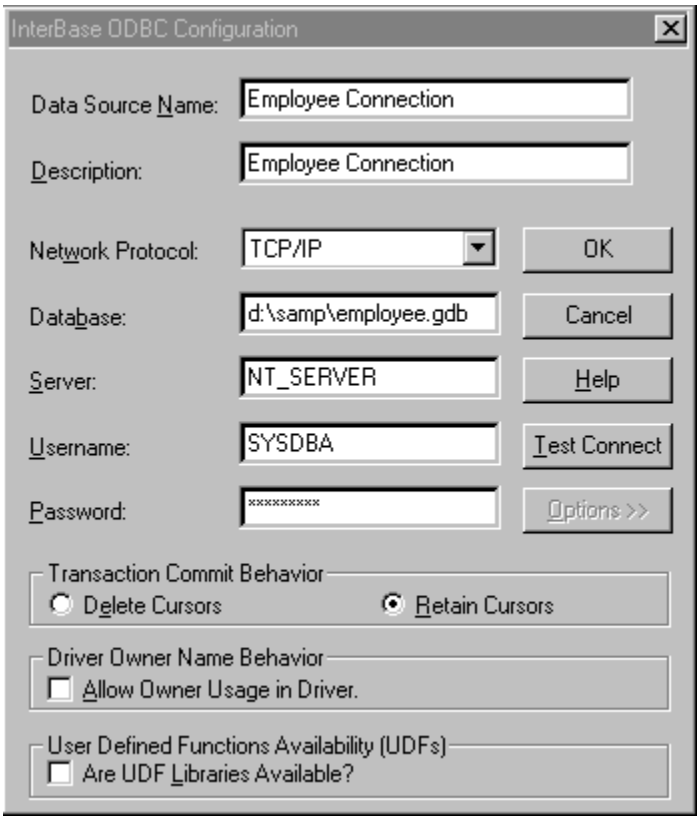
Look below to learn how ODBC data types map to the InterBase driver.

The INTERBASE database support a set of data types. The InterBase driver maps these data types to their appropriate ODBC SQL data types. The following table lists the InterBase data type and its corresponding ODBC SQL data type.

InterBase Type	ODBC SQL Data Type
BLOB	SQL_LONGVARBINARY
BLOB SUB_TYPE 1	SQL_LONGVARCHAR
CHAR	SQL_CHAR
DATE	SQL_TIMESTAMP
DOUBLE PRECISION	SQL_DOUBLE
FLOAT	SQL_REAL
INTEGER	SQL_INTEGER
NUMERIC	SQL_NUMERIC
SMALLINT	SQL_SMALLLINT
VARCHAR	SQL_VARCHAR

ODBC InterBase Setup Dialog Box

Use this dialog box to set up data sources for the InterBase driver. Windows users should always use the ODBC Administrator to add and configure data sources as the ODBC.INI file and Registry entries should never be modified. Click below where the hand appears to find out information about specific areas of the dialog box.



DATA SOURCE NAME

Enter the name of the data source. The data source name is defined by you---that is, it can be any name that you choose.

DESCRIPTION

Enter a description of the data source. This is an optional field that describes the database driver which connects to the data source. The data source description is defined by you---that is, it can be any string that you choose.

NETWORK PROTOCOL

The specification for the network protocol used on remote connection. Must be either 'TCP/IP', 'Novell SPX', 'NetBEUI', or '<local>'.

DATABASE

The name of the InterBase database that contains the tables you want to access.

SERVER

The name of the Server the Database is on.

NOTE: If the Network Protocol is '<local>' then this field will be disabled and the server name will be empty. Otherwise the field may be modified.

USERNAME

Enter your user name. Your user name is the host user ID.

PASSWORD

The Password used for connection. This information is only for testing the connection, and is not stored in the registry.

TEST CONNECTION

This is for testing a connection through the ODBC SQLBrowseConnect Function. This button will be disabled until all of the needed information is entered into the dialog box.

When the user presses 'Test Connection' a dialog will prompt the user to see if the current DSN information can be saved. If 'OK' is chosen, Test connection will proceed by first saving the information to the registry and then testing the connection. If 'Cancel' is chosen, then Test connection performs no further actions and the user is returned to the Setup Dialog.

NOTE: The test connection button will remained disabled for the duration of a setup any time the Data Source Name has changed, or the setup is for a new DSN.

TRANSACTION COMMIT BEHAVIOR

This radio button selection section specifies whether the InterBase ODBC Driver will retain cursors when SQLTransact is called with the commit option, or Delete Cursors for the same call. This option only effects the behavior of commit and does not effect the behavior of rollback. Rollback is always performed as 'Delete Cursors' Option.

DRIVER OWNER NAME BEHAVIOR

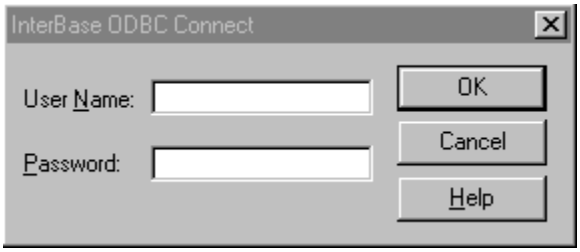
Specifies whether the driver allows the usage of owners in the catalog functions. This should normally be turned off in order to be compliant with programs that try to create SQL using the owner.tablename format. This format is not supported by InterBase, and turning the ownername information on will confuse certain applications.

USER DEFINED FUNCTIONS AVAILABILITY (UDFS)

If this options is on then the GetInfo calls will report that all SQL functions (i.e. Numeric Functions, String Functions, Time/Date Functions) are available for use. This options does not impliment the functions in the DBMS, it only forces the driver to report that they are available. To make the functions work properly they must be created by the user and added on to the DBMS as UDFs (User Defined Functions).

ODBC INTERBASE CONNECT DIALOG BOX

Use this dialog box to connect to an InterBase data source. Click below where the hand appears to find out information about specific areas of the dialog box.



USER NAME

Enter your user name for the host to which you want to connect. The user name is your host user ID.

PASSWORD

Enter your password. This is the password for the host to which you want to connect.

VISIGENIC ODBC DRIVERSET BASICS CONTENTS

This help file contains basic information about the Visigenic ODBC DriverSet.

For Help on Help, Press F1

WHAT IS THE ODBC DRIVERSET?

Look below for basic information about the Visigenic ODBC DriverSet.

ODBC stands for Open Database Connectivity, an interface that allows applications to access data in database management systems (DBMSs) using Structured Query Language (SQL) as a standard for accessing data. The ODBC specification defines a vendor-independent API for accessing data stored in relational and non-relational databases. The Core functions and SQL grammar are based on work done by the X/Open SQL Access Group. The ODBC architecture is made up of four components:

- Database Application. The database application calls functions defined in the ODBC API to access a data source.
- Driver Manager. The Driver Manager implements the ODBC API and provides information to an application (such as a list of available data sources and drivers), loads drivers dynamically as they are needed, and provides argument and state transition checking.
- Drivers. Each driver processes ODBC function calls and manages exchanges between an application and a data source.
- Data Source. A data source contains the data that an application needs to access. The data source includes the data, the database management system (DBMS) in which the data is stored, the platform on which the DBMS resides, and the network (if any) used to access the DBMS.

With the Visigenic DriverSet, the cursor capabilities are implemented using the ODBC Cursor Library. The Cursor Library is a shared library that resides between the Driver Manager and the driver.

An ODBC-compliant driver allows you to communicate between an ODBC-compliant application and a DBMS. For example, the Oracle7 driver allows you to connect your ODBC-compliant application to an Oracle7 database.

The Visigenic ODBC DriverSet is made up of two ODBC components: the Driver Manager and a set of database drivers. With the ODBC DriverSet, you can access, query, and update data in a number of different databases.

UNDERSTANDING THE ODBC.INI FILE

Look below for general information about the ODBC.INI file. **WARNING:** Windows users should never modify the ODBC.INI file (Windows 3.1) or Registry entries (Windows NT and Windows 95) directly. These entries are changed based on the data source set up and modifications you make using the ODBC Administrator. Modifying the ODBC.INI file or Registry directly may result in data source configuration or connection errors

The ODBC.INI file is an initialization file used by the ODBC Driver Manager and ODBC drivers. The file is located in your WINDOWS directory. Although the ODBC.INI file is described in the following sections, Windows users should not modify this file---the ODBC Administrator program modifies it for you. The following sections are intended for informational purposes only.

Windows users should never modify the ODBC.INI file directly. The contents of this file are changed based on the data source setup and modifications you make using the ODBC Administrator. Modifying the ODBC.INI file directly may result in data source configuration errors.

The ODBC.INI file is made up of the following sections:

- ODBC Data Sources. This section lists the name of each data source and describes its associated driver.
- Data Source Specification. For each data source listed in the ODBC Data Sources section, there is a section that contains additional information about that data source.
- Default Data Source Specification. This section is optional and specifies the default data source to use when no data source is specified.
- ODBC Options. This section specifies the ODBC root directory and the ODBC options that may be enabled or disabled.

ODBC DATA SOURCES

Look below for details on the ODBC Data Sources section of the ODBC.INI file. Click on areas of the syntax below for further information.

Each entry in the ODBC Data Sources section lists a data source and a description of the driver it uses. This section has the following format: (Click where the hand appears for more information)

[ODBC Data Sources]
data-source-name ▲ = driver-description
▲ = required

For example, to define an Agencies data source that uses the Oracle7 driver, the ODBC.INI entry would look like the following:

[ODBC Data Sources]
Agencies=Visigenic Oracle7 Driver

DATA SOURCE NAME

The data source name identifies the data source to which the driver connects. You choose this name.

DRIVER DESCRIPTION

The driver description describes the driver to which the data source connects. This field is optional.

DATA SOURCE SPECIFICATION

Look below for details on the Data Source Specification section of the ODBC.INI file. Click on areas of the syntax below for further information.

Each data source listed in the ODBC Data Sources section has its own data source specification section. This section has the following format: (Click where the hand appears for more information)

_____

[data-source-name]
DRIVER = driver-path ▲
KEYWORD = attribute ▲
▲ = required

_____

For example, the data source called Agencies connects to a Sybase SQL Server 10 driver called VSSYB.DLL. The database that Agencies accesses is also called agencies and it resides on the SYBASE10 server. The data source specification entry for the Agencies data source would look like the following:

[Agencies]
Driver=C:\WINDOWS\SYSTEM\VSSYB.DLL
Server=SYBASE10
Database=agencies
UID=marvin
AllowQuotedIdentifiers=1

In this example, the driver-specific keywords for the Sybase SQL Server 10 driver are Server, Database, UID, and AllowQuotedIdentifiers.

DATA SOURCE NAME

The data source name is the name of the data source, as specified in the ODBC Data Sources section of the ODBC.INI file.

DRIVER PATH

The driver path is the dynamic link library.

ATTRIBUTE

The attribute specifies the value for the keyword. Each driver has its own set of keywords.

DEFAULT DATA SOURCE SPECIFICATION

Look below for details on the optional Default Data Source Specification section of the ODBC.INI file.

This section is optional. The Default Data Source specification contains information about the default data source. This data source is called Default and has the same format as any other data source specification section. However, the Default data source is not listed in the ODBC Data Sources section.

For example, the following sample shows a Default data source specification entry for an Oracle database.

```
[Default]
Driver=C:\WINDOWS\SYSTEM\VSORAC.DLL
Server=t:mickey:customers
UID=marvin
```

In this example, the driver-specific keywords for the Oracle7 driver are Server and UID. The Server keyword identifies the SQL*Net connect string for the ORACLE7 server called customers.

ODBC OPTIONS

Look below for details on the ODBC Options section of the ODBC.INI file.

The ODBC Options section indicates whether tracing is enabled or disabled. With tracing, all ODBC function calls made from an application can be logged to the specified trace file. This section has the following format (Click where the hand appears for more information):

```
[ODBC]
Trace = 0|1
TraceFile = tracefile-path
TraceAutoStop = 0|1
```

For example, the following sample shows an entry for the [ODBC] section.

```
[ODBC]
Trace=1
TraceFile=C:\WINDOWS\LOG\TRACE.LOG
TraceAutoStop=0
```

TRACE

Indicates whether tracing is enabled. If the Trace keyword is set to 0, tracing is disabled. If the Trace keyword is set to 1, tracing is enabled.

TRACEFILE

The name of the specified trace file that is logging the ODBC function calls. The tracefile-path specifies the full path to the trace file. If a trace file is not specified and tracing is enabled, logging information is written to the SQL.LOG file located in your current directory.

TRACEAUTOSTOP

Indicates whether tracing is enabled or disabled when an application calls the SQLFreeEnv() function. If the TraceAutoStop keyword is set to 0, tracing is not automatically disabled. If the TraceAutoStop keyword is set to 1, the Trace keyword in the ODBC.INI file is set to 0. On Windows, all other concurrent ODBC applications will have tracing disabled.

ODBC CONFORMANCE LEVELS

Look below to learn the ODBC conformance levels for ODBC drivers.

ODBC defines two different conformance standards for drivers: the API conformance standard and the SQL conformance standard. Each conformance standard is made up of three levels. These levels help application and driver developers establish standard sets of functionality.

API CONFORMANCE LEVELS

Look below for ODBC API conformance levels and API functions.

The API conformance standard is made up of three levels:

- **Core API.** A set of core functions that correspond to the functions in the X/Open SQL Access Group Call Level Interface specification.
- **Level 1 API.** Core API functionality plus all Level 1 functionality.
- **Level 2 API.** Core and Level 1 API functionality plus all Level 2 functionality.

ODBC API Functions
The Visigenic drivers support all Core and Level 1 functions. In addition, each driver supports a key set of the Level 2 functions. For a list of supported Level 2 functions by driver, refer to the appropriate driver's ODBC Conformance Levels section. Click on a level to view the API functions defined to the level.

Core Level Level 1 Level 2

CORE LEVEL API FUNCTIONS

The following are Core API functions:

SQLAllocConnect()	SQLExecute()
SQLAllocEnv()	SQLFetch()
SQLAllocStmt()	SQLFreeConnect()
SQLBindCol()	SQLFreeEnv()
SQLCancel()	SQLFreeStmt()
SQLColAttributes()	SQLGetCursorName()
SQLConnect()	SQLNumResultCols()
SQLDescribeCol()	SQLPrepare()
SQLDisconnect()	SQLRowCount()
SQLError()	SQLSetCursorName()
SQLExecDirect()	SQLTransact()

LEVEL 1 API FUNCTIONS

The following are Level 1 API functions:

SQLBindParameter()
SQLColumns()
SQLDriverConnect()
SQLGetConnectOption()
SQLGetData()
SQLGetInfo()
SQLGetFunctions()
SQLGetStmtOption()
SQLGetTypeInfo()
SQLParamData()
SQLPutData()
SQLSetConnectOption()
SQLSetStmtOption()
SQLSpecialColumns ()
SQLStatistics()
SQLTables()

LEVEL 2 API FUNCTIONS

The following are Level 2 API functions:

SQLBrowseConnect()
SQLColumnPrivileges()
SQLDataSources()
SQLDescribeParam()
SQLDrivers()
SQLExtendedFetch()
SQLForeignKeys()
SQLMoreResults()
SQLNativeSql()
SQLNumParams()
SQLParamOptions()
SQLPrimaryKeys()
SQLProcedureColumns()
SQLProcedures()
SQLSetPos()
SQLSetScrollOptions()
SQLTablePrivileges()

SQL CONFORMANCE LEVELS

Look below to learn the general SQL conformance levels for ODBC.

The SQL conformance standard is made up of three levels:

- **Minimum Conformance.** This level is designed to meet basic ODBC conformance.
- **Core Conformance.** This level roughly corresponds to the X/Open SQL Access Group SQL CAE specification (1995).
- **Extended Conformance.** This level provides common DBMS extensions to SQL.

ODBC SQL Grammar
The Visigenic drivers support Minimum and Core SQL grammar. In addition, each driver supports a number of extended SQL statements, expressions, and data types. For a list of supported Extended SQL grammar by driver, refer to the appropriate driver's ODBC Conformance Levels section. Click on a level for further information.

Minimum Core Extended

MINIMUM SQL GRAMMAR

The Minimum level of SQL grammar consists of the following statements, expressions, and data types:

- Data Definition Language (DDL): CREATE TABLE and DROP TABLE.
- Data Manipulation Language (DML): simple SELECT, INSERT, UPDATE, SEARCHED, and DELETE SEARCHED.
- Expressions: simple (such as A>B+C).
- Data types: CHAR, VARCHAR, or LONG VARCHAR.

CORE SQL GRAMMAR

The Core level of SQL grammar consists of the following statements, expressions, and data types:

- Minimum SQL grammar and data types.
- Data Definition Language (DDL): ALTER TABLE, CREATE INDEX, DROP INDEX, CREATE VIEW, DROP VIEW, GRANT, and REVOKE.
- Data Manipulation Language (DML): full SELECT.
- Expressions: subquery, set functions such as SUM and MIN.
- Data Types: DECIMAL, NUMERIC, SMALLINT, INTEGER, REAL, FLOAT, DOUBLE PRECISION.

EXTENDED SQL GRAMMAR

The Extended level of SQL grammar consists of the following statements, expressions, and data types:

- Minimum and Core SQL grammar and data types.
- Data Manipulation Language (DML): outer joins, positioned UPDATE, positioned DELETE, SELECT FOR UPDATE, and unions.
- Expressions: scalar functions such as SUBSTRING, ABS, DATE, TIME, and TIMESTAMP literals.
- Data types: BIT, TINYINT, BIGINT, BINARY, VARBINARY, LONG VARBINARY, DATE, TIME, TIMESTAMP.
- Batch SQL statements.
- Procedure calls.

ERROR MESSAGES

Look below for an explanation of the structure of ODBC error messages.

Since ODBC is based on a layered architecture, an error message must not only explain the error but also identify the component in which the error occurred. ODBC error messages have two different formats: one format for errors that occur in a data source, and one format for errors that occur in any other ODBC component. ODBC error messages use the format shown below. Note that blue squares indicate items that appear when the error is from a data source, and red squares indicate items that appear when the error is from another ODBC component. Unmarked items appear in both instances. (Click where the hand appears for more information.)

[vendor-identifier] [ODBC-component-identifier]
[data-source-identifier] ▲
data-source-text , ▲
component-text ■

▲ = Error message occurred in the data source
■ = Error message occurred in a component

For example, an error message from a data source might look like the following:

"[Visigenic][ODBC InterBase Driver][InterBase] User's password is not correct for the database server."

Or an error message from a Visigenic driver might look like the following:

"[Visigenic][ODBC ORACLE7 Server Driver]Duplicate cursor name"

VENDOR IDENTIFIER

The vendor of the component in which the error occurred, or that received the error directly from the data source.

ODBC COMPONENT IDENTIFIER

The component in which the error occurred, or that received the error directly from the data source.

DATA SOURCE IDENTIFIER

The data source in which the error occurred. This only appears when the error occurred in the data source.

DATA SOURCE TEXT

Text generated by the data source. This only appears when the error occurs in the data source.

COMPONENT TEXT

Text generated by the ODBC component. This only appears when the error occurs in a component.

ACCESSING TRANSLATION LIBRARIES

Look below for information about how to access translation libraries with Visigenic drivers.

Visigenic drivers are able to call a translation library (on Windows, a Translation Dynamic Link Library). Visigenic drivers can call a translation library using the SQL_TRANSLATE_DLL option for the SQLSetConnectOption() function.

A translation library is used when the client application and the data source store data in different formats. For example, the application might use a different character set than the data source. The translation library enables a driver to translate all data (such as data values, SQL statements, table names, row counts) that pass between the driver and the data source.

You can specify which translation library to use when you configure a data source. The translation library that you specify must be one that you supply---the Visigenic DriverSet does not provide a translation library.

The translation library appears as an option at the bottom of the Setup dialog box. This option appears in the same place on every Setup dialog box.

ADDING AND MODIFYING DATA SOURCES

Look here to discover how to add and modify data sources for the Visigenic ODBC drivers.

A data source definition identifies the location of data that may include a network library, server, database, and other attributes. In order to connect to a data source, the Driver Manager looks at your ODBC.INI file for specific connection information.

The ODBC.INI file is an initialization file used by the Driver Manager and ODBC drivers. This file contains information about each data source and its associated driver. Generally, before you can connect to a data source, its connection information must be added to this file.

You add and configure data sources using the ODBC Administrator. The ODBC Administrator then updates your ODBC.INI file, located in the WINDOWS directory, to reflect your data source connection information. As you add data sources, the ODBC Administrator adds the information to this file for you; that is, you should NEVER modify the ODBC.INI file directly.

Click HERE for steps to add a data source.
Click HERE for steps to modify a data source.

STEPS FOR ADDING A DATA SOURCE

Look below to learn general steps for adding a data source.

1. To start the ODBC Administrator, double-click the ODBC Administrator icon in the Windows Control Panel. A Data Sources dialog box appears.

   NOTE: You can also add data sources when you install the DriverSet. In either case, the steps that you follow are the same.

2. When you see the Data Sources dialog box, click the Add button. The Add Data Source dialog box appears.

3. Select a driver, and then click OK. The Visigenic ODBC Setup dialog box for your driver appears. You will find this dialog box listed on the Contents page of your driver help file.

4. Fill in the fields of the dialog box as appropriate, and click OK to add this data source. When you click OK, the Data Sources dialog box appears.

After you click OK, the ODBC Administrator updates your ODBC.INI file. The values that you enter become the default data source connection values for this data source. That is, when you connect to the data source using either a dialog box or connection string, these values become the default entries for the data source connection.

STEPS FOR MODIFYING A DATA SOURCE

Look below to discover general steps for modifying a data source.

1. Invoke the ODBC Administrator. The Data Sources dialog box appears.
2. In the Data Sources dialog box, select the data source you want to modify and then click the Setup button. The Visigenic ODBC Setup dialog box for the driver you selected appears.
3. Modify the applicable data source fields, and then click OK.

When you have finished modifying the information in this dialog box, the ODBC Administrator updates your ODBC.INI file.

CONNECTING TO A DATA SOURCE

Look below for instructions on connecting to a data source.

An ODBC application can pass connection information in a number of ways. For example, the application may have the driver always prompt the user for connection information. Or the application may expect a connection string that specifies the data source connection. How you connect to a data source depends on the connection method that your ODBC application uses.
One common way of connecting to a data source is through the SQL Data Sources dialog box. That dialog box prompts you for the appropriate data source connection information as shown below.

1. When you see the SQL Data Sources dialog box, select a data source and then click OK. The ODBC Connect dialog box for the appropriate driver appears. You will find this dialog box listed on the Contents page of your driver help file.
2. Fill in the appropriate information for the dialog box as indicated, and then click OK.

Once the connection information is verified, your application can access the information that the data source contains using the appropriate driver.

STRING FUNCTIONS

Click on a function below to find out information about it. The functions begin after you see "Click on a function for more information."

The following are string manipulation functions. Character string literals used as arguments to scalar functions must be bounded by single quotes.

Arguments denoted as *string_exp* can be the name of a column, a string literal, or the result of another scalar function, where the underlying data type can be represented as SQL_CHAR, SQL_VARCHAR, or SQL_LONGVARCHAR.

Arguments denoted as *start*, *length*, *code*, or *count* can be a numeric literal or the result of another scalar function, where the underlying data type can be represented as SQL_TINYINT, SQL_SMALLINT, or SQL_INTEGER.

The string functions listed here are 1-based, that is, the first character in the string is character 1.

Click on a function for more information:

ASCII(*STRING_EXP*)

Returns the ASCII code value of the leftmost character of *string_exp* as an integer.

CHAR(*CODE*)

Returns the character that has the ASCII code value specified by code. The value of code should be between 0 and 255; otherwise, the return value is data source-dependent.

CONCAT(*STRING_EXP1*, *STRING_EXP2*)

Returns a character string that is the result of concatenating *string_exp2* to *string_exp1*. The resulting string is DBMS dependent. For example, if the column represented by *string_exp1* contained a NULL value, DB2 would return NULL, but SQL Server would return the non-NULL string.

DIFFERENCE(*STRING_EXP1*, *STRING_EXP2*)

Returns an integer value that indicates the difference between the values returned by the SOUNDEX function for *string_exp1* and *string_exp2*.

INSERT(*STRING_EXP1*, *START*, *LENGTH*, *STRING_EXP2*)

Returns a character string where *length* characters have been deleted from *string_exp1* beginning at *start* and where *string_exp2* has been inserted into *string_exp1*, beginning at *start*.

LCASE(*STRING_EXP*)

Converts all upper case characters in *string_exp* to lower case.

LEFT(*STRING_EXP*, *COUNT*)

Returns the leftmost *count* of characters of *string_exp*.

LENGTH(*STRING_EXP*)

Returns the number of characters in *string_exp*, excluding trailing blanks and the string termination character.

LOCATE(*STRING_EXP1*, *STRING_EXP2* [*START*])

Returns the starting position of the first occurrence of *string_exp1* within *string_exp2*. The search for the first occurrence of *string_exp1* begins with the first character position in *string_exp2* unless the optional argument, *start*, is specified. If *start* is specified, the search begins with the character position indicated by the value of *start*. The first character position in *string_exp2* is indicated by the value 1. If *string_exp1* is not found within *string_exp2*, the value 0 is returned.

LTRIM(*STRING_EXP*)

Returns the characters of *string_exp*, with leading blanks removed.

REPEAT(*STRING_EXP*, *COUNT*)

Returns a character string composed of *string_exp* repeated *count* times.

REPLACE(*STRING_EXP1*, *STRING_EXP2*, *STRING_EXP3*)

Replaces all occurrences of *string_exp2* in *string_exp1* with *string_exp3*.

RIGHT(*STRING_EXP*, *COUNT*)

Returns the rightmost *count* of characters of *string_exp*.

RTRIM(*STRING_EXP*)

Returns the characters of *string_exp* with trailing blanks removed.

SOUNDEX(*STRING_EXP*)

Returns a data source-dependent character string representing the sound of the words in *string_exp*. For example, SQL Server returns a four digit SOUNDEX code; Oracle returns a phonetic representation of each word.

SPACE(*COUNT*)

Returns a character string consisting of *count* spaces.

SUBSTRING(*STRING_EXP*, *START*, *LENGTH*)

Returns a character string that is derived from *string_exp* beginning at the character position specified by *start* for *length* characters.

UCASE(*STRING_EXP*)

Converts all lower case characters in *string_exp* to upper case.

NUMERIC FUNCTIONS

Click on a function below to find out information about it. The functions begin after you see "Click on a function for more information."

The following are numeric functions that are included in the ODBC scalar function set.

Arguments denoted as *numeric_exp* can be the name of a column, the result of another scalar function, or a numeric literal, where the underlying data type could be represented as SQL_NUMERIC, SQL_DECIMAL, SQL_TINYINT, SQL_SMALLINT, SQL_INTEGER, SQL_BIGINT, SQL_FLOAT, SQL_REAL, or SQL_DOUBLE.

Arguments denoted as *float_exp* can be the name of a column, the result of another scalar function, or a numeric literal, where the underlying data type can be represented as SQL_FLOAT.

Arguments denoted as *integer_exp* can be the name of a column, the result of another scalar function, or a numeric literal, where the underlying data type can be represented as SQL_TINYINT, SQL_SMALLINT, SQL_INTEGER, or SQL_BIGINT.

Click on a function for more information:

ABS(*NUMERIC_EXP*)

Returns the absolute value of *numeric_exp*.

ACOS(*FLOAT_EXP*)

Returns the arccosine of *float_exp* as an angle, expressed in radians.

ASIN(*float_exp*)

Returns the arcsine of *float_exp* as an angle, expressed in radians.

ATAN(*FLOAT_EXP*)

Returns the arctangent of *float_exp* as an angle, expressed in radians.

ATAN2(*FLOAT_EXP1*, *FLOAT_EXP2*)

Returns the arctangent of the x and y coordinates, specified by *float_exp1* and *float_exp2*, respectively, as an angle, expressed in radians.

CEILING(*NUMERIC_EXP*)

Returns the smallest integer greater than or equal to *numeric_exp*.

COS(*FLOAT_EXP*)

Returns the cosine of *float_exp*, where *float_exp* is an angle expressed in radians.

COT(*FLOAT_EXP*)

Returns the cotangent of *float_exp*, where *float_exp* is an angle expressed in radians.

DEGREES(*NUMERIC_EXP*)

Returns the number of degrees converted from *numeric_exp* radians.

EXP(*FLOAT_EXP*)

Returns the exponential value of *float_exp*.

FLOOR(*NUMERIC_EXP*)

Returns largest integer less than or equal to *numeric_exp*.

LOG(*FLOAT_EXP*)

Returns the natural logarithm of *float_exp*.

LOG10(*FLOAT_EXP*)

Returns the base 10 logarithm of *float_exp*.

MOD(*INTEGER_EXP1*, *INTEGER_EXP2*)

Returns the remainder (modulus) of *integer_exp1* divided by *integer_exp2*.

PI( )

Returns the constant value of *pi* as a floating point value.

POWER(*NUMERIC_EXP*, *INTEGER_EXP*)

Returns the value of *numeric_exp* to the power of *integer_exp*.

RADIANS(*NUMERIC_EXP*)

Returns the number of radians converted from *numeric_exp* degrees.

RAND([*INTEGER_EXP*])

Returns a random floating point value using *integer_exp* as the optional seed value.

ROUND(*NUMERIC_EXP*, *INTEGER_EXP*)

Returns *numeric_exp* rounded to *integer_exp* places right of the decimal point. If *integer_exp* is negative, *numeric_exp* is rounded to |*integer_exp*| places to the left of the decimal point.

SIGN(*NUMERIC_EXP*)

Returns an indicator or the sign of *numeric_exp*. If *numeric_exp* is less than zero, –1 is returned. If *numeric_exp* equals zero, 0 is returned. If *numeric_exp* is greater than zero, 1 is returned.

SIN(*FLOAT_EXP*)

Returns the sine of *float_exp*, where *float_exp* is an angle expressed in radians.

SQRT(*FLOAT_EXP*)

Returns the square root of *float_exp*.

TAN(*FLOAT_EXP*)

Returns the tangent of *float_exp*, where *float_exp* is an angle expressed in radians.

TRUNCATE(*NUMERIC_EXP*, *INTEGER_EXP*)

Returns *numeric_exp* truncated to *integer_exp* places right of the decimal point. If *integer_exp* is negative, *numeric_exp* is truncated to |*integer_exp*| places to the left of the decimal point.

TIME AND DATE FUNCTIONS

Click on a function below to find out information about it. The functions begin after you see "Click on a function for more information."

The following are time and date functions that are included in the ODBC scalar function set. Date, time, and timestamp literals must be bounded by single quotes.

Arguments denoted as *timestamp_exp* can be the name of a column, the result of another scalar function, or a time, date, or timestamp literal, where the underlying data type could be represented as SQL_CHAR, SQL_VARCHAR, SQL_TIME, SQL_DATE, or SQL_TIMESTAMP.

Arguments denoted as *date_exp* can be the name of a column, the result of another scalar function, or a date or timestamp literal, where the underlying data type could be represented as SQL_CHAR, SQL_VARCHAR, SQL_DATE, or SQL_TIMESTAMP.

Arguments denoted as *time_exp* can be the name of a column, the result of another scalar function, or a time or timestamp literal, where the underlying data type could be represented as SQL_CHAR, SQL_VARCHAR, SQL_TIME, or SQL_TIMESTAMP.

Values returned are represented as ODBC data types.

Click on a function for more information:

CURDATE( )

Returns the current date as a date value.

CURTIME( )

Returns the current local time as a time value.

DAYNAME(*DATE_EXP*)

Returns a character string containing the data source-specific name of the day (for example, Sunday, through Saturday or Sun. through Sat. for a data source that uses English, or Sonntag through Samstag for a data source that uses German) for the day portion of *date_exp*.

DAYOFMONTH(*DATE_EXP*)

Returns the day of the month in *date_exp* as an integer value in the range of 1 to 31.

DAYOFWEEK(*DATE_EXP*)

Returns the day to the week in *date_exp* as an integer value in the range of 1 to 7, where 1 represents Sunday.

DAYOFYEAR(*DATE_EXP*)

Returns the day of the year in *date_exp* as an integer value in the range of 1 to 366.

HOUR(*TIME_EXP*)

Returns the hour in *time_exp* as an integer value in the range of 0 to 23.

MINUTE(*TIME_EXP*)

Returns the minute in *time_exp* as an integer value in the range of 0 to 59.

MONTH(*DATE_EXP*)

Returns the month in *date_exp* as an integer value in the range of 1 to 12.

MONTHNAME(*DATE_EXP*)

Returns a character string containing the data-source-specific name of the month (for example, January through December or Jan. through Dec. for a data source that uses English, or Januar through Dezember for a data source that uses German) for the month portion of *date_exp*.

NOW( )

Returns current date and time as a timestamp value.

QUARTER(*DATE_EXP*)

Returns the quarter in *date_exp* as an integer value in the range of 1 to 4, where 1 represents January 1 through March 31.

SECOND(*TIME_EXP*)

Returns the second in *time_exp* as an integer value in the range of 0 to 59.

TIMESTAMPADD(*INTERVAL*, *INTEGER_EXP*, *TIMESTAMP_EXP*)

Returns the timestamp calculated by adding *integer_exp* intervals of type interval to *timestamp_exp*. Valid values of interval are the following keywords where fractional seconds are expressed in billionths of a second:

- SQL_TSI_FRAC_SECOND
- SQL_TSI_SECOND
- SQL_TSI_MINUTE
- SQL_TSI_HOUR
- SQL_TSI_DAY
- SQL_TSI_WEEK
- SQL_TSI_MONTH
- SQL_TSI_QUARTER
- SQL_TSI_YEAR

For example, the following SQL statement returns the name of each employee and their one-year anniversary dates:

```
SELECT NAME,
{fn TIMESTAMPADD(SQL_TSI_YEAR,1, HIRE_DATE)} FROM EMPLOYEES
```

If *timestamp_exp* is a time value and interval specifies days, weeks, months, quarters, or years, the date portion of *timestamp_exp* is set to the current date before calculating the resulting timestamp.

If *timestamp_exp* is a date value and interval specifies fractional seconds, seconds, minutes, or hours, the time portion of timestamp_exp is set to 0 before calculating the resulting timestamp.
An application determines which intervals a data source supports by calling SQLGetInfo() with the SQL_TIMEDATE_ADD_INTERVALS option.

TIMESTAMPDIFF(*INTERVAL*, *TIMESTAMP_EXP1*, *TIMESTAMP_EXP2*)

Returns the integer number of intervals of type interval by which *timestamp_exp2* is greater than *timestamp_exp1*. Valid values of interval are the following keywords where fractional seconds are expressed in billionths of a second:

- SQL_TSI_FRAC_SECOND
- SQL_TSI_SECOND
- SQL_TSI_MINUTE
- SQL_TSI_HOUR
- SQL_TSI_DAY
- SQL_TSI_WEEK
- SQL_TSI_MONTH
- SQL_TSI_QUARTER
- SQL_TSI_YEAR

For example, the following SQL statement returns the name of each employee and the number of years they have been employed.

```
SELECT NAME,
{fn TIMESTAMPDIFF(SQL_TSI_YEAR,{fn CURDATE()}, HIRE_DATE)} FROM EMPLOYEES
```

If either timestamp expression is a time value and interval specifies days, weeks, months, quarters, or years, the date portion of that timestamp is set to the current date before calculating the difference between the timestamps.

If either timestamp expression is a date value and interval specifies fractional seconds, seconds, minutes, or hours, the time portion of that timestamp is set to 0 before calculating the difference between the timestamps.

An application determines which intervals a data source supports by calling SQLGetInfo() with the SQL_TIMEDATE_DIFF_INTERVALS option.

WEEK(*DATE_EXP*)

Returns the week of the year in *date_exp* as an integer value in the range of 1 to 53.

YEAR(*DATE_EXP*)

Returns the year in *date_exp* as an integer value. The range is data source dependent.

SYSTEM FUNCTIONS

Click on a function below to find out information about it. The functions begin after you see "Click on a function for more information."

The following are system functions that are included in the ODBC scalar function set.

Arguments denoted as exp can be the name of a column, the result of another scalar function, or a literal, where the underlying data type could be represented as SQL_NUMERIC, SQL_DECIMAL, SQL_TINYINT, SQL_SMALLINT, SQL_INTEGER, SQL_BIGINT, SQL_FLOAT, SQL_REAL, SQL_DOUBLE, SQL_DATE, SQL_TIME, or SQL_TIMESTAMP.

Arguments denoted as value can be a literal constant, where the underlying data type can be represented as SQL_NUMERIC, SQL_DECIMAL, SQL_TINYINT, SQL_SMALLINT, SQL_INTEGER, SQL_BIGINT, SQL_FLOAT, SQL_REAL, SQL_DOUBLE, SQL_DATE, SQL_TIME, or SQL_TIMESTAMP.

Values returned are represented as ODBC data types.

Click on a function for more information:

DATABASE( )

Returns the name of the database corresponding to the connection handle (hdbc). (The name of the database is also available by calling SQLGetConnectOption() with the SQL_CURRENT_QUALIFIER connection option.)

IFNULL(*EXP*, *VALUE*)

If *exp* is null, *value* is returned. If *exp* is not null, *exp* is returned. The possible data type(s) of *value* must be compatible with the data type of *exp*.

USER( )

Returns the user's authorization name. (The user's authorization name is also available via SQLGetInfo() by specifying the information type: SQL_USER_NAME.)

EXPLICIT DATA TYPE CONVERSION

Look below for information about data type conversion across the Visigenic DriverSet.

Explicit data type conversion is specified in terms of ODBC SQL data type definitions.

The ODBC syntax for the explicit data type conversion function does not restrict conversions. The validity of specific conversions of one data type to another data type will be determined by each driver-specific implementation. The driver will, as it translates the ODBC syntax into the native syntax, reject those conversions that, although legal in the ODBC syntax, are not supported by the data source. The ODBC function SQLGetInfo() provides a way to inquire about conversions supported by the data source.

The format of the CONVERT() function is:

CONVERT(*value_exp*, *data_type*)

The function returns the value specified by *value_exp* converted to the specified *data_type*, where *data_type* is one of several keywords.

The ODBC syntax for the explicit data type conversion function does not support specification of conversion format. If specification of explicit formats is supported by the underlying data source, a driver must specify a default value or implement format specification.

The argument *value_exp* can be a column name, the result of another scalar function, or a numeric or string literal.

For example, the following example converts the output of the CURDATE() scalar function to a character string.

{ fn CONVERT({ fn CURDATE() }, SQL_CHAR) }

MORE EXAMPLES

DATA TYPE KEYWORDS

The keywords can be any of the following:

SQL_BIGINT	SQL_BINARY
SQL_BIT	SQL_CHAR
SQL_DATE	SQL_DECIMAL
SQL_DOUBLE	SQL_FLOAT
SQL_INTEGER	SQL_LONGVARBINARY
SQL_LONGVARCHAR	SQL_REAL
SQL_SMALLINT	SQL_TIME
SQL_TIMESTAMP	SQL_TINYINT
SQL_VARBINARY	SQL_VARCHAR

EXAMPLES USING THE CONVERT()FUNCTION

Look below for more examples on using the Convert() function.

The following two examples illustrate the use of the CONVERT() function. These examples assume the existence of a table called employees, with an empno column of type SQL_SMALLINT and an empname column of type SQL_CHAR.

Example #1
If an application specifies the following:

```
SELECT empno FROM employees WHERE
--(*vendor(Microsoft),product(ODBC)
    fn CONVERT(empno,SQL_CHAR)*)--
LIKE '1%'
```

or its equivalent in shorthand form:

```
SELECT empno FROM employees WHERE
{fn CONVERT(empno,SQL_CHAR)} LIKE '1%'
```

A driver that supports an ORACLE DBMS would translate the request to:

```
SELECT empno FROM employees
 WHERE to_char(empno) LIKE '1%'
```

A driver that supports a Sybase SQL Server DBMS would translate the request to:

```
SELECT empno FROM employees
WHERE convert(char,empno) LIKE '1%'
```

Example #2
If an application specifies the following:

```
SELECT
--(*vendor(Microsoft),product(ODBC)
    fn ABS(empno)*)--,
--(*vendor(Microsoft),product(ODBC)
    fn CONVERT(empname,SQL_SMALLINT)*)--
FROM employees WHERE empno <> 0
```

or its equivalent in shorthand form:

```
SELECT {fn ABS(empno)},
{fn CONVERT(empname,SQL_SMALLINT)}
FROM employees WHERE empno <> 0
```

A driver that supports an Oracle DBMS would translate the request to:

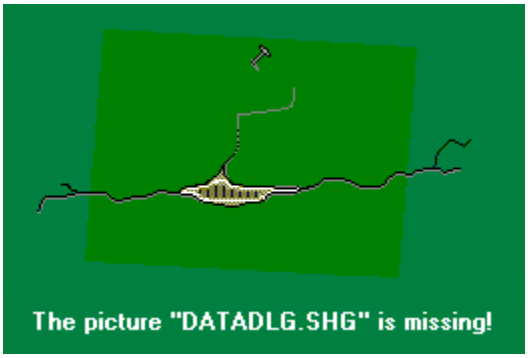
```
SELECT abs(empno), to_number(empname)
FROM employees
WHERE empno <> 0
```

A driver that supports a Sybase SQL Server DBMS would translate the request to:

```
SELECT abs(empno), convert(smallint, empname)
FROM employees
WHERE empno != 0
```

DATA SOURCES DIALOG BOX

Use this dialog to add, modify, or delete data sources. Click below where the hand appears to find out information about specific areas of the dialog box.



SETUP…

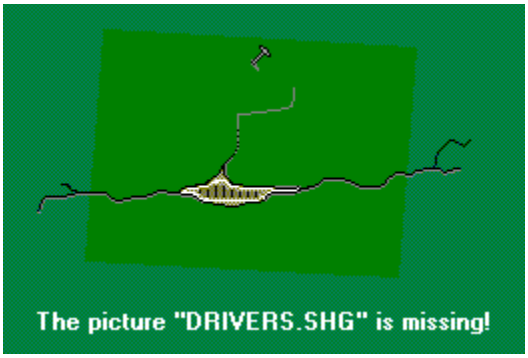
Click this button to view the Setup dialog box for the data source you selected. In the Setup dialog box, you can create or modify a data source.

ADD...

Click this button to add a data source to the list. You will see the Add Data Source dialog box.

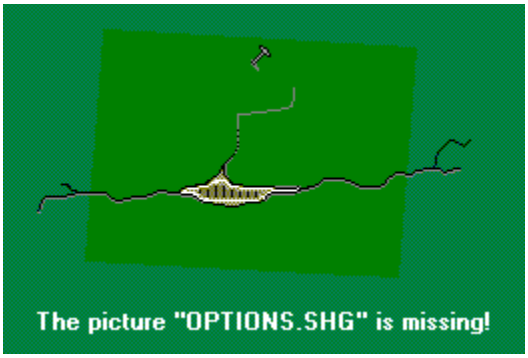
DRIVERS…

You see this dialog when you click the Drivers... button in the Data Sources dialog box. From this dialog, you may choose to add or delete drivers from your system.



OPTIONS…

You see this dialog when you click on the Options... button in the Data Sources dialog box. From this dialog, you may choose to enable or disable tracing.



ADD DATA SOURCE DIALOG BOX

Use this dialog box to select a driver for which you want to add a data source. When you click OK, the ODBC Setup dialog box for your driver appears where you can set up your data source.