Go To

DataDirect ODBC Drivers Reference
Table of Contents

Go To

List of Tables

Go To

Using this Manual

This manual, your reference to the INTERSOLV DataDirect ODBC drivers, accompanies the INTERSOLV DataDirect products.

This manual contains the following chapters:

- Chapter 1, “Getting Started,” on page 1 explains the DataDirect drivers and ODBC, discusses environment-specific subjects, and explains the error messages returned by the drivers.
- A chapter for each database driver. Each driver‘s chapter is structured so that the information you need to reference first is at the beginning of the chapter. First, it lists system requirements for your operating environment. Next, it explains how to configure a data source. Finally, it explains how to connect to that data source.

This manual also includes six appendixes that provide information on technical topics:

- Appendix A, “SQL for Flat-File Drivers,” on page 299 explains the SQL statements that you can use with Btrieve, dBASE, Excel, Paradox, and text files.
- Appendix B, “Using Indexes,” on page 325 provides general guidelines on how to improve performance when querying a database system.
- Appendix C, “ODBC API and Scalar Functions,” on page 332 lists the ODBC API functions that each driver supports. Any exceptions are listed in each driver chapter, in the section “ODBC Conformance Level.” This appendix also lists the ODBC scalar functions.

- Appendix D, “Locking and Isolation Levels,” on page 342 provides a general discussion of isolation levels and locking.
- Appendix E, “ODBC.INI,” on page 346 explains the structure of ODBC.INI as defined by the ODBC specification.
- Appendix F, “Designing Performance-Oriented ODBC Applications,” on page 354 provides guidelines for designing performance-oriented ODBC applications.

If you are writing programs to access ODBC drivers, you need to obtain a copy of the *ODBC Programmer’s Reference* for the Microsoft Open Database Connectivity Software Development Kit, available from Microsoft Corporation.

Conventions Used in This Manual

This manual uses various conventions to aid in its usability. The typography, terminology, and callouts to environment-specific information used are intended to make this manual easy to use, regardless of the operating environment you are using. The following sections describe these conventions.

Typography

This manual uses various typefaces, fonts, and characters to indicate certain types of information, as follows:

Convention	Explanation
italics	Used to introduce new terms that you may not be familiar with, and occasionally for emphasis.
bold	Used to emphasize important information.

Convention	Explanation
UPPERCASE	Indicates the name of a file. For operating environments that use case-sensitive filenames, the correct capitalization is used in information specific to those environments.
`monospace`	Syntax examples, values that you specify, or results that you receive.
`monospaced` *`italics`*	Names that are placeholders for values you specify; for example, *`filename`*.
vertical rule |	Indicates an OR separator to delineate items.
brackets []	Indicate optional items; for example, `SELECT [DISTINCT ]`. In this example, `DISTINCT` is an optional keyword.
braces {}	Indicate that you must select one item. For example, `{yes | no }`. In this example, you must specify either yes or no.

This manual uses the following terminology:

- The term *ODBC.INI* refers to the ODBC.INI file format as defined by the Microsoft ODBC specification. Appendix E, “ODBC.INI,” on page 346 discusses how ODBC.INI is implemented in all supported environments.
- The suffix .DLL refers to a dynamic link library file. Your operating system may use the term shared object or shared library files instead.

Environment-Specific Information

This manual supports users of various operating environments. Wherever it provides information that is not applicable to all supported environments, the following symbols are used to identify that information:

Symbol	Environment
	Windows. Information specific to the Microsoft Windows environment is identified by this symbol, which is a trademark of Microsoft Corporation.
95	**Windows 95**. Information specific to the Microsoft Windows 95 environment is identified by the Windows symbol and the numbers “95.”
NT	**Windows NT**. Information specific to the Microsoft Windows NT environment is identified by the Windows symbol and the letters “NT.”
OS/2	**OS/2.** Information specific to the OS/2 environment is identified by this symbol. OS/2 is a registered trademark of IBM Corporation.
	Macintosh. Information specific to Macintosh users is identified by this symbol, which is a registered trademark of Apple Computer, Inc.
UNIX	**UNIX.** Information specific to UNIX environments is identified by this symbol, which applies to all UNIX environments supported. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

This manual shows dialog boxes that are specific to Windows. If you are using the drivers on Windows 95, Windows NT, OS/2, and Macintosh, the dialog box you see may differ slightly from the Windows version. The UNIX environment does not currently support a graphical user interface.

Go To

1 Getting Started

This chapter contains the following sections:

- About INTERSOLV DataDirect ODBC Drivers
- Environment-Specific Information
- Error Messages

About INTERSOLV DataDirect ODBC Drivers

DataDirect database drivers and ODBC

The INTERSOLV DataDirect ODBC drivers are compliant with the Microsoft® Open Database Connectivity (ODBC) specification. ODBC is a specification for an application program interface (API) that enables applications to access multiple database management systems using Structured Query Language (SQL).

ODBC permits maximum interoperability—a single application can access many different database management systems. This enables an ODBC developer to develop, compile, and ship an application without targeting a specific type of data source. Users can then add the database drivers, which link the application to the database management systems of their choice.

INTERSOLV provides ODBC drivers for both relational and flat-file database systems. The flat-file drivers provide full SQL support as discussed in Appendix A, “SQL for Flat-File Drivers,” on page 299.

Support for Multiple Environments

DataDirect drivers offer multi-platform support

INTERSOLV provides ODBC-compliant database drivers on Windows, Windows 95, Windows NT, OS/2, Macintosh, and the following UNIX platforms: Solaris for SPARC, Solaris for x86, HP-UX, and AIX.

The next section “Database Systems Supported” lists the drivers that are supported under each operating environment.

Note: Database drivers are continually being added to each operating environment. See the README file shipped with your INTERSOLV product for an up-to-date list of drivers and for the most current driver information.

Network protocol requirements vary, depending on the database drivers you use. To connect to a relational database system, you need the appropriate network software from the database vendor. See the “System Requirements” section in the appropriate driver chapter for additional information.

The section “Environment-Specific Information” on page 6 explains the environment-specific differences that you should be aware of when using the database drivers in your operating environment.

Database Systems Supported

INTERSOLV drivers support over 30 database systems

To access a database system, you must use the correct driver. Table 1-1 lists the database systems that INTERSOLV supports, the driver you use to access that system, and the operating environments on which each driver is available.

Table 1-1. Database Systems Supported

Note: Database drivers are continually being added to each operating environment. See the README file shipped with your INTERSOLV product for an up-to-date list of drivers and for the most current driver information.

Database System	Driver	Win	Win 95	Win NT	OS/2	UNIX*	Mac
ALLBASE/SQL	ALLBASE	Y	—	—	—	—	—
Btrieve 5 and 6.*x*	Btrieve	Y	—	—	—	—	—
Clipper files	dBASE	Y	Y	Y	Y	All	Y
DB2 (MDI gateway)	Micro-Decisionware	Y	—	—	Y	—	—
DB2 (SQLHost gateway)	SQLBase	Y	Y	Y	Y	—	—
DB2 (Sybase Net gateway)	SQL Server	Y	Y	Y	Y	All except Solaris (x86)	—
DB2/2	DB2/2	Y	—	—	Y	AIX only	—
DB2/6000	DB2/2	Y	—	—	Y	AIX only	—
DB2/MVS (DDCS/2 or DDCS/6000 gateways)	DB2/2	Y	—	—	Y	AIX only	—
dBASE II files	dBASE	Y	Y	Y	—	—	—
dBASE III, IV, V files	dBASE	Y	Y	Y	Y	Y	Y
Excel 2, 3, and 4 .XLS files	Excel 4	Y	Y	Y	—	—	Y
Excel 5 .XLS Workbook files	Excel 5	Y	Y	Y	—	—	Y
FoxBASE files	dBASE	Y	Y	Y	Y	All	Y
FoxPro files 1, 2.5	dBASE	Y	Y	Y	Y	All	Y
IMAGE/SQL	ALLBASE	Y	—	—	—	—	—

***** The UNIX platforms supported are: AIX 3.2.5, HP-UX 9 or greater, Solaris 2.x for SPARC, and Solaris 2.x for x86.

Table 1-1. Database Systems Supported (cont.)

Note: Database drivers are continually being added to each operating environment. See the README file shipped with your INTERSOLV product for an up-to-date list of drivers and for the most current driver information.

Database System	Driver	Win	Win 95	Win NT	OS/2	UNIX*	Mac
InfoHub (MDI gateway)	Micro-Decisionware	Y	—	—	Y	—	—
INFORMIX 4	INFORMIX 4 or 5	Y	—	—	—	—	Y
INFORMIX 5	INFORMIX 5	Y	Y	Y	Y	AIX and Solaris (SPARC)	—
INFORMIX 7	INFORMIX 7	—	—	—	—	All except Solaris (x86)	—
INGRES 6.4/03	INGRES	Y	—	—	—	—	—
INGRES 6.4/04 or higher	INGRES 6.4/04	Y	Y	Y	Y	AIX and Solaris (SPARC)	—
InterBase	InterBase	Y	—	—	—	—	—
Oracle 6	Oracle 6	Y	—	—	Y	All except Solaris (x86)	Y
Oracle 7	Oracle 7	Y	Y	Y	Y	All	Y
Paradox 3-4.5	Paradox 4	Y	—	—	—	—	—
Paradox 3-5	Paradox 5	Y	—	—	—	—	—
PROGRESS 6	PROGRESS	Y	—	—	—	—	—
Scalable SQL 3.*x*	Scalable SQL	Y	—	—	—	—	—
SQL Server	SQL Server	Y	Y	Y	Y	All except Solaris (x86)	Y

***** The UNIX platforms supported are: AIX 3.2.5, HP-UX 9 or greater, Solaris 2.x for SPARC, and Solaris 2.x for x86.

Table 1-1. Database Systems Supported (cont.)

Note: Database drivers are continually being added to each operating environment. See the README file shipped with your INTERSOLV product for an up-to-date list of drivers and for the most current driver information.

Database System	Driver	Win	Win 95	Win NT	OS/2	UNIX*	Mac
SQL/400 (DDCS/2 or DDCS/6000 gateways)	DB2/2	Y	—	—	Y	AIX only	—
SQL/DS (DDCS/2 or DDCS/6000 gateways)	DB2/2	Y	—	—	Y	AIX only	—
SQLBase	SQLBase	Y	Y	Y	Y	—	—
Sybase System 10	Sybase System 10	Y	Y	Y	Y	All	Y
Text files	Text	Y	Y	Y	Y	All	Y
XDB 2.41, 3.4, 4.0	XDB	Y	—	—	—	—	—

***** The UNIX platforms supported are: AIX 3.2.5, HP-UX 9 or greater, Solaris 2.x for SPARC, and Solaris 2.x for x86.

Installing the ODBC Drivers

The DataDirect ODBC drivers are installed by the Setup program for the product with which they are shipped. For instructions on running the Setup program, see the *Installation Guide* that accompanies the product.

ODBC Developer Program

Special driver access and support for ODBC developers

If you are an ODBC developer, you may be interested in INTERSOLV's ODBC Developer Program. This program provides you with

- Early access to database drivers for development and testing of your ODBC-compliant applications
- API-level technical support
- Access to our senior support representatives
- Early access to Field Test versions of our products

To learn more about the ODBC Developer Program, contact INTERSOLV's Sales Department at (800) 876-3101.

Environment-Specific Information

The following topics contain information specific to your operating environment, such as filenames, system requirements, ODBC.INI implementation, etc. Information is provided for Windows, Windows 95, Windows NT, OS/2, Macintosh, and UNIX systems.

For Windows Users

On Microsoft Windows systems, the ODBC drivers are 16-bit drivers. All required network software supplied by your database system vendors must be 16-bit compliant. Consult the "System Requirements" section of the appropriate chapter for specific requirements for each ODBC driver.

ODBC.INI

ODBC.INI is a text file located in the Windows directory. This file is maintained by the ODBC Administrator, located within the Windows Control Panel.

The structure of this file is described in Appendix E, “ODBC.INI,” on page 346.

Starting the ODBC Administrator

The section “Configuring Data Sources” in each subsequent driver chapter instructs you to start the ODBC Administrator. To start the ODBC Administrator under Windows, double-click the ODBC icon in the Windows Control Panel.

Driver Names

The prefix for all ODBC driver filenames on Windows is “QE.” The file extension is .DLL. This indicates that they are dynamic link libraries. For example, the Oracle 7 driver filename is QEOR7*nn*.DLL, where *nn* is the revision number of the driver.

See the README file shipped with your INTERSOLV DataDirect product for the filename of each driver supported on Windows.

Disk Space and Memory Requirement s

Disk space requirements are 6 MB of free disk space on the disk drive where the Windows system is installed.

Memory requirements vary, depending on the database driver. If you are using a flat-file database driver, you need at least 4 MB of memory. If your system will host a relational database system, 6 to 8 MB of memory may be required. Consult your relational database documentation to determine the exact memory requirements.

For Windows 95 and Windows NT Users

On Windows 95 and Windows NT systems, the ODBC drivers are 32-bit drivers. All required network software supplied by your database system vendors must be 32-bit compliant. Consult the “System Requirements” section for specific requirements for each relational database driver.

ODBC.INI

ODBC.INI is a subkey of the HKEY_CURRENT_USER key within the Windows 95 or Windows NT registry. When you access the registry using this subkey, the ODBC structure is the same as described in Appendix E, “ODBC.INI,” on page 346.

The ODBC.INI subkey is maintained by the ODBC Administrator, which is located in the Windows 95 or Windows NT Control Panel. Because Windows 95 and Windows NT can support multiple users, the ODBC.INI subkey is stored under unique user keys in the registry. First run the ODBC Setup program to install the ODBC.INI subkey and then configure data sources for each user.

Starting the ODBC Administrator

The section “Configuring Data Sources” in each subsequent driver chapter instructs you to start the ODBC Administrator. To start the ODBC Administrator under Windows 95 or Windows NT, double-click the ODBC icon in the Windows 95 or Windows NT Control Panel. Or, double-click the ODBC Administrator icon in the DataDirect product group in the Program Manager.

Driver Names

The prefix for all ODBC driver filenames on Windows 95 and Windows NT is “IV.” The file extension is .DLL. This indicates that they are dynamic link libraries. For example, the Oracle 7 driver filename is IVOR7*nn*.DLL, where *nn* is the revision number of the driver.

See the README file shipped with your INTERSOLV DataDirect product for the filename of each driver supported on Windows 95 and Windows NT.

You must modify the CONFIG.SYS file to add the fully qualified path to the environment variable LIBPATH. For example, if you install the ODBC drivers in the directory C:\ODBC, then the fully qualified path is C:\ODBC. This enables your applications to dynamically load the ODBC drivers at runtime.

Disk Space and Memory Requirements

Disk space requirements are 6 MB of free disk space on the disk drive where Windows 95 or Windows NT is installed.

Memory requirements vary, depending on the database driver. If you are using a flat-file database driver, you need at least 8 MB of memory on Windows 95 or at least 16 MB of memory on Windows NT. If your system is hosting a relational database system, additional memory may be required. Consult your relational database documentation to determine the exact memory requirements.

For OS/2 Users

On OS/2 systems, the ODBC drivers are 32-bit drivers. The required network software supplied by your database system vendors must be 32-bit compliant. Consult the "System Requirements" section for specific requirements for each relational database driver.

OS/2 version 2.*x* or higher is required to use the ODBC drivers.

ODBC.INI

ODBC.INI is an operating system binary INI file located in the OS/2 directory. The structure of this file is described in Appendix E, "ODBC.INI," on page 346. Since this file is binary, you cannot edit it with a text editor.

Starting the ODBC Administrator

The section "Configuring Data Sources" in each subsequent driver chapter instructs you to start the ODBC Administrator. To start the ODBC Administrator under OS/2, double-click the ODBC Administrator icon.

If you are using ODBC Pack, this icon is in the ODBC Pack folder. If you are using the INTERSOLV DataDirect Developer’s Toolkit, this icon is in the folder for that product.

Driver Names

The prefix for all ODBC driver filenames on OS/2 is “IV.” The file extension is .DLL. This indicates that they are dynamic link libraries. For example, the Oracle 7 driver filename is IVOR7*nn*.DLL, where *nn* is the revision number of the driver.

See the README file shipped with your INTERSOLV DataDirect product for the filename of each driver supported on OS/2.

You must modify the CONFIG.SYS file to add the fully qualified path to the environment variable LIBPATH. For example, if you install the ODBC drivers in the directory C:\ODBC, then the fully qualified path is C:\ODBC. This enables your applications to dynamically load the ODBC drivers at runtime.

Disk Space and Memory Requirement s

Disk space requirements are 6 MB of free disk space on the disk drive where OS/2 is installed.

Memory requirements vary, depending on the database driver. If you are using a flat-file database driver, you need at least 4 MB of memory. If your system will host a relational database, additional memory may be required. Consult your relational database documentation to determine the exact memory requirements.

For Macintosh Users

On Macintosh systems, the ODBC drivers are 32-bit drivers. They are built using the Apple Shared Library Manager and are stored in the Extensions folder. The Apple Shared Library Manager is required to use the drivers, and is not provided by INTERSOLV. To determine if the Apple Shared Library Manager is installed, look for Shared Library Manager in the Extensions folder. The operating system required is System 7 or greater.

ODBC.INI

ODBC.INI is a text file called ODBC Preferences located in the Preferences folder. Although this file is maintained by the ODBC Administrator, it can be modified, if necessary, using TeachText. The structure of ODBC Preferences is described in Appendix E, "ODBC.INI," on page 346.

Starting the ODBC Administrator

The section "Configuring Data Sources" in each subsequent driver chapter instructs you to start the ODBC Administrator. To start the ODBC Administrator, double-click the ODBC Setup Control Panel, which is located in the Control Panels folder under the System folder.

Driver Names

The database driver names are all lowercase and end with a suffix of .DLL. This indicates that they are dynamic link libraries (called *shared libraries* on the Macintosh).

See the README file shipped with your INTERSOLV DataDirect product for the filename of each driver supported on the Macintosh.

Disk Space and Memory Requirements

Disk space requirements are 6 MB of free disk space on the disk where the operating system is installed.

Memory requirements vary, depending on the database driver. If you are using a flat-file database driver, you need at least 4 MB of memory. If your system will host a relational database system, additional memory may be required. Consult your relational database documentation to determine the exact memory requirements.

For UNIX Users

On UNIX systems, the ODBC drivers are 32-bit drivers. The required network software supplied by your database system vendors must be 32-bit compliant. Consult the “System Requirements” section for specific requirements for each relational database driver.

The UNIX platforms supported are: AIX 3.2.5, HP-UX 9 or greater, Solaris 2.*x* for SPARC, and Solaris 2.*x* for x86.

ODBC.INI

ODBC.INI is a plain text file called .odbc.ini. The structure of this file is described in Appendix E, “ODBC.INI,” on page 346.

Since UNIX is a multiuser operating system, each user can have a private copy of .odbc.ini that resides in his or her home directory. System administrators can also maintain a centralized copy of .odbc.ini that they can control. To maintain a centralized copy, set the environment variable ODBC_INI to point to the fully qualified pathname of the centralized .odbc.ini file.

For example, if the fully qualified path is /opt/odbc/system_odbc.ini and you are using the C shell, you would enter the following:

```
setenv ODBC_INI /opt/odbc/system_odbc.in    i
```

If you are using the Korn or Bourne shells, you would enter the following:

```
ODBC_INI=/opt/odbc/system_odbc.in   i
export ODBC_IN I
```

This makes /opt/odbc/system_odbc.ini the centralized copy of the file. If a user has a copy of the .odbc.ini file in his or her home directory, then that file is used instead of the centralized version.

Starting the ODBC Administrator

The section “Configuring Data Sources” in each subsequent driver chapter instructs you to start the ODBC Administrator. You cannot do this under UNIX. Instead, edit the .odbc.ini file with a text editor. Each driver chapter contains a section on connecting using a connection string which lists the attributes and values you can set for each data source.

Driver Names

The ODBC drivers are 32-bit ODBC API-compliant dynamic link libraries, referred to in UNIX as *shared objects*. The prefix for all ODBC driver filenames on UNIX is “qe.” On UNIX the driver filenames are lowercase and the extension is .so or .sl. This is the standard form for a shared object. For example, the Oracle 7 driver filename is qeor7*nn*.so, where *nn* is the revision number of the driver.

In this manual, however, the convention is to list the driver names in uppercase with the extension .DLL, so UNIX users should keep this distinction in mind.

See the README file shipped with your INTERSOLV DataDirect product for the filename of each driver supported on UNIX.

Setting the Library Path Environment Variable

You must include the full path to the dynamic link libraries in the environment variable LD_LIBRARY_PATH (on Solaris), LIBPATH (on AIX), and SHLIB_PATH (on HP-UX). For example, if you install the ODBC drivers in the system directory /opt, then the fully qualified path for the ODBC Pack is /opt/odbc/dlls. During installation, a shell startup script is created and stored in the odbc directory. This shell script sets up the odbc environment for you.

For C shell users, the shell startup script is called .odbc.csh. This script can be sourced from a user’s own .login script. For example:

```
source /opt/odbc/.odbc.csh
```

For Bourne or Korn shell users, the shell startup script is called .odbc.sh. This script can also be sourced from a user’s own .profile script. For example:

```
.  /opt/odbc/.odbc.s  h
```

If you do not include the path /opt/odbc in the environment variable LD_LIBRARY_PATH (on Solaris), LIBPATH (on AIX), and SHLIB_PATH (on HP-UX), then your applications are unable to load the ODBC drivers dynamically at runtime.

Disk Space and Memory Requirement s

Disk space requirements are between 6 and 10 MB of free disk space on the disk where the UNIX system is installed.

Memory requirements vary, depending on the database driver. If you are using a flat-file database driver, you need at least 4 MB of memory. If your system will host a relational database system, additional memory may be required. Consult your relational database documentation to determine the exact memory requirements.

Error Messages

Error messages may come from

- An ODBC driver
- The database system
- The ODBC driver manager

An error reported on an ODBC driver has the following format:

```
[vendor] [ODBC_component] messag   e
```

ODBC_component is the component in which the error occurred. For example, an error message from INTERSOLV’s SQL Server driver would look like this:

```
[INTERSOLV] [ODBC SQL Server driver] Invalid precision
specified .
```

If you get this type of error, check the last ODBC call your application made for possible problems or contact your ODBC application vendor.

An error that occurs in the data source includes the data source name, in the following format:

```
[vendor] [ODBC_component] [data_source] messag    e
```

With this type of message, *ODBC_component* is the component that received the error from the data source indicated. For example, you may get the following message from an Oracle data source:

```
[INTERSOLV] [ODBC Oracle driver] [Oracle] ORA-0919:
specified length too long for CHAR colum    n
```

If you get this type of error, you did something incorrectly with the database system. Check your database system documentation for more information or consult your database administrator. In this example, you would check your Oracle documentation.

The driver manager is a DLL that establishes connections with drivers, submits requests to drivers, and returns results to applications. An error that occurs in the driver manager has the following format:

```
[vendo r] [ODBC DLL]  messag e
```

vendor can be Microsoft, Apple, or INTERSOLV. For example, an error from the Microsoft driver manager might look like this:

```
[Microsoft] [ODBC DLL] Driver does not support this
functio n
```

If you get this type of error, consult the *Programmer’s Reference* for the Microsoft ODBC Software Development Kit available from Microsoft.

UNIX Error Handling

UNIX error handling is performed according to the X/Open XPG3 messaging catalog system. Localized error messages are stored in the subdirectory locale/*localized_territory_directory*/LC_MESSAGES, where *localized_territory_directory* depends on your language.

For instance, German localization files are stored in locale/de/LC_MESSAGES, where de is the locale for German.

If localized error messages are not available for your locale, then they will contain message numbers instead of text. For example:

```
[INTERSOLV] [ODBC 20101 driver] 3004   0
```

Go To

2 ALLBASE Driver

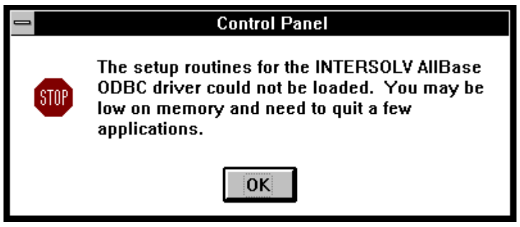
The ALLBASE driver supports the HP ALLBASE/SQL and HP IMAGE/SQL database systems in the Windows environment.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the ALLBASE driver.

System Requirements

The ALLBASE driver requires that you use the HP ALLBASE PC API product. Consult the *HP ALLBASE PC API User's Guide* for proper installation instructions.

If you attempt to configure a data source and you do not have SQLAPIW.DLL on your path or in your Windows SYSTEM directory, the message similar to the following appears.


Control Panel

The setup routines for the INTERSOLV AllBase ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

Configuring Data Sources

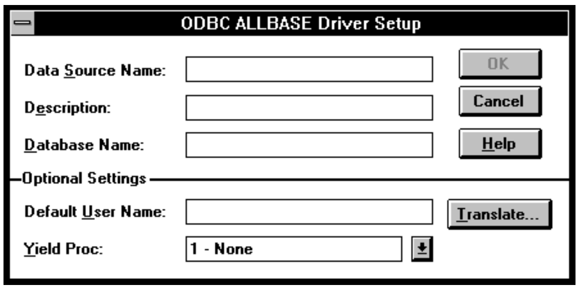
To configure an ALLBASE data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the AllBase driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.


ODBC ALLBASE Driver Setup

Data Source Name:
Description:
Database Name:
Optional Settings
Default User Name:
Yield Proc: 1 - None

3 Specify values as follows:

Data Source Name:A string that identifies this ALLBASE data source configuration in ODBC.INI. Examples include “Accounting” or “ALLBASE-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “ALLBASE Database on Server #1.”

Database Name: The name of the database to which you want to connect by default.

The following settings are optional:

Default User Name: The default user name used to connect to your ALLBASE database. This user name is case-sensitive. A user name is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the Logon dialog box or connection string.

Yield Proc:A value of 0, 1, 2, or 3 that determines whether you can work in other applications when ALLBASE is busy. This attribute is useful to users of ODBC applications. Valid values are:

- 0 (peek and dispatch), which causes the driver to check the Windows message queue and send any messages to the appropriate Windows application.
- 1 (no yielding, the default), which does not let you work in other applications.
- 2 (ALLBASE's yield procedure), which uses ALLBASE's default yield procedure.
- 3 (dispatch via Windows Yield function), which turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window.

It is recommended that you use the value 1.

4. Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

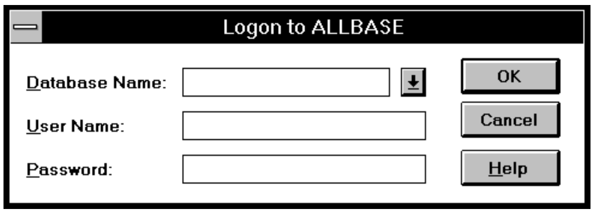
   The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

5. Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the

defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For ALLBASE, the dialog box is as follows:



In this dialog box, do the following:

1 If you want to access a database other than the one specified in ODBC.INI or if no default database is specified, type the name of the database you want to access, or select the name from the Database Name drop-down list.

2 If required, type your case-sensitive user name.

3 If required, type your case-sensitive password.

4 Click **OK** to complete the logon. The name of the database and your user name are saved in ODBC.INI and displayed the next time you log on.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for ALLBASE is

```
DSN=ACCOUNTING;DB=PAYROLL;UID=JIM;PWD=XYZZY
```

Table 2-1 gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 2-1. ALLBASE Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies an ALLBASE data source configuration in ODBC.INI. Examples include “Accounting” or “ALLBASE-Serv1.”
LogonID (UID)	The default Logon ID (user name) used to connect to your ALLBASE database. This ID is case-sensitive. A Logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your Logon ID.

Table 2-1. ALLBASE Connection String Attributes (cont.)

Attribute	Description
Password (PWD)	A case-sensitive password.
Database (DB)	A string that identifies the name of the database to which you want to connect.
YieldProc (YLD)	YieldProc={0|1|2|3}. This attribute determines whether you can work in other applications when ALLBASE is busy. It is useful to users of ODBC applications. • YieldProc=0 (peek and dispatch) causes the driver to check the Windows message queue and send any messages to the appropriate Windows application • YieldProc=1 (no yielding) does not let you work in other applications. • YieldProc=2 (default yield procedure) uses ALLBASE's default yield procedure. • YieldProc=3 (dispatch via Windows Yield function) turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window. It is recommended that you use YieldProc=1, the initial default. YieldProc=0, YieldProc=2, and YieldProc=3 do not work with all Windows applications.

Data Types

Table 2-2 shows how ALLBASE data types map to the standard ODBC data types.

Table 2-2. ALLBASE Data Types

ALLBASE	ODBC
Char	SQL_VARCHAR
Varchar	SQL_VARCHAR
Date	SQL_DATE
Datetime	SQL_TIMESTAMP
Float(24)	SQL_REAL
Double Precision	SQL_DOUBLE
Decimal	SQL_DECIMAL
Integer	SQL_INTEGER
Smallint	SQL_SMALLINT
Time	SQL_TIME

Isolation and Lock Levels Supported

ALLBASE supports isolation levels 1 (read committed) and 3 (serializable). The default is 1. ALLBASE supports page-level locking. See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The ALLBASE driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. Also, the ALLBASE driver supports the SQLSetPos function, as well as backward and random fetching in SQLExtendedFetch.

The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The HP/ALLBASE and HP/IMAGE database systems support a single connection and multiple statements per connection.

Go To

3 Btrieve Driver

The Btrieve driver supports Btrieve version 5.0 and higher in the Windows environment. The driver executes SQL statements directly on Btrieve databases.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the Btrieve driver.

System Requirements

To access a Btrieve database, you must be using the following software:

- WDDLSVCS.DLL dynamic link library.
- WBTRCALL.DLL dynamic link library. There are two versions of WBTRCALL.DLL: networked and non-networked. INTERSOLV supplies the non-networked version when you install the Btrieve driver. If you want network access, you must get the networked version from Btrieve Technologies.

Before you attempt to access Btrieve files, you must follow these steps:

1. Incorporate existing Btrieve files into a Scalable SQL database. To do this, see the section “Defining Table Structure” on page 31.
2. Make sure the WBTRCALL.DLL and WDDLSVCS.DLL dynamic link libraries are in a directory on your DOS path or in the Windows SYSTEM directory.

Managing Databases

If you already use Scalable SQL, the Btrieve driver can access your Scalable SQL databases directly. If not, your Btrieve files must be incorporated into a Scalable SQL database.

A Scalable SQL database is composed of data files that contain your records and data dictionary files that describe the database. The data files are Btrieve files. The data dictionary files are special Btrieve files that contain descriptions of the data files, views, fields, and indexes in your database.

All Btrieve files in a Scalable SQL database must reside in the same directory. In addition to the Btrieve data files, the three data dictionary files, FILE.DDF, FIELD.DDF, and INDEX.DDF, also must be in the directory.

Incorporating a Btrieve file into a Scalable SQL database does not change the Btrieve file in any way. You can continue to access the file directly with any existing Btrieve application.

Transactions

The Btrieve driver supports *transactions*. A transaction is a series of database changes that is treated as a single unit. In applications that don't use transactions, the Btrieve driver immediately executes Insert, Update, and Delete statements on the database files and the changes are automatically committed when the SQL statement is executed. You cannot undo these changes. In applications that use transactions, the Btrieve driver holds inserts, updates, and deletes until you issue a Commit or Rollback. A Commit saves the changes to the database file; a Rollback undoes the changes.

Transactions affect the removal of record locking. All locks are removed when SQLTransact is called with the Commit or Rollback option to end the active transaction.

Installing Btrieve to Process Transactions

To use the Btrieve driver's transaction processing capabilities, you must change the required settings for Btrieve in the WIN.INI file; if you are using Btrieve 6.0, use the file NOVDB.INI; if you are using Btrieve 6.15, use the file BTI.INI. Edit the appropriate .INI file to contain the following options setting in the [btrieve] section:

```
options=/m:64 /p:4096 /t:c:  \windows_director y\bbt.trn /
n:12
```

This setting specifies the following options:

/m	The total size, in kilobytes, of Btrieve's data area. 64 is the maximum value that can be specified.
/p	The maximum page size of any Btrieve file you want to access. 4096 is the maximum value that can be specified.
/t	The pathname to the transaction control file used to process transactions. In a network environment, the transaction control file should be on a network disk drive, and all users who start Btrieve should use the same file.
/n	The maximum number of files involved per transaction. This number must not exceed 18.

The complete set of options are documented in the *Btrieve for Windows Installation and Operations Manual*, in the section "Starting the Record Manager."

For example, to have a maximum of 16 files per transaction and to place the transaction control file in F:\BTRANS\BTRIEVE.TRN, use these settings in the options specification:

```
/n:16 /t:f:\btrans\btrieve.tr  n
```

Add a [brequestDPMI] section to the appropriate .INI file to initialize the requester DLL. For example:

```
[brequestDPMI]
datalength=4096
chkparms=no
local={yes | no}
tasks=1 0
```

The datalength setting must be at least as large as the largest record to be accessed.

Set local=no if you are accessing network files and WBTRCALL.DLL is the networked version. Set local=yes if you are accessing local files and the non-networked version of WBTRCALL.DLL is on your disk. If you want to access both network and local files, you must have the networked version of WBTRCALL.DLL, rename the local version of WBTRCALL.DLL to WBTRLOCL.DLL, and set local=yes. If you are using a requester with a version earlier than 6.0A, the local=yes line must not have anything after the yes (no spaces, no comment).

Configuring Data Sources

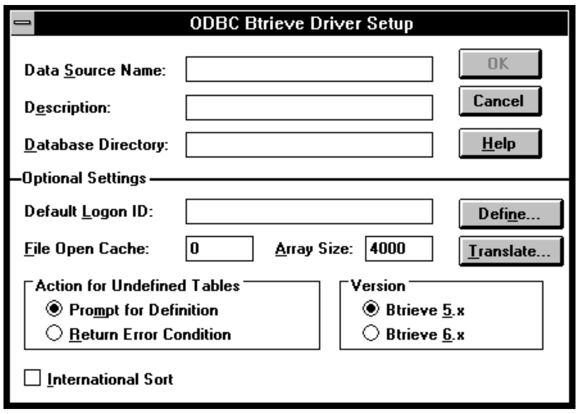
To configure a Btrieve data source:

1. Start the ODBC Administrator. A list of data sources appears.
2. If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Btrieve driver and click **OK**.

   If you are configuring an existing data source, select the data source name and click **Setup**.

   The setup dialog box appears.

.



3 Specify values as follows:

Data Source Name:A string that identifies this Btrieve data source configuration in ODBC.INI. Examples include “Accounting” or “Btrieve Files.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “Btrieve files in C:\ACCOUNTS.”

Database Directory: The full pathname of the directory that contains the Btrieve files and the data dictionary files (.DDF). Data dictionary files describe the structure of Btrieve data. If no directory is specified, the current working directory is used.

The following settings are optional:

Default Logon ID: The default logon ID used to connect to your Btrieve database. A logon ID is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in a connection string.

File Open Cache A numeric value to specify the maximum number of used file handles to cache. For example, the value 4 specifies that when a user opens and closes four tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is increased performance. The default is 0, which means no file open caching.

Action for Undefined Tables: A setting to indicate whether the driver should prompt the user when it encounters a table for which it has no structure information. Set the Prompt for Definition radio button to prompt the user; set the Return Error Condition radio button (the default) to return an error.

Version: The version of Btrieve you want to access. Choose either the Btrieve 5.*x* radio button (the default) or the Btrieve 6.*x* radio button.

International Sort: A setting to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Select this check box to use the international sort order as defined by your operating system. International sort order is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (see your operating system documentation). Leave this box blank to use ASCII sort order. ASCII sort order is case-sensitive, where uppercase letters precede lowercase letters (*B* precedes *a*).

4 Click **Define** to define table structure. See the section "Defining Table Structure" on page 31 for step-by-step instructions for how to define table structure.

5 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

6 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the

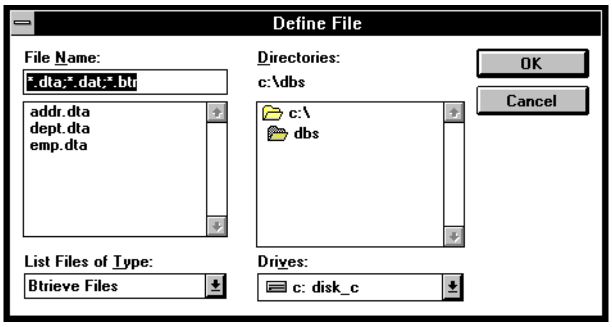
defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Defining Table Structure

Since Btrieve does not store any column information in the data file, you must define its structure through the Btrieve setup dialog box, which you can access through the ODBC Administrator.

To define the structure of a file, take the following steps:

1 Click **Define** in the setup dialog box. The Define File dialog box appears.


Define File

File Name: *.dta;*.dat;*.btr

addr.dta
dept.dta
emp.dta

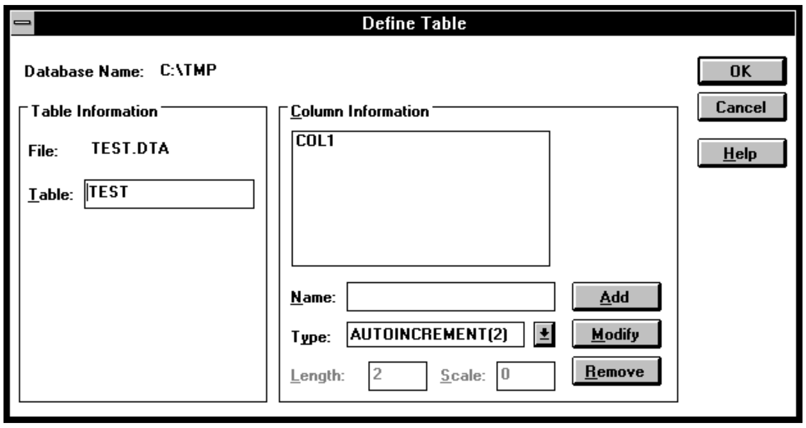
Directories: c:\dbs

c:\
dbs

List Files of Type: Btrieve Files

Drives: c: disk_c

2 Select the file you want to define and click **OK.** The Define Table dialog box appears.


Define Table

Database Name: C:\TMP

Table Information
File: TEST.DTA
Table: TEST

Column Information
COL1
Name:
Type: AUTOINCREMENT(2)
Length: 2 Scale: 0

In this dialog box, Database Name and File display the name of the directory containing the Scalable SQL data dictionary files and the name of the file, respectively.

3 In the Table Information section of this dialog box, type the name of the table to be returned by SQLTables in the Table box. The name can be up to 20 characters and may not be the same as another defined table in the database. This field is required.

4 The Column Information section of this dialog box lets you add, modify, and delete columns. The box at the top of this section displays the defined columns. To add a column definition:

a In the Name box, type the name of the column.

b In the Type box, open the drop-down list and select the column's data type.

c In the Length box, type the column's length, if applicable.

d In the Scale box, type the column's scale, if applicable.

e Click **Add**.

To modify a column's definition, select the column you want to modify, type the new definition in the appropriate boxes (refer to steps 4a-d), and click **Modify**.

To remove a column's definition, select the column you want to remove and click **Remove**.

5 Click **OK** to define the table.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for Btrieve is

```
DSN=BTRIEVE FILES;DB=EMP;UID=JOHN;PWD=XYZZY
```

Table 3-1 lists the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 3-1. Btrieve Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies a Btrieve data source configuration in ODBC.INI. Examples include "Accounting" or "Btrieve Files."
Database (DB)	The directory in which the Btrieve files and the data dictionary files (.DDF) are stored. Data dictionary files describe the structure of Btrieve data. These files must exist to connect with the driver.
UndefinedTable (UT)	UndefinedTable={PROMPT|ERROR}. This attribute determines whether the driver should prompt the user when it encounters a table for which it has no structure information. Set this option to PROMPT to prompt the user; set it to ERROR to return an error. The initial default is to return an error.
FileOpenCache (FOC)	An integer value that determines the maximum number of used file handles to cache. For example, when FileOpenCache=4, and a user opens and closes 4 tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The initial default is 0, which means no file open caching.
LogonID (UID)	The default logon ID used to connect to your Btrieve database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.

Table 3-1. Btrieve Connection String Attributes (cont.)

Attribute	Description
Password (PWD)	The password that you must enter if your Scalable SQL data dictionary files have security restrictions imposed.
Version (VER)	Version={5|6}. The version of Btrieve you want to access. The initial default is 5.
IntlSort (IS)	IntlSort={0|1}. This attribute determines the order that records are retrieved when you issue a Select statement with an Order By clause. If IntlSort=1, the driver uses the international sort order as defined by your operating system. The sort is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (see your operating system documentation). If IntlSort=0 (the initial default), the driver uses the ASCII sort order, where uppercase letters precede lowercase letters (*B* precedes *a*).
ModifySQL (MS)	ModifySQL={0|1}. This attribute is provided for backward compatibility. Specify ModifySQL=0 to have the driver understand SQL dialects found in earlier drivers. Specify ModifySQL=1 (the default) to have the driver modify the SQL statement to conform to ODBC specifications.

Table 3-1. Btrieve Connection String Attributes (cont.)

Attribute	Description
DeferQueryEvaluation (DQ)	DeferQueryEvaluation={0|1}. This attribute determines when a query is evaluated—after all records are read or each time a record is fetched. If DeferQueryEvaluation=0, the driver generates a result set when the first record is fetched. The driver reads all records, evaluates each one against the Where clause, and compiles a result set containing the records that satisfy the search criteria. This process slows performance when the first record is fetched, but activity performed on the result set after this point is much faster because the result set has already been created. You do not see any additions, deletions, or changes in the database that occur while working with this result set. If DeferQueryEvaluation=1 (the default), the driver evaluates the query each time another record is fetched and stops reading through the records when it finds one that matches the search criteria. This setting avoids the slowdown while fetching the first record, but each fetch takes longer because of the evaluation taking place. The data you retrieve reflects the latest changes to the database. However, a result set is still generated if the query is a Union of multiple Select statements, if it contains the Distinct keyword, or if it has an Order By or Group By clause.
ArraySize (AS)	An integer value that enables the driver to retrieve an array of records from the Btrieve engine and in most cases results in better performance for the application. The value of ArraySize is the number of bytes in the array. The default ArraySize is 4096 bytes and the maximum is 65535 bytes.

Data Types

Table 3-2 shows how the Btrieve data types map to the standard ODBC data types. The Btrieve data types are used when you incorporate Btrieve files into a Scalable SQL database.

Table 3-2. Btrieve Data Types

Btrieve	ODBC
Autoincrement(2)	SQL_SMALLINT
Autoincrement(4)	SQL_INTEGER
Bfloat(4)	SQL_REAL
Bfloat(8)	SQL_DOUBLE
Char	SQL_CHAR
Date	SQL_DATE
Decimal	SQL_DECIMAL
Float(4)	SQL_REAL
Float(8)	SQL_DOUBLE
Integer(1)	SQL_TINYINT
Integer(2)	SQL_SMALLINT
Integer(4)	SQL_INTEGER
Logical(1)	SQL_BIT
Logical(2)	SQL_BIT
Lstring	SQL_VARCHAR
Money	SQL_DECIMAL
Note	SQL_LONGVARCHAR
Numeric	SQL_NUMERIC
Numericsts	SQL_NUMERIC
Time	SQL_TIME
Zstring	SQL_VARCHAR

Indexes

Note: If you define an index using the INTERSOLV Btrieve driver, the index will not have the restrictions discussed here.

For query optimization, the Btrieve driver does not use

- Indexes containing all-segment-null keys or any-segment-null keys.
- Any index key that is marked case-insensitive.
- Any index keys where the data type of the index key does not match the data type of the field. The one exception is if the index key is declared as an unsigned integer and the field in the file is declared as signed integer, or vice versa, then the driver assumes the field contains only unsigned quantities and uses the index. Note that this can lead to incorrect results if the field in fact does contain signed quantities.

The Btrieve driver only uses an alternate-collating-sequence (ASC) index key for equality lookups. Additionally, if an ASC key is part of a segmented index, the other index segments are not used for query optimization unless the Where clause contains an equality condition for the ASC key.

Column Names

Column names in SQL statements (such as, Select, Insert, etc.) can be up to 20 characters long. If column names are in all lowercase, a combination of upper and lowercase, contain blank spaces, or are reserved words, they must be surrounded by the grave character (`) (ASCII 96). For example:

```
SELECT `name` FROM em  p
```

Select Statement

You use the SQL Select statement to specify the columns and records to be read. Btrieve Select statements support all the Select statement clauses as described in Appendix A, “SQL for Flat-File Drivers,” on page 299. This section describes the information that is specific to Btrieve, which is ROWID.

ROWID Pseudo-Colum

Each Btrieve record contains a special column named ROWID. This field contains a unique number that indicates the record’s sequence in the database. You can use ROWID in Where and Select clauses.

ROWID is particularly useful when you are updating records. You can retrieve the ROWID of the records in the database along with the other field values. For example:

```
SELECT last_name, first_name, salary, rowid FROM em    p
```

Then you can use the ROWID of the record that you want to update to ensure that you are updating the correct record and no other. For example:

```
UPDATE emp set salary = 40000 FROM emp WHERE rowid=2    1
```

The fastest way of updating a single row is to use a Where clause with the ROWID. You cannot update the ROWID column.

Select statements that use the ROWID pseudo-column in the Where clause achieve maximum performance only for exact equality matches. If you use range scans instead of exact equality matches, a full table scan is performed. For example:

```
SELECT * FROM emp WHERE rowid=21  //fast searc    h

SELECT * FROM emp WHERE rowid <=25  //full table sca    n
```

Create and Drop Index Statements

The Btrieve driver supports SQL statements to create and delete indexes. The Create Index statement is used to create indexes and the Drop Index is used to delete indexes.

Create Index

The Create Index statement for Btrieve files has the form:

```
CREATE [UNIQUE] INDEX   index_name ON table_name
    ([field_name [ASC | DESC] [ ,field_name [ASC |
DESC]]... )
```

UNIQUE means that Btrieve does not let you insert two records with the same index values.

index_name is the name of the index.

table_name is the name of the table on which the index is to be created.

ASC tells Btrieve to create the index in ascending order. DESC tells Btrieve to create the index in descending order. By default, indexes are created in ascending order.

For example:

```
CREATE INDEX lname ON emp (last_name    )
```

Drop Index

The form of the Drop Index statement is

```
DROP INDEX table_name.index_name
```

table_name is the name of the table from which the index is to be dropped.

index_name is the name of the index.

For example:

```
DROP INDEX emp.lname
```

Isolation and Lock Levels Supported

Btrieve supports isolation level 1 (read committed) only. Btrieve supports page-level locking. See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The Btrieve driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 function is supported: SQLSetPos.

The Btrieve driver also supports backward and random fetching in SQLExtendedFetch. The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

Btrieve files support a single connection and multiple statements per connection.

Go To

4 DB2/2 Drivers

Two DB2/2 drivers are available:

- The new DB2/2 driver supports the use of the Client Application Enabler (CAE) 1.2 interfaces.
- The previous DB2/2 driver, QEDBM*nn*.DLL, supports the use of CAE 1.0 interfaces and PCDRDLL.DLL in the Windows environment only. Note that the previous DB2/2 driver is supported in this release but will not be supported in the next release of INTERSOLV ODBC drivers.

The new DB2/2 driver supports the following database systems in the Windows, OS/2, and AIX environments:

- DB2/2 for OS/2 (formerly called Database Manager for OS/2)
- DB2/6000
- DB2/MVS (through the DDCS/2 or DDCS/6000 gateway)
- SQL/400 (through the DDCS/2 or DDCS/6000 gateway)
- SQL/DS (through the DDCS/2 or DDCS/6000 gateway)

The previous DB2/2 driver, QEDBM*nn*.DLL, supports all of the database systems listed above on Windows only.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the DB2/2 driver.

System Requirements

You must have a DB2 Client Application Enabler (CAE) or DOS/Windows Requester to access the DB2 family of databases.

Windows

There are three distinct types of DB2/2 clients for Windows:

- The DOS/Windows Requester (formerly Database Manager Client) and CAE versions prior to 1.2
- DB2/6000 or DDCS/6000 using CAE
- CAE version 1.2

System requirements are as follows:

- The DB2/2 Server must be installed as the Server Version (*not* the Local Version).
- To use the DDCS/2 gateway, you must install the DDCS/2 Multi-User Edition (*not* the Single-User Edition).
- DB2/2 requires network protocols from products such as Communications Manager from Extended Services version 1.0, Communications Manager/2 version 1.0, Network Transport Services/2 version 1.0, LAN Services, Novell NetWare, or numerous TCP/IP vendors.
- To connect to DB2/6000, use the CAE for DOS and Windows that comes with DB2/6000 or the CAE 1.2. You cannot use any version of the CAE prior to 1.2. DB2/6000 does not require a gateway. The DB2/6000 CAE for DOS/Windows and the CAE 1.2 connect directly to DB2/6000 via TCP/IP.
- To connect to DDCS/6000, use the CAE for DOS and Windows that comes with DB2/6000 or CAE 1.2. You cannot use any version of the DOS/Windows Requester prior to 1.2. The DB2/6000 CAE for DOS/Windows and the CAE 1.2 connect directly to DB2/6000 via TCP/IP.

For Users of QEDBMnn.DLL Only

To use versions of the DB2/2 CAE prior to 1.2, you must first modify the Windows Client Enabler configuration file, DBDRQLIB.CFG, located in the DBDRQLIB directory. Three parameters must be changed from their default values for the Database Requester to work properly, as follows:

```
SQLSIZE=4096/51200  0
SQLCODEPAGE =code_page_numbe r
SQLNAME =YOUR_NAM E
```

Change SQLSIZE exactly as shown.

The SQLCODEPAGE setting must match the code page number in which the DB2/2 database was created.

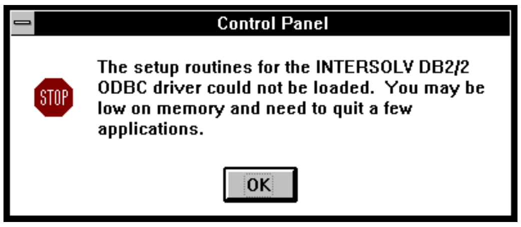
SQLNAME must be set to the SQL name assigned to you by your system administrator and must be entered in uppercase letters.

Note: When using DB2/6000 or DDCS/6000 with CAE, or when using CAE version 1.2 for DB2/2, you must catalog the protocol node (usually TCP/IP, NetBios, or IPX/SPX) before you catalog the database. You need to use the DB2 command line interface that comes from the CAE. For details, see the *DATABASE 2 Client Application Enabler/DOS User's Guide.*

CAE version 1.2 and the DB2/6000 CAE do not have a DBDRQLIB.CFG file.

PCDRDLL.DLL must be either in your Windows SYSTEM directory or the DBDRQLIB directory must be on your search path.

If you attempt to configure a data source and you do not have PCDRDLL.DLL on your path or in your Windows SYSTEM directory, a message similar to the following appears.



The setup routines for the INTERSOLV DB2/2 ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

OS/2

System requirements are as follows:

- The DB2/2 Server must be installed as the Server Version for remote clients to use it. The Local Version supports only OS/2 applications running on the server.
- To use the DDCS/2 gateway, you must install the DDCS/2 Multi-User Edition for remote clients to use it. The Single-User Version supports only OS/2 applications running on the server.
- For remote database access, DB2/2 requires either Communications Manager from Extended Services version 1.0, Communications Manager/2 version 1.0, Network Transport Services/2 version 1.0, or LAN Services.

UNIX (AIX)

There are three distinct types of DB2 clients for UNIX:

- The local DB2 client, where the DB2 RDBMS resides on the same computer as the client.

  Required software:

  - IBM DATABASE 2 for UNIX (AIX, Solaris, HP-UX)
  - The RDBMS

- The remote DB2 client, where the DB2 RDBMS resides on a remote computer from the DB2 client.

Required software for the client side:

- IBM DATABASE 2 Client Application Enabler

Required software for the server side:

- IBM DATABASE 2 for UNIX (AIX, Solaris, HP-UX)
- IBM DATABASE 2 Client Support for UNIX (AIX, Solaris, HP-UX)

- The remote DB2 client, where the remote RDBMS is either DB2 for MVS, SQL/DS for VM or VSE, or DB2/400 for OS/400. This enables the client to access DRDA-compliant RDBMS.

  Required software for the client side:

  - IBM DATABASE 2 Client Application Enabler

  Required software for the server side:

  - IBM DATABASE Client Support for UNIX (AIX, Solaris, HP-UX)
  - IBM Distributed Database Connection Services for UNIX (AIX, Solaris, HP-UX).

Configuring Data Sources

For Users of QEDB2nn.DLL Only

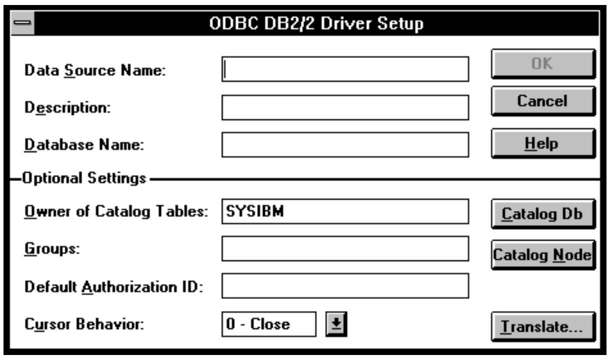
To configure a DB2/2 data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the DB2/2 driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.



Owner of Catalog Tables: SYSIBM
Cursor Behavior: 0 - Close

3 Specify values as follows:

Data Source Name: A string that identifies this DB2/2 data source configuration in ODBC.INI. Examples include “Accounting” or “DB2/2-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “DB2/2 on Server #1.”

Database Name: The name of the database to which you want to connect by default.

The following values are optional:

Owner of Catalog Tables: On most DB2/2 systems, SYSIBM is the owner of the catalog system tables. If you have read access to the system tables, you do not need to change this option.

If you do not have read access, the system administrator must create a view of the system tables in another account and give you permission to use that view. In this case, specify the Authorization ID for the account that owns the views of the system tables.

If you are using DDCS/2 to connect to a SQL/400 database, specify a comma-separated list of the Collection IDs to which you want to have access in your ODBC application.

If you are using DDCS/2 to connect to a SQL/DS database, specify SYSTEM as the owner.

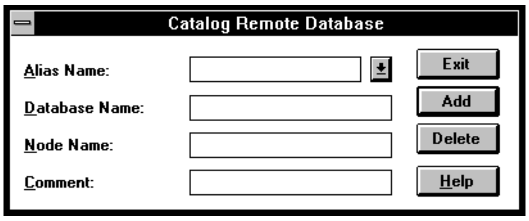
Groups: A value to determine which tables you can access. Your system administrator may have placed you in a “group” of users and granted table access to the entire group. If this is the case, specify the names of any groups to which you belong; separate each name with a comma. Alternatively, specify the word ALL so that you see all table names even if you cannot access the table.

Default Authorization ID: The default Logon ID used to connect to your DB2/2 database. A Logon ID is required only if security is enabled on your database. Your ODBC application may override this value or you may change this value in the logon dialog box.

Cursor Behavior: Select Preserve if you want cursors to be held at the current position when the transaction ends. Otherwise, leave this set to Close. Selecting Preserve may impact the performance of your database operations. This setting does not apply to SQL/DS.

When you select Preserve, the driver returns SQL_CB_PRESERVE from SQLGetInfo (SQL_CURSOR_COMMIT_BEHAVIOR). But only Select statements and prepared Update or Delete . . . Where Current of Cursor statements are preserved when the transaction ends. All other prepared statements are closed and deleted.

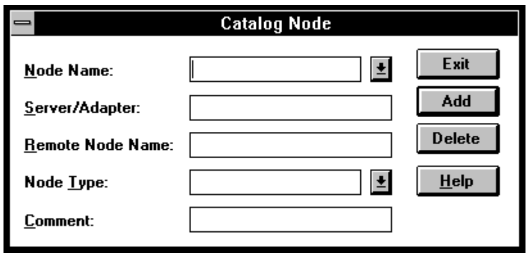
4 You must catalog any remote database you want to access. To do so, click the **Catalog Db** button. The Catalog Remote Database dialog box appears.



In this dialog box, type an alias name for the remote database, the name of the remote database, and an alias for the node (the node name could describe the remote node name and the communication protocol used to access it). Optionally, you can enter a comment. Click **Add**. The alias name you added will be listed in the Database Name drop-down list in the logon dialog box when you exit this dialog box.

This dialog box also lets you uncatalog remote databases. To do this, select the alias you want to delete from the Alias Name drop-down list and click **Delete**.

5 To provide information about the remote site for the DB2/2 databases, click **Catalog Node**. The Catalog Node dialog box appears.



In this dialog box, select a node name (only names of previously cataloged nodes are available), the name of the server or number of the adapter (NetBIOS nodes), the name of the remote node where the DB2 server is installed, and the type of communication protocol to use (IPX/SPX, NetBios, or TCP/IP). Optionally, you can enter a comment. Click **Add**.

This dialog box also lets you uncatalog nodes. To do this, select the node you want to delete from the Node Name drop-down list and click **Delete**.

6 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

Note: In the UNIX environment, there is no graphical interface. To perform a translation, you must edit the .odbc.ini file and add to the data source section the TranslationDLL keyword and optionally the TranslationOption keyword:

Keyword	Definition
TranslationDLL	Full path of translation DLL.
TranslationOption	ASCII representation of the 32-bit integer translation option.

The INTERSOLV OEM ANSI translator is shipped on UNIX to provide an example of how to create a translation DLL.

7 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

For Users of QEDBMnn.DLL Only

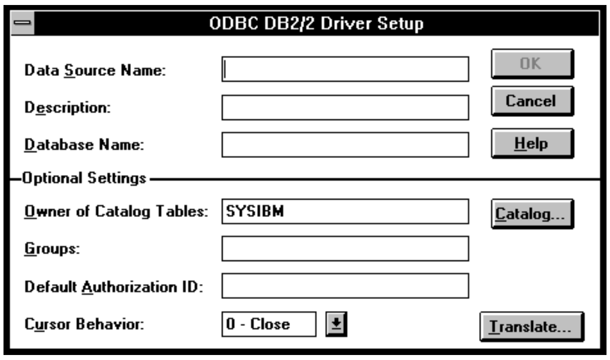
To configure a DB2/2 data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Database Manager driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.


ODBC DB2/2 Driver Setup

Data Source Name:	
Description:	
Database Name:	
Optional Settings	
Owner of Catalog Tables:	SYSIBM
Groups:	
Default Authorization ID:	
Cursor Behavior:	0 - Close

3 Specify values as follows:

Data Source Name: A string that identifies this DB2/2 data source configuration in ODBC.INI. Examples include “Accounting” or “DB2/2-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “DB2/2 on Server #1.”

Database Name: The name of the database to which you want to connect by default.

The following values are optional:

Owner of Catalog Tables: On most DB2/2 systems, SYSIBM is the owner of the catalog system tables. If you have read access to the system tables, you do not need to change this option.

If you do not have read access, the system administrator must create a view of the system tables in another account and give you permission to use that view. In this case, specify the Authorization ID for the account that owns the views of the system tables.

If you are using DDCS/2 to connect to a SQL/400 database, specify a comma-separated list of the Collection IDs to which you want to have access in your ODBC application.

If you are using DDCS/2 to connect to a SQL/DS database, specify SYSTEM as the owner.

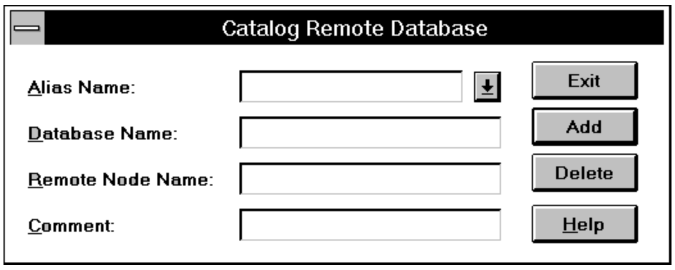
Groups: A value to determine which tables you can access. Your system administrator may have placed you in a “group” of users and granted table access to the entire group. If this is the case, specify the names of any groups to which you belong; separate each name with a comma. Alternatively, specify the word ALL so that you see all table names even if you cannot access the table.

Default Authorization ID: The default Logon ID used to connect to your DB2/2 database. A Logon ID is required only if security is enabled on your database. Your ODBC application may override this value or you may change this value in the logon dialog box.

Cursor Behavior: Select Preserve if you want cursors to be held at the current position when the transaction ends. Otherwise, leave this set to Close. Selecting Preserve may impact the performance of your database operations. This setting does not apply to SQL/DS.

When you select Preserve, the driver returns SQL_CB_PRESERVE from SQLGetInfo (SQL_CURSOR_COMMIT_BEHAVIOR). But only Select statements and prepared Update or Delete . . . Where Current of Cursor statements are preserved when the transaction ends. All other prepared statements are closed and deleted.

4 You must catalog any database you want to access. To do so, click the **Catalog** button. The Catalog Remote Database dialog box appears.



In this dialog box, type an alias name, the name of the database, and the name of the remote node (the name of the node where the server is installed). Click **Add**. The alias name you added will be listed in the Database Name drop-down list in the logon dialog box when you exit this dialog box.

This dialog box also lets you uncatalog remote databases. To do this, select the alias you want to delete from the Alias Name drop-down list and click **Delete**.

5 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

UNIX

Note: In the UNIX environment, there is no graphical interface. To perform a translation, you must edit the .odbc.ini file and add to the data source section the TranslationDLL keyword and optionally the TranslationOption keyword:

Keyword	Definition
TranslationDLL	Full path of translation DLL
TranslationOption	ASCII representation of the 32-bit integer translation option

The INTERSOLV OEM ANSI translator is shipped on UNIX to provide an example of how to create a translation DLL.

6 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

DB2/2 and DDCS/2 Gateway

Access to DB2/2 databases, or other databases via the DDCS/2 gateway, requires that you bind to tables and grant privileges to all users. These processes are described in the following sections.

Binding

Before you can access a table, you must bind to the database. You must bind to every database you intend to access. To do this, enter the following commands on the server:

```
SQLBIND QECSV1.BND   database_name
SQLBIND QERRV1.BND   database_name
SQLBIND QEURV1.BND   database_name
SQLBIND QECSWHV1.BND   database_name
SQLBIND QERRWHV1.BND   database_name
SQLBIND QEURWHV1.BND   database_name
```

You may need to copy the bind files from the installation directory to the server.

database_name is the name of the database you want to access.

Note for SQL/DS Users For binding to SQL/DS databases, enter only the first three of these commands. The files QECSWHV1.BND, QERRWHV1.BND, and QEURWHV1.BND will not work with SQL/DS.

For SQL/400 Users

If you receive the error

```
'QSYS.QE' is an undefined name
```

you must create a collection named QE. To do this, enter the following command on the host AS/400:

```
CREATE COLLECTION QE
```

Then, issue the SQLBIND commands again.

Granting Privileges

Users must be granted the necessary privileges before they can access a database or table through a gateway.

DB2/2 users are assigned an Authorization ID. There are three categories of Authorization IDs: administrator, local administrator, and user. The third category, user, must be granted the privilege to access DB2/2 tables.

To grant the necessary privileges, an administrator must run Query Manager or the DBM Command Line Interface and execute the appropriate GRANT statement.

For DB2/2 databases and databases that reside on DDCS/2 connecting to DB2:

```
GRANT EXECUTE, BIND ON PROGRAM QE.QECSV1 TO
authorization_lis t
GRANT EXECUTE, BIND ON PROGRAM QE.QERRV1 TO
authorization_lis t
GRANT EXECUTE, BIND ON PROGRAM QE.QEURV1 TO
authorization_lis t
GRANT EXECUTE, BIND ON PROGRAM QE.QECSWHV1 TO
 authorization_lis t
GRANT EXECUTE, BIND ON PROGRAM QE.QERRWHV1 TO
 authorization_lis t
GRANT EXECUTE, BIND ON PROGRAM QE.QEURWHV1 TO
 authorization_lis t
```

For databases that reside on the DDCS/2 gateway connecting to SQL/DS:

```
GRANT RUN ON QE.QECSV1 TO   authorization_lis t
GRANT RUN ON QE.QERRV1 TO   authorization_lis t
GRANT RUN ON QE.QEURV1 TO   authorization_lis t
GRANT RUN ON QE.QECSWHV1 TO   authorization_lis t
GRANT RUN ON QE.QERRWHV1 TO   authorization_lis t
GRANT RUN ON QE.QEURWHV1 TO   authorization_lis t
```

For databases that reside on the DDCS/2 gateway connecting to SQL/400:

```
GRANT RUN ON PROGRAM QE.QECSV1 TO    authorization_lis t
GRANT RUN ON PROGRAM QE.QERRV1 TO    authorization_lis t
GRANT RUN ON PROGRAM QE.QEURV1 TO    authorization_lis t
GRANT RUN ON PROGRAM QE.QECSWHV1 TO    authorization_lis t
GRANT RUN ON PROGRAM QE.QERRWHV1 TO    authorization_lis t
GRANT RUN ON PROGRAM QE.QEURWHV1 TO    authorization_lis t
```

authorization_list is a list of comma-separated Authorization IDs or Group IDs, or PUBLIC if you want to grant access to all users.

DB2/6000 and DDCS/6000 Gateway

Before you can access a table via DB2/6000 or the DDCS/6000 gateway, you must bind to the database and grant each user access to it. Enter the following commands on the client computer from the DBM Command Line Interface:

```
BIND QECSV1.BND BLOCKING ALL GRANT    authorization_lis t
BIND QERRV1.BND BLOCKING ALL GRANT    authorization_lis t
BIND QEURV1.BND BLOCKING ALL GRANT    authorization_lis t
BIND QECSWHV1.BND BLOCKING ALL GRANT    authorization_lis t
BIND QERRWHV1.BND BLOCKING ALL GRANT    authorization_lis t
BIND QEURWHV1.BND BLOCKING ALL GRANT    authorization_lis t
```

authorization_list is a list of comma-separated Authorization IDs or Group IDs, or PUBLIC if you want to grant access to all users.

Note for SQL/DS Users For binding to SQL/DS databases, enter only the first three of these commands. The files QECSWHV1.BND, QERRWHV1.BND, and QEURWHV1.BND will not work with SQL/DS.

For SQL/400 Users

If you receive the error

```
'QSYS.QE' is an undefined nam   e
```

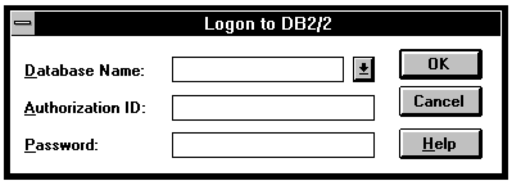
you must create a collection named QE. To do this, enter the following command on the host AS/400:

```
CREATE COLLECTION Q  E
```

Then, issue the BIND commands again.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For DB2/2, the dialog box is as follows:



In this dialog box, do the following:

1 Type the name of the remote database or select the name of the remote database from the Database Name drop-down list.

You must have cataloged any database you want to access from the client. (See the section "Configuring Data Sources" on page 47 for information on how to do this.)

2 If required, type your user name (authorization ID).

3 If required, type your password.

If you leave your user name and password blank, the ODBC application assumes you have already logged on using SQLLOGN2 (under DOS) or using User Profile Management (under OS/2). If you have not, the application returns an error. You must either type your user name and password in the dialog box or log on using SQLLOGN2 and STARTDRQ (under DOS) or using User Profile Management (under OS/2). See the *IBM Extended Services for OS/2 Guide to DB2/2 Clients* for more information.

4 Click **OK** to complete the logon and to update the values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for DB2/2 is

```
DSN=DB22 TABLES;DB=PAYROLL;UID=JOHN;PWD=XYZZY;GRP=ACCTNG
```

Table 4-1 gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 4-1. DB2/2 Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies a DB2/2 data source configuration in ODBC.INI. Examples include "Accounting" or "DB2/2-Serv1."
Database (DB)	The name of the database to which you want to connect

Table 4-1. DB2/2 Connection String Attributes (cont.)

Attribute	Description
Sysibm (SI)	On most DB2/2 systems, SYSIBM is the owner of the catalog system tables. If you have read access to the system tables, you do not need to change this option. If you do not have read access, the database administrator must create a view of the system tables in another account and give you permission to use that view. In this case, specify the Authorization ID for the account that owns the views of the system tables. If you are using DDCS/2 to connect to a SQL/400 database, specify a comma-separated list of the Collection IDs to which you want to have access in your ODBC application. If you are using DDCS/2 to connect to a SQL/DS database, specify SYSTEM as the owner.
Groups (GRP)	A value that determines which tables you can access. Your system administrator may have placed you in a “group” of users and granted table access to the entire group. If this is the case, set Groups to the names of any groups to which you belong; separate each name with a comma. Alternatively, Groups=ALL lets your application see all table names even if you cannot access the table.
LogonID (UID)	The default logon ID used to connect to your DB2/2 database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID. For DB2/6000 on UNIX, normal UNIX security is used. The LogonID value is your UNIX user ID.

Table 4-1. DB2/2 Connection String Attributes (cont.)

Attribute	Description
Password (PWD)	Password.
CursorBehavior (CB)	CursorBehavior={0|1}. This attribute determines whether cursors are preserved or closed at the end of each transaction. The initial default is 0 (close). Set this attribute to 1 if you want cursors to be held at the current position when the transaction ends. Doing so may impact the performance of your database operations. This attribute is not valid for SQL/DS. When CursorBehavior=1, the driver returns SQL_CB_PRESERVE from SQLGetInfo (SQL_CURSOR_COMMIT_BEHAVIOR). But only Select statements and prepared Update or Delete . . . Where Current of Cursor statements are preserved when the transaction ends. All other prepared statements are closed and deleted.

Data Types

Table 4-2 shows how the DB2/2 data types map to the standard ODBC data types.

Table 4-2. DB2/2 Data Types

DB2/2	ODBC
Char	SQL_CHAR
Char() for Bit Data	SQL_BINARY
Date	SQL_DATE

Table 4-2. DB2/2 Data Types (cont.)

DB2/2	ODBC
Decimal	SQL_DECIMAL
Float	SQL_DOUBLE
Integer	SQL_INTEGER
Long Varchar[1]	SQL_LONGVARCHAR
Long Varchar for Bit Data[1]	SQL_LONGVARBINARY
Smallint	SQL_SMALLINT
Time	SQL_TIME
Timestamp	SQL_TIMESTAMP
Varchar[2]	SQL_VARCHAR
Varchar[3]	SQL_LONGVARCHAR
Varchar() for Bit Data[2]	SQL_VARBINARY
Varchar() for Bit Data[3]	SQL_LONGVARBINARY

[1]Not supported for SQL/400

[2]For DB2/2, DB2/6000, and SQL/400; SQL/DS and MVS DB2 up to 254

[3]For SQL/DS and MVS DB2 over 254

Note: The Graphic, Vargraphic, and Long Vargraphic data types are not supported

Isolation and Lock Levels Supported

DB2/2 supports isolation levels 0 (read uncommitted), 1 (read committed), and 2 (repeatable read). It supports record-level locking. See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The DB2/2 driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, "ODBC API and Scalar Functions," on page 332. The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The DB2/2 family of database systems supports a single connection and multiple statements per connection.

Go To

5 dBASE Driver

The dBASE driver supports the following files in the following operating environments:

File Type	Operating Environments
dBASE II	Windows, Windows 95, and Windows NT
dBASE III, IV, and V	Windows, Windows 95, Windows NT, OS/2, Macintosh, and UNIX
Clipper	Windows, Windows 95, Windows NT, OS/2, Macintosh, and UNIX
FoxPro and FoxBASE	Windows, Windows 95, Windows NT, OS/2, Macintosh, and UNIX

The dBASE driver executes the SQL statements directly on dBASE-compatible files. You do not need to own the dBASE product to access these files.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the dBASE driver.

Configuring Data Sources

UNIX

Note: In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the .odbc.ini file using the attributes in Table 5-1 on page 73. For information about this file, see Appendix E, “ODBC.INI,” on page 346.

To configure a dBASE data source:

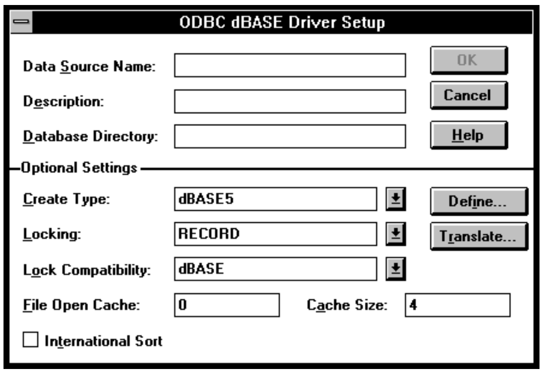
1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the dBASE driver and click **OK**.

On the Macintosh, select dBase Driver from the list of installed drivers.

If you are configuring an existing data source, select the data source name and click **Setup** (if you are using Apple’s ODBC Driver Manager on the Macintosh, this button is labeled **Modify**).

The setup dialog box appears.



Data Source Name:
Description:
Database Directory:
Optional Settings
Create Type: dBASE5
Locking: RECORD
Lock Compatibility: dBASE
File Open Cache: 0
Cache Size: 4
☐ International Sort

3 Specify values as follows:

Data Source Name: A string that identifies this dBASE data source configuration in ODBC.INI. Examples include “Accounting” or “dBASE Files.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “dBASE files in C:\ACCOUNTS.”

Database Directory: A path specification to the directory that contains the database files. If none is specified, the current working directory is used.

The following values are optional:

Create Type: The type of table or index to be created on a Create Table or Create Index statement. Select dBASE II, dBASE III, dBASE IV, dBASE V, Clipper, FoxBASE, FoxPro1, or FoxPro25. The default is dBASE V for all environments except the Macintosh. *dBASE II is not supported for OS/2, UNIX, or Macintosh.*

On the Macintosh, the default create type is FoxPro25.

Locking: The level of locking for the database file (FILE, RECORD, or NONE). FILE locks all of the records in the table. RECORD (the default) locks only the records affected by the statement. NONE offers the best performance but is intended only for single-user environments.

On the Macintosh, the default locking level is NONE.

Lock Compatibility: The locking scheme the driver uses when locking records. Select dBASE, Q+E, Q+EVirtual, Clipper, or Fox. The default is dBASE. These values determine locking support as follows:

- dBASE specifies Borland-compatible locking.
- Q+E specifies that locks be placed on the actual bytes occupied by the record. Only applications that use the dBASE driver can read and write to the database. Other applications are locked out of the table completely (they cannot even read other records). This locking is

compatible with earlier versions of Q+E products.

- Q+EVirtual specifies that locks be placed on bytes beyond the physical end-of-file. Q+EVirtual is the same as Q+E except that other applications can open the table and read the data.

  The advantage of using a Q+E locking scheme over dBASE locking is that, on Inserts and Updates, Q+E locks only individual index tags, while dBASE locks the entire index.

- Clipper specifies Clipper-compatible locking.
- Fox specifies FoxPro- and FoxBASE-compatible locking.

  If you are accessing a table with an application that uses the dBASE driver, your locking scheme does not have to match the Create Type. However, if you are accessing a table with two applications, and only one uses the dBASE driver, set your locking scheme to match the other application. For example, you do not have to set this value to Fox to work with a FoxPro table. But if you are using a FoxPro application simultaneously with an application using the dBASE driver on the same set of tables, set this value to Fox to ensure that your data does not get corrupted.

On the Macintosh, the default lock compatibility is Fox.

File Open Cache: An integer value to specify the maximum number of used file handles to cache. For example, the value 4 specifies that when a user opens and closes four tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is increased performance. The disadvantage is that a user who specifies file locking on open may get a locking conflict even though no one appears to have the file open. The default is 0, which means no file open caching.

Cache Size: The number of 64K blocks the driver uses to cache database records. The higher the number of blocks, the better the performance. The maximum number of blocks you can set depends on the system memory available. If the cache size is greater than 0, when browsing backwards, you will not be able to see updates made by other users until you reexecute the Select statement. The default is 4.

International Sort: A setting to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Select this check box to use the international sort order as defined by your operating system. International sort order is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (see your operating system documentation). Leave this box blank to use the ASCII sort order. ASCII sort order is case-sensitive, where uppercase letters precede lowercase letters (*B* precedes *a*).

Mac File Info: On the Macintosh, click this button to specify the creator and file types. A dialog box is displayed to enable you to specify 4-character, case-sensitive values for the following:

- Creator (default is FOXX)
- Data Base file (default is F+DB)
- Memo file (default is F+DT)
- Single Entry Index file (default is F+IX)
- Multiple Entry Index file (default is FCDX).

Use Long Names Set this check box to use long filenames as table names. The maximum table name length is specific to the environment in which you are running (for example, in Windows 95, the maximum table name length is 128).

Use Long Qualifiers Set this check box to use long pathnames as table qualifiers. When you set this check box, pathnames can be up to 255 characters. The default length for pathnames is 128 characters.

4 Click **Define** to define the index attributes for your data files. See the section "Defining Index Attributes" on page 70 for step-by-step instructions.

5 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a

translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

UNIX

Note: In the UNIX environment, there is no graphical interface. To perform a translation, you must edit the .odbc.ini file and add to the data source section the TranslationDLL keyword and optionally the TranslationOption keyword:

Keyword	**Definition**
TranslationDLL	Full path of translation DLL
TranslationOption	ASCII representation of the 32-bit integer translation option

The INTERSOLV OEM ANSI translator is shipped on UNIX to provide an example of how to create a translation DLL.

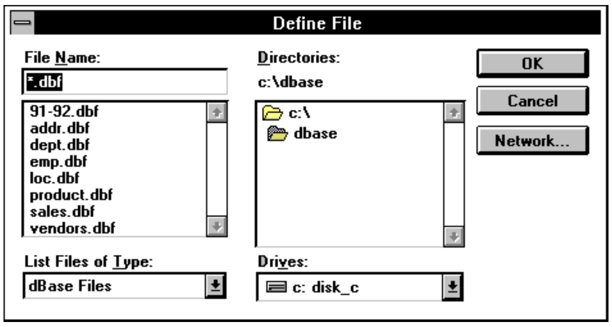
6 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Defining Index Attributes

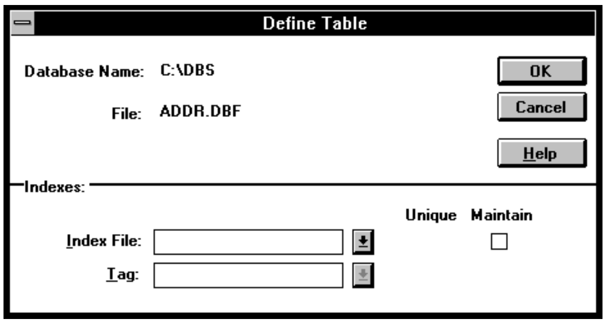
Because dBASE lets you create index files that have different names than their corresponding data files, you must tell the driver what index files are associated with the dBASE file. The driver updates the indexes for you, which ensures that they match the records in the dBASE file. The driver also makes indexes available for optimizing queries. It is not necessary to mark production .MDX files or structured .CDX files as maintained, as the driver maintains them for you. However, you may wish to use this method to mark a tag as unique.

To define the index files that are associated with a dBASE file, take the following steps:

1 Click **Define** in the dBASE setup dialog box, which you can access through the ODBC Administrator. The standard file open dialog box for your system appears. The following figure shows the Windows Define File dialog box.



2 Select a dBASE file and click **OK** to define the special indexes using the Define Table dialog box.



3 The upper section of the dialog box displays the directory name and filename that contains the data file.

4 The lower section of the dialog box displays the index information for the data file. The Index File drop-down list lets you select any index file in the database directory. If the index file is in a different directory, you must provide the full pathname.

Select the Maintain check box to associate this index file with your dBASE file.

To specify that an index file is unique, select the Unique check box that appears at the right of the index filename.

5 If the selected index has an .MDX or .CDX extension, you cannot mark the index file as unique. Instead, you may mark the tags within the index as unique. To do so, select the tag name in the Tag drop-down list and select the Unique check box that appears at the right of the tag name.

6 Click **OK** to save this information.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to the ODBC.INI file.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for dBASE is

```
DSN=DBASE FILES;LCK=NONE;IS=  0
```

Table 5-1 gives the long and short names for each attribute, as well as a description. To configure a data source in the UNIX environment, you must edit the .odbc.ini file. This file accepts only long names for attributes.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 5-1. dBASE Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies a dBASE data source configuration in ODBC.INI. Examples include “Accounting” or “dBASE Files.”
Database (DB)	The directory in which the dBASE files are stored
CreateType (CT)	CreateType={dBASE2|dBASE3|dBASE4|dBASE5|Clipper|FoxBASE|FoxPro1|FoxPro25}. The type of table or index to be created on a Create Table or Create Index statement. The initial default is dBASE5 for all environments except the Macintosh; on the Macintosh, the initial default is FoxPro25. *dBASE2 is not supported in the OS/2, UNIX, or Macintosh operating environments .*

Table 5-1. dBASE Connection String Attributes (cont.)

Attribute	Description
LockCompatibility (LCOMP)	LockCompatibility={Q+E|Q+EVirtual|dBASE|Clipper|Fox}. The locking scheme to be used in your dBASE tables. • LockCompatibility=Q+E specifies that locks be placed on the actual bytes occupied by the record. Only applications that use the dBASE driver can read and write to the database. Other applications are locked out of the table completely (they cannot even read other records). This locking is compatible with earlier versions of Q+E products. • LockCompatibility=Q+EVirtual specifies that locks be placed on bytes beyond the physical end-of-file. Q+EVirtual is the same as Q+E except that other applications can open the table and read the data The advantage of using a Q+E locking scheme over dBASE locking is that, on Inserts and Updates, Q+E locks only individual index tags, while dBASE locks the entire index. • LockCompatibility=dBASE specifies Borland-compatible locking. This is the initial default • LockCompatibility=Clipper specifies Clipper-compatible locking. • LockCompatibility=Fox specifies FoxPro- and FoxBASE-compatible locking. This is the initial default on the Macintosh.

Table 5-1. dBASE Connection String Attributes (cont.)

Attribute	Description
LockCompatibility (LCOMP) (*cont.*)	If you are accessing a table with an application that uses the dBASE driver, your locking scheme does not have to match the Create Type. However, if you are accessing a table with two applications, and only one uses the dBASE driver, set your locking scheme to match the other application. For example, you don’t have to set LCOMP=Fox to work with a FoxPro table. But if you are using a FoxPro application simultaneously with an application using the dBASE driver on the same set of tables, set LCOMP=Fox to ensure that your data does not get corrupted.
Locking (LCK)	Locking={NONE|RECORD|FILE}. This attribute determines the level of locking for the database tables. Locking=NONE offers the best performance but is intended only for single-user environments. This is the initial default on the Macintosh. Locking=RECORD locks only the records affected by the statement. This is the initial default. Locking=FILE locks all of the records in the table.
FileOpenCache (FOC)	The maximum number of used file handles to cache. For example, when FileOpenCache=4, and a user opens and closes four files, the files are not actually closed. The driver keeps them open so that if another query uses one of these files, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The disadvantage is that a user who tries to open the file exclusively may get a locking conflict even though no one appears to have the file open. The initial default is 0.

Table 5-1. dBASE Connection String Attributes (cont.)

Attribute	Description
CacheSize (CSZ)	The number of 64K blocks the driver uses to cache database records. The higher the number of blocks, the better the performance. The maximum number of blocks you can set depends on the system memory available. If the cache size is greater than 0, when browsing backwards, you will not be able to see updates made by other users until you reexecute the Select statement. The initial default is 4.
IntlSort (IS)	IntlSort={0|1}. This attribute determines the order that records are retrieved when you issue a Select statement with an Order By clause. If IntlSort=1, the driver uses the international sort order as defined by your operating system. The sort is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (see your operating system documentation). If IntlSort=0 (the initial default), the driver uses the ASCII sort order, where uppercase letters precede lowercase letters (*B* precedes *a*).
ModifySQL (MS)	ModifySQL={0|1}. This attribute is provided for backward compatibility with earlier versions of Q+E products. It determines whether the driver modifies SQL statements to conform to ODBC specifications or passes the SQL statement directly to dBASE. Specify ModifySQL=1 to have the driver modify the SQL statement to conform to ODBC specifications. Specify ModifySQL=0 to have the driver understand SQL dialects found in earlier drivers. The default is 1.
Compatibility (COMP)	Compatibility={DBASE|ANSI}. This attribute is provided for backward compatibility with earlier versions of Q+E products. Use Compatibility=DBASE for backward compatibility; use Compatibility=ANSI for portability. The default is ANSI.

Table 5-1. dBASE Connection String Attributes (cont.)

Attribute	**Description**
UltraSafeCommit (USF)	UltraSafeCommit={0|1}. This attribute specifies when the driver flushes the file cache. If UltraSafeCommit=1, the driver updates directory entries after each Commit. This decreases performance. If UltraSafeCommit=0 (the default) the driver updates the directory entry when the file is closed. In this case, a machine "crash" before closing the file causes newly inserted records to be lost.
ExtensionCase (EC)	ExtensionCase={LOWER|UPPER}. This attribute specifies whether upper- or lowercase file extensions are accepted. If ExtensionCase=LOWER, lowercase extensions are accepted. If ExtensionCase=UPPER (the default), uppercase extensions are accepted.
MacFileInfo (MFI)	On Macintosh systems, four-character, case-sensitive values that specify the following in the order shown: • Creator (initial default is FOXX) • Data Base file (initial default is F+DB) • Memo file (initial default is F+DT) • Single Entry Index file (initial default is F+IX) • Multiple Entry Index file (initial default is FCDX). The values are specified in a comma-separated list. For example, MacFileInfo=ABCD,EFGH,IJKL,MNOP,QRST.

Table 5-1. dBASE Connection String Attributes (cont.)

Attribute	Description
UseLongNames (ULN)	UseLongNames={0|1}. This attribute specifies whether the driver uses long filenames as table names. The default is 0, do not use long filenames. If UseLongNames=1, the driver uses long filenames. The maximum table name length is specific to the environment in which you are running (for example, in Windows 95, the maximum table name length is 128).
UseLongQualifiers (ULQ)	UseLongQualifiers={0|1}. This attribute specifies whether the driver uses long pathnames as table qualifiers. The default is 0, do not use long pathnames (the default length of pathnames is 128 characters). If UseLongQualifiers=1, the driver uses long pathnames (up to 255 characters).

Data Types

Table 5-2 shows how dBASE data types map to the standard ODBC data types. These dBASE data types can be used in a Create Table statement. For the syntax of the Create Table statement, see Appendix A, "SQL for Flat-File Drivers," on page 299.

Note: A few products can create dBASE files with numbers that do not conform to the precision and scale of the Number column. For example, these products can store 100000 in a column declared as NUMBER(5,2). When this occurs, the dBASE driver displays error 1244, "Unsupported decimal format." To remedy this situation, multiply the nonconforming column by 1, which converts it to the Float data type. For example:

```
SELECT BADCOL * 1 FROM BADFILE
```

BADCOL * 1 is evaluated as an expression and is returned as a float value.

Table 5-2. dBASE Data Types

dBASE	ODBC
Binary[1]	SQL_LONGVARBINARY
Char[2]	SQL_CHAR
Date[3]	SQL_DATE
Float[4]	SQL_DECIMAL
General[5]	SQL_LONGVARBINARY
Logical	SQL_BIT
Memo[3]	SQL_LONGVARCHAR
Numeric	SQL_DECIMAL
Picture[6]	SQL_LONGVARBINARY

[1] dBASE V only
[2] 254 characters maximum (1024 for Clipper)
[3] dBASE III, IV, V, FoxPro, FoxBASE, and Clipper
[4] dBASE IV and V only
[5] FoxPro 2.5 and dBASE V only
[6] FoxPro, FoxBASE, and Clipper

Column Names

Column names in SQL statements (such as, Select, Insert, etc.) can be up to 10 characters long. A column name can contain alphanumeric characters and the hyphen character (-). The first character must be a letter (a through z).

Select Statement

You use a SQL Select statement to specify the columns and records to be read. dBASE Select statements support all of the Select statement clauses as described in Appendix A, "SQL for Flat-File Drivers," on page 299. This section describes the information that is specific to dBASE, which is ROWID.

ROWID Pseudo-Colum

Each dBASE record contains a special column named ROWID. This field contains a unique number that indicates the record's sequence in the database. For example, a table that contains 50 records has ROWID values from 1 to 50 (if no records are marked deleted). You can use ROWID in Where and Select clauses.

ROWID is particularly useful when you are updating records. You can retrieve the ROWID of the records in the database along with the other field values. For example:

```
SELECT last_name, first_name, salary, rowid FROM emp
```

Then you can use the ROWID of the record that you want to update to ensure that you are updating the correct record and no other. For example:

```
UPDATE emp set salary = 40000 FROM emp WHERE rowid=21
```

The fastest way of updating a single row is to use a Where clause with the ROWID. You cannot update the ROWID column.

Select statements that use the ROWID pseudo-column in the Where clause achieve maximum performance only for exact equality matches. If you use range scans instead of exact equality matches, a full table scan is performed. For example:

```
SELECT * FROM emp WHERE rowid=21  //fast search

SELECT * FROM emp WHERE rowid <=25  //full table scan
```

Create Index Statement

The type of index you create is determined by the value of the CreateType attribute, which you set in the setup dialog box or as a connection string option. The index can be

- dBASE II or III (.NDX)
- dBASE IV (.MDX)
- Clipper (.NTX)
- FoxBASE (.IDX)
- FoxPro (.CDX)

The syntax for creating an index is

```
CREATE [UNIQUE] INDEX   index_name ON base_table_name
    (field_name [ASC | DESC] [ ,field_name [ASC | DESC]]
...)
```

index_name is the name of the index file. For FoxPro 2.5 and dBASE IV, this is a tag, which is required to identify the indexes in an index file. Each index for a table must have a unique name.

UNIQUE means that the driver creates an ANSI-style unique index over the column and ensures uniqueness of the keys. Use of unique indexes improves performance. ANSI-style unique indexes are different from dBASE-style unique indexes. With ANSI-style unique indexes, you receive an error message when you try to insert a duplicate value into an indexed field. With dBASE-style unique indexes, you do not see an error message when you insert a duplicate value into an indexed field. This is because only one key is inserted in the index file.

base_table_name is the name of the database file whose index is to be created. The .DBF extension is not required; the driver automatically adds it if it is not present. By default, dBASE IV index files are named *base_table_name.*MDX and FoxPro 2.5 indexes are named *base_table_name.*CDX.

field_name is a name of a column in the dBASE table. You can substitute a valid dBASE-style index expression for the list of field names.

ASC tells dBASE to create the index in ascending order. DESC tells dBASE to create the index in descending order. By default, indexes are created in ascending order. You cannot specify both ASC and DESC orders within a single Create Index statement. For example, the following statement is invalid:

```
CREATE INDEX emp_i ON emp (last_name ASC, emp_id DESC     )
```

Table 5-3 shows the attributes of the different index files supported by the dBASE driver. For each type supported, it provides the following details:

- Whether dBASE-style unique indexes are supported
- Whether descending order is supported
- The maximum size supported for key columns
- The maximum size supported for the column specification in the Create Index statement
- Whether production/structural indexes are supported

Table 5-3. dBASE-Compatible Index Summary

Create Type/Ext	dBASE UNIQUE	DESC	Max Size of Key Column	Max Size of Column Spec.	Production / Structural Indexes
dBASE II .NDX	No	No	100	99	No
dBASE III .NDX	Yes	No	100	219	No
Clipper .NTX	Yes	Yes	250	255	No

Table 5-3. dBASE-Compatible Index Summary (cont.)

Create Type/Ext.	dBASE UNIQUE	DESC	Max Size of Key Column	Max Size of Column Spec.	Production / Structural Indexes
FoxBASE .IDX*	Yes	No	100	219	No
dBASE IV and V .MDX	Yes	Yes	100	220	Yes
FoxPro 2.5 .IDX**	Yes	Yes	240	255	No
FoxPro 2.5 .CDX	Yes	Yes	240	255	Yes

* These IDX indexes are also created as the default for FoxPro 1.0

**Compact IDX indexes have the same internal structure as a tag in a CDX file. These indexes can be created if the IDX extension is included with the index name in the Create Index statement.

Drop Index Statement

The syntax for dropping an index is as follows:

```
DROP INDEX  table_name.index_name
```

table_name is the name of the dBASE file without the extension.

For FoxPro 2.5 and dBASE IV, *index_name* is the tag. Otherwise, *index_name* is the name of the index file without the extension.

To drop the index EMPHIRE.NDX, issue the following statement:

```
DROP INDEX emp.emphire
```

Pack Statement

When records are deleted from a dBASE file, they are not removed from the file. Instead, they are marked as having been deleted. Also, when memo fields are updated, space may be wasted in the files. To remove the deleted records and free the unused space from updated memo fields, you must use the Pack statement. It has the following form:

```
PACK filename
```

filename is the name of the dBASE file to be packed. The .DBF extension is not required; the driver automatically adds the extension if it is not present. For example:

```
PACK emp
```

You cannot pack a file that is opened by another user, and you cannot use the Pack statement in manual commit mode.

For the specified file, the Pack statement does the following:

- Removes all deleted records from the file
- Removes the entries for all deleted records from .CDX and .MDX files having the same name as the file
- Compresses unused space in the memo (.DBT or .FPT) file

Locking

With the dBASE driver, you can build and run applications that share dBASE database files on a network. Whenever more than one user is running an application that accesses a shared database file, the applications should lock the records that are being changed. Locking a record prevents other users from locking, updating, or deleting the record.

Levels of Database Locking

The dBASE driver supports three levels of database locking: NONE, RECORD, and FILE. You can set these levels in

- The connection string (LCK=)
- The setup dialog box

No locking offers the best performance but is intended only for single-user environments.

With record or file locking, the system locks the database tables during Insert, Update, Delete, or Select...For Update statements. The locks are released when the user commits the transaction. The locks prevent other users from modifying the locked objects, but they do not lock out readers.

With record locking, only records affected by the statement are locked. Record locking provides better concurrency with other users who also want to modify the table.

With file locking, all the records in the table are locked. File locking has lower overhead and may work better if records are modified infrequently, if records are modified primarily by one user, or if a large number of records are modified.

Using Locks on Local Files

If you use database locking and are accessing files locally (not on a network), run the DOS utility SHARE.EXE before running Windows. If you add SHARE.EXE to your AUTOEXEC.BAT file, it runs automatically each time you boot your computer.

On the Macintosh, if file sharing is not enabled, then all locking is done at the file level. If file sharing is enabled and the file is shared, the locking level implemented is the one specified in the Setup dialog box or the connection string.

Limit on Number of Locks

There is a limit on the number of locks that can be placed on a file. If you are accessing a dBASE file from a server, the limit depends on the server (see your server documentation). If you are accessing a dBASE file locally, the limit depends on the buffer space allocated when SHARE.EXE was loaded (see your DOS documentation). If you are exceeding the number of locks available, you may want to switch to file locking.

How Transactions Affect Record Locks

When an Update or Delete statement is executed, the driver locks the records affected by that statement. The locks are released after the driver commits the changes. Under manual commit mode, the locks are held until the application commits the transaction. Under autocommit mode, the locks are held until the statement is executed.

When a Select...For Update statement is executed, the driver locks a record only when the record is fetched. If the record is updated, the driver holds the lock until the changes are committed. Otherwise, the lock is released when the next record is fetched.

Isolation and Lock Levels Supported

dBASE supports isolation level 1. It supports both file- and record-level locking. See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The dBASE driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 function is supported: SQLSetPos.

The dBASE driver also supports backward and random fetching in SQLExtendedFetch. The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

dBASE supports multiple connections and multiple statements per connection.

Go To

6 Excel Drivers

There are two drivers that support Excel .XLS files. One supports Excel versions 2, 3, and 4 and for the purposes of this book is called the Excel 4 driver. The other supports Excel version 5 and is called the Excel 5 driver.

The Excel 4 and Excel 5 drivers are supported in the Windows, Windows 95, Windows NT, and Macintosh environments.

See the README file shipped with your INTERSOLV DataDirect product for the filenames of the Excel drivers.

System Requirements

There are no software requirements for using the Excel 4 driver, because you do not need the Excel program to use it. It works wherever your Excel version 2, 3, or 4 .XLS files are present.

To use the Excel 5 driver, you must have version 5 of the Excel program installed on the system where you intend to use an Excel 5 database. There are two main reasons for this restriction:

- The Excel 5 driver requires two DLLs to operate. These DLLs, STORAGE.DLL and COMPOBJ.DLL, are installed when you install Excel version 5.

- Because the Excel 5 driver is able only to read data from Excel 5 .XLS files, you must use the Excel 5 program for any manipulation of your Excel 5 databases.

For the Macintosh, you must have OLE for the Macintosh installed.

Using Excel Databases

The way that you create and use Excel databases differs according to which version of Excel and corresponding Excel driver you use. The different methods are described in the following sections.

Excel 4 Databases

The Excel 4 driver lets you create databases using Excel version 2, 3, or 4. An Excel 4 database is a directory of component .XLS worksheet files, each of which contains a single table.

With the Excel 4 driver, you can open any Excel worksheet file that contains a database section. To create a database section in a worksheet, you select a range to be used as a database and choose Excel's Set Database command from the Data menu. The driver accesses the rows and columns from the database section.

The Excel 4 driver allows some database manipulation via SQL statements. The Insert statement is limited to use with the Create Table statement only. The Drop Table statement is also supported. You may not execute Update or Delete statements.

Excel 5 Databases

An Excel 5 database is any Excel 5 .XLS workbook file that contains one or more named lists containing record data. Each named list is treated as a table within the workbook database. The lists must be set up as follows:

- The first row in each list must contain the column labels that identify the fields in each record.
- The name of each list must be a book-level name, not a sheet-level name.
- Although the Excel 5 driver recognizes one list named “Database” per .XLS file, it is recommended that you avoid using this name—both to ensure that the Excel 5 driver recognizes other lists in the file and to prevent future naming conflicts.

See your Excel 5 documentation for more information on creating and naming lists in Excel.

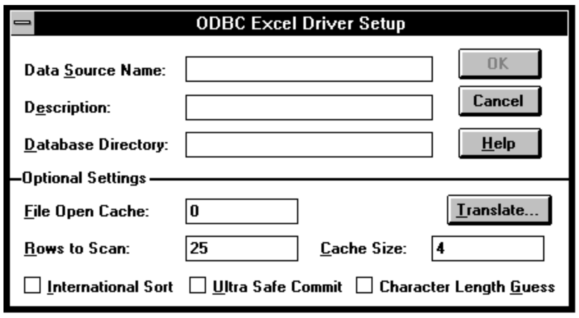
Configuring Data Sources

To configure an Excel data source:

1. Start the ODBC Administrator. A list of data sources appears.
2. If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Excel driver and click **OK**.

   If you are configuring an existing data source, select the data source name and click **Setup** (if you are using Apple’s ODBC Driver Manager on the Macintosh, this button is labeled **Modify**).

The Setup dialog box appears.


ODBC Excel Driver Setup

Data Source Name:
Description:
Database Directory:
Optional Settings
File Open Cache: 0
Rows to Scan: 25
Cache Size: 4
☐ International Sort ☐ Ultra Safe Commit ☐ Character Length Guess

3 Specify values as follows:

Data Source Name:A string that identifies this Excel data source configuration in ODBC.INI. Examples include "Accounting" or "Excel Database."

Description: An optional long description of a data source name. For example, "My Accounting Database" or "Excel .XLS file data."

Database Directory*or* Database Workbook:For the Excel 4 driver, this identifies the directory in which the Excel files are stored. If none is specified, the current working directory is used. For the Excel 5 driver, this identifies the workbook file containing the Excel database.

The following values are optional:

File Open Cache: An integer value that specifies the maximum number of unused file opens to cache. For example, the value 4 specifies that when a user opens and closes four tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is increased performance. The default is 0, which means no file open caching.

Cache Size: The number of 64K blocks the driver uses to cache database records. The higher the number of blocks, the better the performance. The maximum number of blocks you can set depends on the system memory available. If the cache size is greater than 0, when browsing backwards, you will not be able to see updates made by other users until you reexecute the Select statement. The default is 4.

Rows to Scan: An integer value that specifies the number of rows to scan to determine the column data types. The default is 25 rows. A value of 0 means to scan the whole table.

International Sort:A setting to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Select this check box to use the international sort order as defined by your operating system. International sort order is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (see your operating system documentation). Do not set this check box to use the ASCII sort order. ASCII sort order is case-sensitive, where uppercase letters precede lowercase letters (*B* precedes *a*).

Ultra Safe Commit Excel 4 driver attribute only. A value of 0 or 1 that determines when the driver flushes the file cache. If set to 1, the driver updates directory entries after each Commit. This decreases performance. If set to 0 (the default), the driver updates the directory entry when the file is closed. In this case, a machine “crash” before closing the file causes newly inserted records to be lost.

Character Length Guess A value of 0 or 1 that determines whether the driver tries to guess the length of character columns. If set to 0 (the default), the driver does not try to guess the length of character columns; instead, it assumes that all character columns have a length of 255. If set to 1, the driver tries to guess the length.

Mac File Info: Click this button to specify the creator and file type. A dialog box is displayed to enable you to specify 4-character, case-sensitive values for the following:

- Creator (default is XCEL)
- File Type (default is XLS4 for Excel 4.0)

Use Long Qualifiers Set this check box to use long pathnames as table qualifiers. When you set this check box, pathnames can be up to 255 characters. The default length for pathnames is 128 characters.

4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

   The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the default values by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for Excel is

```
DSN=EXCEL FILES;FOC= 4
```

Table 6-1 gives the long and short names for each attribute, as well as a description. To configure a data source in the UNIX environment, you must edit the .odbc.ini file. This file accepts only long names for attributes.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 6-1. Excel Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies an Excel data source configuration in ODBC.INI. Examples include “Accounting” or “Excel Files.”
Database (DB)	For the Excel 4 driver, the directory in which the Excel files are stored. For the Excel 5 driver, the name of the .XLS workbook file containing the Excel database.
ScanRows (SR)	The number of rows to scan to determine the column data types. A value of 0 means to scan the whole table. The initial default is 25.
FileOpenCache (FOC)	The maximum number of unused file opens to cache. For example, when FileOpenCache=4, and a user opens and closes four files, the files are not actually closed. The driver keeps them open so that if another query uses one of these files, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The initial default is 0.
CacheSize (CSZ)	The number of 64K blocks the driver uses to cache database records. The higher the number of blocks, the better the performance. The maximum number of blocks you can set depends on the system memory available. If the cache size is greater than 0, when browsing backwards, you will not be able to see updates made by other users until you reexecute the Select statement. The initial default is 4.

Table 6-1. Excel Connection String Attributes (cont.)

Attribute	**Description**
IntlSort (IS)	IntlSort={0|1}. This attribute determines the order in which records are retrieved when you issue a Select statement with an Order By clause. If IntlSort=1, the driver uses the international sort order as defined by your operating system. The sort is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (see your operating system documentation). If IntlSort=0 (the initial default), the driver uses the ASCII sort order, where uppercase letters precede lowercase letters (*B* precedes *a*).
FieldGuessing (FG)	FieldGuessing={0|1}. This attribute determines whether the driver will try to guess column data types. If FieldGuessing=1 (the default), the driver attempts to guess the data types. If FieldGuessing=0, the driver assumes that all data types are SQL_CHAR.
UltraSafeCommit (USF) Excel 4 attribute only	UltraSafeCommit={0|1}. This attribute determines when the driver flushes the file cache. If UltraSafeCommit=1, the driver updates directory entries after each Commit. This decreases performance. If UltraSafeCommit=0 (the default) the driver updates the directory entry when the file is closed. In this case, a machine “crash” before closing the file causes newly inserted records to be lost.
CharacterLength Guess (CLG)	CharacterLengthGuess={0|1}.This attribute determines whether the driver tries to guess the length of character columns. If set to 0 (the initial default), the driver does not try to guess the length of character columns; instead, it assumes that all character columns have a length of 255. If set to 1, the driver tries to guess the length.

Table 6-1. Excel Connection String Attributes (cont.)

Attribute	**Description**
ModifySQL (MS)	ModifySQL={0|1}. This attribute is provided for backward compatibility with earlier versions of Q+E products. It determines whether the driver modifies SQL statements to conform to ODBC specifications or passes the SQL statement directly to Excel. Specify ModifySQL=1 to have the driver modify the SQL statement to conform to ODBC specifications. Specify ModifySQL=0 to have the driver understand SQL dialects found in earlier drivers. The default is 1.
MacFileInfo (MFI)	On Macintosh systems, 4-character, case-sensitive values that specify the following in the order shown: • Creator (initial default is XCEL) • File Type (initial default is XLS4 for Excel 4) The values are specified in a comma-separated list. For example, MacFileInfo=ABCD,EFGH.
UseLongQualifiers (ULQ)	UseLongQualifiers={0|1}. It specifies whether the driver uses long pathnames as table qualifiers. The default is 0, do not use long pathnames (the default length of pathnames is 128 characters). If UseLongQualifiers=1, the driver uses long pathnames (up to 255 characters).

Data Types

Table 6-2 shows how Excel data types map to the standard ODBC data types. These Excel data types can be used in a Create Table statement with the Excel 4 driver. For the syntax of the Create Table statement, see Appendix A, "SQL for Flat-File Drivers," on page 299.

Table 6-2. Excel Data Types

Excel	ODBC
Char	SQL_VARCHAR
Date	SQL_DATE
Logical	SQL_BIT
Number Float	SQL_DOUBLE
Time	SQL_TIME
Timestamp	SQL_TIMESTAMP

Table and Column Names

Table names are case-sensitive when using the Excel 5 driver. They are not case-sensitive when using the Excel 4 driver.

Excel files contain column names in the first row of the database section or named list that is used as a table. Use these names as the column names in a Select statement. Column names are limited to 32 characters. If column names are lowercase, contain upper- and lowercase, contain blank spaces, or are reserved words, surround them with the grave character (`) (ASCII 96).

For example:

```
SELECT `name` FROM em  p
```

Select Statement

You use a SQL Select statement to specify the columns and records to be read. Excel Select statements support all of the Select statement clauses as described in Appendix A, “SQL for Flat-File Drivers,” on page 299.

Create Table Statement

The Excel 5 driver does not support the Create Table statement.

The Excel 4 driver supports the Create Table statement. The Insert statement can be used to load the newly created table, but this must be done during the same connection in which the Create Table statement was issued. Once the connection used to create the table is closed, the Insert statement no longer operates on that table. Future rows need to be inserted directly through Excel. The Excel driver creates tables in Microsoft Excel version 4.0 (BIFF3) format.

ODBC Conformance Level

The Excel drivers support the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 function is supported: SQLSetPos.

The Excel drivers support backward and random fetching in SQLExtendedFetch. The drivers support the minimum SQL grammar.

Number of Connections and Statements Supported

The Excel drivers support multiple connections and multiple statements per connection.

Go To

7 INFORMIX 4 Driver

The INFORMIX 4 driver supports the INFORMIX database system in the Windows environment using version 4.*x* of the INFORMIX-Net or INFORMIX-Star product.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the INFORMIX 4 driver.

System Requirements

INFORMIX 4 system requirements are as follows:

- You must have version 4.*x* of the INFORMIX-Net or INFORMIX-Star product from INFORMIX.
- The message files RDS.IEM and SQL.IEM must be located in the INFORMIX subdirectory MSG.
- The environment variable INFORMIXDIR must be set to the directory where you have installed the INFORMIX client. For example:

  ```
  SET INFORMIXDIR=C:\INFORMIX
  ```

  You can include this line in your AUTOEXEC.BAT file.
- The Informix-Net package includes LDLLSQLW.DLL. The path to this DLL must be in your PATH environment variable.

Configuring Data Sources

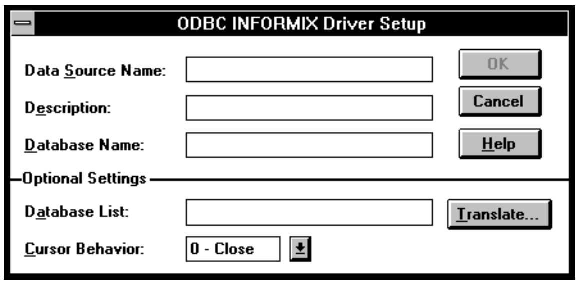
To configure an INFORMIX data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the INFORMIX driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.


ODBC INFORMIX Driver Setup

Data Source Name:

Description:

Database Name:

Optional Settings

Database List:

Cursor Behavior: 0 - Close

3 Specify values as follows:

Data Source Name:A string that identifies this INFORMIX data source configuration in ODBC.INI. Examples include “Accounting” or “INFORMIX on Serv #1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “INFORMIX tables on server number 1.”

Database Name: The name of the database to which you want to connect by default.

The following values are optional:

Database List: The list of databases that will be available in the logon dialog box. Separate the names with commas.

Cursor Behavior: Select Preserve if you want cursors to be held at the current position when the transaction ends. Otherwise, leave this set to Close. Selecting Preserve may impact the performance of your database operations.

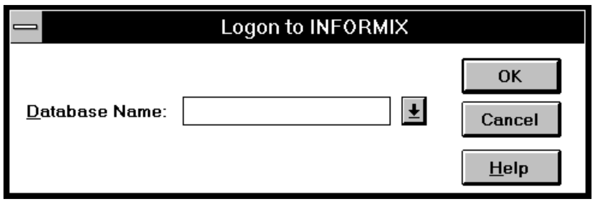
4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

   The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For INFORMIX, the dialog box is as follows:



In this dialog box, do the following:

1 Type the name of the database you want to access or select the name from the Database Name drop-down list, which displays the names you specified in the setup dialog box.

2 Click **OK** to complete the logon and to update this value in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_nam e[;attribute=valu e[;attribute=valu e].
..]
```

An example of a connection string for INFORMIX 4 is

```
DSN=INFORMIX on Serv #1;DB=PAYROL   L
```

Table 7-1 gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 7-1. INFORMIX 4 Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies an INFORMIX data source configuration in ODBC.INI. Examples include “Accounting” or “INFORMIX on Serv #1.”
Database (DB)	The name of the database to which you want to connect
CursorBehavior (CB)	CursorBehavior={0|1}. This attribute determines whether cursors are preserved or closed at the end of each transaction. The initial default is 0 (close). Set this attribute to 1 if you want cursors to be held at the current position when the transaction ends. Doing so may impact the performance of your database operations

Data Types

Table 7-2 shows how INFORMIX 4 data types map to the standard ODBC data types.

Table 7-2. INFORMIX 4 Data Types

INFORMIX 4	ODBC
Byte*	SQL_LONGVARBINARY
Char	SQL_CHAR
Date	SQL_DATE

Table 7-2. INFORMIX 4 Data Types (cont.)

INFORMIX 4	ODBC
Datetime year to fraction(5)	SQL_TIMESTAMP
Decimal	SQL_DECIMAL
Float	SQL_DOUBLE
Integer	SQL_INTEGER
Money	SQL_DECIMAL
Serial	SQL_INTEGER
Smallfloat	SQL_REAL
Smallint	SQL_SMALLINT
Text*	SQL_LONGVARCHAR
Varchar*	SQL_VARCHAR

* Not supported for Standard Engine Databases

Note: The Interval data type is not supported. Existing columns of this type are mapped to the ODBC SQL_CHAR data type.

Isolation and Lock Levels Supported

If connected to an Online Server, INFORMIX 4 supports isolation levels 0 (read uncommitted), 1 (read committed), and 3 (serializable). The default is 1. The Standard Engine supports isolation level 0 (read uncommitted) only.

INFORMIX 4 also supports an alternative isolation level 1, called cursor stability. Your ODBC application can use this isolation level by calling SQLSetConnectOption (1040,1).

Additionally, if transaction logging has not been enabled for your database, then transactions are not supported by the driver (the driver is always in auto-commit mode).

INFORMIX 4 supports page-level locking.

See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The INFORMIX 4 driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 functions are supported:

- SQLBrowseConnect
- SQLProcedures

The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The INFORMIX 4 database system supports a single connection and multiple statements per connection.

Go To

8 INFORMIX 5 and INFORMIX 7 Drivers

The INFORMIX 5 driver supports multiple connections to the INFORMIX database system version 5 in the Windows, Windows 95, Windows NT, OS/2, and Macintosh environments using version 5.x of the INFORMIX-Net or INFORMIX-Star product.

The INFORMIX 5 driver supports only single connections to the INFORMIX database systems version 5 and 6 in the following two UNIX environments: AIX and Solaris for SPARC.

The INFORMIX 7 driver supports multiple connections to the INFORMIX database system version 7 in the following three UNIX environments: AIX, HP-UX, and Solaris for SPARC.

See the README file shipped with your INTERSOLV DataDirect product for the filenames of the INFORMIX 5 and INFORMIX 7 drivers.

System Requirements

To access remote INFORMIX databases you need version 5.*x* of the INFORMIX-Net or INFORMIX-Star product from INFORMIX.

The message files RDS.IEM and SQL.IEM must be located in the INFORMIX subdirectory MSG.

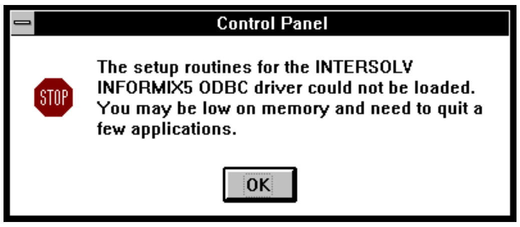
Windows, Windows 95, and Windows NT

The environment variable INFORMIXDIR must be set to the directory where you have installed the INFORMIX client. For example:

```
SET INFORMIXDIR=C:\INFORMIX
```

This line can be included in your AUTOEXEC.BAT file.

The INFORMIX-Net 5.0.1 package includes ISQLI501.DLL. The path to this DLL must be in your PATH environment variable. If it is not, a message similar to the following appears:


Control Panel

The setup routines for the INTERSOLV INFORMIX5 ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

OS/2

The environment variable INFORMIXDIR must be set to the directory where you have installed the INFORMIX client. For example:

```
SET INFORMIXDIR=C:\INFORMIX
```

This line can be included in your CONFIG.SYS file.

The INFORMIX-Net 5.0.1 package includes ISQLI501.DLL. The path to this DLL must be in your LIBPATH environment variable defined in CONFIG.SYS.

UNIX (AIX, HP-UX, and Solaris for SPARC)

The environment variable INFORMIXDIR must be set to the directory where you have installed the INFORMIX client.

Go To

For example, the following syntax is valid for C-shell users:

```
setenv INFORMIXDIR /databases/informi   x
```

For Bourne- or Korn-shell users, the following syntax is valid:

```
INFORMIXDIR=/databases/informix;export INFORMIXDI    R
```

For information about how to access a remote server, refer to *INFORMIX-NET/INFORMIX-STAR Installation and Configuration Guide—Client/Server Interface Version 5.0.*

If Version 6 of INFORMIX Online or SE is being used on the workstation, two additional environment variables must be set. The INFORMIXSERVER variable must be set to the name of the INFORMIX server (as defined in your $INFORMIXDIR/ext/sqlhosts file). The SQLEXEC variable must be set to $INFORMIXDIR/lib/sqlrm. For further details, refer to the *INFORMIX Online Administrator's Manual* Volume 2 or the *INFORMIX UNIX Installation Guide for Version 6*.

Macintosh

System requirements are as follows:

- 8 MB of memory.
- For systems prior to version 7.5, you must use MacTCP version 1.1 or greater. For system 7.5, you must use MacTCP version 2.0.6.
- The INFORMIX-Net for Macintosh provides a Control Panel called setnet. This Control Panel sets the environment variables used by INFORMIX-Net. The INFORMIXDIR environment variable must be set to the folder where you have installed the INFORMIX client software.
- You also must select the MacTCP INET you want to use.
- MacTCP INET version 5.02 or later must be running to access remote databases.

Configuring Data Sources

UNIX

Note: In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the .odbc.ini file using the attributes in Table 8-1. For information about this file, see Appendix E, “ODBC.INI,” on page 346.

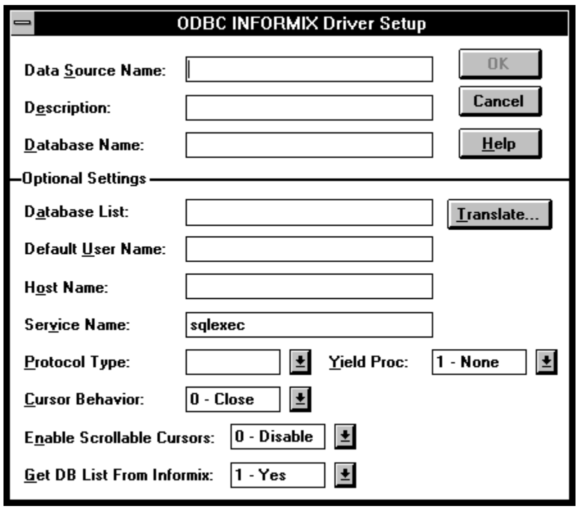
To configure an INFORMIX 5 or INFORMIX 7 data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the INFORMIX 5 or INFORMIX 7 driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.



Data Source Name:	
Description:	
Database Name:	

Optional Settings

Database List:	
Default User Name:	
Host Name:	
Service Name:	sqlexec
Protocol Type:	
Yield Proc:	1 - None
Cursor Behavior:	0 - Close
Enable Scrollable Cursors:	0 - Disable
Get DB List From Informix:	1 - Yes

3 Specify values as follows:

Data Source Name:A string that identifies this INFORMIX data source configuration in ODBC.INI. Examples include “Accounting” or “INFORMIX-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “INFORMIX 5 files on Server number 1.”

Database Name: The name of the database to which you want to connect by default.

The following values are optional:

Database List: The list of databases that will be displayed in the logon dialog box if Get DB List From Informix is set to 0. If Get DB List From Informix is set to 1, the list of databases that will be displayed in the logon dialog box is created from the database list returned from the INFORMIX server.

When you specify databases, separate the names with commas.

Default User Name: The name of the user as specified on the INFORMIX server.

Host Name: The name of the machine on which the INFORMIX server resides.

Service Name: The name of the service as it appears on the host machine. This service is assigned by the system administrator. The name you specify is displayed in the INFORMIX Server Options dialog box.

Protocol Type: The protocol used to communicate with the server. Specify one or more values; separate the names with commas. Values can be FTP, IPX, LANMAN, TCP-IP. This list is displayed in the Logon Options dialog box.

Yield Proc: A value of 0, 1, 2, or 3 that determines whether you can work in other Windows applications when INFORMIX is busy. This attribute is useful to users of ODBC applications.

- 0 (peek and dispatch), which causes the driver to check the Windows message queue and send any messages to the appropriate Windows application.
- 1 (no yielding, the default), which does not let you work in other applications.
- 2 (INFORMIX's yield procedure), which uses INFORMIX's default yield procedure.
- 3 (dispatch via Windows Yield function), which turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window.

It is recommended that you use the value 1.

Cursor Behavior: Select Preserve if you want cursors to be held at the current position when the transaction ends. Otherwise, leave this set to Close. Selecting Preserve may impact the performance of your database operations.

Enable Scrollable Cursors: A value of 0 or 1 that determines whether the driver provides scrollable cursors. The default value is 0 (no use of scrollable cursors). The INFORMIX drivers can use scollable cursors only if there are no long columns (SQL_LONGVARCHAR or SQL_LONGVARBINARY) in a Select list. If you set this option to use scrollable cursors (a value of 1), you must not include long columns in the Select list.

Get DB List From Informix: A value of 0 or 1 that determines whether the driver requests the database list to be returned from the INFORMIX server or from the database list that the user entered at driver setup.

When set to 1, the default, the driver requests the database list from the INFORMIX server. When set to 0, it uses the list that was entered by the user at driver setup.

4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

Note: In the UNIX environment, there is no graphical interface. To perform a translation, you must edit the .odbc.ini file and add to the data source section the TranslationDLL keyword and optionally the TranslationOption keyword:

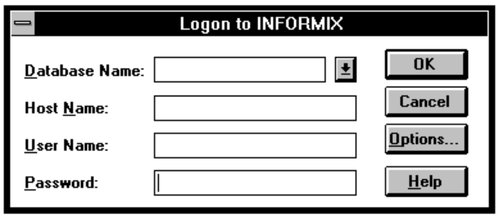
Keyword	**Definition**
TranslationDLL	Full path of translation DLL
TranslationOption	ASCII representation of the 32-bit integer translation option

The INTERSOLV OEM ANSI translator is shipped on UNIX to provide an example of how to create a translation DLL.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For INFORMIX 5 or INFORMIX 7, the dialog box is as follows:



In this dialog box, do the following:

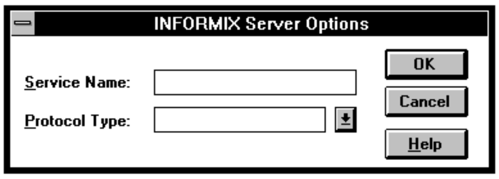
1 Type the name of the database you want to access or select the name from the Database Name drop-down list, which displays the names you specified in the setup dialog box if during setup you specified a value of 0 for the connection option Get DB List From Informix. Otherwise, the names displayed in this drop-down list are returned from the INFORMIX server.

2 Type the name of the server (host name) on which INFORMIX resides.

3 If required, type your user name as specified on the INFORMIX server.

4 If required, type your password.

5 Optionally, click **Options** to display the INFORMIX Server Options dialog box. This dialog box enables you to change the Service Name and Protocol Type that you specified in the setup dialog box. Click **OK** to accept any changes you make.



6 Click **OK** to complete the logon and to update these values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for INFORMIX is

```
DSN=INFORMIX TABLES;DB=PAYROLL
```

Table 8-1 gives the long and short names for each attribute, as well as a description. To configure a data source in the UNIX environment, you must edit the .odbc.ini file. This file accepts only long names for attributes.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 8-1. INFORMIX5 and INFORMIX7 Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies an INFORMIX data source configuration in ODBC.INI. Examples include “Accounting” or “INFORMIX-Serv1.”
Database (DB)	The name of the database to which you want to connect
HostName (HOST)	The name of the machine on which the INFORMIX server resides.

Table 8-1. INFORMIX5 and INFORMIX7 Connection String Attributes (cont.)

Attribute	**Description**
LogonID (UID)	Your user name as specified on the INFORMIX server.
Password (PWD)	A password.
Service (SERV)	The name of the service as it appears on the host machine. This service is assigned by the system administrator.
Protocol (PRO)	Protocol={FTP|IPX|LANMAN|TCP-IP}. The protocol used to communicate with the server. You can specify one or more values; separate the names with commas.
YieldProc (YLD)	YieldProc={0|1|2|3}. This attribute determines if you can work in other Windows applications when INFORMIX is busy. It is useful to users of ODBC applications. • YieldProc=0 (peek and dispatch) causes the driver to check the Windows message queue and send any messages to the appropriate Windows application. • YieldProc=1 (no yielding, the initial default) does not let you work in other applications. • YieldProc=2 (INFORMIX's yield procedure) uses INFORMIX's default yield procedure. • YieldProc=3 (dispatch via Windows Yield function) turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window. It is recommended that you use YieldProc=1. YieldProc=0, YieldProc=2, and YieldProc=3 do not work with all Windows applications.

Table 8-1. INFORMIX5 and INFORMIX7 Connection String Attributes (cont.)

Attribute	Description
CursorBehavior (CB)	CursorBehavior={0|1}. This attribute determines whether cursors will be preserved or closed at the end of each transaction. The initial default is 0 (close). Set this attribute to 1 if you want cursors to be held at the current position when the transaction ends. The value CursorBehavior=1 may impact the performance of your database operations
EnableScrollable Cursors (ESC)	EnableScrollableCursors={0|1}. This attribute determines whether the driver provides scrollable cursors. The initial default value is 0 (no use of scrollable cursors). The INFORMIX driver can use scollable cursors only if there are no long columns (SQL_LONGVARCHAR or SQL_LONGVARBINARY) in a Select list. If you set this option to use scrollable cursors (EnableScrollableCursors=1), you must not include long columns in the Select list
GetDBListFrom Informix (GDBLFI)	GetDBListFromInformix={0|1}. This attribute determines whether the driver requests the database list to be returned from the INFORMIX server or from the database list that the user entered at driver setup. When set to 1, the initial default, the driver requests the database list from the INFORMIX server. When set to 0, it uses the list that was entered by the user at driver setup

Data Types

Table 8-2 shows how the INFORMIX 5 and INFORMIX 7 data types map to the standard ODBC data types.

Table 8-2. INFORMIX5 and INFORMIX7 Data Types

INFORMIX5 and 7	ODBC
Byte*	SQL_LONGVARBINARY
Char	SQL_CHAR
Date	SQL_DATE
Datetime year to fraction(5)	SQL_TIMESTAMP
Decimal	SQL_DECIMAL
Float	SQL_DOUBLE
Integer	SQL_INTEGER
Money	SQL_DECIMAL
Serial	SQL_INTEGER
Smallfloat	SQL_REAL
Smallint	SQL_SMALLINT
Text*	SQL_LONGVARCHAR
Varchar*	SQL_VARCHAR

* Not supported for Standard Engine Databases

Note: The Interval data type is not supported. Existing columns of this type are mapped to the ODBC SQL_CHAR data type.

Isolation and Lock Levels Supported

If connected to an Online Server, INFORMIX 5 and INFORMIX 7 support isolation levels 0 (read uncommitted), 1 (read committed), and 3 (serializable). The default is 1. The Standard Engine supports isolation level 0 (read uncommitted) only.

INFORMIX 5 and INFORMIX 7 also support an alternative isolation level 1, called cursor stability. Your ODBC application can use this isolation level by calling SQLSetConnectOption (1040,1).

Additionally, if transaction logging has not been enabled for your database, then transactions are not supported by the driver (the driver is always in auto-commit mode).

INFORMIX 5 and INFORMIX 7 support page-level locking.

See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The INFORMIX 5 and INFORMIX 7 drivers supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 functions are supported:

- SQLBrowseConnect
- SQLProcedures

The driver also supports scrollable cursors with SQLExtendedFetch if the connection attribute EnableScrollableCursors is set to 1. The driver supports the core SQL grammar.

Number of Connections and Statements Supported

The INFORMIX 5 driver supports multiple connections to the INFORMIX database system in the Windows, Windows 95, Windows NT, OS/2, and Macintosh environments and multiple statements per connection. The INFORMIX 5 driver supports only single connections to the INFORMIX database system in the AIX and Solaris environments.

The INFORMIX 7 driver supports multiple connections to the INFORMIX database system in the AIX, HP-UX, and Solaris environments.

Go To

9 INGRES Drivers

Two INGRES drivers are available:

- One supports INGRES Release 6.4/04 in the Windows, Windows 95, Windows NT, OS/2, AIX, and Solaris environments.
- The other supports Release 6.4/03 and earlier in the Windows environment.

See the README file shipped with your INTERSOLV DataDirect product for the filenames of the INGRES drivers.

System Requirements

To access INGRES databases from your client workstation you must have the INGRES/Net or INGRES/Star product from Ingres installed on both your client and server nodes.

Windows

To access INGRES, you must have the following software installed:

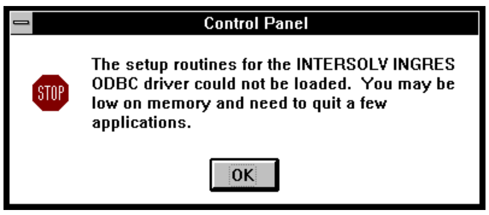
- For INGRES Release 6.4/03 and earlier, you must have Networking for DOS Release 6.4/01 (net.msd/00) and Networking for Windows Release 6.4/01 (net.win/02).
- For INGRES Release 6.4/04, you must have Networking for Windows Release 6.4/04 (net.win/00).

You must have the environment variable II_SYSTEM set to the directory where you installed the INGRES/Net or INGRES/Star product. For example:

```
SET II_SYSTEM= C:
```

In the [386Enh] section of your SYSTEM.INI file, set the NetHeapSize attribute to a number from 16 to 64. When using the NetBIOS protocol, include the line II_NETBIOS_NODE in your AUTOEXEC.BAT file for proper multiuser access to INGRES.

If you are using INGRES 6.4/03 and attempt to configure a data source without the files CLW.DLL, ADFW.DLL, LIBQW.DLL, and GCCW.DLL either on your path or in your Windows SYSTEM directory, a message similar to the following appears:


Control Panel

The setup routines for the INTERSOLV INGRES ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

If you are using INGRES 6.4/04 and attempt to configure a data source without the files IIW3ING.DLL, IIW3CL.DLL, IIW3PRN.DLL, IIW3GCC.DLL, IIW3ADF.DLL, IIGCCW.DLL, and IIW3LIBQ.DLL either on your path or in your Windows SYSTEM directory, a message similar to the above message appears.

Windows NT

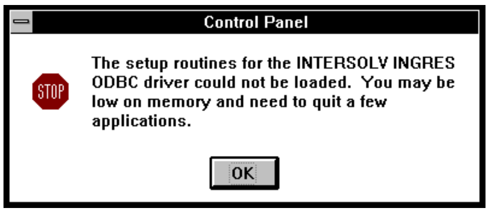
To access INGRES databases from your client workstation, you must have the INGRES/Net or INGRES/Star product installed on both your client and server nodes.

To access INGRES, you must have Networking for Windows NT Release 6.4/05 (int.wnt/01).

You must have the environment variable II_SYSTEM set to the directory where you installed the INGRES/Net or INGRES/Star product. For example:

```
SET II_SYSTEM=C:\INGRE  S
```

If you attempt to configure a data source without the files LIBQ32.DLL, ADF32.DLL, CL32_1.DLL, GCA32.DLL, and MSVCRT10.DLL either on your path, or in your Windows 95 SYSTEM directory or the Windows NT SYSTEM32 directory, a message similar to the following appears:


Control Panel

The setup routines for the INTERSOLV INGRES ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

Windows 95

See the README file shipped with your INTERSOLV DataDirect product for the system requirements for Windows 95.

OS/2

Only INGRES 6.4/04 is supported for OS/2.

To access INGRES, you must have Networking for OS/2 Release 6.4/04 (net.os2/02).

All required DLLs must reside in directories that are listed in the LIBPATH specification in your CONFIG.SYS file.

UNIX

The remote or host INGRES databases must be INGRES 6.4/04 ar later.

UNIX

You must have the environment variable II_SYSTEM set to the directory above the directory where you installed the INGRES client.

For example, if you have installed your INGRES product in /databases/ ingres, the following syntax is valid for C-shell users:

```
setenv II_SYSTEM /database   s
```

For Bourne- or Korn-shell users, the following syntax is valid:

```
II_SYSTEM=/databases;export II_SYSTE   M
```

Configuring Data Sources

UNIX

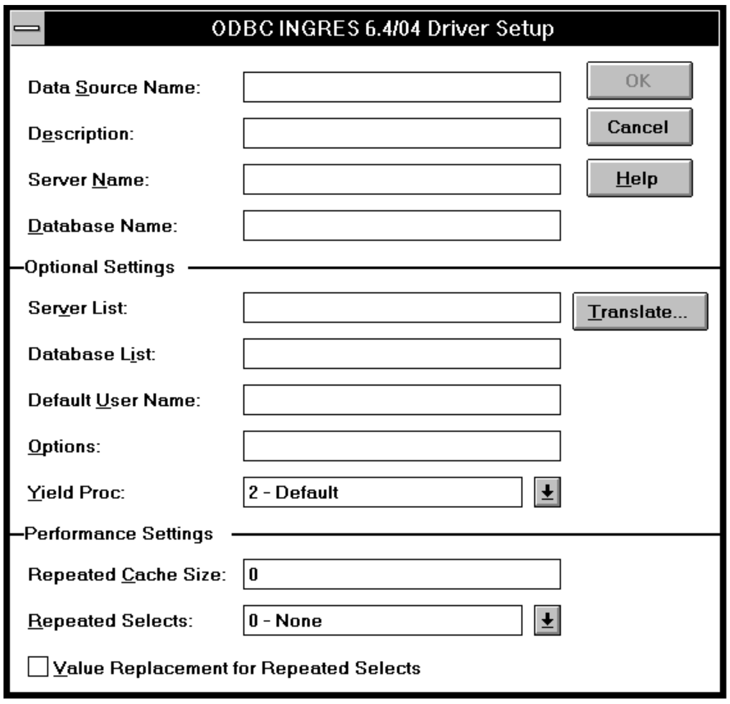
Note: In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the .odbc.ini file using the attributes in Table 9-1. For information about this file, see Appendix E, "ODBC.INI," on page 346.

To configure an INGRES data source:

1. Start the ODBC Administrator. A list of data sources appears.
2. If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the INGRES driver and click **OK**.

   If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.



ODBC INGRES 6.4/04 Driver Setup

Data Source Name:

Description:

Server Name:

Database Name:

Optional Settings

Server List:

Database List:

Default User Name:

Options:

Yield Proc: 2 - Default

Performance Settings

Repeated Cache Size: 0

Repeated Selects: 0 - None

☐ Value Replacement for Repeated Selects

3 Specify values as follows:

Data Source Name:A string that identifies this INGRES data source configuration in ODBC.INI. Examples include “Accounting” or “INGRES-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “INGRES on Server number 1.”

Server Name: The name of the virtual node that you defined using the INGRES NETU utility. This virtual node tells INGRES which system to call, how to call it, and the user's name and password.

Database Name: The name of the database to which you want to connect by default.

The following values are optional:

Server List: The list of servers (virtual nodes) that will be available in the Logon dialog box. Separate the server names with commas.

Database List: The list of databases that will be available in the logon dialog box. Separate the names with commas.

Default User Name: The default user name used to connect to your INGRES database. A user name is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the logon dialog box.

Options: Any flag allowed on the INGRES SQL command line. Some examples are

- -l (locks the database exclusively)
- -u (logs on as username)
- +w or -w (waits/doesn't wait for the database if a user has already opened it exclusively)
- +U or -U (enables/disables user updating of the system tables and locks the database exclusively)
- +Y or -Y (enables/disables user updating of the system tables but does not lock the database exclusively)

Yield Proc: A value of 0, 1, 2, or 3 that determines whether you can work in other applications when INGRES is busy. This attribute is useful to users of ODBC applications.

- 0 (peek and dispatch), which causes the driver to check the Windows message queue and send any messages to the appropriate Windows application.
- 1 (no yielding), which does not let you work in other applications. This

is the default for the INGRES 6.4/03 driver, and it is recommended that you use this value with that driver. This value is not valid for the INGRES 6.4/04 driver.

- 2 (INGRES' yield procedure), which uses INGRES' default yield procedure. This is the default for the INGRES 6.4/04 driver, and it is recommended that you use this value with that driver.
- 3 (dispatch via Windows Yield function), which turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window.

Repeated Cache Size An integer value that determines whether all Update and Insert statements are to be executed as repeated statements. This option improves the performance of applications that repeat the same set of SQL statements. When set to 0, the default, no set of statements is repeated. The recommended setting is 100.

To repeat a single statement rather than all statements, use the INGRES REPEATED syntax.

Repeated Selects A value of 0, 1, or 2 that determines whether the driver optimizes Select statements or executes them as repeated queries. When set to 0, the default, the driver executes all Select statements as it did in previous versions of the product.

When set to 1, the driver optimizes Select statements that return only one result row. When set to 2, the driver executes all Select statements as repeated queries. If this attribute is set to 1 or 2, the Repeated Cache Size option must be set to greater than zero.

Setting this option to 1 or 2:

- limits the driver to one active statement and one active connection
- has no effect on Select statements containing a For Update clause

Value Replacement for Repeated Selects A value of 0 or 1 that determines whether the driver substitutes parameters for hardcoded values in repeated statements. This option is convenient in applications that do not use dynamic parameters.

When set to 0, the default, the driver does not substitute parameters. When set to 1, the driver substitutes parameter markers for hardcoded values and the RepeatedCacheSize attribute must be greater than zero or the REPEATED INGRES keyword must be used.

This option has no effect upon Select statements that contain a For Update clause that requires a cursor or upon statements that already use parameter markers.

This attribute supports only a subset of standard ODBC SQL grammar. It is intended for performance and does not utilize a full SQL parser. For example, subselects are not supported and use of “is NULL” for columns other than character columns is not supported.

4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

UNIX

Note: In the UNIX environment, there is no graphical interface. To perform a translation, you must edit the .odbc.ini file and add to the data source section the TranslationDLL keyword and optionally the TranslationOption keyword:

Keyword	**Definition**
TranslationDLL	Full path of translation DLL
TranslationOption	ASCII representation of the 32-bit integer translation option

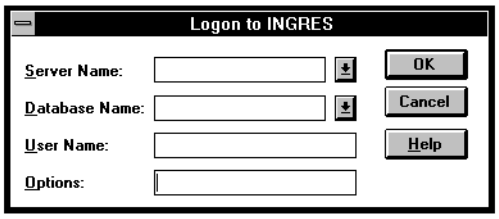
The INTERSOLV OEM ANSI translator is shipped on UNIX to provide an example of how to create a translation DLL.

5 Click **OK** to write these values to ODBC.INI. These values are now the

defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For INGRES, the dialog box is as follows:



In this dialog box, do the following:

1 Type the server name of the computer containing the INGRES database you want to access or select the name from the Server Name drop-down list, which displays the server names you specified in the setup dialog box. Server Name must be a valid virtual node that has been added using the INGRES NETU utility.

2 Type the name of the database to which you want to connect or select the name from the Database Name drop-down list, which displays the names you specified in the setup dialog box.

3 If required, type your user name.

4 Type any options for the connection. The options can be any flags allowed on the INGRES SQL command line. See Table 9-1 for examples of the flags allowed.

5 Click **OK** to log on to INGRES and to update the values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for INGRES is

```
DSN=INGRES TABLES;SRVR=QESERV;DB=PAYROLL;UID=JOHN
```

Table 9-1 gives the long and short names for each attribute, as well as a description. To configure a data source in the UNIX environment, you must edit the .odbc.ini file. This file accepts only long names for attributes.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 9-1. INGRES Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies an INGRES data source configuration in ODBC.INI. Examples include “Accounting” or “INGRES-Serv1.”

Table 9-1. INGRES Connection String Attributes (cont.)

Attribute	Description
ServerName (SRVR)	The name of the virtual node that you defined using the INGRES NETU utility. This virtual node tells INGRES which system to call, how to call it, and the user's name and password.
Database (DB)	The name of the database to which you want to connect
LogonID (UID)	The default logon ID (user name) used to connect to your INGRES database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.
Options (OPTS)	The flags allowed on the INGRES SQL command line. Some examples are • -l (locks the database exclusively) • -u (logs on as username) • +w or -w (waits/doesn't wait for the database if someone has already opened it exclusively) • +U or -U (enables/disables user updating system tables and locks the database exclusively) • +Y or -Y (enables/disables user updating system tables but does not lock the database exclusively)

Table 9-1. INGRES Connection String Attributes (cont.)

Attribute	Description
YieldProc (YLD)	YieldProc={0|1|2|3}. This attribute determines if you can work in other applications when INGRES is busy. This attribute is useful to users of ODBC applications. • YieldProc=0 (peek and dispatch) causes the driver to check the Windows message queue and send any messages to the appropriate Windows application • YieldProc=1 (no yielding), which does not let you work in other applications. This is the default for the INGRES 6.4/03 driver, and it is recommended that you use this value with that driver. This value is not valid for the INGRES 6.4/04 driver. • YieldProc=2 (INGRES' yield procedure), which uses INGRES' default yield procedure. This is the default for the INGRES 6.4/04 driver, and it is recommended that you use this value with that driver. • YieldProc=3 (dispatch via Windows Yield function) turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window.
RepeatedCache Size (RCS)	An integer value that determines whether all Update and Insert statements are to be executed as repeated statements. This attribute improves the performance of applications that repeat the same set of SQL statements. When set to 0, the initial default, no statements are repeated. The recommended setting for this attribute is 100 (RepeatedCacheSize=100) To repeat a single statement rather than all statements, use the INGRES REPEATED syntax.

Table 9-1. INGRES Connection String Attributes (cont.)

Attribute	Description
RepeatedSelects (RS)	RepeatedSelects={0|1|2}. This attribute determines whether the driver optimizes Select statements or executes them as repeated queries. When set to 0, the initial default, the driver executes all Select statements as it did in previous versions of the product. When set to 1, the driver optimizes Select statements that return only one result row. When set to 2, the driver executes all Select statements as repeated queries. If this attribute is set to 1 or 2, the RepeatedCacheSize attribute must be set to greater than zero. Setting this option to 1 or 2: • limits the driver to one active statement and one active connection • has no effect on Select statements containing a For Update clause
ValueReplacement (VR)	ValueReplacement={0|1}. This attribute determines whether the driver substitutes parameters for hardcoded values in repeated statements. This option is convenient in applications that do not use dynamic parameters. When set to 0, the initial default, the driver does not substitute parameters. When set to 1, the driver substitutes parameter markers for hardcoded values and the RepeatedCacheSize attribute must be greater than zero or the REPEATED INGRES keyword must be used. This option has no effect upon Select statements that contain a For Update clause that requires a cursor or upon statements that already use parameter markers. This attribute supports only a subset of standard ODBC SQL grammar. It is intended for performance and does not utilize a full SQL parser. For example, subselects are not supported and use of "is NULL" for columns other than character columns is not supported.

Data Types

Table 9-2 shows how INGRES data types map to the standard ODBC data types.

Table 9-2. INGRES Data Types

INGRES	ODBC
Char	SQL_CHAR
Date	SQL_TIMESTAMP
Float	SQL_DOUBLE
Float4	SQL_REAL
Integer	SQL_INTEGER
Integer1	SQL_TINYINT
Money	SQL_DECIMAL
Smallint	SQL_SMALLINT
Varchar	SQL_VARCHAR

Note that INGRES Date values do not directly map to the ODBC type SQL_TIMESTAMP. If interval data is placed in Date columns, then the driver raises an error when attempting to read the value.

Isolation and Lock Levels Supported

INGRES supports isolation levels 1 (read committed, the default) and 3 (serializable). INGRES supports page-level locking. See Appendix D, "Locking and Isolation Levels," on page 342 for a discussion of these topics.

ODBC Conformance Level

The INGRES driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the SQLBrowseConnect Level 2 function is supported.

The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The INGRES database system supports multiple connections and multiple statements per connection.

Note that if you set the RepeatedSelects connection string attribute to 1 or 2, the driver is limited to one active statement and one active connection.

Go To

10 InterBase Driver

The InterBase driver supports version 4.0 of the InterBase database system in the Windows environment.

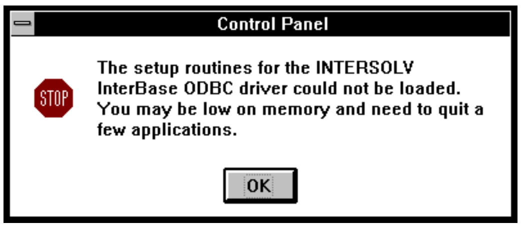
See the README file shipped with your INTERSOLV DataDirect product for the filename of the InterBase driver.

System Requirements

System requirements are as follows:

- InterBase 4.0.
- The new Client-Library requires a PC user to purchase the Windows client piece for InterBase databases. Also, the InterBase client uses TCP/IP as the communications protocol to connect to a UNIX server.

If you attempt to configure a data source and you do not have the InterBase directory on your path or the driver DLLs in your Windows SYSTEM directory, a message similar to the following appears.


Control Panel

The setup routines for the INTERSOLV InterBase ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

Configuring Data Sources

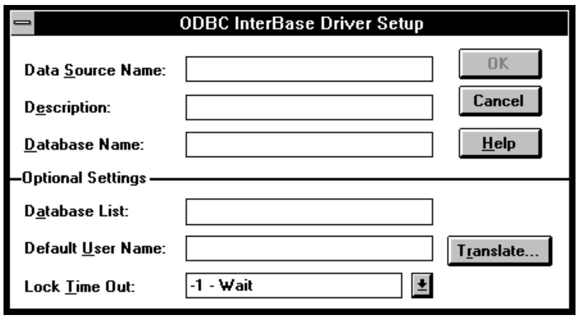
To configure a InterBase data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the InterBase driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.



3 Specify values as follows:

Data Source Name:A string that identifies this InterBase data source configuration in ODBC.INI. Examples include “Accounting” or “InterBase-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “InterBase on Server #1.”

Database Name: The name of the database to which you want to connect by default.

The following values are optional:

Database List The list of databases that will be displayed in the logon dialog box. Separate the names with commas.

Default User Name: The default user name used to connect to your InterBase database. This ID is case-sensitive. Your ODBC application may override this value or you may override this value in the logon dialog box.

Lock Time Out:A value that specifies whether InterBase should wait for a lock to be freed before raising an error when processing a Select...For Update Of statement. Values can be -1 (Wait) or 0 (Don’t wait). The default is -1.

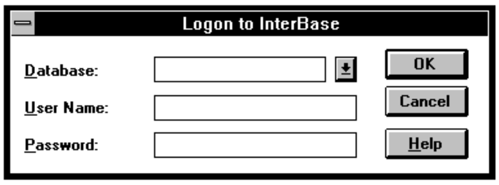
4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For InterBase, the dialog box is as follows:



In this dialog box, do the following:

1 Type the name of the server containing the InterBase tables you want to access (case-sensitive) or click the arrow to the right of the box to select a server name you specified in the setup dialog box.

2 If required, type your case-sensitive user name.

3 If required, type your case-sensitive password for the system.

4 Click **OK** to log on to the InterBase database installed on the server you specified and to update the values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section of ODBC.INI to use for the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_nam e[;attribute=valu e[;attribute=valu e].
..]
```

An example of a connection string for InterBase is

```
DSN=ACCOUNTING;DB=PAYRLL;UID=JOHN;PWD=XYZZY
```

Table 10-1 gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 10-1. InterBase Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies an InterBase data source configuration in ODBC.INI. Examples include “Accounting” or “InterBase-Serv1.”
Description	An optional long description of a data source name. For example, “My Accounting Database” or “InterBase on Server #1.”
Database (DB)	The name of the database to which you want to connect
LogonID (UID)	The case-sensitive user name used to connect to your InterBase database.
Password (PWD)	A case-sensitive password.
LockTimeOut (LTO)	LockTimeOut={0|-1}. This attribute specifies whether InterBase should wait for a lock to be freed before raising an error when processing a Select...For Update Of statement. LockTimeOut=-1 means wait forever (the initial default). LockTimeOut=0 means no waiting for locks (return error)

Table 10-1. InterBase Connection String Attributes (cont.)

Attribute	Description
BinarySegmentSize (BSS)	An integer value that determines the segment size used to create a binary blob. The size of the segment specifies how many bytes the segment can hold. However, this value does not reflect the maximum size of the blob. The default is 80.
CharSegmentSize (CSS)	An integer value that determines the segment size used to create a text blob. Blobs are stored in segments. The size of the segment specifies how many bytes the segment can hold. However, this value does not reflect the maximum size of the blob. The default is 80.

Data Types

Table 10-2 shows how the InterBase data types are mapped to the standard ODBC data types.

Table 10-2. InterBase Data Types

InterBase	ODBC
Blob (SegmentSize*, 0)	SQL_LONGVARBINARY
Blob (SegmentSize*, 1)	SQL_LONGVARCHAR
Char	SQL_CHAR
Date	SQL_DATE
Decimal	SQL_DECIMAL
Double Precision	SQL_DOUBLE
Float	SQL_REAL
Integer	SQL_INTEGER

Table 10-2. InterBase Data Types (cont.)

InterBase	ODBC
Smallint	SQL_SMALLINT
Varchar	SQL_VARCHAR

*Determined by the value set for the BinarySegmentSize (BSS) or CharSegmentSize (CSS) attribute.

Notes: The Blob data types cannot be used in a Where clause and cannot be inserted into a column as a string.

The array data type is not supported.

Isolation and Lock Levels Supported

InterBase supports isolation levels 1 (read committed), 3 (serializable, the default), and 4 (versioning).

InterBase performs *optimistic locking*. This means that InterBase does not really obtain locks on a record; instead, it uses a comparison method. Whenever an Update operation is performed, the server compares the user's value of the record with the most current value in the table. If the values are the same, the update succeeds. Differing values indicate conflict, and the update fails.

See Appendix D, "Locking and Isolation Levels," on page 342 for a discussion of these topics.

ODBC Conformance Level

The InterBase driver supports the Core, Level 1, and Level 2 API functic listed in Appendix C, "ODBC API and Scalar Functions," on page 332. Ir addition, the following Level 2 functions are supported:

- SQLPrimaryKeys
- SQLProcedureColumns
- SQLProcedures

Number of Connections and Statements Supported

The InterBase database system supports multiple connections and multi statements per connection.

Go To

11 MicroDecisionware Driver

The MicroDecisionware driver supports the IBM Database 2 (DB2) and InfoHub database systems in the Windows and OS/2 environments. The MicroDecisionware Database Gateway makes an InfoHub database resemble a DB2 database; therefore, this chapter refers to all databases accessed through the MicroDecisionware Gateway as DB2 databases. Catalog functions are not currently supported for InfoHub.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the MicroDecisionware driver.

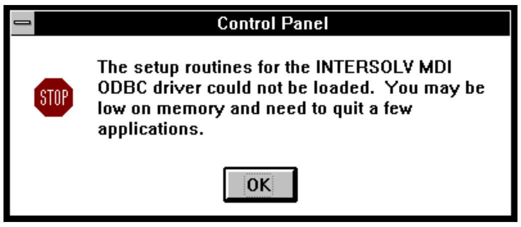
System Requirements

The MicroDecisionware driver must be used with MicroDecisionware's Database Gateway running on a Windows NT server to access DB2 and InfoHub databases or an OS/2 server to access DB2.

Before you can use the MicroDecisionware driver, you must install the MicroDecisionware Database gateway and MicroDecisionware Gateway client appropriate to your workstation.

You can test your connection to the database gateway by using ISQL or SAF.

If you attempt to configure a data source and you do not have W3DBLIB.DLL on your path or in your Windows SYSTEM directory, a message similar to the following appears.


Control Panel

The setup routines for the INTERSOLV MDI ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

Configuring Data Sources

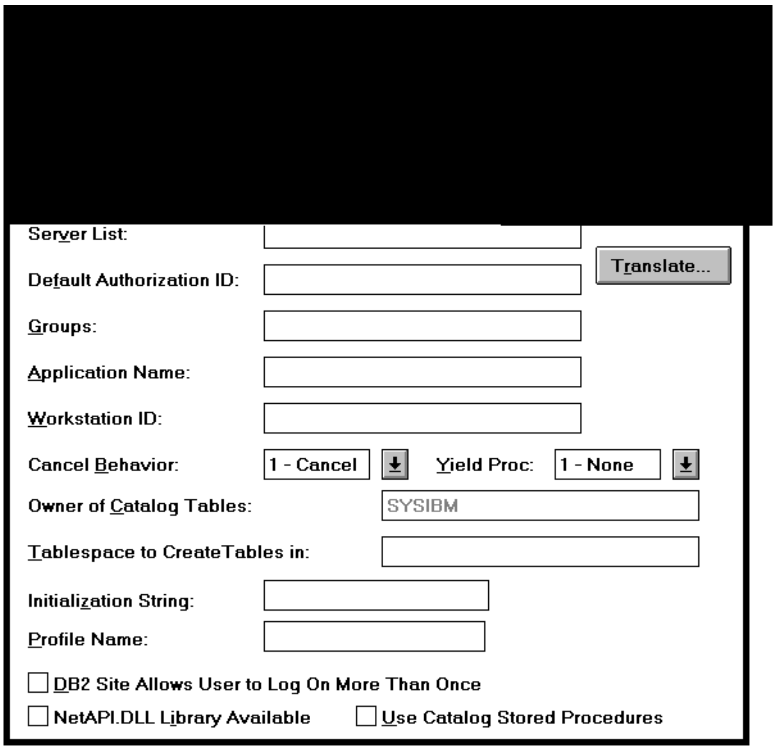
To configure a MicroDecisionware DB2 data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the MDI driver and click **OK**.

  If you are configuring an existing data source, select the data source name and click **Setup**.

  The setup dialog box appears.



Server List:

Default Authorization ID:

Groups:

Application Name:

Workstation ID:

Cancel Behavior: 1 - Cancel

Yield Proc: 1 - None

Owner of Catalog Tables: SYSIBM

Tablespace to CreateTables in:

Initialization String:

Profile Name:

☐ DB2 Site Allows User to Log On More Than Once

☐ NetAPI.DLL Library Available

☐ Use Catalog Stored Procedures

3 Specify values as follows:

Data Source Name: A string that identifies this MicroDecisionware data source configuration in ODBC.INI. Examples include “Accounting” or “MicroDecisionware Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “MicroDecisionware on Server number 1.”

Server Name: The name of the OS/2 server running the MicroDecisionware Database Gateway. This value will appear as the initial default in the logon dialog box.

The following values are optional:

Server List: The list of servers that will appear in the logon dialog box. Separate the server names with commas.

Default Authorization ID: The default authorization ID used to connect to your DB2 database. This ID is case-sensitive. An authorization ID is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the logon dialog box.

Groups: A list of the groups to which you belong, used to determine which tables you can access. Separate each group name with a comma. Alternatively, enter the word ALL so that you can see all table names even if you cannot access the table.

Application Name: The name used to identify your application.

Workstation ID: The workstation ID used by the MicroDecisionware Database Gateway.

Cancel Behavior: A value of 0, 1, or 2 that specifies how a previously executed statement should be canceled.

- 0 fetches all remaining records if the statement was a Select.
- 1 cancels the statement by calling dbcancel. This is the default.
- 2 closes the connection to the server for the statement.

Yield Proc: A value of 0, 1, or 3 that determines whether you can work in other applications when the gateway is busy. This attribute is useful to users of ODBC applications.

- 0 (peek and dispatch), which causes the driver to check the Windows message queue and send any messages to the appropriate Windows application.

- 1 (no yielding, the default), which does not let you work in other applications.
- 3 (dispatch via Windows Yield function), which turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window.

It is recommended that you use the value 1. The values 0 and 3 do not work with all Windows applications.

Owner of Catalog Tables: The owner of views of the system tables. The tables themselves are owned by SYSIBM. If you have read access to the system tables, this option does not have to be changed. If you do not have read access, the database administrator must create a view of the system tables in another account and give you permission to use the view. Enter the authorization ID for the account that owns the views for the system tables.

Tablespace to Create Tables in: The name of the tablespace in which new tables are stored. For example, DSNDB04.

Initialization String Any SQL Server command you wish to execute when the data source connection is initialized.

Profile Name: For InfoHub only. The InfoHub profile name. If you specify a profile name, the driver issues a "USE *profilename*" statement at data source connect time.

DB2 Site Allows User to Log On More Than Once: A check box that specifies that your site lets a user log on to DB2 more than once. Selecting this check box may improve performance.

NETAPI.DLL Library Available: A check box that specifies that the driver can use NETAPI.DLL to get the name of your workstation. Most major PC networks support this feature. If your network does not support this capability, deselect this option. If you supply a workstation ID, this setting is ignored.

Use Catalog Stored Procedures: A check box that specifies whether the driver uses Catalog Stored Procedures or direct SQL to obtain system catalog information. The default is to use direct SQL. If you want to use Catalog Stored Procedures, deselect this option.

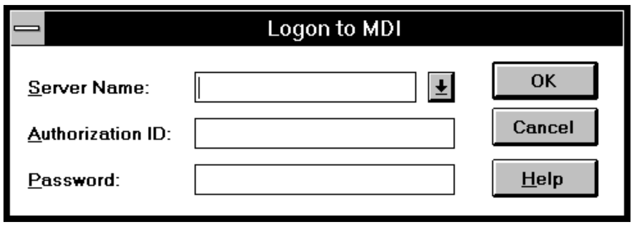
4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For DB2, the dialog box is as follows:



In this dialog box, do the following:

1. Type the name of the OS/2 server running the Database Gateway you wish to access or select the name from the Gateway Server box, which displays the server names specified in the Setup dialog box.
2. If required, type your Authorization ID (case-sensitive).
3. If required, type your password for the system (case-sensitive).
4. Click **OK** to complete the logon and to update the values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_nam e[;attribute=valu e[;attribute=valu e].
..]
```

An example of a connection string for the MicroDecisionware Database Gateway is

```
DSN=DB2 VIA MDI;SRVR=QESRVR;UID=JOHN;PWD=XYZZY;APP=Q      4
```

Table 11-1 gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 11-1. MicroDecisionware Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies a MicroDecisionware data source configuration in ODBC.INI. Examples include "Accounting" or "MicroDecisionware Serv1."
ServerName (SRVR)	The name of the OS/2 server running the MicroDecisionware Database Gateway.
LogonID (UID)	The default logon ID (authorization ID) used to connect to your DB2 database. This ID is case-sensitive. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.
Password (PWD)	A case-sensitive password.
Groups (GRP)	A list of the group to which you belong, used to determine which tables you can access. Set Groups to the names of any groups to which you belong; separate each name with a comma. Groups=ALL lets your application see all table names even if you cannot access the table.
ApplicationName (APP)	The name used to identify your application.
WorkstationID (WKID)	The workstation ID used by the MicroDecisionware Database Gateway.

Table 11-1. MicroDecisionware Connection String Attributes (cont.)

Attribute	**Description**
Cancel (CAN)	Cancel={0|1|2}. This attribute specifies how a previously executed statement should be canceled. • Cancel=0 fetches all of the remaining records if the statement was a Select. • Cancel=1 cancels the statement by calling dbcancel. Set Cancel=1 if dbcancel is supported in your client/server configuration. This is the initial default. • Cancel=2 closes the connection to the server for the statement. Set Cancel=2 only if dbcancel is not supported for your configuration and the performance of fetching all remaining records is unacceptable.
Sysibm (SI)	The authorization ID for the system tables. The initial default is SYSIBM.
InitializationString (IS)	Any SQL Server command you want to execute when the data source connection is initialized.
CreateDB (CDB)	The name of the tablespace in which new tables are stored. For example, CreateDB=DSNDB04.
MultipleLogon (MULT)	MultipleLogon={0|1}. This attribute specifies whether the DB2 site lets users log on more than once. Set this attribute to 1 if this is the case; 0 otherwise.
Netapi (NAPI)	Netapi={0|1}. This attribute specifies whether NETAPI.DLL is available. Most major PC networks support this feature. Netapi=0 indicates it is not available; Netapi=1 indicates it is available. If you supply a value for the attribute WorkstationID, this attribute is ignored.

Table 11-1. MicroDecisionware Connection String Attributes (cont.)

Attribute	Description
YieldProc (YLD)	YieldProc={0|1|3}. This attribute determines if you can work in other applications when the gateway is busy, is useful to users of ODBC applications. • YieldProc=0 (peek and dispatch) causes the driver to check the Windows message queue and to send any messages to the appropriate Windows application. • YieldProc=1 (no yielding, the initial default) does not let you work in other applications. • YieldProc=3 (dispatch via Windows Yield function) turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window. It is recommended that you use YieldProc=1. YieldProc=0 and YieldProc=3 do not work with all Windows applications
ModifySQL (MS)	ModifySQL={0|1}. This attribute is provided for backward compatibility. Specify ModifySQL=0 to have the driver understand SQL dialects found in earlier drivers. Specify ModifySQL=1 (the default) to have the driver modify the SQL statement to conform to ODBC specifications
UseCatalogStored Procedures (UCSP)	UseCatalogStoredProcedures={0|1}. This attribute determines if the driver uses Catalog Stored Procedures or direct SQL to obtain system catalog information. If set to 1, the driver uses Catalog Stored Procedures. If set to 0, the driver uses direct SQL. The initial default is 0.
ProfileName (PN)	For InfoHub only. This attribute is the InfoHub profile name. If you specify a name, the driver issues a "USE *profilename*" statement at data source connect time.

Go To

12 Oracle Drivers

Two separate Oracle drivers support the Oracle 6 and Oracle 7 database systems.

- The Oracle 6 driver is supported in the Windows, OS/2, Macintosh, and UNIX (not including Solaris for x86) environments.
- The Oracle 7 driver is supported in the Windows, Windows 95, Windows NT, OS/2, Macintosh, and UNIX environments.

See the README file shipped with your INTESOLV DataDirect product for the filenames of the Oracle drivers.

System Requirements

The Oracle SQL*Net product is required to access remote Oracle databases.

Windows

In Windows, there are separate requirements for Oracle 6 and Oracle 7.

Oracle 6 Requirements

For the DOS-based SQL*Net TSRs, you must start the SQL*Net TSRs before you start Windows. You also must include a statement in your AUTOEXEC.BAT file to enable Oracle to find the Oracle configuration file, CONFIG.ORA. For example, if the file is located in the ORACLE6 directory, then the following statement must be included in your AUTOEXEC.BAT file:

```
SET CONFIG=C:\ORACLE6\CONFIG.OR   A
```

To use the Windows-based SQL*Net DLLs with ORA6WIN.DLL, the directory containing the DLLs must be on your path. Generally, these DLLs are installed in C:\ORAWIN. The Oracle configuration file for Windows, ORACLE.INI, is placed in the Windows directory.

To use ORA6WIN.DLL with the SQL*Net DLLs and have it find its error message file, you must include a statement in your AUTOEXEC.BAT file. For example, if your Windows home directory is WIN31, then the following statement must be included in your AUTOEXEC.BAT file:

```
SET CONFIG=C:\WIN31\ORACLE.IN   I
```

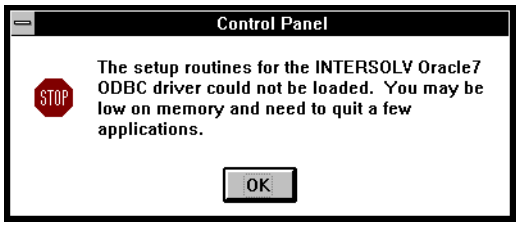
Oracle 7 Requirements

If you are using an Oracle 7 data source and have SQL*Net 2.0, you need ORA71WIN.DLL, CORE3WIN.DLL, and NLS23WIN.DLL.

If you do not have SQL*Net 2.0, you need ORA7WIN.DLL, ORA71WIN.DLL, COREWIN.DLL, CORE3WIN.DLL, USDMEM.DLL, and NLS23WIN.DLL.

These DLLs are included with the drivers on your INTERSOLV release disks. By default, the Setup program installs these files in your Windows SYSTEM directory. If you install them in another directory, that directory must be on your path.

If you attempt to configure an Oracle data source and you do not have the above listed DLLs on your path or in your Windows SYSTEM directory, a message similar to the following appears.

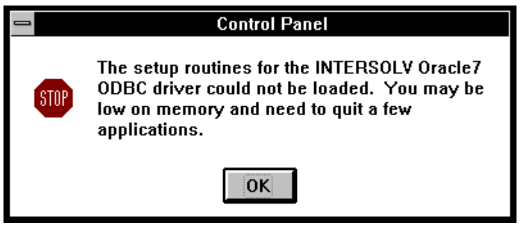


The setup routines for the INTERSOLV Oracle7 ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

Windows 95 and Windows NT

Windows 95 and Windows NT support Oracle 7 only.

ORA7NT.DLL, ORA71NT.DLL, CORENT.DLL, and CORENT23.DLL must be on your path or in your Windows 95 SYSTEM directory or your Windows NT SYSTEM32 directory. If you attempt to configure an Oracle 7 data source and you do not have these DLLs on your path or in the appropriate system directory, a message similar to the following appears.


Control Panel

The setup routines for the INTERSOLV Oracle7 ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

OS/2

SQL*Net DLLs are required to access remote Oracle databases.

The following line must be included in your CONFIG.SYS file:

```
SET CONFIG=C:\ORACLE\CONFIG.OR   A
```

Oracle 6 requires ORA6032.DLL. Oracle 7 requires ORA7032.DLL. These 32-bit DLLs are supplied by Oracle along with other required DLLs. All the DLLs must be in directories that are included in the LIBPATH specification in your CONFIG.SYS file.

UNIX

UNIX

Before you can use the Oracle data source, you must have the Oracle SQL*Net drivers you plan to use installed on your workstation in the $ORACLE_HOME source tree. ORACLE_HOME is an environment variable created by the Oracle installation process that identifies the location of your Oracle client components.

Oracle refers to the runtime Oracle component as “Oracle RDBMS.” From the Oracle RDBMS product, the Oracle driver depends on the executables in $ORACLE_HOME/bin and the interface libraries in $ORACLE_HOME/rdbms/lib.

Set the environment variable ORACLE_HOME to the directory where you installed the Oracle RDBMS or SQL*Net product. For example, for C-shell users, the following syntax is valid:

```
setenv ORACLE_HOME /databases/oracle
```

For Bourne- or Korn-shell users, the following syntax is valid:

```
ORACLE_HOME=/databases/oracle;export ORACLE_HOME
```

Building the Required Oracle SQL*Net Driver

The Oracle 6 and 7 drivers require a one-time site linking to build an Oracle SQL*Net driver on AIX and Solaris only, not HP-UX. On HP-UX, the driver (libqeoracle.sl) is shipped with the INTERSOLV ODBC drivers. This site linking binds your unique Oracle SQL*Net configuration into the file, which is used by the Oracle drivers to access local and remote Oracle databases.

Before you build the Oracle SQL*Net DLL, install Oracle and set the environment variable ORACLE_HOME to the directory where you installed Oracle.

A make file is provided that builds the Oracle SQL*Net driver. This make file is in the bin directory. The Oracle SQL*Net driver should be either built in or copied to the directory in which the other INTERSOLV ODBC DLLs are installed, usually qelib/dlls or odbc/dlls.

The following example builds the Oracle SQL*Net driver if you are in the qelib/bin or odbc/bin directory**:**

```
%make -f ./qeor an.mk EXE=../dll s
```

n is 6 if you are connecting to an Oracle 6 database and 7 if you are connecting to an Oracle 7 database.

The EXE= directive builds the Oracle SQL*Net DLL in the location you specify.

Macintosh

System requirements are as follows:

- 8 MB of memory
- MacTCP version 1.1 or greater

The Oracle 6 driver requires that you install qe_oci in the Drivers folder under the Oracle 6 home directory.

Configuring Data Sources

UNIX

Note: In the UNIX environment, there is no ODBC administrator. To configure a data source in the UNIX environment, you must edit the .odbc.ini file using the attributes in Table 12-1 on page 166. For information about this file, see Appendix E, “ODBC.INI,” on page 346.

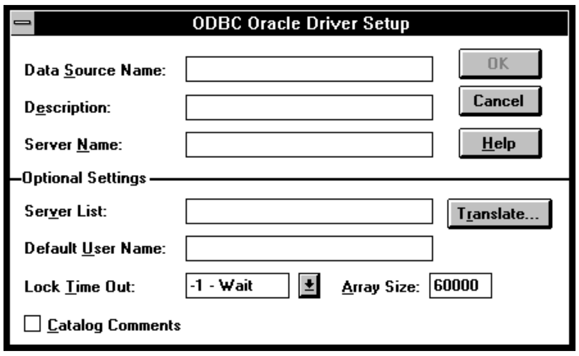
To configure an Oracle data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Oracle driver of your choice and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup** (if you are using Apple's ODBC Driver Manager on the Macintosh, this button is labeled **Modify**).

The setup dialog box appears.



3 Specify values as follows:

Data Source Name:A string that identifies this Oracle data source configuration in ODBC.INI. Examples include "Accounting" or "Oracle-Serv1."

Description: An optional long description of a data source name. For example, "My Accounting Database" or "Oracle on Server number 1."

Server Name: The SQL*Net connection string designating the server and database to be accessed. The information required varies depending on the SQL*Net driver you are using. The section "Connecting to a Data Source Using a Connection String" on page 165 gives the format of the SQL*Net connection string.

The following values are optional:

Server List: The list of SQL*Net connection strings that will appear in the logon dialog box. Separate the strings with commas. If the SQL*Net connection string contains a comma, enclose it in quotation marks; for example, "Serv,1", "Serv,2", "Serv,3."

Default User Name: The default user name used to connect to your Oracle database. A default user name is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the logon dialog box or connection string.

Lock Time Out: A value of 0 or -1 that specifies whether Oracle should wait for a lock to be freed before raising an error when processing a Select...For Update statement. Values can be -1 (wait forever) or 0 (don't wait). The default is -1.

Array Size: The number of bytes the driver uses for fetching multiple rows. Values can be an integer from 0 to 65536; the default is 60000. Larger values increase throughput by reducing the number of times the driver fetches data across the network. Smaller values increase response time, as there is less of a delay waiting for the server to transmit data.

Catalog Comments: Check this box if you want to retrieve the contents of the COMMENTS column for catalog functions. Retrieving the COMMENTS column may reduce the performance of your data catalog operations.

4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

UNIX

Note: In the UNIX environment, there is no graphical interface. To perform a translation, you must edit the .odbc.ini file and add to the data source section the TranslationDLL keyword and optionally the TranslationOption keyword:

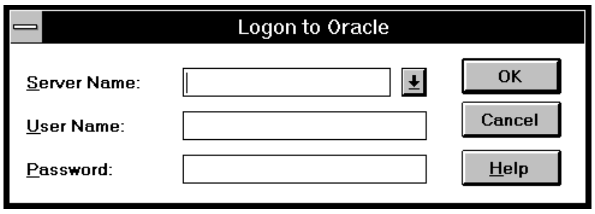
Keyword	**Definition**
TranslationDLL	Full path of translation DLL
TranslationOption	ASCII representation of the 32-bit integer translation option

The INTERSOLV OEM ANSI translator is shipped on UNIX to provide an example of how to create a translation DLL.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For Oracle, the dialog box is as follows:



In this dialog box, do the following:

1 Type the SQL*Net connection string of the computer containing the Oracle database tables you want to access or select the string from the Server Name drop-down list, which displays the names you specified in the setup dialog box.

2 If required, type your Oracle user name.

3 If required, type your Oracle password.

4 Click **OK** to log on to the Oracle database installed on the server you specified and to update the values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_nam e[;attribute=valu e[;attribute=valu e].
..]
```

An example of a connection string for Oracle is

```
DSN=Accounting;SRVR=X:QESRVR;UID=JOHN;PWD=XYZZ     Y
```

If the server name contains a semicolon, enclose it in quotation marks:

```
DSN=Accounting;SRVR="X:QE;SRVR";UID=JOHN;PWD=XYZZ     Y
```

Table 12-1 gives the long and short names for each attribute, as well as a description. To configure a data source in the UNIX environment, you must edit the .odbc.ini file. This file accepts only long names for attributes.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 12-1. Oracle Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies an Oracle data source configuration in ODBC.INI. Examples include "Accounting" or "Oracle-Serv1."
LogonID (UID)	The logon ID (user name) that the application uses to connect to your Oracle database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.
Password (PWD)	Your password.
LockTimeOut (LTO)	A value that specifies whether Oracle should wait for a lock to be freed before raising an error when processing a Select...For Update statement. Values can be -1 (wait forever, the initial default) or 0 (don't wait).
ArraySize (AS)	The number of bytes the driver uses for fetching multiple rows. Values can be an integer from 0 to 65536. The initial default is 60000. Larger values increase throughput by reducing the number of times the driver fetches data across the network. Smaller values increase response time, as there is less of a delay waiting for the server to transmit data.

Table 12-1. Oracle Connection String Attributes (cont.)

Attribute	Description

ServerName (SRVR)	The SQL*Net connection string designating the server and database to be accessed. The information required varies depending on the SQL*Net driver you are using.
	
	For remote servers, the SQL*Net connection string has the following form:
	
	`driver_prefix:computer_name[:sid]`
	
	driver_prefix identifies the network protocol being used. The driver prefix can be as follows: P (named pipes), X (SPX), B (NetBIOS), T (TCP/IP), D (DECNet), A (Oracle Async), AT (AppleTalk), or TNS (SQL*net 2.0). Check your Oracle documentation for other protocols.
	
	computer_name is the name of the Oracle Listener on your network.
	
	sid is the Oracle System Identifier and refers to the instance of Oracle running on the host. This item is required when connecting to systems that support more than one instance of an Oracle database.
	
	For local servers, the SQL*Net connection string has the form
	
	`database_name`
	
	database_name identifies your Oracle database.
	
	If the SQL*Net connection string contains semicolons, enclose it in quotation marks. See your SQL*Net documentation for more information.

CatalogComments (CC)	CatalogComments={0|1}. This attribute specifies whether the driver returns the contents of the COMMENTS column for catalog functions. CatalogComments=1 returns COMMENTS. Retrieving the COMMENTS column may reduce the performance of your data catalog operations. CatalogComments=0 does not return COMMENTS (the initial default).

Oracle 6 Data Types

Table 12-2 shows how the Oracle 6 data types are mapped to the standard ODBC data types.

Table 12-2. Oracle 6 Data Types

Oracle 6	ODBC
Char	SQL_VARCHAR
Date	SQL_TIMESTAMP
Long	SQL_LONGVARCHAR
Long Raw	SQL_LONGVARBINARY
Number	SQL_DOUBLE
Number(p,s)	SQL_DECIMAL
Raw	SQL_VARBINARY

Oracle 7 Data Types

Table 12-3 shows how the Oracle 7 data types are mapped to the standard ODBC data types.

Table 12-3. Oracle 7 Data Types

Oracle 7	ODBC
Char	SQL_CHAR
Date	SQL_TIMESTAMP
Long	SQL_LONGVARCHAR

Table 12-3. Oracle 7 Data Types (cont.)

Oracle 7	**ODBC**
Long Raw	SQL_LONGVARBINARY
Number	SQL_DOUBLE
Number(p,s)	SQL_DECIMAL
Raw	SQL_VARBINARY
Varchar2	SQL_VARCHAR

Isolation and Lock Levels Supported

Oracle 6 and 7 both support isolation level 2 (repeatable read) only. Both versions support record-level locking.

See Appendix D, "Locking and Isolation Levels," on page 342 for a discussion of these topics.

ODBC Implementation Level

Both Oracle drivers support the Core, Level 1, and Level 2 API functions listed in Appendix C, "ODBC API and Scalar Functions," on page 332.

The Oracle 6 driver also supports the Level 2 function SQLBrowseConnect.

The Oracle 7 driver also supports the following Level 2 functions:

- SQLBrowseConnect
- SQLProcedures
- SQLPrimaryKeys

Both drivers support the core SQL grammar.

Number of Connections and Statements Supported

Oracle 6 and Oracle 7 support multiple connections and multiple statem
per connection.

Go To

13 Paradox 4 Driver

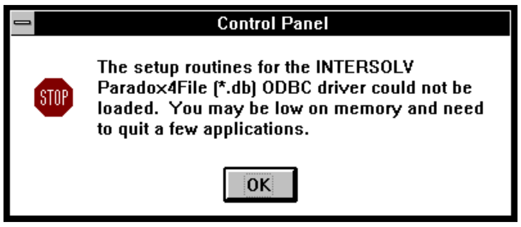
The Paradox 4 driver supports Paradox 3.0, 3.5, 4.0, and 4.5 tables in the Windows environment.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the Paradox 4 driver.

System Requirements

You must install the Windows version of the Paradox Engine’s dynamic link library (PXENGWIN.DLL) on your computer or on your network server before you can open Paradox tables. INTERSOLV does not provide PXENGWIN.DLL. You must purchase Borland’s Paradox Engine version 3.0 or later, which contains this DLL.

If you attempt to configure a data source and you do not have PXENGWIN.DLL on your path or in your Windows SYSTEM directory, a message similar to the following appears.



The setup routines for the INTERSOLV Paradox4File (*.db) ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

In addition, the DOS SHARE program must be loaded to access Paradox tables.

Multiuser Access to Tables

You can query Paradox tables that reside in a shared directory on a network or that are to be shared among applications running on a local workstation. If the tables are on a network server, multiple users can query these tables simultaneously. To share Paradox tables among multiple users, the tables must be located in a shared directory on your network server.

Because the process of establishing shared directories varies from network to network, contact your system administrator to set up a shared directory and to configure each user's environment to access the directory properly. See the *Borland Paradox Engine User's Guide* for information on specifying the file-sharing characteristics of the tables accessed by Paradox. You specify these settings using the Paradox Engine Configuration utility (PXENGCFG.EXE).

Whenever you open a Paradox table that another user opens at the same time, the consistency of the data becomes an issue if both individuals are updating the table.

Locking

The Paradox 4 driver has two locking levels: record locking and table locking. Tables that have no primary key always have a prevent write lock placed on the table when the table is opened. The table lock is escalated to a write lock when an operation that changes the table is attempted.

Tables that have primary keys use record locking. The locking level is escalated from record locking to a table write lock if the transaction runs out of record locks. You can configure the number of record locks permitted by using the Paradox Engine Configuration utility (PXENGCFG.EXE). The maximum number of record locks is 128.

If you set the connection string attribute ReadConsistency to 1, record locking is turned off and the locking that exists is the same as tables with no primary key. When this attribute is set to 0, record locking is used on tables that have a primary key.

Primary keys provide the greatest concurrency for tables that are accessed and modified by multiple users.

Note: If a table lock is placed on a Paradox table, the Paradox 4 driver prevents users from updating and deleting records but does not prevent them from inserting records into the locked table.

Configuring Data Sources

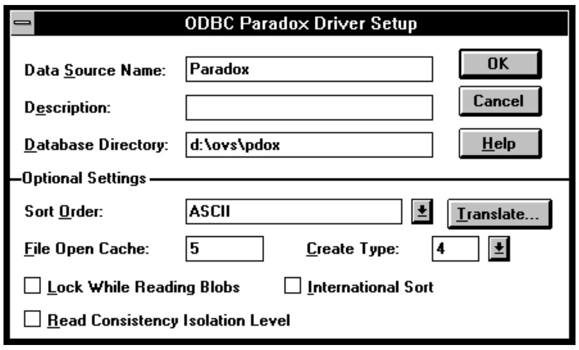
To configure a Paradox 4 data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Paradox 4 driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.



ODBC Paradox Driver Setup

Data Source Name: Paradox

Description:

Database Directory: d:\ovs\pdox

Optional Settings

Sort Order: ASCII

File Open Cache: 5

Create Type: 4

☐ Lock While Reading Blobs

☐ International Sort

☐ Read Consistency Isolation Level

3 Specify values as follows:

Data Source Name: A string that identifies this Paradox data source configuration in ODBC.INI. Examples include “Accounting” or “Paradox-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “Paradox Files on Server number 1.”

Database Directory: The directory in which the Paradox files are stored. If a directory is not specified, the current working directory is used.

The following values are optional:

Sort Order: A value of ASCII (the default), INTL, NORDAN, NORDAN4, or SWEDFIN that specifies the order in which records are sorted in a Paradox index. This option is used for compatibility with international versions of Paradox. NORDAN4 is only for Paradox 4.0 and 4.5 tables.

File Open Cache: An integer value to specify the maximum number of unused file opens to cache. For example, the value 4 specifies that when a user opens and closes four tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is increased performance. The disadvantage is that a user who specifies file locking on open may get a locking conflict even though no one appears to have the file open. The default is 0, which means no file open caching.

Create Type: A value of 3 or 4 (the default) that determines the table version for Create Table statements. Specify 4 for Paradox 4.0 or 4.5 tables. Specify 3 for Paradox 3.0 or 3.5 tables.

Lock While Reading Blobs: This check box, when selected, indicates that records containing blob data types (BLOB, OLE, Graphic, Formatted Memo, and Memo) are to be locked before they are fetched.

International Sort: A setting to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Select this check box to use the international sort order as defined by your operating system. International sort order is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (see your operating system documentation). Leave this box blank to use ASCII sort order. ASCII sort order is case-sensitive, where uppercase letters precede lowercase letters (*B* precedes *a*).

Leave this box blank to use the ASCII sort order, where uppercase letters precede lower case letters (*B* precedes *a*).

Read Consistency Isolation Level: This check box, when selected, indicates that tables are locked to assure that all records can be reread within a statement.

4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

   The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for Paradox is

```
DSN=PARADOX TABLES;DB=C:\ODBC\EMP;PW=ABC,DEF,GH    I
```

Table 13-1 gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 13-1. Paradox 4 Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies a Paradox data source configuration in ODBC.INI. Examples include “Accounting” or “Paradox-Serv1.”
Database (DB)	The directory in which the Paradox files are stored

Table 13-1. Paradox 4 Connection String Attributes (cont.)

Attribute	Description
FileOpenCache (FOC)	An integer value that determines the maximum number of unused table opens to cache. For example, when FileOpenCache=4, and a user opens and closes 4 tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is increased performance. The disadvantage is that a user who tries to open the table exclusively may get a locking conflict even though no one appears to have the table open. The initial default is 0.
CreateType (CT)	CreateType={3|4}. The table version for Create Table statements. Specify CreateType=3 to create Paradox 3.0 or 3.5 tables, CreateType=4 for Paradox 4.0 or 4.5 tables (the initial default).
SortOrder (SO)	SortOrder={ASCII|INTL|NORDAN|NORDAN4, SWEDFIN}. This attribute determines the order in which records are sorted in a Paradox index. This option is used for compatibility with international versions of Paradox. ASCII is the initial default. NORDAN4 is only for Paradox 4.0 and 4.5 tables.
LockToReadBlob (LTRB)	LockToReadBlob={0|1}. This attribute affects only tables that have blob data types (BLOB, OLE, Graphic, Formatted Memo, and Memo). With LockToReadBlob=1, in shared environments, records with blob data types are locked while the data type is cloned to assure a consistent snapshot of the data type and record. Disabling this feature by setting LockToReadBlob=0 (the initial default) may improve performance. See the section "LockToReadBlob" on page 180 for more information on this attribute.

Table 13-1. Paradox 4 Connection String Attributes (cont.)

Attribute	Description
ReadConsistency (RC)	ReadConsistency={0|1}. This attribute specifies whether tables are locked to assure that all records can be reread within a statement. ReadConsistency=1 implies the table is always locked to prevent other writes. This assures that a record can be reread within a statement. ReadConsistency=0, the initial default, disables this behavior if it was specified during data source configuration.
IntlSort (IS)	IntlSort={0|1}. This attribute determines the order in which records are retrieved when you issue a Select statement. If IntlSort=1, the driver uses the international sort order as defined by your operating system. The sort is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (check your operating system documentation). If IntlSort=0 (the initial default), the driver uses the ASCII sort order, where uppercase letters precede lowercase letters (*B* precedes *a*).
UltraSafeCommit (USF)	UltraSafeCommit={0|1}. This attribute determines when the driver flushes its changes to disk. If UltraSafeCommit=1, the driver does this at each COMMIT. This decreases performance. The initial default is 0. This means that the driver flushes its changes to disk when the table is closed or when internal buffers are full. In this case, a machine “crash” before closing a table may cause recent changes to be lost.
Passwords (PW)	A password or list of passwords. You can add 1 to 50 passwords into the system using a comma-separated list of passwords. Passwords are case-sensitive. For example, Passwords=psw1, psw2, psw3.

Table 13-1. Paradox 4 Connection String Attributes (cont.)

Attribute	**Description**
ModifySQL (MS)	ModifySQL={0|1}. This attribute is provided for backward compatibility with earlier versions of Q+E products. It determines whether the driver modifies SQL statements to conform to ODBC specifications or passes the SQL statement directly to Paradox. Specify ModifySQL=1 to have the driver modify the SQL statement to conform to ODBC specifications. Specify ModifySQL=0 to have the driver understand SQL dialects found in earlier drivers. The default is 1.
DeferQueryEvaluation (DQ)	DeferQueryExecution={0|1}. This attribute determines when a query is evaluated—after all records are read or each time a record is fetched. If DeferQueryEvaluation=0, the driver generates a result set when the first record is fetched. The driver reads all records, evaluates each one against the Where clause, and compiles a result set containing the records that satisfy the search criteria. This process slows performance when the first record is fetched, but activity performed on the result set after this point is much faster, because the result set has already been created. You do not see any additions, deletions, or changes in the database that occur while working from this result set. If DeferQueryEvaluation=1 (the default), the driver evaluates the query each time a record is fetched, and stops reading through the records when it finds one that matches the search criteria. This setting avoids the slowdown while fetching the first record, but each fetch takes longer because of the evaluation taking place. The data you retrieve reflects the latest changes to the database. However, a result set is still generated if the query is a Union of multiple Select statements, if it contains the Distinct keyword, or if it has an Order By or Group By clause.

LockToReadBlob

The LockToReadBlob connection attribute determines whether the Paradox 4 driver locks records containing a blob data type. LockToReadBlob affects only tables containing blob data types (that is, BLOB, OLE, Graphic, Formatted Memo, and Memo) for which you use record locking.

LockToReadBlob=1 ensures greater data integrity but tends to be slow with large Paradox tables. With LockToReadBlob=1, when the driver reads a record containing a blob data type, it locks the record, duplicates (clones) the corresponding blob data type, and then unlocks the record. This precaution ensures that the user receives a consistent snapshot of a record and its corresponding field.

With LockToReadBlob=0, the driver neither locks the record nor duplicates the blob data type. This setting improves performance but causes an error when the following sequence occurs:

1. You read part of a record containing a blob data type.
2. Another user reads the same record and updates the blob data type of this record.
3. You attempt to read the blob data type to finish reading the record.

   When this sequence occurs, the following error is displayed:

   ```
   Error on input or output to file:  Another user modified the
   BLOB.
   ```

   If you receive this message, you must re-execute the query.

Data Types

Table 13-2 shows how the Paradox data types are mapped to the standard ODBC data types. These Paradox data types can be used in a Create Table statement. For the syntax of the Create Table statement, see Appendix A, "SQL for Flat-File Drivers," on page 299.

Table 13-2. Paradox Data Types

Paradox	ODBC
Alphanumeric	SQL_VARCHAR
BLOB	SQL_LONGVARBINARY
Currency	SQL_DOUBLE
Date	SQL_DATE
Formatted Memo	SQL_LONGVARBINARY
Graphic	SQL_LONGVARBINARY
Memo	SQL_LONGVARCHAR
Number	SQL_DOUBLE
OLE	SQL_LONGVARBINARY
Short	SQL_SMALLINT

Note: BLOB, Formatted Memo, Graphic, Memo, and OLE data types are valid only for Paradox 4.0 or greater. With the exception of the Memo data type, these data types cannot be used in a Where clause or as key fields of an index. The Memo data type can be used in Where clauses but only the first 512 characters are used for comparison purposes.

Column Names

Column names in SQL statements (such as, Select, Insert, etc.) can be up to 25 characters long and are case-insensitive. If a column name contains a special character, does not begin with a letter, or is a reserved word, surround it with the grave character (`) (ASCII 96). For example:

```
SELECT `last name` FROM em  p
```

Select Statement

You use a SQL Select statement to specify the columns and records to be read. The Paradox 4 driver supports all of the Select statement clauses as described in Appendix A, “SQL for Flat-File Drivers,” on page 299.

Password Protection

Paradox supports two types of passwords: master and auxiliary. The Paradox 4 driver supports master passwords only and can manage up to 50 passwords.

Paradox tables may be encrypted to provide limited access to users who do not know the password. The driver maintains a list of passwords for each connection. The driver can access only encrypted tables for which a password appears in this list. You can supply a password in three ways: by typing it in the Password dialog box (which appears when the driver does not have the password to open an encrypted table), by including it in a connection string (with the Passwords attribute), or by using the Add Password statement.

Paradox provides five statements that manage passwords for Paradox tables. These statements are specific to the Paradox driver:

```
ENCRYPT filename USING PASSWORD password

ADD PASSWORD password

DECRYPT filename USING PASSWORD password

REMOVE PASSWORD password

REMOVE ALL PASSWORDS
```

filename can be a simple filename or a full pathname. If a simple filename is given, the file must be in the directory specified with the Database connection string attribute. The .DB extension is not required.

password is a case-sensitive text string up to 15 characters in length, including blanks. If your password includes lower-case letters or non-alphanumeric characters, enclose it in single quotation marks (').

Encrypting a Paradox Table

The Encrypt statement associates a password with a table. For example:

```
ENCRYPT emp USING PASSWORD test
```

Accessing an Encrypted Paradox Table

To access an encrypted Paradox table, add the password to the list of passwords Paradox maintains for that connection. To do so, you can

- Issue an Add Password statement before you access the table. For example:

  ```
  ADD PASSWORD test
  SELECT * FROM emp
  ```

- Specify the passwords using the Passwords attribute at connection time.

If you do not add the password, the driver prompts you for it when you access the table.

Decrypting a Paradox Table

The Decrypt statement disassociates a password from a table. You no longer need to enter the password to open the table. For example:

```
DECRYPT emp USING PASSWORD test
```

Removing a Password from Paradox

The Remove Password statement removes a password from the list Paradox maintains for the connection. For example:

```
REMOVE PASSWORD test
```

Removing All Passwords from Paradox

The Remove All Passwords statement removes the list of passwords Paradox maintains.

If you remove a password from Paradox and do not decrypt the table, you must continue entering the password to open the table.

Index Files

An index is used to read records in sorted order and to improve performance when selecting records and joining tables. Paradox indexes are stored in separate files and are either *primary* or *non-primary*.

Primary Index

A primary index is made up of one or more fields from the Paradox table. The primary key fields of a primary index consist of one or more consecutive fields in the table, beginning with the first field in the table. A table can have only one primary index.

Collectively, the primary key fields uniquely identify each record in the Paradox table. Thus, no two records in a Paradox table can share the same values in their primary key fields. Once a primary index is created for a Paradox table, the table’s records are re-ordered based on the primary key fields. At the time a primary index is created, if any records have matching primary key field values, Paradox deletes all but the first record. Paradox creates this index as maintained; that is, if you modify, add, or delete records in the table, the primary index is updated automatically to reflect these changes.

A primary index is a single file with the same name as the table on which it is based but with a .PX extension.

To lock records, you must have a primary index.

Non-Primary Index

Unlike a primary key field of a primary index, a non-primary key field does not uniquely identify each record. Thus, two or more records in a Paradox table can share the same value in their non-primary key field. A Paradox table can have more than one non-primary index, as long as each one is based on different fields.

A non-primary index is defined by specifying one or more fields in the Paradox table that constitute the non-primary key fields. It allows Paradox to sort each record in the table according to the value of the non-primary key fields.

There are two kinds of non-primary indexes: maintained and non-maintained. The difference between the two is that a maintained index is automatically updated when the table is changed, whereas a non-maintained index is not. Instead, a non-maintained index is tagged out-of-date and is updated when the index is used again.

You must have a primary index on a table before you create a maintained, non-primary index. Although you can create a non-maintained index on a table that has no primary index, the non-maintained index is ignored. The index also is ignored during statements that modify the table because the index is not updated and is therefore incorrect. The only time a non-maintained index is used is when you issue a read-only Select statement on a locked table that has a primary index.

For Paradox 3.*x*, a single non-primary index consists of a pair of files with the same name as the table on which the non-primary index is based; one of these files has an .X*nn* extension while the other has a .Y*nn* extension (where the hexadecimal number *nn* corresponds to the field number of the non-primary key field for the non-primary index).

For Paradox 4.*x*, single-field non-primary indexes that are case sensitive have the same name as their associated table and are assigned file extensions .X01 through .XFF, depending on the number of the field on which the index is based. Single-field non-primary indexes that are case insensitive and composite indexes have the same name as the table on which they are based. They are assigned file extensions sequentially starting with .XG0 (with hexadecimal increments).

Create and Drop Index Statements

The Paradox driver supports SQL statements to create and delete indexes. The Create Index Primary statement is used to create primary indexes. The Create Index statement is used to create non-primary indexes. The Drop Index statement is used to delete indexes.

Create Index Primary Statement

The syntax for creating a primary index is

```
CREATE [UNIQUE] INDEX PRIMARY
    ON table_name (column [,column...])
```

The UNIQUE keyword is optional; the index is unique whether or not you include this keyword.

table_name is the name of the table on which the index is to be based.

column is the name of a column that is included as a the key field for the index. The column list must contain one or more consecutive fields in the table, beginning with the first field in the table.

For example:

```
CREATE UNIQUE INDEX PRIMARY ON emp (emp_id    )
```

Be careful when you create a primary key because any rows that have a primary key duplication are deleted when you execute the Create Index Primary statement.

Create Index Statement

The syntax for creating a non-primary index is

```
CREATE INDEX  index_name [/NON_MAINTAINED]   [/
CASE_INSENSITIVE]
    ON table_name (column [, column...])
```

index_name identifies the index. If the name contains blanks or special characters, or does not begin with a letter, surround it with the grave character (`) (ASCII 96).

The NON_MAINTAINED switch makes the index non-maintained. The default is to create a maintained index.

The CASE_INSENSITIVE switch makes the index case-insensitive. The default is to create a case-sensitive index.

table_name is the name of the table on which the index is to be based.

column is the name of a column that is included as a the key field for the index.

Paradox 3.0 and 3.5 tables cannot have composite or case-insensitive indexes. When you create a non-primary index for Paradox 3.0 and 3.5 tables, follow these rules:

- Specify only one column name.
- Do not use the CASE_INSENSITIVE switch.
- Use the column name as the index name.

For example:

```
CREATE INDEX last_name ON emp (last_name    )
```

Drop Index Statement

The syntax for dropping a primary index is

```
DROP INDEX  path_name.PRIMARY
```

For example:

```
DROP INDEX emp.PRIMARY
```

The syntax for dropping a non-primary index is

```
DROP INDEX  path_name.index_name
```

path_name is the name of the table from which the index is being dropped. The pathname can be either the fully qualified pathname or, if the table is specified with the Database attribute of the connection string, a simple table name.

index_name is the name that was given to the index when it was created. If the name contains blanks or special characters, or does not begin with a letter, surround it with the grave character (`) (ASCII 96). Use the column name as the index name when dropping indexes from Paradox 3.5 or 3.0 tables.

For example:

```
DROP INDEX emp.last_nam  e
```

Transactions

The Paradox 4 driver supports transactions. A transaction is a series of database changes that is treated as a single unit. In applications that don't use transactions, the Paradox driver immediately executes Insert, Update, and Delete statements on the database tables and the changes are automatically committed when the SQL statement is executed. There is no way to undo such changes. In applications that use transactions, inserts, updates, and deletes are held until a Commit or Rollback is specified. A Commit saves the changes to the database file; a Rollback discards the changes.

Transactions affect the removal of record locking. All locks are removed when SQLTransact is called with the Commit or Rollback option to end the active transaction.

Isolation and Lock Levels Supported

Paradox supports isolation level 1 (read committed). It supports record- and table-level locking. See Appendix D, "Locking and Isolation Levels," on page 342 for a discussion of these topics.

ODBC Conformance Level

The Paradox 4 driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 functions are supported:

- SQLSetPos
- SQLPrimaryKeys

The Paradox 4 driver also supports backward and random fetching in SQLExtendedFetch. The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The Paradox database system supports multiple connections and multiple statements per connection.

Go To

14 Paradox 5 Driver

The Paradox 5 driver supports Paradox 3.0, 3.5, 4.0, 4.5, and 5.0 tables in the Windows environment.

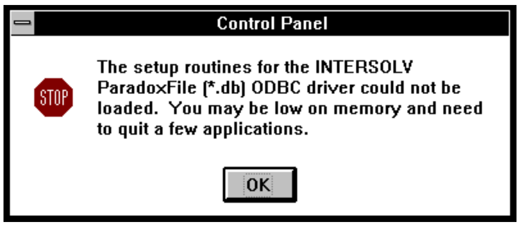
See the README file shipped with your INTERSOLV DataDirect product for the filename of the Paradox 5 driver.

System Requirements

To use the Paradox 5 driver, you must install the Borland Database Engine, which conforms to the IDAPI programming interface. If you have the most recent version of Paradox for Windows or dBASE for Windows, then you should already have the necessary files, which include the following:

- IDAPI01.DLL
- IDPDX01.DLL
- IDRL0009.DLL
- ILD01.DLL

If you attempt to configure a data source and you do not have these files on your path or in your Windows SYSTEM directory, a message similar to the following appears:


Control Panel

The setup routines for the INTERSOLV ParadoxFile (*.db) ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

Multiuser Access to Tables

You can query Paradox tables that reside in a shared directory on a network or that are to be shared among applications running on a local workstation. If the tables are on a network server, multiple users can query these tables simultaneously. To share Paradox tables among multiple users, the tables must be located in a shared directory on your network server.

Two connection attributes identify the Paradox database you are accessing: Database (database directory) and NetDir (network directory). The Database setting specifies the directory of Paradox tables that is the database. If the Database setting you specify is a shared network directory, then Paradox requires a NetDir specification as well. This value identifies the directory containing the PARADOX.NET file that corresponds to the Database setting you have specified.

Every user who accesses the same shared directory of tables must set the NetDir value to point to the same PARADOX.NET file. If your connection string does not specify a NetDir value, then Paradox uses the NetDir value specified in the Paradox section of the IDAPI configuration file. This makes it important that the NetDir specification in each user's IDAPI configuration file be set correctly.

Whenever you open a Paradox table that another user opens at the same time, the consistency of the data becomes an issue if both individuals are updating the table.

Locking

The Paradox 5 driver has two locking levels: record locking and table locking. Tables that have no primary key always have a prevent write lock placed on the table when the table is opened. The table lock is escalated to a write lock when an operation that changes the table is attempted.

Tables that have primary keys use record locking. The locking level is escalated from record locking to a table write lock if the transaction runs out of record locks.

Primary keys provide the greatest concurrency for tables that are accessed and modified by multiple users.

Note: If a table lock is placed on a Paradox table, the Paradox 5 driver prevents users from updating and deleting records but does not prevent them from inserting records into the locked table.

Configuring Data Sources

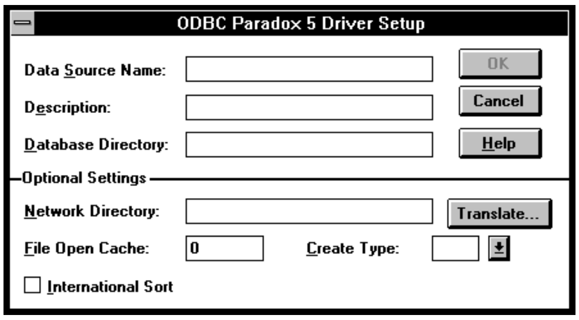
To configure a Paradox 5 data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Paradox 5 driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.



3 Specify values as follows:

Data Source Name:A string that identifies this Paradox data source configuration in ODBC.INI. Examples include “Accounting” or “Paradox5-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “Paradox Files on Server number 1.”

Database Directory: The directory in which the Paradox files are stored. If a directory is not specified, the current working directory is used. You can also specify aliases that are defined in your IDAPI configuration file, if you have one. To do this, enclose the alias name in colons. For example, to use the alias MYDATA, specify “:MYDATA:”

The following values are optional:

Network Directory: The directory containing the PARADOX.NET file that corresponds to the database you have specified. If the Paradox database you are using is shared on a network, then every user who accesses it must set this value to point to the same PARADOX.NET file. If not set here, this value is determined by the NetDir setting in the Paradox section of the IDAPI configuration file. If you are not sure how to set this value, contact your network administrator.

File Open Cache: An integer value to specify the maximum number of unused file opens to cache. For example, the value 4 specifies that when a user opens and closes four tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is increased performance. The disadvantage is that a user who specifies file locking on open may get a locking conflict even though no one appears to have the file open. The default is 0, which means no file open caching.

Create Type: The Paradox table version desired for any Create Table statements that you execute. You can specify 3, 4, 5, or leave the box blank and use the default, which is determined by the Level setting in the Paradox section of the IDAPI configuration file. The numeric values map to the major revision numbers of the Paradox family of products.

4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

   The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value].
..]
```

An example of a connection string for Paradox is

```
DSN=PARADOX TABLES;DB=C:\ODBC\EMP;PW=ABC,DEF,GH     I
```

Table 14-1 gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 14-1. Paradox 5 Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies a Paradox data source configuration in ODBC.INI. Examples include “Accounting” or “Paradox5-Serv1.”
Database (DB)	The directory in which the Paradox files are stored For this attribute, you can also specify aliases that are defined in your IDAPI configuration file, if you have one. To do this, enclose the alias name in colons. For example, to use the alias MYDATA, specify “Database=:MYDATA:”
NetDir (ND)	The directory containing the PARADOX.NET file that corresponds to the database you have specified. If the Paradox database you are using is shared on a network, then every user who accesses it must set this value to point to the same PARADOX.NET file. If not specified, this value is determined by the NetDir setting in the Paradox section of the IDAPI configuration file. If you are not sure how to set this value, contact your network administrator.

Table 14-1. Paradox 5 Connection String Attributes (cont.)

Attribute	Description
FileOpenCache (FOC)	The maximum number of unused table opens to cache. For example, when FileOpenCache=4, and a user opens and closes 4 tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The disadvantage is that a user who tries to open the table exclusively may get a locking conflict even though no one appears to have the table open. The initial default is 0.
CreateType (CT)	CreateType={3|4|5|null (blank)}. This attribute specifies the table version for Create Table statements. There are four valid values for this connection string: 3, 4, 5 and null (blank). The numeric values map to the major revision numbers of the Paradox family of products. To override another CreateType setting chosen during data source configuration with the default create type determined by the Level setting in the Paradox section of the IDAPI configuration file, set CreateType= (null).
IntlSort (IS)	IntlSort={0|1}. This attribute determines the order that records are retrieved when you issue a Select statement. If IntlSort=1, the driver uses the international sort order as defined by your operating system. The sort is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (check your operating system documentation). If IntlSort=0 (the default), the driver uses the ASCII sort order, where uppercase letters precede lowercase letters (*B* precedes *a*).

Table 14-1. Paradox 5 Connection String Attributes (cont.)

Attribute	Description
UltraSafeCommit (USF)	UltraSafeCommit={0|1}. This attribute determines when the driver flushes its changes to disk. If UltraSafeCommit=1, the driver does this at each COMMIT. This decreases performance. The default is 0. This means that the driver flushes its changes to disk when the table is closed or when internal buffers are full. In this case, a machine “crash” before closing a table may cause recent changes to be lost.
DeferQueryEvaluation (DQ)	DeferQueryEvaluations={0|1}. This attribute determines when a query is evaluated—after all records are read or each time a record is fetched. If DeferQueryEvaluation=0, the driver generates a result set when the first record is fetched. The driver reads all records, evaluates each one against the Where clause, and compiles a result set containing the records that satisfy the search criteria. This process slows performance when the first record is fetched, but activity performed on the result set after this point is much faster, because the result set has already been created. You do not see any additions, deletions, or changes in the database that occur while working from this result set. If DeferQueryEvaluation=1 (the default), the driver evaluates the query each time a record is fetched, and stops reading through the records when it finds one that matches the search criteria. This setting avoids the slowdown while fetching the first record, but each fetch takes longer because of the evaluation taking place. The data you retrieve reflects the latest changes to the database. However, a result set is still generated if the query is a Union of multiple Select statements, if it contains the Distinct keyword, or if it has an Order By or Group By clause.

Table 14-1. Paradox 5 Connection String Attributes (cont.)

Attribute	Description
Passwords (PW)	A password or list of passwords. You can add 1 to 50 passwords into the system using a comma-separated list of passwords. Passwords are case-sensitive. For example, Passwords=psw1, psw2, psw3.

Data Types

Table 14-2 shows how the Paradox 5 data types are mapped to the standard ODBC data types. It also identifies the types supported by Paradox versions 3.*x* and 4.*x*. These Paradox 5 data types can be used in a Create Table statement. For the syntax of the Create Table statement, see Appendix A, "SQL for Flat-File Drivers," on page 299.

Table 14-2. Paradox 5 Data Type

Paradox	ODBC	3.*x* Support	4.*x* Support
Alpha	SQL_VARCHAR	Yes	Yes
AutoIncrement	SQL_INTEGER	No	No
BCD	SQL_DECIMAL	No	No
Binary*	SQL_LONGVARBINARY	No	Yes
Bytes*	SQL_BINARY	No	No
Date	SQL_DATE	Yes	Yes
Formatted Memo*	SQL_LONGVARBINARY	No	Yes
Graphic*	SQL_LONGVARBINARY	No	Yes
Logical*	SQL_BIT	No	No

Table 14-2. Paradox 5 Data Type (cont.)

Long Integer	SQL_INTEGER	No	No
Memo*	SQL_LONGVARCHAR	No	Yes
Money	SQL_DOUBLE	Yes	Yes
Number	SQL_DOUBLE	Yes	Yes
OLE*	SQL_LONGVARBINARY	No	Yes
Short	SQL_SMALLINT	Yes	No
Time	SQL_TIME	No	No
TimeStamp	SQL_TIMESTAMP	No	No

*Cannot be used in an index. Of these types, only Logical can be used in a Where clause.

Select Statement

You use a SQL Select statement to specify the columns and records to be read. The Paradox 5 driver supports all of the Select statement clauses as described in Appendix A, “SQL for Flat-File Drivers,” on page 299. This section describes the information that is specific to the Paradox 5 driver, which is column names.

Column Names

Paradox column names are case-insensitive and their maximum length is 25 characters. If a column name contains a special character, does not begin with a letter, or is a reserved word, surround it with the grave character (`) (ASCII 96). For example:

```
SELECT `last name` FROM emp
```

Password Protection

Paradox supports two types of passwords: master and auxiliary. The Paradox 5 driver supports master passwords only and can manage up to 50 passwords.

Paradox tables may be encrypted to provide limited access to users who do not know the password. The driver maintains a list of passwords for each connection. The driver can access only encrypted tables for which a password appears in this list. You can supply a password in three ways: by typing it in the Password dialog box (which appears when the driver does not have the password to open an encrypted table), by including it in a connection string (with the Passwords attribute), or by using the Add Password statement.

Paradox provides five statements that manage passwords for Paradox tables. These statements are specific to the Paradox driver:

```
ENCRYPT filename USING PASSWORD password
```

```
ADD PASSWORD password
```

```
DECRYPT filename USING PASSWORD password
```

```
REMOVE PASSWORD password
```

```
REMOVE ALL PASSWORDS
```

filename can be a simple filename or a full pathname. If a simple filename is given, the file must be in the directory specified with the Database connection string attribute. The .DB extension is not required.

password is a case-sensitive text string up to 15 characters in length, including blanks. If your password includes lower-case letters or nonalphanumeric characters, enclose it in single quotation marks (').

Encrypting a Paradox Table

The Encrypt statement associates a password with a table. For example:

```
ENCRYPT emp USING PASSWORD tes   t
```

Accessing an Encrypted Paradox Table

To access an encrypted Paradox table, add the password to the list of passwords Paradox maintains for that connection. To do so, you can

- Issue an Add Password statement before you access the table. For example:

  ```
  ADD PASSWORD test
  SELECT * FROM em p
  ```

- Specify the passwords using the Passwords attribute at connection time.

If you do not add the password, the driver prompts you for it when you access the table.

Decrypting a Paradox Table

The Decrypt statement disassociates a password from a table. You no longer need to enter the password to open the table. For example:

```
DECRYPT emp USING PASSWORD tes   t
```

Removing a Password from Paradox

The Remove Password statement removes a password from the list Paradox maintains for the connection. For example:

```
REMOVE PASSWORD tes  t
```

Removing All Passwords from Paradox

The Remove All Passwords statement removes the list of passwords Paradox maintains.

If you remove a password from Paradox and do not decrypt the table, you must continue entering the password to open the table.

Index Files

An index is used to read records in sorted order and to improve performance when selecting records and joining tables. Paradox indexes are stored in separate files and are either *primary* or *non-primary*.

Primary Index

A primary index is made up of one or more fields from the Paradox table. The primary key fields of a primary index consist of one or more consecutive fields in the table, beginning with the first field in the table. A table can have only one primary index.

Collectively, the primary key fields uniquely identify each record in the Paradox table. Thus, no two records in a Paradox table can share the same values in their primary key fields. Once a primary index is created for a Paradox table, the table’s records are re-ordered based on the primary key fields. At the time a primary index is created, if any records have matching primary key field values, Paradox deletes all but the first record. Paradox creates this index as maintained; that is, if you modify, add, or delete records in the table, the primary index is updated automatically to reflect these changes.

A primary index is a single file with the same name as the table on which it is based but with a .PX extension.

To lock records, you must have a primary index.

Non-Primary Index

Unlike a primary key field of a primary index, a non-primary key field does not uniquely identify each record. Thus, two or more records in a Paradox table can share the same value in their non-primary key field. A Paradox table can have more than one non-primary index, as long as each one is based on different fields.

A non-primary index is defined by specifying one or more fields in the Paradox table that constitute the non-primary key field. It allows Paradox to sort each record in the table according to the values of the non-primary key fields.

There are two kinds of non-primary indexes: maintained and non-maintained. The difference between the two is that a maintained index is automatically updated when the table is changed, whereas a non-maintained index is not. Instead, a non-maintained index is tagged out-of-date and is updated when the index is used again.

You must have a primary index on a table before you create a maintained, non-primary index.

The Paradox 5 driver uses non-maintained indexes only for read-only queries on locked tables. A primary index is not required for the non-maintained index to be used.

For Paradox 3.*x*, a single non-primary index consists of a pair of files with the same name as the table on which the non-primary index is based; one of these files has an .X*nn* extension while the other has a .Y*nn* extension (where the hexadecimal number *nn* corresponds to the field number of the non-primary key field for the non-primary index).

For Paradox 4.*x* and 5, single-field non-primary indexes that are case sensitive have the same name as their associated table and are assigned file extensions .X01 through .XFF, depending on the number of the field on which the index is based. Single-field non-primary indexes that are case insensitive and composite indexes have the same name as the table on which they are based. They are assigned file extensions sequentially starting with .XG0 (with hexadecimal increments).

Create and Drop Index Statements

The Paradox driver supports SQL statements to create and delete indexes. The Create Index Primary statement is used to create primary indexes. The Create Index statement is used to create non-primary indexes. The Drop Index statement is used to delete indexes.

Create Index Primary Statement

The syntax for creating a primary index is

```
CREATE [UNIQUE] INDEX PRIMARY
    ON table_name (column [,column...])
```

The UNIQUE keyword is optional; the index is unique whether or not you include this keyword.

table_name is the name of the table on which the index is to be based.

column is the name of a column that is included as a the key field for the index. The column list must contain one or more consecutive fields in the table, beginning with the first field in the table.

For example:

```
CREATE UNIQUE INDEX PRIMARY ON emp (emp_id    )
```

Be careful when you create a primary key because any rows that have a primary key duplication are deleted when you execute the Create Index Primary statement.

Create Index Statement

The syntax for creating a non-primary index is

```
CREATE INDEX  index_name [/NON_MAINTAINED]   [/
CASE_INSENSITIVE]
    ON table_name (column [, column...])
```

index_name identifies the index. If the name contains blanks or special characters, or does not begin with a letter, surround it with the grave character (`) (ASCII 96).

The NON_MAINTAINED switch makes the index non-maintained. The default is to create a maintained index.

The CASE_INSENSITIVE switch makes the index case-insensitive. The default is to create a case-sensitive index.

table_name is the name of the table on which the index is to be based.

column is the name of a column that is included as a the key field for the index.

Paradox 3.0 and 3.5 tables cannot have composite or case-insensitive indexes. When you create a non-primary index for Paradox 3.0 and 3.5 tables, follow these rules:

- Specify only one column name.
- Do not use the CASE_INSENSITIVE switch.
- Use the column name as the index name.

For example:

```
CREATE INDEX last_name ON emp (last_name    )
```

Drop Index Statement

The syntax for dropping a primary index is

```
DROP INDEX  path_name.PRIMARY
```

For example:

```
DROP INDEX emp.PRIMARY
```

The syntax for dropping a non-primary index is

```
DROP INDEX  path_name.index_name
```

path_name is the name of the table from which the index is being dropped. The pathname can be either the fully qualified pathname or, if the table is specified with the Database attribute of the connection string, a simple table name.

index_name is the name that was given to the index when it was created. If the name contains blanks or special characters, or does not begin with a letter, surround it with the grave character (`) (ASCII 96). Use the column name as the index name when dropping indexes from Paradox 3.5 or 3.0 tables.

For example:

```
DROP INDEX emp.last_name
```

Transactions

The Paradox 5 driver supports transactions. A transaction is a series of database changes that is treated as a single unit. In applications that don’t use transactions, the Paradox 5 driver immediately executes Insert, Update, and Delete statements on the database tables and the changes are automatically committed when the SQL statement is executed. There is no way to undo such changes. In applications that use transactions, inserts, updates, and deletes are held until a Commit or Rollback is specified. A Commit saves the changes to the database file; a Rollback discards the changes.

Transactions affect the removal of record locking. All locks are removed when SQLTransact is called with the Commit or Rollback option to end the active transaction.

Isolation and Lock Levels Supported

Paradox 5 supports isolation levels 1 (read committed) and 3, (serializable). It supports record- and table-level locking. See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The Paradox 5 driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 functions are supported:

- SQLSetPos
- SQLPrimaryKeys

The Paradox 5 driver also supports backward and random fetching in SQLExtendedFetch. The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The Paradox database system supports multiple connections and multiple statements per connection.

Go To

15 PROGRESS Driver

The PROGRESS driver supports version 6 of the PROGRESS database system in the Windows environment.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the PROGRESS driver.

System Requirements

You must have the PROGRESS Windows Runtime client software to access a local- or server-based PROGRESS database.

If you are using LAN Workplace TCP/IP, HP TCP/IP, or Wollongong TCP/IP, or if you are reinstalling a PROGRESS driver that uses the network protocols NetBIOS, SPX, or FTP TCP/IP, you must follow these steps after you install the driver but before you configure a data source:

1 Exit Windows.

2 Change to your install directory, C:\QEODBC (for the DataDirect ODBC Pack), C:\QELINK (for DataDirect MultiLink/VB), or C:\QELIB (for the DataDirect Developer's Tool Kit).

3 Enter the following command.

```
C:> PROGRESS \WINDOWS\SYSTEM option
```

You must specify the Windows SYSTEM directory. If your machine has a different name for the Windows directory, use that name. The option you choose depends on the network protocol you are using.

If you have	option is
NetBIOS, SPX, FTP TCP/IP	WS
LAN Workplace TCP/IP	LAN
HP TCP/IP	HP
Wollongong TCP/IP	WOL

Configuring Data Sources

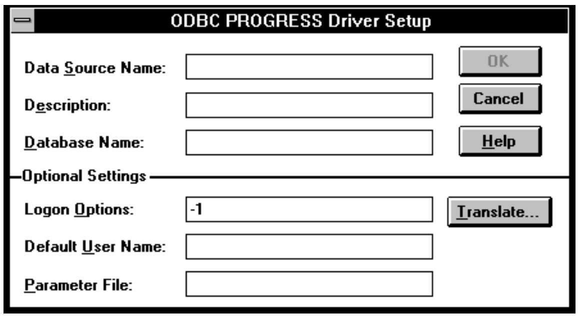
To configure a PROGRESS data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the PROGRESS driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.


ODBC PROGRESS Driver Setup

Data Source Name:

Description:

Database Name:

Optional Settings

Logon Options: -1

Default User Name:

Parameter File:

3 Specify values as follows:

Data Source Name: A string that identifies this PROGRESS data source configuration in ODBC.INI. Examples include “Accounting” or “PROG-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “PROGRESS on Server number 1.”

Database Name: The name of the database to which you want to connect by default.

The following values are optional:

Logon Options: The startup options. The format for specifying these options is

```
[-db database] [-u userid] [-p password] [-1]
```

Use the **-1** option (note that this is the number 1) to access a local database.

Alternatively, you can specify these options in a parameter file. In this case, you need not specify startup options here; instead, you specify the parameter filename in the Parameter File field.

Default User Name: The default user name used to connect to your PROGRESS database. A user name is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the logon dialog box or connection string.

Parameter File: The name of the parameter file that contains the logon options. See also the description of the Logon Options field.

4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from

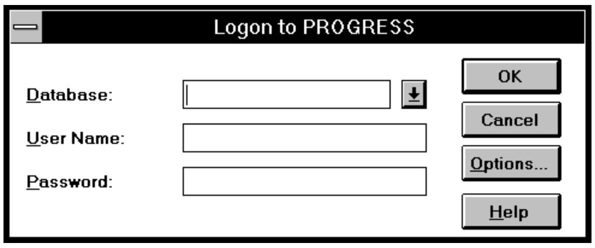
the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For PROGRESS, the dialog box is as follows:



In this dialog box, do the following:

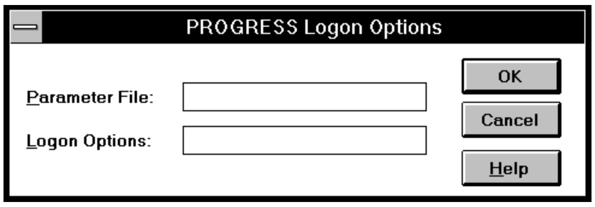
1 Type the name of the database to which you want to connect.

2 If required, type your user name.

3 If required, type your password.

User name and password are optional parameters. You can log on to the database without these parameters, but you may not be able to create, delete, or manipulate the data.

4 Click **Options** to display the PROGRESS Logon Options dialog box.



If the options are stored in a parameter file, then type its name in the Parameter File box. The options are read from that file. To display the options from the file, move to the Logon Options box.

To access more than one database at a time, you must specify all the database names. The format is

```
[-db database] [-u userid] [-p password] [-1]
```

Use the **-**1 option (note that this is the number 1) to access a local database.

If you change the options and click **OK**, the modified options are saved to the parameter file. You can click **Cancel** and not save the changes to the file, but the options you specified will not be used for this connection.

5 Click **OK** to complete the logon and to update the values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_nam e[;attribute=valu e[;attribute=valu e].
..]"
```

An example of a connection string for PROGRESS is

```
DSN=PROGRESS ON
PION1;DB=PAYROLL;UID=JOHN;PWD=XYZZY;PF=OPTS     2
```

Table 15-1 gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 15-1. PROGRESS Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies a PROGRESS data source configuration in ODBC.INI. Examples include "Accounting" or "PROG-Serv1."
Database (DB)	The name of the database to which you want to connect

Table 15-1. PROGRESS Connection String Attributes (cont.)

Options (OPTS)	The startup options. The format is `[-db database] [-u userid] [-p password] [-1]` Use the -1 option (the number 1) to access a local database. See the PROGRESS manual *System Administration II* for available startup options. Alternatively, you can specify these options in a parameter file. In this case, you need not specify startup options here; instead, you specify the name of the parameter file using the PfName attribute.
LogonID (UID)	The default logon ID (user name) used to connect to your PROGRESS database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID. This ID is case-sensitive
Password (PWD)	A case-sensitive password.
PfName (PF)	The name of parameter file containing the startup options

Data Types

Table 15-2 shows how the PROGRESS data types are mapped to the standard ODBC data types.

Table 15-2. PROGRESS Data Types

PROGRESS	ODBC
Character	SQL_CHAR
Date	SQL_DATE

Table 15-2. PROGRESS Data Types (cont.)

PROGRESS	ODBC
Decimal	SQL_DECIMAL
Integer	SQL_INTEGER
Logical	SQL_BIT
Numeric	SQL_NUMERIC
Smallint	SQL_INTEGER

Isolation and Lock Levels Supported

PROGRESS supports isolation level 1 (read committed). PROGRESS supports record-level locking. See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The PROGRESS driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. The driver supports the core SQL grammar.

Connections and Statements Supported

The PROGRESS database system supports a single connection and multiple statements per connection.

Go To

16 Scalable SQL Driver

The Scalable SQL driver supports the Scalable SQL database system in the Windows environment.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the Scalable SQL driver.

System Requirements

You must start the Scalable SQL Requester Utility, NSREQ.EXE, before running Windows (except if you are using Scalable SQL 3.0.1—version 3.0.1 does not require NSREQ to be running). NSREQ.EXE must have a minimum data size setting of 4096, which is the default value. For example, to start NSREQ with a data size of 32K, enter the following at the DOS prompt:

```
NSREQ /D:3276 7
```

To gain access to Scalable SQL databases using a client system, you must have the following files: WXQLCALL.DLL and NSREQ.EXE.

Configuring Data Sources

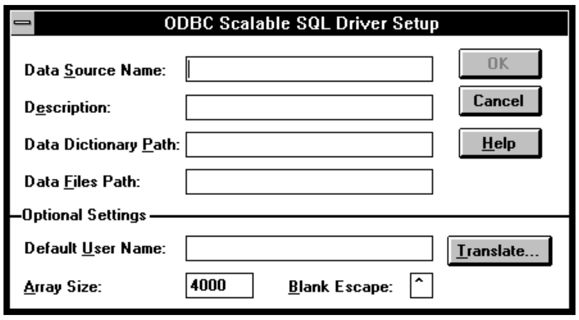
To configure a Scalable SQL data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Scalable SQL driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.


ODBC Scalable SQL Driver Setup

Data Source Name:
Description:
Data Dictionary Path:
Data Files Path:
Optional Settings
Default User Name:
Array Size: 4000
Blank Escape: ^

3 Specify values as follows:

Data Source Name: A string that identifies this Scalable SQL data source configuration in ODBC.INI. Examples include “Accounting” or “Scalable SQL-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “Scalable SQL on Server number 1.”

Data Dictionary Path: The directory or path of the data dictionary files you want to access. Your ODBC application uses this path to connect to Scalable SQL and give you access to the data dictionary files in that directory. This may be the same as your Data Files Path.

Data Files Path: The directory or path of the data files you want to access. Your ODBC application uses this path to connect to Scalable SQL and give you access to the data files in that directory.

The following values are optional:

Default User Name: The default user name used to connect to your Scalable SQL database. This value is case-sensitive. Your ODBC application may override this value or you may override this value in the Logon dialog box or connection string. A user name is required only if security is enabled on your database.

Array Size: The number of bytes the driver uses for fetching multiple rows. Values can be integers from 0 to 65536 bytes; the default is 4000. Larger values increase throughput by reducing the number of times the driver fetches data across the network. Smaller values increase response time, as there is less of a delay waiting for the server to transmit data.

Blank Escape: Scalable SQL lets you specify blanks as part of table or field names. The blank escape character is used to replace these blanks. The default blank escape character is a carat (^). Other values are tilde (~) and underscore (_).

4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

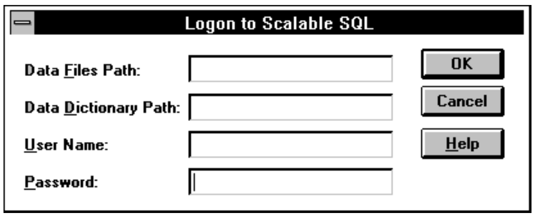
   The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the

defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For Scalable SQL, the dialog box is as follows:



In this dialog box, do the following:

1 Type the database name or the complete pathname of the directory containing your data files in the Data Files Path box.

2 Type the database name or the complete pathname of the directory containing your data dictionary files in the Data Dictionary Path box.

3 If security has been enabled on your Scalable SQL database, type your user name and password.

4 Click **OK** to complete the logon and to update the values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for Scalable SQL is

```
DSN=SCALSQL TABLES;UID=JOHN;PWD=XYZZY
```

Table 16-1 gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 16-1. Scalable SQL Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies a Scalable SQL data source configuration in ODBC.INI. Examples include “Accounting” or “Scalable SQL-Serv1.”
DataDictPath (DICT)	The directory or path of the data dictionary files that you want to access. Your ODBC application uses this path to connect to Scalable SQL and give you access to the data dictionary files in that directory. This value may be the same as the value of DataFilesPath.
DataFilesPath (FILES)	The directory or path of the data files you want to access. Your ODBC application uses this path to connect to Scalable SQL and give you access to the data files in that directory.
LogonID (UID)	The default logon ID (user name) used to connect to your Scalable SQL database. This value is case-sensitive. This setting is required only if security is enabled on your database system. If so, contact your system administrator to get your logon ID.
Password (PWD)	A case-sensitive password.
BlankEscape (BLANK)	Scalable SQL lets you specify blanks as part of table or field names. The blank escape character is used to replace these blanks. The default blank escape character is a carat (^). Other values are tilde (~) and underscore (_).

Table 16-1. Scalable SQL Connection String Attributes (cont.)

Attribute	Description
ArraySize (AS)	The number of bytes the driver uses for fetching multiple rows. Values can be integers from 0 to 65536. The initial default is 4000. Larger values increase throughput by reducing the number of times the driver fetches data across the network. Smaller values increase response time, as there is less of a delay waiting for the server to transmit data.
ModifySQL (MS)	ModifySQL={0|1}. This attribute is provided for backward compatibility with earlier versions of Q+E products. It determines whether the driver modifies SQL statements to conform to ODBC specifications or passes the SQL statement directly to Scalable SQL. Specify ModifySQL=1 to have the driver modify the SQL statement to conform to ODBC specifications. Specify ModifySQL=0 to have the driver understand SQL dialects found in earlier drivers. The default is 1.

Data Types

Table 16-2 shows how the Scalable SQL data types are mapped to the standard ODBC data types.

Table 16-2. Scalable SQL Data Types

Scalable SQL	ODBC
Autoinc	SQL_SMALLINT
Autoinc(4)	SQL_INTEGER
Bfloat(4)	SQL_REAL

Table 16-2. Scalable SQL Data Types (cont.)

Scalable SQL	**ODBC**
Bfloat(8)	SQL_DOUBLE
Bit	SQL_BIT
Character	SQL_CHAR
Date	SQL_DATE
Decimal	SQL_DECIMAL
Float(4)	SQL_REAL
Float(8)	SQL_DOUBLE
Int	SQL_SMALLINT
Int(1)	SQL_TINYINT
Int(4)	SQL_INTEGER
Logical	SQL_BIT
Logical(2)	SQL_BIT
Lstring	SQL_VARCHAR
Lvar	SQL_LONGVARCHAR
Money	SQL_DECIMAL
Note	SQL_LONGVARCHAR
Numeric	SQL_NUMERIC
Numericsts	SQL_NUMERIC
Time	SQL_TIME
Zstring	SQL_VARCHAR

Isolation and Lock Levels Supported

Scalable SQL supports isolation levels 1 (read committed). Scalable SQL supports page-level locking.

See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The Scalable SQL driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 functions are supported:

- SQLBrowseConnect
- SQLProcedures

The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The Scalable SQL database system version 3.1 supports multiple connections and multiple statements per connection. Versions of the Scalable SQL database system prior to version 3.1 support single connections and multiple statements per connection.

For Programmers Only

Scalable SQL allows multiple applications to share a single connection to the database. To implement this functionality, your application must retrieve the session ID value assigned to the initial connection, and then pass that ID as part of the connection string from other applications that share the connection.

To do this, first establish a connection by logging on to Scalable SQL with the standard DataDictPath, DataFilesPath, LogonID, and Password values. Then, get the session ID assigned to that connection by calling SQLGetConnectOption with 1051 as the *fOption* argument.

To share the connection, pass its session ID in the connection strings of other applications. The attribute SessionID (SID) is provided for this purpose.

Go To

17 SQL Server Driver

The SQL Server driver supports the SQL Server database system available from Microsoft and Sybase, Inc. in the Windows, Windows 95, Windows NT, OS/2, Macintosh, and the UNIX (not including Solaris for x86) environments.

It supports the Sybase Net gateway to DB2 in the Windows, Windows 95, Windows NT, OS/2, and UNIX (not including Solaris for x86) environments.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the SQL Server driver.

System Requirements

You must have the appropriate DB-Library and Net-Library installed to gain access to Microsoft SQL Server or Sybase SQL Server databases.

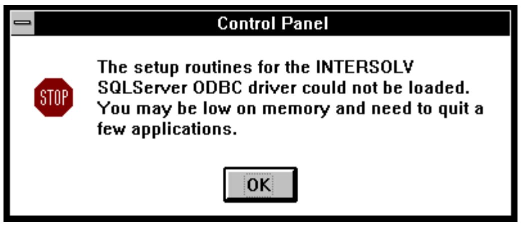
Your database must support catalog stored procedures.

Windows

The DB-Library is W3DBLIB.DLL. The Net-Library you need depends on the network protocol used to connect to SQL Server. For example, Named Pipes requires DBNMP3.DLL, FTP's TCP/IP requires WDBFTPTC.DLL, and Novell's LAN Workplace for DOS requires WDBNOVTC.DLL. Contact your Microsoft SQL Server or Sybase SQL Server vendor to obtain the appropriate DB-Library and Net-Library.

WDBPING is a Windows tool provided with net-libraries to test connectivity from your client workstation to the database server. Use this tool to test your connection.

If you attempt to configure a data source and you do not have W3DBLIB.DLL on your path or in your Windows SYSTEM directory, a message similar to the following appears:

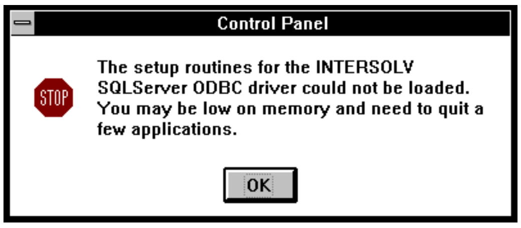

Control Panel

The setup routines for the INTERSOLV SQLServer ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

Windows 95 and Windows NT

The DB-Library is NTWDBLIB.DLL. The Net-Library you need depends on the network protocol used to connect to the SQL Server. For example, Named Pipes requires DBNMPNT.DLL. Contact your Microsoft SQL Server or Sybase SQL Server vendor to obtain the appropriate DB-Library and Net-Library.

If you attempt to configure a data source and you do not have NTWDBLIB.DLL on your path or in your Windows 95 or Windows NT SYSTEM directory, a message similar to the following appears:


Control Panel

The setup routines for the INTERSOLV SQLServer ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

OS/2

System requirements are as follows:

- Net-Library version 2.0 (32-bit) or higher, which is required to set up your network environment. Contact your Sybase SQL Server vendor for more information on Net-Library.
- DB-Library version 4.6 (32-bit) or higher, available from Sybase.

All DLLs from DB-Library and Net-Library must be in directories listed in the environment variable LIBPATH in your CONFIG.SYS file.

Macintosh

System requirements are as follows:

- 8 MB of memory
- MacTCP version 1.1 or greater
- The appropriate TCP/IP software

UNIX (AIX, HP-UX, and Solaris for SPARC)

Before you can use the SQL Server data source, you must have the Sybase Open Client Net-Libraries you plan to use installed on your workstation in the $SYBASE source tree.

Set the environment variable SYBASE to the directory where you installed the Sybase Open Client Net-Libraries. For example, for C-shell users, the following syntax is valid:

```
setenv SYBASE /databases/sybase
```

For Bourne- or Korn-shell users, the following syntax is valid:

```
SYBASE=/databases/sybase;export SYBASE
```

You must include the directory containing the System 10 client-shared libraries in the environment variable LD_LIBRARY_PATH (on Solaris), LIBPATH (on AIX), and SHLIB_PATH (on HP-UX). For example, for C-shell users, the following syntax is valid:

```
setenv LD_LIBRARY_PATH /databases/sybase/
lib:$LD_LIBRARY_PAT  H
```

For Bourne- or Korn-shell users, the following syntax is valid:

```
LD_LIBRARY_PATH=/databases/sybase/
lib:$LD_LIBRARY_PATH;export LD_LIBRARY_PAT    H
```

Building the Required Sybase Open Client Net-Libraries

The Sybase SQL Server driver requires a one-time site linking to build the Sybase Open Client Net-Libraries on Solaris and AIX. On HP-UX, these libraries are provided in the System 10 Open Client from Sybase.

Before you build these libraries, install Sybase SQL Server and set the environment variable SYBASE to the directory where you installed Sybase SQL Server.

A make file is provided that builds the Sybase Open Client Net-Libraries. This make file (qesybase.mk) is in the bin directory. These Sybase libraries should be either built in or copied to the directory in which the other INTERSOLV ODBC DLLs are installed, usually qelib/dlls or odbc/dlls.

The following example builds the Sybase Open Client-Net Libraries if you are in the qelib/bin or odbc/bin directory**:**

```
%make -f qesybase.mk EXE=../dll   s
```

The EXE= directive builds the Sybase Open Client-Net Libraries in the location you specify.

Stored Procedures

This driver requires stored procedure support that may or may not be installed on your database. To test to see if the required stored procedures have been installed in your database, issue the following commands from isql:

```
%sp_server_inf o
%go
```

If you get an error that says that the stored procedure “sp_server_info” does not exist, then you must install those stored procedures for the Sybase driver to connect. Within this release you will find a subdirectory called sqlsrv. In the sqlsrv subdirectory is a CSH shell script to install the required stored procedures. The script install.csh has help built into it.

Issue the following commands to get a list of how to use the install.csh script:

```
%cd sqlsr v
% ./install.csh hel  p
```

Configuring Data Sources

UNIX

Note: In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the .odbc.ini file using the attributes in Table 17-1. For information about this file, see Appendix E, “ODBC.INI,” on page 346.

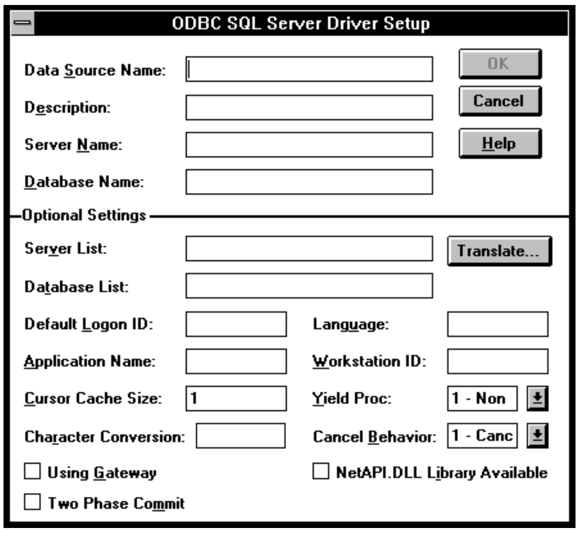
To configure a SQL Server data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select INTERSOLV SQLServer and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup** (if you are using Apple's ODBC Driver Manager on the Macintosh, this button is labeled **Modify**).

The setup dialog box appears.



3 Specify values as follows:

Data Source Name:A string that identifies this SQL Server data source configuration in ODBC.INI. Examples include “Accounting” or “SQL Server-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “SQL Server on Server number 1.”

Server Name: The name of the server that contains the desired database. On UNIX, the name of a server from your $SYBASE/interfaces file.

Database Name: The name of the database to which you want to connect by default. If you do not specify a value, the default database defined by SQL Server is used.

The following values are optional:

Server List: A comma-separated list of servers that will appear in the logon dialog box.

Database List: The databases that will be available in the SQL Server Logon Options dialog box. Separate the names with commas.

Default Logon ID: The default logon ID used to connect to your SQL Server database. This ID is case-sensitive. A logon ID is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the logon dialog box or connection string.

Language: The national language to be used by the client. The default is English.

Application Name: The name SQL Server uses to identify your application.

Workstation ID: The workstation ID used by the client.

Cursor Cache Size: The number of cursors the cursor cache can hold. The driver creates a cache of statements; each statement represents an open connection to SQL Server. The cursor cache increases performance but uses database resources. The default is 1 (one cursor).

Yield Proc: A integer value of 0, 1, or 3 that determines whether you can work in other applications when SQL Server is busy. This attribute is useful to users of ODBC applications.

- 0 (peek and dispatch), which causes the driver to check the Windows message queue and send any messages to the appropriate Windows application.
- 1 (no yielding, the default), which does not let you work in other

applications.

- 3 (dispatch via Windows Yield function), which turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window.

It is recommended that you use the value 1.

Character Conversion: This value controls the character set conversion between SQL Server (version 4.8 or later) and a client application. If you omit this value, no character conversion takes place on your server.

Common values include iso-1 for ISO-8859-1, cp850 for Code Page 850, roman8 for Roman8 character set, and SJIS for a Japanese character set. See your SQL Server documentation for a complete list of values.

Cancel Behavior: An integer value of 0, 1, or 2 that specifies how a previously executed statement should be canceled.

- 0 fetches all of the remaining records if the statement was a Select.
- 1 cancels the statement by calling dbcancel. This is the default (in all environment except the Macintosh—the default the Macintosh is 0) and should be used if dbcancel is supported in your client/server configuration.
- 2 closes the connection to the server for the statement. Use this value only if dbcancel is not supported for your configuration and the performance of fetching all remaining records is unacceptable.

Using Gateway: Select this check box if you are using Sybase Net-Gateway to access a DB2 database with this data source.

NETAPI.DLL Library Available: The driver uses NETAPI.DLL to get the name of your workstation. Most major PC networks support this feature. If your network supports this capability, select this option. If you supply a workstation ID, this field is ignored.

Two-Phase Commit This check box, when selected, enables you to have two active statements within a transaction, using the SQL Server two-phase commit services. The active statements may deadlock if they reference the same SQL Server table.

4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

UNIX

Note: In the UNIX environment, there is no graphical interface. To perform a translation, you must edit the .odbc.ini file and add to the data source section the TranslationDLL keyword and optionally the TranslationOption keyword:

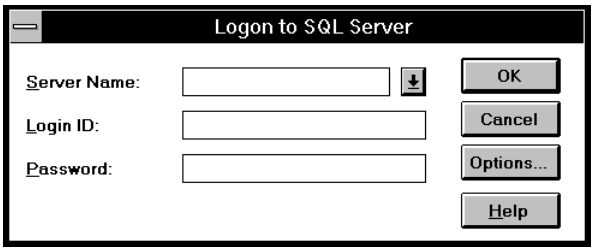
Keyword	**Definition**
TranslationDLL	Full path of translation DLL
TranslationOption	ASCII representation of the 32-bit integer translation option

The INTERSOLV OEM ANSI translator is shipped on UNIX to provide an example of how to create a translation DLL.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For SQL Server, the dialog box is as follows:



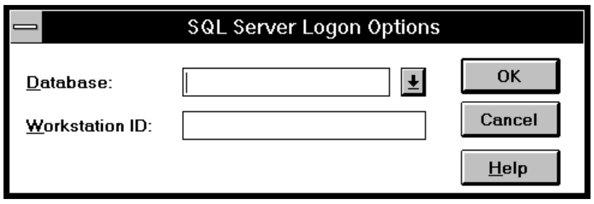
In this dialog box, do the following:

1 Type the name of the server containing the SQL Server database tables you want to access (case-sensitive) or select the name from the Server Name drop-down list, which displays the server names you specified in the setup dialog box.

2 If required, type your case-sensitive login ID.

3 If required, type your case-sensitive password for the system.

4 Optionally, click **Options** to display the SQL Server Logon Options dialog box and specify the initial SQL Server database to connect to and the name of your workstation.



5 Click **OK** to log on to the SQL Server database installed on the server you specified and to update the values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for SQL Server is

```
DSN=Accounting;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

Table 17-1 gives the long and short names for each attribute, as well as a description. To configure a data source in the UNIX environment, you must edit the .odbc.ini file. This file accepts only long names for attributes.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 17-1. SQL Server Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies a SQL Server data source configuration in ODBC.INI. Examples include “Accounting” or “SQL Server-Serv1.”
ServerName (SRVR)	The name of the server containing the SQL Server tables you want to access. On UNIX, the name of a server from your $SYBASE/interfaces file.
Database (DB)	The name of the database to which you want to connect
LogonID (UID)	The case-sensitive logon ID used to connect to your SQL Server database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.
Password (PWD)	A case-sensitive password.
Language (LANG)	The national language to be used by the client. The initial default is English.
ApplicationName (APP)	The name SQL Server uses to identify your application
WorkstationID (WKID)	The workstation ID used by the client.
CursorCacheSize (CCS)	The number of cursors the cursor cache can hold. The driver creates a cache of statements; each statement represents an open connection to SQL Server. The cursor cache increases performance but uses database resources. The initial default is 1.

Table 17-1. SQL Server Connection String Attributes (cont.)

Attribute	Description
YieldProc (YLD)	YieldProc={0|1|3}. This attribute determines if you can work in other applications when SQL Server is busy. This attribute is useful to users of ODBC applications. Valid values are: • YieldProc=0 (peek and dispatch) causes the driver to check the Windows message queue and send any messages to the appropriate Windows application. • YieldProc=1 (no yielding, the initial default) does not let you work in other applications. • YieldProc=3 (dispatch via Windows Yield function) turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window. It is recommended that you use YieldProc=1. YieldProc=0 and YieldProc=3 do not work with all Windows applications.
CharConv (CC)	A value that controls the character set conversion between SQL Server (version 4.8 or later) and a client application. Common values include iso-1 for ISO-8859-1, cp850 for Code Page 850, roman8 for the Roman8 character set, and SJIS for a Japanese character set. See your SQL Server documentation for a complete list of values.
Cancel (CAN)	Cancel={0|1|2}. This attribute specifies how a previously executed statement should be canceled. Valid values are: • Cancel=0 fetches all remaining records if the statement was a Select. • Cancel=1 cancels the statement by calling dbcancel. Set Cancel=1 if dbcancel is supported in your client/server configuration. This is the initial default. • Cancel=2 closes the connection to the server for the statement. Set Cancel=2 only if dbcancel is not supported for your configuration and the performance of fetching all remaining records is unacceptable.

Table 17-1. SQL Server Connection String Attributes (cont.)

Attribute	Description
Gateway (GW)	Gateway={0|1}. This attribute specifies whether you are using the Sybase Net-Gateway to access a DB2 database with this data source. Set Gateway=1 if this is the case. Otherwise, set Gateway=0 (the initial default).
TwoPhaseCommit (TPC)	TwoPhaseCommit={0|1}. This attribute lets you have two or more active statements within a transaction, using the SQL Server two-phase commit services. Set TwoPhaseCommit=1 to use two-phase commit. The active statements may deadlock if they reference the same SQL Server table. Otherwise, set TwoPhaseCommit=0 (the initial default)
Netapi (NAPI)	Netapi={0|1}. This attribute specifies whether NETAPI.DLL is available. Netapi=0, the initial default, indicates it is not available; Netapi=1 indicates it is available. If you supply a value for the WorkstationID attribute, this attribute is ignored
ModifySQL (MS)	ModifySQL={0|1}. This attribute is provided for backward compatibility. It determines whether the driver modifies SQL statements to conform to ODBC specifications or passes the SQL statement directly to SQL Server. Specify ModifySQL=1 to have the driver modify the SQL statement to conform to ODBC specifications. Specify ModifySQL=0 to have the driver understand SQL dialects found in earlier drivers. The default is 1.

Data Types

Table 17-2 shows how the SQL Server data types are mapped to the standard ODBC data types.

Table 17-2. SQL Server Data Types

SQL Server	ODBC Data Type
binary	SQL_BINARY
bit	SQL_BIT
char	SQL_CHAR
datetime	SQL_TIMESTAMP
float	SQL_FLOAT
image	SQL_LONGVARBINARY
int	SQL_INTEGER
money	SQL_DECIMAL
real	SQL_REAL
smalldatetime	SQL_TIMESTAMP
smallint	SQL_SMALLINT
smallmoney	SQL_DECIMAL
sysname	SQL_VARCHAR
text	SQL_LONGVARCHAR
timestamp	SQL_VARBINARY
tinyint	SQL_TINYINT
varbinary	SQL_VARBINARY
varchar	SQL_VARCHAR

Isolation and Lock Levels Supported

SQL Server supports isolation level 1. SQL Server supports page-level locking. See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The SQL Server driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 functions are supported:

- SQLBrowseConnect
- SQLColumnPrivileges
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedureColumns
- SQLProcedures
- SQLTablePrivileges

The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The SQL Server database system supports multiple connections. With two-phased commit, SQL Server supports multiple statements per connection. Otherwise, SQL Server supports a single statement per connection if SQL_AUTOCOMMIT is 0 and multiple statements per connection if SQL_AUTOCOMMIT is 1.

Go To

18 SQLBase Driver

The SQLBase driver supports the Gupta SQLBase database system in the Windows, Windows 95, Windows NT, and OS/2 environments. It also supports Gupta's SQLHost gateway to DB2.

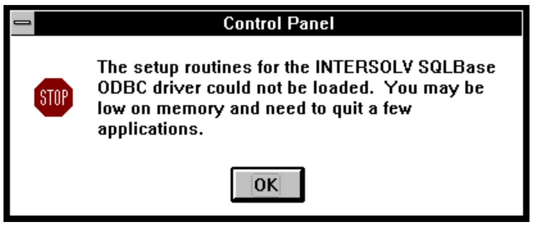
See the README file shipped with your INTERSOLV DataDirect product for the filename of the SQLBase driver.

System Requirements

Windows

The SQLBase driver requires the SQLBase SQLAPIW.DLL and one or more SQLBase communication files (such as DBWSERVR.EXE for local databases and SQLNBIOW.DLL for NetBIOS) to provide access to local and remote SQLBase databases. The directory containing these files must be on your path.

If you attempt to configure a data source and you do not have SQLAPIW.DLL on your path or in your Windows SYSTEM directory, a message similar to the following appears.

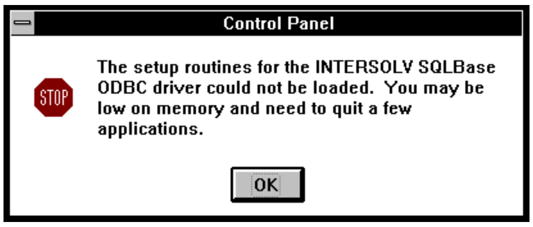


The setup routines for the INTERSOLV SQLBase ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

Windows 95 andWindows NT

The SQLBase driver requires a communication DLL (for example, SQLNPIPE.DLL for named pipes) to communicate with the server. The directory containing this file must be on your path.

If you attempt to configure a data source and you do not have SQLWNTM.DLL and SQLNGCI.DLL on your path or in your Windows 95 or Windows NT SYSTEM directory, a message similar to the following appears:



The setup routines for the INTERSOLV SQLBase ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

You must have version 5.2 or higher of SQLBase.

OS/2

The SQLBase driver requires the SQLBase client DLLs, SQLOS22.DLL and SQLGCI.DLL. It also requires one or more SQLBase communication files (such as SQLPIPE.DLL) to provide access to local and remote SQLBase databases. The directory containing these files must be on the LIBPATH specification in CONFIG.SYS.

The SQLBase driver is a 32-bit DLL and requires version 5.2 or greater of SQLBase client DLLs.

Configuring Data Sources

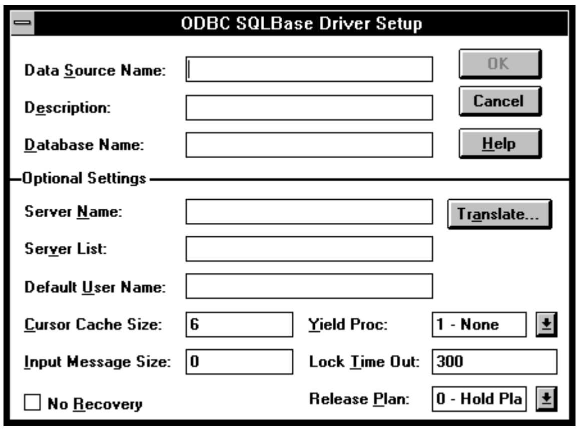
To configure a SQLBase data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the SQLBase driver and click **OK**.

  If you are configuring an existing data source, select the data source name and click **Setup**.

The setup dialog box appears.



Data Source Name:
Description:
Database Name:
Server Name:
Server List:
Default User Name:
Cursor Cache Size: 6
Yield Proc: 1 - None
Input Message Size: 0
Lock Time Out: 300
No Recovery
Release Plan: 0 - Hold Pla

3 Specify values as follows:

Data Source Name:A string that identifies this SQLBase data source configuration in ODBC.INI. Examples include “Accounting” or “SQLBase-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “SQLBase on Server number 1.”

Database Name: The name of the database to which you want to connect by default.

The following values are optional:

Server Name: The name of the server that contains the desired database. Specify the word LOCAL if you are using the local server.

The server name is not required to log on. The logon dialog box appears more quickly if you do not specify a server name. If you do specify a server name, the Database Name drop-down list in the logon dialog box is populated with the names of the databases available on that server.

Server List: A comma-separated list of servers that will appear in the Logon dialog box. Specify LOCAL to add the local server to the list.

Default User Name: The default user name used to connect to your SQLBase database. A user name is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the logon dialog box or connection string.

Cursor Cache Size: The number of cursors the cursor cache can hold. The default is 6.

Yield Proc: An integer value of 0, 1, 2, or 3 that determines whether you can work in other applications when SQLBase is busy. This attribute is useful to users of ODBC applications.

- 0 (peek and dispatch), which causes the driver to check the Windows message queue and send any messages to the appropriate Windows application.
- 1 (no yielding, the default), which does not let you work in other applications.
- 2 (SQLBase's yield procedure), which uses SQLBase's default yield procedure.
- 3 (dispatch via Windows Yield function), which turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window.

It is recommended that you use the value 1.

Note that the Yield Proc setting is the only setup option that applies to all connections to SQLBase. Therefore, the first connection will determine the Yield Proc setting for all other SQLBase connections, until all SQLBase connection handles have been freed, not just disconnected.

Input Message Size: The number of bytes in the input message buffer. The default is determined by SQLBase. Increasing this value retrieves more records across the network in a single fetch.

Lock Time Out: The number of seconds SQLBase should wait for a lock to be freed before raising an error. Values can be -1 (wait forever) to 1800; the default is 300.

No Recovery: Select this check box to disable transaction recovery. Selecting this box is dangerous because your database can become inconsistent in the event of a system crash.

Release Plan: A value of 0 or 1 that determines whether a lock is maintained on a table when the cursors accessing the table are freed. Freeing the lock on the table results in a request to the server, which can decrease performance but will always allow you to Drop or Alter the table.

- 0 - Hold Plan, the default—no locks on tables are freed.
- 1 - Free Plan—locks are freed.

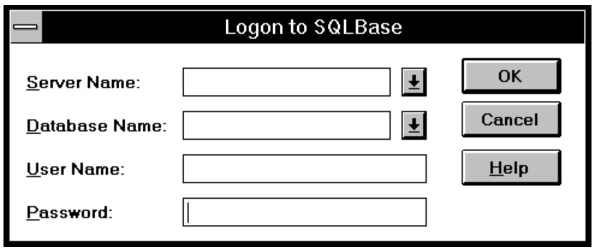
4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. For SQLBase, the dialog box is as follows:



In this dialog box, do the following:

1 Optionally, type the name of the server containing the SQLBase database tables you want to access or select the name from the Server Name drop-down list, which displays the server names you specified in the Setup dialog box. Specify LOCAL to access a local SQLBase database.

2 Type the name of the database you want to access. If you specified a server name, you can select the name from the Database Name drop-down list.

3 If required, type your user name.

4 If required, type your password.

5 Click **OK** to complete the logon and to update the values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_nam e[;attribute=valu e[;attribute=valu e].
..]
```

An example of a connection string for SQLBase is

```
DSN=SQLBASE
TABLES;SRVR=QESRVR;DB=PAYROLL;UID=JOHN;PWD=XYZZ      Y
```

Table 18-1 gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 18-1. SQLBase Connection String Attributes

Attribute	**Description**
DataSourceName (DSN)	A string that identifies a SQLBase data source configuration in ODBC.INI. Examples include “Accounting” or “SQLBase-Serv1.”
Database (DB)	The name of the database to which you want to connect
LogonID (UID)	The default logon ID (user name) used to connect to your SQLBase database. If so, contact your system administrator to get your logon ID. A logon ID is required only if security is enabled on your database.
ServerName (SRVR)	The name of the server containing the SQLBase database tables you want to access. Specify ServerName=LOCAL if you are using the local server.
Servers (SRVRLIST)	A comma-separated list of servers with which to prompt the user in a Logon dialog box. Specify LOCAL to add the local server to the list.
Password (PWD)	A case-sensitive password.

Table 18-1. SQLBase Connection String Attributes (cont.)

Attribute	**Description**
CursorCacheSize (CCS)	The number of cursors the cursor cache can hold. The cursor cache increases performance and uses few database resources. The initial default is 6.
InputMessageSize (IMS)	An integer value that determines the number of bytes of the input message buffer. Increasing this value retrieves more records across the network in a single fetch. The initial default is determined by SQLBase.
LockTimeOut (LTO)	The number of seconds SQLBase should wait for a lock to be freed before raising an error. Values can be -1 to 1800; -1 means wait forever. The initial default is 300.
DB2IsolationLevel (DIL)	DB2IsolationLevel={CS|RR}. This attribute determines the DB2 isolation level SQLHost is using, provided for use with SQLHost connections. Set DB2IsolationLevel=CS when DB2 is using the Cursor Stability isolation level; DB2IsolationLevel=RR (the initial default) when DB2 is using Repeatable Read.
ReleasePlan (RP)	ReleasePlan={0|1}. This attribute determines whether a lock is maintained on a table when the cursors accessing the table are freed. Freeing the lock on the table results in a request to the server, which can decrease performance but will always allow you to Drop or Alter the table. When set to 0, the initial default, no locks on tables are freed. When set to 1, locks are freed.

Table 18-1. SQLBase Connection String Attributes (cont.)

Attribute	Description
YieldProc (YLD)	YieldProc={0|1|2|3}. This attribute determines whether you can work in other applications when SQLBase is busy. This attribute is useful to users of ODBC applications • YieldProc=0 (peek and dispatch) causes the driver to check the Windows message queue and send any messages to the appropriate Windows application • YieldProc=1 (no yielding, the initial default) does not let you work in other applications. • YieldProc=2 (SQLBase's yield procedure) uses SQLBase's default yield procedure. • YieldProc=3 (dispatch via Windows Yield function) turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window. It is recommended that you use YieldProc=1. YieldProc=0, YieldProc=2, and YieldProc=3 do not work with all Windows applications. Note that the YieldProc attribute is the only SQLBase connection string attribute that applies to all connections to SQLBase. Therefore, the first connection will determine the YieldProc setting for all other SQLBase connections, until all SQLBase connection handles have been freed, not just disconnected.
NoRecovery (NR)	NoRecover={0|1}. This attribute enables or disables transaction recovery. NoRecovery=0 (the initial default) enables recovery; NoRecovery=1 disables recovery. NoRecovery=1 improves performance but is dangerous because your database can become inconsistent in the event of a system crash. See your SQLBase documentation for information on this option.

Data Types

Table 18-2 shows how the SQLBase data types are mapped to the standard ODBC data types.

Table 18-2. SQLBase Data Types

SQLBase	ODBC
Char	SQL_VARCHAR
Date	SQL_DATE
Decimal	SQL_DECIMAL
Double Precision	SQL_DOUBLE
Integer	SQL_INTEGER
Long Varchar	SQL_LONGVARCHAR
Number	SQL_DOUBLE
Real	SQL_REAL
Smallint	SQL_SMALLINT
Time	SQL_TIME
Timestamp	SQL_TIMESTAMP
Varchar	SQL_VARCHAR

Note: The SQLBase Real data type is replaced by Float(21) when using DB2 via the SQLHost gateway. The Graphic, Vargraphic, and Long Vargraphic DB2 data types are not supported.

Isolation and Lock Levels Supported

SQLBase supports isolation levels 1 (read committed, the default) and 3 (serializable). SQLBase also supports an alternative isolation level 1 called cursor stability. Your ODBC application can use this isolation level by calling SQLSetConnectOption (1040,1).

SQLBase supports page-level locking.

See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The SQLBase driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 functions are supported:

- SQLBrowseConnect
- SQLColumnPrivileges
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedures
- SQLSetPos
- SQLTablePrivileges

The driver also supports backward and random fetching in SQLExtendedFetch. It supports the core SQL grammar.

Number of Connections and Statements Supported

The SQLBase database system supports multiple connections and multiple statements per connection.

Go To

19 Sybase System 10 Driver

The Sybase System 10 driver supports the SQL Server System 10 database system from Sybase in the Windows, Windows 95, Windows NT, OS/2, Macintosh, and UNIX environments.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the Sybase System 10 driver.

System Requirements

You must install the Sybase Open Client-Library and the appropriate Sybase Net-Library to gain access to System 10.

Windows

WDBPING is a Windows tool that is provided with net-libraries to test connectivity from your client workstation to the database server. Use this tool to test your connection.

Windows 95 and Windows NT

SYBPING is a tool that is provided to test connectivity from your client workstation to the database server (servers that are added through SQLEdit). Use this tool to test your connection.

SQLEdit is a tool that allows you to define servers and adds them to SQL.INI.

Set the environment variable SYBASE to the directory where you installed the System 10 client. For example, set SYBASE=C:\SQL10. You set this environment variable in the Control Panel under System.

OS/2

SYBPING is a tool that is provided to test connectivity from your client workstation to the database server (servers that are added through SQLEdit). Use this tool to test your connection.

SQLEdit is a tool that allows you to define servers and adds them to SQL.INI.

Set the environment variable SYBASE to the directory where you installed the System 10 client. For example, set SYBASE=C:\SQL10. You set this environment variable in the config.sys file.

UNIX

Before you can use the System 10 data source, you must have the Sybase Open Client Net-Libraries you plan to use installed on your workstation in the $SYBASE source tree.

Set the environment variable SYBASE to the directory where you installed the System 10 client. For example, for C-shell users, the following syntax is valid:

```
setenv SYBASE /databases/sybas   e
```

For Bourne- or Korn-shell users, the following syntax is valid:

```
SYBASE=/databases/sybase;export SYBAS    E
```

You must include the directory containing the System 10 client-shared libraries in the environment variable LD_LIBRARY_PATH (on Solaris), LIBPATH (on AIX), and SHLIB_PATH (on HP-UX). For example, for C-shell users, the following syntax is valid:

```
setenv LD_LIBRARY_PATH /databases/sybase/
lib:$LD_LIBRARY_PAT  H
```

For Bourne- or Korn-shell users, the following syntax is valid:

```
LD_LIBRARY_PATH=/databases/sybase/
lib:$LD_LIBRARY_PATH;export LD_LIBRARY_PAT    H
```

Building the Required Sybase Open Client Net-Libraries

The Sybase System 10 driver requires a one-time site linking to build the Sybase Open Client Net-Libraries for AIX only. On Solaris and HP-UX, these libraries are provided from Sybase.

Before you build the Sybase Open Client Net-Libraries, install Sybase System 10 and set the environment variable SYBASE to the directory where you installed Sybase System 10.

A make file (qesyb10.mk) is provided that builds the Sybase Open Client Net-Libraries. This make file is in the bin directory. The Sybase Open Client Net-Libraries should be either built in or copied to the directory in which the other INTERSOLV ODBC DLLs are installed, usually qelib/dlls or odbc/dlls.

The following example builds the Sybase Open Client-Net Libraries if you are in the qelib/bin or odbc/bin directory**:**

```
%make -f qesyb10.mk EXE=../dll   s
```

The EXE= directive builds the Sybase Open Client Net-Libraries in the location you specify.

Macintosh

System requirements are as follows:

- 8 MB of memory
- MacTCP version 1.1 or greater

Double-click the Sybase Config Control Panel and select your interface file. See your System 10 documentation for more information.

You can use SybPing to test the connection to the database server.

Configuring Data Sources

UNIX

Note: In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the .odbc.ini file using the attributes in Table 19-1 on page 265. For information about this file, see Appendix E, “ODBC.INI,” on page 346.

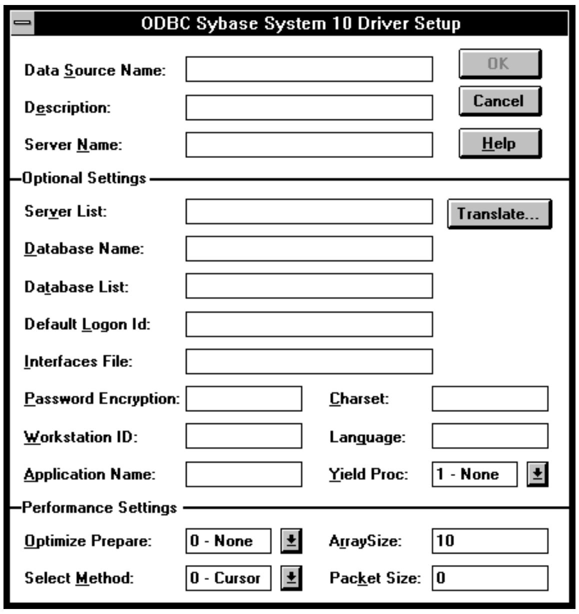
To configure a System 10 data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Sybase System 10 driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup** (if you are using Apple’s ODBC Driver Manager on the Macintosh, this button is labeled **Modify**).

The setup dialog box appears.



Data Source Name:
Description:
Server Name:
Optional Settings
Server List:
Database Name:
Database List:
Default Logon Id:
Interfaces File:
Password Encryption:
Charset:
Workstation ID:
Language:
Application Name:
Yield Proc: 1 - None
Performance Settings
Optimize Prepare: 0 - None
ArraySize: 10
Select Method: 0 - Cursor
Packet Size: 0

3 Specify values as follows:

Data Source Name:A string that identifies this Sybase System 10 data source configuration in ODBC.INI. Examples include “Accounting” or “Sys10-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Database” or “System 10 on Server number 1.”

Server Name: The name of the server that contains the System 10 tables you want to access. If not supplied, the server name in the DSQUERY environment variable is used. On UNIX, the name of a server from your $SYBASE/interfaces file.

The following values are optional:

Server List: The list of servers that appear in the logon dialog box. Separate the server names with commas.

Database Name: The name of the database to which you want to connect by default. If you do not specify a value, the default is the database defined by the system administrator for each user.

Database List: The databases that appear in the logon dialog box. Separate the names with commas.

Default Logon ID: The default logon ID used to connect to your System 10 database. This ID is case-sensitive. A logon ID is required only if security is enabled for the database you are connecting to. Your ODBC application may override this value or you may override this value in the logon dialog box or connection string.

Interfaces File: The pathname of the interfaces file. The default is the normal Sybase interfaces file.

Password Encryption A value of 0 or 1 that determines whether password encryption can be performed from the Open Client Library to the server. A value of 1 enables this password encryption; a value of 0, the default, does not.

Charset: The name of a character set corresponding to a subdirectory in $SYBASE/charsets. The default is the setting on the System 10 server.

Workstation ID: The workstation ID used by the client.

Language: The national language corresponding to a subdirectory in $SYBASE/locales. The default is English.

Application Name: The name used by System 10 to identify your application.

Yield Proc: An integer value of 0, 1, or 3 that determines whether you can work in other applications when Sybase System 10 is busy. This attribute is useful to users of ODBC applications.

- 0 (peek and dispatch), which causes the driver to check the Windows message queue and send any messages to the appropriate Windows application.
- 1 (no yielding, the default), which does not let you work in other applications.
- 3 (dispatch via Windows Yield function), which turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window.

It is recommended that you use the value 1.

The following are performance settings:

Optimize Prepare: A value of 0, 1, or 2 that determines whether stored procedures are created on the server for every call to SQLPrepare.

When set to 0, stored procedures are created for every call to SQLPrepare. This setting can result in bad performance.

When set to 1, the initial default, the driver creates stored procedures only if the statement contains parameters. Otherwise, the statement is cached and executed directly at SQLExecute time.

When set to 2, the driver never creates stored procedures.

Array Size: The number of rows the driver retrieves when fetching from the server. This is not the number of rows given to the user. The default is 10 rows.

Select Method: A value of 0 or 1 that determines whether database cursors are used for Select statements. When set to 0, the default, database cursors are used; when set to 1, Select statements are executed directly without using database cursors. A setting of 1 limits the data source to one active statement and one active connection.

Packet Size: A value of -1, 0, or *x* that determines the number of bytes per network packet transferred from the database server to the client. The correct setting of this attribute can improve performance.

When set to 0, the default, the driver uses the default packet size as specified in the System 10 server configuration.

When set to -1, the driver computes the maximum allowable packet size on the first connect to the data source and saves the value in the odbc.ini file.

When set to *x*, an integer from 1 to 10, which indicates a multiple of 512 bytes (for example, 6 means to set the packet size to 6 * 512 = 3072 bytes).

For you to take advantage of this connection attribute, you must configure the System 10 server for a maximum network packet size greater than or equal to the value you specified for PacketSize. For example,

```
sp_configure “maximum network packet size”, 5120
reconfigure
Restart System 10 Server
```

Note that the ODBC specification identifies a connect option, SQL_PACKET_SIZE, that offers this same functionality. To avoid conflicts with applications that may set both the connection string attribute and the ODBC connect option, they have been defined as mutually exclusive. If PacketSize is specified, you will receive a message “Driver Not Capable” if you attempt to call SQL_PACKET_SIZE. If you do not set PacketSize, then application calls to SQL_PACKET_SIZE are accepted by the driver.

4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

UNIX

Note: In the UNIX environment, there is no graphical interface. To perform a translation, you must edit the .odbc.ini file and add to the data source section the TranslationDLL keyword and optionally the TranslationOption keyword:

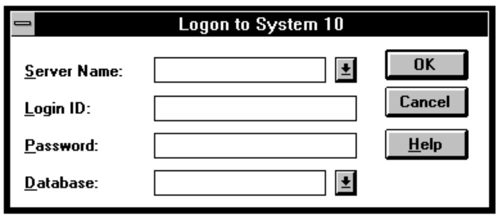
Keyword	Definition
TranslationDLL	Full path of translation DLL
TranslationOption	ASCII representation of the 32-bit integer translation option

The INTERSOLV OEM ANSI translator is shipped on UNIX to provide an example of how to create a translation DLL.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a Logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For System 10, the dialog box is as follows:



In this dialog box, do the following:

1 Type the case-sensitive name of the server containing the System 10 database tables you want to access or select the name from the Server Name drop-down list, which displays the server names you specified in the setup dialog box.

2 If required, type your case-sensitive login ID.

3 If required, type your case-sensitive password for the system.

4 Type the name of the database you want to access (case-sensitive) or select the name from the Database drop-down list, which displays the names you specified in the setup dialog box.

5 Click **OK** to complete the logon and to update the values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for Sybase System 10 is

```
DSN=SYS10 TABLES;SRVR=QESRVR;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

Table 19-1 gives the long and short names for each attribute, as well as a description. To configure a data source in the UNIX environment, you must edit the .odbc.ini file. This file accepts only long names for attributes.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 19-1. Sybase System 10 Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies a single connection to a System 10 database. Examples include “Accounting” or “Sys10-Serv1.”
ServerName (SRVR)	The name of the server containing the System 10 tables you want to access. If not supplied, the initial default is the server name in the DSQUERY environment variable On UNIX, the name of a server from your $SYBASE/interfaces file.
LogonID (UID)	The default logon ID used to connect to your System 10 database. This ID is case-sensitive. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.
Password (PWD)	A case-sensitive password.
Database (DB)	The name of the database to which you want to connect
Language (LANG)	The national language corresponding to a subdirectory in $SYBASE/locales.
Charset (CS)	The name of a character set corresponding to a subdirectory in $SYBASE/charsets.
WorkstationID (WKID)	The workstation ID used by the client
ApplicationName (APP)	The name used by System 10 to identify your application
InterfacesFile (IFILE)	The pathname to the interfaces file.

Table 19-1. Sybase System 10 Connection String Attributes (cont.)

Attribute	Description
OptimizePrepare (OP)	OptimizePrepare={0|1|2}. This attribute determines whether stored procedures are created on the server for every call to SQLPrepare. When set to 0, stored procedures are created for every call to SQLPrepare. This setting can result in bad performance When set to 1, the initial default, the driver creates stored procedures only if the statement contains parameters. Otherwise, the statement is cached and executed directly at SQLExecute time. When set to 2, the driver never creates stored procedures
SelectMethod (SM)	SelectMethod={0|1}. This attribute determines whether database cursors are used for Select statements. When set to 0, the initial default, database cursors are used. In some cases performance degradation can occur when performing large numbers of sequential Select statements because of the amount of overhead associated with creating database cursors. When set to 1, Select statements are executed directly without using database cursors. When set to 1, the data source is limited to one active statement and one active connection.
ArraySize (AS)	The number of rows the driver retrieves when fetching from the server. This is not the number of rows given to the user. This increases performance by reducing network traffic. The initial default is 10 rows.

Table 19-1. Sybase System 10 Connection String Attributes (cont.)

Attribute	**Description**
YieldProc (YLD)	YieldProc={0|1|3}. This attribute determines whether you can work in other applications when Sybase System 10 is busy. This attribute is useful to users of ODBC applications • YieldProc=0 (peek and dispatch) causes the driver to check the Windows message queue and to send any messages to the appropriate Windows application • YieldProc=1 (no yielding, the initial default) does not let you work in other applications. • YieldProc=3 (dispatch via Windows Yield function) turns control over to the Windows kernel. The Windows kernel checks the message queue and sends any messages to the appropriate application window. It is recommended that you use YieldProc=1. YieldProc=0 and YieldProc=3 do not work with all Windows applications
PasswordEncryption (PE)	PasswordEncryption={0|1}. This attribute determines whether password encryption can be performed from the Open Client Library to the server (PasswordEncryption=1). When set to 0, the initial default, this cannot be done

Table 19-1. Sybase System 10 Connection String Attributes (cont.)

Attribute	Description
PacketSize (PS)	PacketSize={-1|0|*x*}. This attribute determines the number of bytes per network packet transferred from the database server to the client. The correct setting of this attribute can improve performance. When set to 0, the initial default, the driver uses the default packet size as specified in the System 10 server configuration. When set to -1, the driver computes the maximum allowable packet size on the first connect to the data source and saves the value in the odbc.ini file. When set to *x*, an integer from 1 to 10, which indicates a multiple of 512 bytes (for example, PacketSize=6 means to set the packet size to 6 * 512 = 3072 bytes). For you to take advantage of this connection attribute, you must configure the System 10 server for a maximum network packet size greater than or equal to the value you specified for PacketSize. For example, sp_configure "maximum network packet size", 5120 reconfigure Restart System 10 Server Note that the ODBC specification specifies a connect option, SQL_PACKET_SIZE, that offers this same functionality. To avoid conflicts with applications that may set both the connection string attribute and the ODBC connect option, they have been defined as mutually exclusive. If PacketSize is specified, you will receive a message "Driver Not Capable" if you attempt to call SQL_PACKET_SIZE. If you do not set PacketSize, then application calls to SQL_PACKET_SIZE are accepted by the driver.

Data Types

Table 19-2 shows how the System 10 data types are mapped to the standard ODBC data types.

Table 19-2. Sybase System 10 Data Types

System 10	ODBC
binary	SQL_BINARY
bit	SQL_BIT
char	SQL_CHAR
datetime	SQL_TIMESTAMP
decimal	SQL_DECIMAL
float	SQL_FLOAT
image	SQL_LONGVARBINARY
int	SQL_INTEGER
money	SQL_DECIMAL
numeric	SQL_NUMERIC
real	SQL_REAL
smalldatetime	SQL_TIMESTAMP
smallint	SQL_SMALLINT
smallmoney	SQL_DECIMAL
sysname	SQL_VARCHAR
text	SQL_LONGVARCHAR
timestamp	SQL_VARBINARY
tinyint	SQL_TINYINT
varbinary	SQL_VARBINARY
varchar	SQL_VARCHAR

Isolation and Lock Levels Supported

System 10 supports isolation levels 1 (read committed, the default) and 3 (serializable). It supports page-level locking. See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The Sybase System 10 driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 functions are supported:

- SQLBrowseConnect
- SQLColumnPrivileges
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedureColumns
- SQLProcedures
- SQLTablePrivileges

The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The System 10 database system supports multiple connections and multiple statements per connection.

Go To

20 Text Driver

The Text driver supports ASCII text files in the Windows, Windows 95, Windows NT, OS/2, Macintosh, and UNIX environments. These files can be printed directly or edited with text editors or word processors, because none of the data is stored in a binary format.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the text driver.

The Text driver executes SQL statements directly on the text files. The driver supports Insert statements, and inserts the record at the end of the file. However, you may not execute Update or Delete statements.

System Requirements

Windows

Windows users who are accessing the same text file must be running SHARE.EXE. Otherwise, new records can be overwritten if more than one user inserts a new record at the same time.

Macintosh

Macintosh users who are accessing the same text file must have file sharing enabled.

Formats for Text Files

Some common formats for text files are listed in Table 20-1.

Table 20-1. Common Text File Formats

Format	Description
Comma-separated values	Commas separate column values, and each line is a separate record. Column values can vary in length. These files often have the .CSV extension.
Tab-separated values	Tabs separate column values, and each line is a separate record. Column values can vary in length
Character-separated values	Any printable character except single or double quotation marks can separate column values, and each line is a separate record. Column values can vary in length.
Fixed	No character separates column values. Instead, values start at the same position and have the same length in each line. The values appear in fixed columns if you display the file. Each line is a separate record.

Comma-, tab-, and character-separated files are called character-delimited files, because values are separated by a special character.

Configuring Data Sources

UNIX

Note: In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the .odbc.ini file using the attributes in Table 20-4 on page 285. For information about this file, see Appendix E, “ODBC.INI,” on page 346.

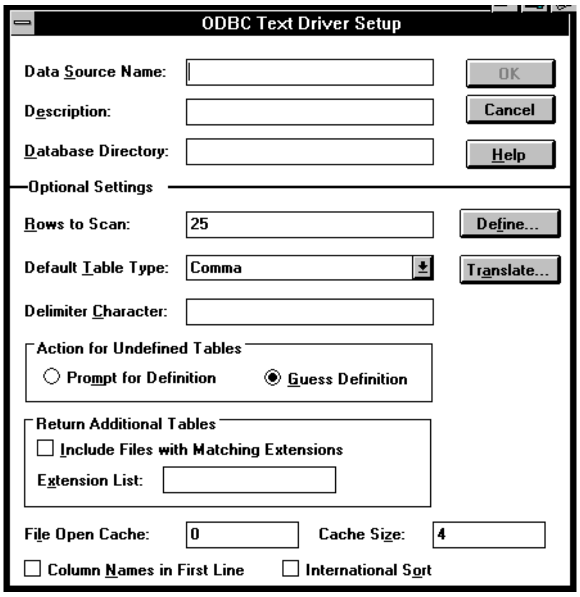
To configure a Text data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Text driver and click **OK**.

If you are configuring an existing data source, select the data source name and click **Setup** (if you are using Apple's ODBC Driver Manager on the Macintosh, this button is labeled **Modify**).

The setup dialog box appears.



Data Source Name:
Description:
Database Directory:
Optional Settings
Rows to Scan: 25
Default Table Type: Comma
Delimiter Character:
Action for Undefined Tables
○ Prompt for Definition ● Guess Definition
Return Additional Tables
☐ Include Files with Matching Extensions
Extension List:
File Open Cache: 0 Cache Size: 4
☐ Column Names in First Line ☐ International Sort

3 Specify values as follows:

Data Source Name:A string that identifies this Text data source configuration in ODBC.INI. Examples include "Accounting" or "Text Files."

Description: An optional long description of a data source name. For example, “My Accounting Files” or “My Text Files in the Accounting Directory.”

Database Directory: The directory in which the text files are stored. If none is specified, the current working directory is used.

The following values are optional:

Rows to Scan: The number of rows in a text file that the driver scans to determine the data types in the file. If the value is 0, all rows in the file are scanned. The default is 25.

Default Table Type: The type of text file: comma-separated, tab-separated, character-separated, or fixed length. This value tells the driver the default type, which is used when creating a new table and opening an undefined table.

Delimiter Character: The character used as a delimiter for character-separated files. It can be any printable character. The default is a comma (,).

Action for Undefined Tables: Two radio buttons that indicate what action the driver should take when it encounters a file that has not been defined. Set the Prompt for Definition radio button, if you want the driver to prompt the user when it encounters a file whose format is not defined. Otherwise, set the Guess Definition radio button; in this case, the driver guesses the file’s format.

Return Additional Tables: Select this check box to tell the driver to return files you have defined in functions like SQLTables, SQLColumns, etc. *and* files with a given extension. In Extension List, specify a comma-separated list of the extensions. To have files with no extensions returned, specify NONE. For example, if some of your files have the extensions TXT and CSV and others have no extension, specify TXT,CSV,NONE.

By default, when an application requests a list of tables, only files that have been defined are returned.

File Open Cache: An integer value that specifies the maximum number of unused file opens to cache. For example, the value 4 specifies that when a user opens and closes four tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is increased performance. The disadvantage is that a user who specifies file locking on open may get a locking conflict even though no one appears to have the file open. The default is 0, which means no file open caching.

Cache Size: The number of 64K blocks the driver uses to cache database records. The higher the number of blocks, the better the performance. The maximum number of blocks you can set depends on the system memory available. If the cache size is greater than 0, when browsing backwards, you will not be able to see updates made by other users until you re-execute the Select statement. The default is 4.

Column Names in First Line: Select this check box to tell the driver to look for column names in the first line of the file.

International Sort: A setting to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Select this check box to use the international sort order as defined by your operating system. International sort order is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (see your operating system documentation). Leave this box blank to use the ASCII sort order. ASCII sort order is case-sensitive, where uppercase letters precede lowercase letters (*B* precedes *a*).

Mac File Info: On the Macintosh, click this button to specify the creator and file type. A dialog box is displayed to enable you to enter four-character, case-sensitive values for the following:

- Creator (default is ttxt)
- File Type (default is TEXT)

Note: The Rows to Scan, Default Table Type, Delimiter Character, and Column Names in First Line settings apply only to tables *not* previously defined. These fields also determine the attributes of new tables created with the Create Table statement.

Use Long Qualifiers Set this check box to use long pathnames as table qualifiers. When you set this check box, pathnames can be up to 255 characters. The default length for pathnames is 128 characters.

4 Click **Define** in this dialog box to define the structure of your text files. The following section discusses how to define the column names and data types in a table.

5 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

Note: In the UNIX environment, there is no graphical interface. To perform a translation, you must edit the .odbc.ini file and add to the data source section the Translation keyword and optionally the TranslationOption keyword:

Keyword	**Definition**
TranslationDLL	Full path of translation DLL
TranslationOption	ASCII representation of the 32-bit integer translation option

The INTERSOLV OEM ANSI translator is shipped on UNIX to provide an example of how to create a translation DLL.

6 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the

defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

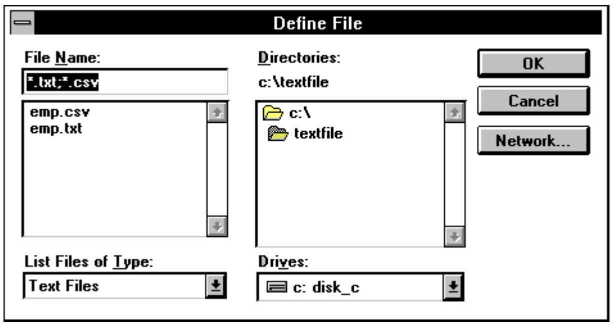
Defining Table Structure

Note that this section does not apply to the UNIX platforms. See “Defining Table Structure on UNIX Platforms” on page 280 for how to define table structure on the UNIX platforms.

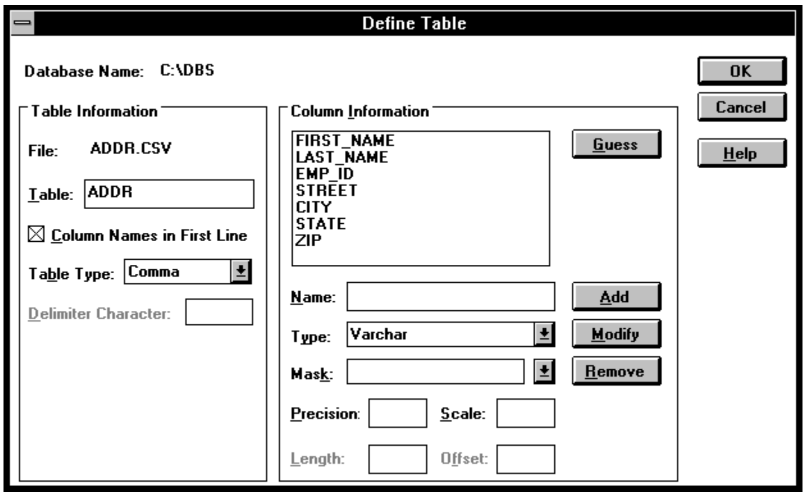
Because text files do not all have the same structure, the driver provides the option of defining the structure of an existing file. Although defining the structure is not mandatory, because the driver can attempt to guess the names and types of the columns, this feature is extremely useful.

Define the structure of a file as follows:

1 Click **Define** in the ODBC Text Driver Setup dialog box, which you can access through the ODBC Administrator. The standard file open dialog box for your system appears. The following figure shows the Windows Define File dialog box.


Define File
File Name: *.txt;*.csv
emp.csv
emp.txt
Directories: c:\textfile
c:\
textfile
List Files of Type: Text Files
Drives: c: disk_c

2 Select the correct file and click **OK** to define the file’s structure using the Define Table dialog box.


Define Table

Database Name: C:\DBS

Table Information

File: ADDR.CSV

Table: ADDR

☒ Column Names in First Line

Table Type: Comma

Delimiter Character:

Column Information

FIRST_NAME
LAST_NAME
EMP_ID
STREET
CITY
STATE
ZIP

Name:

Type: Varchar

Mask:

Precision: Scale:

Length: Offset:

In this dialog box, Database Name and File display the name of the database directory that contains the file and the name of the file you previously selected, respectively.

3 In the Table Information section of this dialog box, specify information about the overall structure of the file:

a In the Table box, type a table name. Values may be up to 32 characters in length and may not be the same as another defined table in the database. This name is returned by SQLTables. By default, it is the filename without its extension.

b Select the Column Names in First Line check box if the first line of the file contains column names. Otherwise, do not select this check box.

c Open the Table Type box and select a table type: comma, tab, fixed, or character.

d If the table type is character-separated, type the character that separates the values in the Delimiter Character box.

4 In the Column Information section, define the types and names of the columns in the table. The box in the upper-left corner of this section lists the defined columns.

If you specified a comma-separated, tab-separated, or character-separated table type, the **Guess** button appears in this section. Click **Guess** to tell the driver to guess the fields. The driver then displays what it thinks the fields are. You can then modify the field definitions by specifying values in the Name, Type, Mask, Precision, and Scale boxes (as appropriate) and clicking **Modify**. If you do not want the driver to guess the fields, take the following steps to define the fields:

a In the Name box, type the name of the field.

b Open the Type drop-down list and select the data type of the field. If the field type is date, then you must select a date mask for the field or type one in. See the following section, “Date Masks,” for more information.

c In the Precision box, type the precision of the field.

d In the Scale box, type the scale of the field.

The precision and scale values determine how numeric data is to be returned.

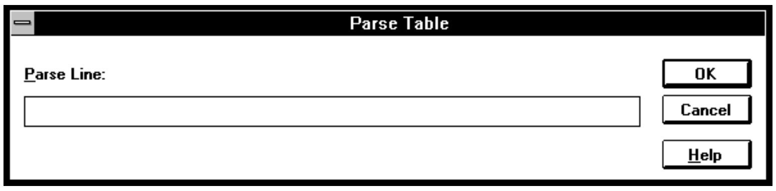
e If you specified a fixed-length table type, you may specify the length and offset in the Length and Offset boxes, or you can specify a parse string. The length is the number of bytes the data takes up in storage; the offset is the number of bytes from the start of the table to the start of the field.

f Click **Add** to add this field definition to the list box.

To modify the selected field in the list box, click **Modify**.

To remove the selected field in the list box, click **Remove**.

If you specified a fixed-length table type, the **Parse** button is displayed. If you have not entered values in the length and offset boxes, click **Parse** to define the columns of the table. The Parse Table dialog box appears.



This dialog box displays the first line of the file. You must mark where each field begins and ends by enclosing it in brackets. These brackets indicate the position and length of each field value in the record. You must do this for all the fields in the table. Click **OK** when you are done.

5 Click **OK** to define the table.

Defining Table Structure on UNIX Platforms

Because text files do not all have the same structure, the driver provides the option of defining the structure of an existing file. Although defining the structure is not mandatory, because the driver can attempt to guess the names and types of the columns, this feature is extremely useful.

To define the structure of a text file, you create a QETXT.INI file using any plain text editor, such as vi. The filename must be in uppercase. All of the tables you wish to define are specified in the QETXT.INI file. When you specify table attributes in QETXT.INI, you override the attributes specified in .odbc.ini or in the connection string.

Define the QETXT.INI file as follows:

1 Create a [Defined Tables] section and list all of the tables you are defining. Specify the text filename (in either upper- or lowercase, depending on the file) followed by the name you want to give the table, for example:

```
emptxt.txt=EMP
```

Table names can be up to 32 characters in length and cannot be the same as another defined table in the database. This name is returned by SQLTables. By default, it is the filename without its extension.

2 For each table listed in the [Defined Tables] section, you must specify the text file (FILE=), the table type (TT=), whether the first line of the file contains column names (FLN=), and the delimiter character (DC=).

Specify the text filename. For example:

```
FILE=emptxt.txt
```

To define the table type, specify how the fields are separated (comma, tab, fixed, or character). For example:

```
TT=COMMA
```

If the table type is CHARACTER, specify the delimiter character. For example, if the fields are separated by semicolons,

```
DC=;
```

Specify whether the first line of the file contains column names, using 1 for yes and 0 for no. For example:

```
FLN=0
```

3 Define the fields in the table, beginning with FIELD1. For each field, specify the field name, field type, precision, scale, length, offset (for fixed tables), and date/time mask. See the DATE MASKS section for information about masks.

Separate the values with commas. For example, to define 2 fields,

```
FIELD1=EMP_ID,VARCHAR,6,0,6,0  ,
FIELD2=HIRE_DATE,DATE,10,0,10,0,m/d/y   y
```

4 Save the file as QETXT.INI. The driver looks for this file in the directory specified by the “Database” attribute in odbc.ini, or in the current directory.

Example of QETXT.IN

The following is an example of a QETXT.INI file. This file defines the structure of the emptext.txt file, which is a sample data file shipped with the DataDirect ODBC Text file.

```
[Defined Tables ]
emptext.txt=EM P

[EMP ]
FILE=emptext.tx t
FLN= 1
TT=Comm a
Charset=ANS I
FIELD1=FIRST_NAME,VARCHAR,10,0,10,0   ,
FIELD2=LAST_NAME,VARCHAR,9,0,9,0   ,
FIELD3=EMP_ID,VARCHAR,6,0,6,0   ,
FIELD4=HIRE_DATE,DATE,10,0,10,0,m/d/y    y
FIELD5=SALARY,NUMERIC,8,2,8,0   ,
FIELD6=DEPT,VARCHAR,4,0,4,0   ,
FIELD7=EXEMPT,VARCHAR,6,0,6,0   ,
FIELD8=INTERESTS,VARCHAR,136,0,136,0    ,
```

Date Masks

Date masks tell the driver how a date is stored in a text file. When a value is inserted into a text file, the date is formatted so that it matches the mask. When reading a text file, the driver converts the formatted date into a date data type.

Table 20-2 lists the symbols to use when specifying the date mask.

Table 20-2. Date Masks for Text Driver

Symbol	Description
m	Output the month's number (1-12).
mm	Output a leading zero if the month number is less than 10.
mmm, Mmm, MMM	Output the three-letter abbreviation for the month depending on the case of the M's (that is, jan, Jan, JAN).
mmmm, Mmmm, MMMM	Output the full month name depending on the case of the M's (that is, january, January, JANUARY).
d	Output the day number (1-31).
dd	Output a leading zero if the day number is less than 10.
ddd, Ddd, DDD	Output the three-letter day abbreviation depending on the case of the D's (that is, mon, Mon, MON).
dddd, Dddd, DDDD	Output the day depending on the case of the D's (that is, monday, Monday, MONDAY).
yy	Output the last two digits of the year.
yyyy	Output the full four digits of the year.
J	Output the Julian value for the date. The Julian value is the number of days since 4712 BC.
\ - . : , (space)	Special characters used to separate the parts of a date.
\	Output the next character. For example, if the mask is mm/dd/yyyy \A\D, the value appears as 10/01/1993 AD in the text file.
"string", 'string'	Output the string in the text file.

Table 20-3 shows some example date values, masks, and how the date appears in the text file.

Table 20-3. Date Mask Examples

Date	Mask	Value
1993-10-01	yyyy-mm-dd	1993-10-01
	m/d/yy	10/1/93
	Ddd, Mmm dd, yyyy	Fri, Oct 01, 1993

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for text files is

```
DSN=TEXT FILES;TT=CHARACTER;DC=&
```

Table 20-4 gives the long and short names for each attribute, as well as a description. To configure a data source in the UNIX environment, you must edit the .odbc.ini file. This file accepts only long names for attributes.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in ODBC.INI. If you specified a value for the attribute when configuring the data source, that value is your default.

Table 20-4. Text Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies a Text data source configuration in ODBC.INI. Examples include “Accounting” or “Text Files.”
Database (DB)	The directory in which the text files are stored.
ScanRows (SR)	The number of rows in a text file that the driver scans to determine the column types in the file. If the value is 0, all rows in the file are scanned. The initial default is 25.
TableType (TT)	TableType={Comma|Tabl|Character|Fixed}. The Text driver supports four types: comma-separated, tab-separated, character-separated, and fixed length. Setting this value tells the driver the default type, which is used when creating a new table and opening an undefined table.
Delimiter (DC)	The character used as a delimiter for character-separated files. It can be any printable character except single or double quotes. The initial default is a comma (,).
UndefinedTable (UT)	The Text driver can perform two operations when it encounters a file that has not been defined. UndefinedTable=PROMPT tells the driver to display a dialog box that allows the user to describe the file’s format. UndefinedTable=GUESS tells the driver to guess the file’s format. This is the initial default.
ExtraExtensions (EE)	A list of additional filename extensions to be recognized as text tables. When an application requests a list of tables, only files that have been defined are returned. To have the driver also return names of undefined files, specify a comma-separated list of file extensions. To specify files with no extension, use the keyword NONE.

Table 20-4. Text Connection String Attributes (cont.)

Attribute	Description
FileOpenCache (FOC)	The maximum number of unused file opens to cache. For example, when FileOpenCache=4, and a user opens and closes four files, the files are not actually closed. The driver keeps them open so that if another query uses one of these files, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The disadvantage is that a user who tries to open the file exclusively may get a locking conflict even though no one appears to have the file open. The initial default is 0.
CacheSize (CSZ)	The number of 64K blocks the driver uses to cache database records. The higher the number of blocks, the better the performance. The maximum number of blocks you can set depends on the system memory available. If the cache size is greater than 0, when browsing backwards, you will not be able to see updates made by other users until you re-execute the Select statement. The initial default is 4.
FirstLineNames (FLN)	FirstLineNames={0|1}. This attribute determines whether the driver looks for column names in the first line of the file. If FirstLineNames=1, the driver looks for column names in the first line of the file. If FirstLineNames=0 (the initial default), the first line is interpreted as the first record in the file.
IntlSort (IS)	IntlSort-{0|1}. This attribute determines the order that records are retrieved when you issue a Select statement with an Order By clause. If IntlSort=1, the driver uses the international sort order as defined by your operating system. The sort is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (see your operating system documentation). If IntlSort=0 (the initial default), the driver uses the ASCII sort order, where uppercase letters precede lowercase letters (*B* precedes *a*).

Table 20-4. Text Connection String Attributes (cont.)

Attribute	Description
MacFileInfo (MFI)	On Macintosh systems, four-character, case-sensitive values that specify the following in the order shown: • Creator (default is ttxt) • File Type (default is TEXT) The values are specified in a comma-separated list. For example, MacFileInfo=ABCD,EFGH.
UseLongQualifiers (ULQ)	UseLongQualifiers={0|1}. It specifies whether the driver uses long pathnames as table qualifiers. The default is 0, do not use long pathnames (the default length of pathnames is 128 characters). If UseLongQualifiers=1, the driver uses long pathnames (up to 255 characters).

Note: The ScanRows, TableType, Delimiter, FirstLineNames, and Charset attributes apply to tables that have *not* been defined. These attributes also determine the characteristics of new tables created with the Create Table statement

Data Types

Table 20-5 shows how the text file data types are mapped to the standard ODBC data types.

Table 20-5. Text Data Types

Text	**ODBC**
Numeric	SQL_NUMERIC
Date	SQL_DATE
Varchar	SQL_VARCHAR

Select Statement

You use the SQL Select statement to specify the columns and records to be read. The driver supports all Select statement clauses as described in Appendix A, “SQL for Flat-File Drivers,” on page 299.

ODBC Conformance Level

The Text driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 function is supported: SQLSetPos.

The Text driver also supports backward and random fetching in SQLExtendedFetch. The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

Text files support multiple connections and multiple statements per connection.

Go To

21 XDB Driver

The XDB driver supports the XDB database system in the Windows environment.

See the README file shipped with your INTERSOLV DataDirect product for the filename of the XDB driver.

System Requirements

The XDB driver requires that you have

- XDB version 2.41 or higher.
- A protocol supported by XDB (such as NetBIOS) if you are using the client-server version of XDB. If you are using the local version of XDB, a protocol is not required.
- XDB client software to access the server.

To have access to network DLLs, you must run the CLIENT.BAT file supplied with the XDB server. This file must be run on each client.

Modify the [XDB] section of your WIN.INI file. For example, if XDB is installed on drive C, modify WIN.INI as follows:

```
xdbcfg=C:\XD B
xsrvcfg=C:\XD B
```

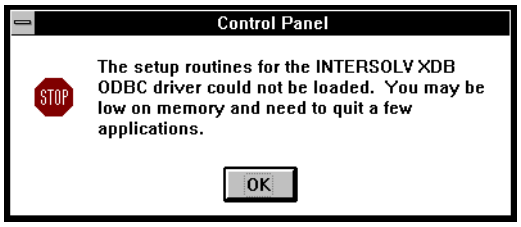
These settings must also be defined as environment variables in your AUTOEXEC.BAT file. For example:

```
SET xdbcfg=C:\XD B
SET xsrvcfg=C:\XD B
```

If you are using the local version of XDB, you must also define XDBSERVE as an environment variable:

```
SET xdbserve=loca  l
```

If you attempt to configure a data source and do not have XUTILW.DLL, MEMRESW.DLL, MEMR30WD.DLL, and WINROUTE.DLL on your path or in your Windows SYSTEM directory, a message similar to the following appears.



The setup routines for the INTERSOLV XDB ODBC driver could not be loaded. You may be low on memory and need to quit a few applications.

When setting up your User ID in XDB user administration, the user type must be set to install user.

Configuring Data Sources

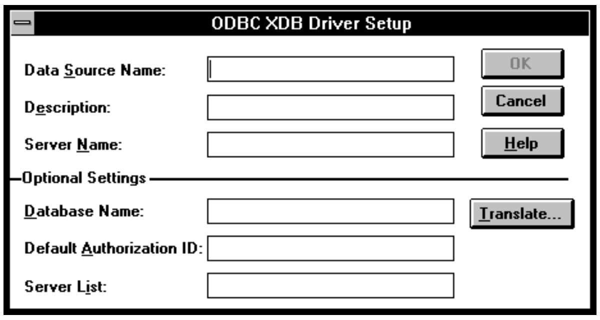
To configure an XDB data source:

1 Start the ODBC Administrator. A list of data sources appears.

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the XDB driver and click **OK**.

   If you are configuring an existing data source, select the data source name and click **Setup**.

   The setup dialog box appears.



3 Specify values as follows:

Data Source Name:A string that identifies this XDB data source configuration in ODBC.INI. Examples include “Accounting” or “XDB-Serv1.”

Description: An optional long description of a data source name. For example, “My Accounting Files” or “XDB Database on Server number 1.”

Server Name: The name of the server containing the XDB database tables you want to access. You can leave this empty if you are using the local server.

The following values are optional:

Database Name: The name of the database to which you want to connect by default.

Default Authorization ID: The default authorization ID used to connect to your XDB database. An authorization ID is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the logon dialog box or connection string.

Server List: The list of servers that appear in the logon dialog box. Separate the server names with commas.

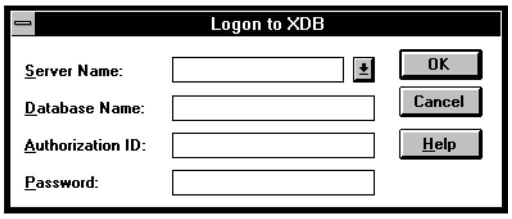
4 Click **Translate** to perform a translation of your data from one character set to another. The Select Translator dialog box appears in which you can select a translator to perform the translation. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set. Click **OK** to leave this dialog box and perform the translation.

   The translators that are listed in this dialog box are determined by the values listed in the ODBC Translators section of your odbcinst.ini file.

5 Click **OK** to write these values to ODBC.INI. These values are now the defaults when you connect to the data source. You can change the defaults by configuring your data source again. You can override the defaults by connecting to the data source using a connection string with alternate values.

Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. For XDB, the dialog box is as follows:



In this dialog box, do the following:

1 Type the server name of the computer containing the XDB database tables you want to access or select the name from the Server Name drop-down list, which displays the server names you specified in the setup dialog box. Server names contain the information necessary for an XDB

network redirector to locate the database to be used.

2 Type the name of the XDB database containing the tables you want to access.

3 If required, type your user name.

4 If required, type your password.

5 Click **OK** to log on to the XDB database on the computer you specified and to update the values in ODBC.INI.

Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which ODBC.INI section to use for the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in ODBC.INI. These values are not written to ODBC.INI.

You can specify either long or short names in the connection string. The connection string has the form

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

An example of a connection string for XDB is

```
DSN=XDB TABLES;SRVR=QESRVR;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

Table 21-1 gives the long and short names for each attribute, as well as a description. To configure a data source in the UNIX environment, you must edit the .odbc.ini file. This file accepts only long names for attributes.

Table 21-1. XDB Connection String Attributes

Attribute	Description
DataSourceName (DSN)	A string that identifies an XDB data source configuration in ODBC.INI. Examples include “Accounting” or “XDB-Serv1.”
LogonID (UID)	The default logon ID (authorization ID) used to connect to your XDB database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.
Password (PWD)	Your password.
ServerName (SRVR)	The name of the server containing the XDB database tables you want to access.
Database (DB)	The name of the database to which you want to connect

Data Types

Table 21-2 shows how the XDB data types are mapped to the standard ODBC data types.

Table 21-2. XDB Data Types

XDB	ODBC
Char	SQL_CHAR
Date	SQL_DATE
Decimal	SQL_DECIMAL

Table 21-2. XDB Data Types (cont.)

XDB	ODBC
Float	SQL_DOUBLE
Int	SQL_INTEGER
Long Varchar	SQL_LONGVARCHAR
Money	SQL_DECIMAL
Smallint	SQL_SMALLINT
Time	SQL_TIME
Timestamp	SQL_TIMESTAMP
Varchar	SQL_VARCHAR

Isolation and Lock Levels Supported

XDB supports isolation levels 0 (read uncommitted), 1 (read committed, the default), and 2 (repeatable read). XDB supports record-level locking. See Appendix D, “Locking and Isolation Levels,” on page 342 for a discussion of these topics.

ODBC Conformance Level

The XDB driver supports the Core, Level 1, and Level 2 API functions listed in Appendix C, “ODBC API and Scalar Functions,” on page 332. In addition, the following Level 2 function is supported: SQLBrowseConnect.

The driver supports the core SQL grammar.

Number of Connections and Statements Supported

The XDB database system supports a single connection and multiple statements within that connection.

Go To

A SQL for Flat-File Drivers

This appendix describes the SQL statements that you can use with the flat-file drivers (Btrieve, dBASE, Excel, Paradox, and Text). The database drivers parse SQL statements and translate them into a form that the database can understand. The SQL statements in this appendix let you

- Read, insert, update, and delete records from a database
- Create new tables
- Drop existing tables

These SQL statements allow your application to be portable across other databases.

Select Statement

The form of the Select statement supported by the flat-file drivers is

```
SELECT [DISTINCT] {* | column_expression, ...}
FROM table_names [table_alias] ...
[ WHERE expr1 rel_operator expr2 ]
[ GROUP BY {column_expression, ...} ]
[ HAVING expr1 rel_operator expr2 ]
[ UNION [ALL] (SELECT...) ]
[ ORDER BY {sort_expression [DESC | ASC]}, ... ]
[ FOR UPDATE [OF {column_expression, ...}] ]
```

Select Clause

Follow Select with a list of column expressions you want to retrieve or an asterisk (*) to retrieve all fields.

```
SELECT [DISTINCT] {* | column_expression, ...}
```

column_expression can be simply a field name (for example, LAST_NAME). More complex expressions may include mathematical operations or string manipulation (for example, SALARY * 1.05). See the section “SQL Expressions” later in this appendix.

Separate multiple column expressions with commas (for example, LAST_NAME, FIRST_NAME, HIRE_DATE).

Field names can be prefixed with the table name or alias. For example, EMP.LAST_NAME or E.LAST_NAME, where E is the alias for the table EMP.

The Distinct operator can precede the first column expression. This operator eliminates duplicate rows from the result of a query. For example:

```
SELECT DISTINCT dep FROM emp
```

Aggregate Functions

Aggregate functions can also be a part of a Select clause. Aggregate functions return a single value from a set of records. An aggregate can be used with a field name (for example, AVG(SALARY)) or in combination with a more complex column expression (for example, AVG(SALARY * 1.07)). The column expression can be preceded by the Distinct operator. The Distinct operator eliminates duplicate values from an aggregate expression. For example:

```
COUNT (DISTINCT last_name)
```

In this example, only distinct last name values are counted.

Table A-1 lists valid aggregates.

Table A-1. Aggregate Functions

Aggregate	Returns
SUM	The total of the values in a numeric field expression. For example, SUM(SALARY) returns the sum of all salary field values
AVG	The average of the values in a numeric field expression. For example, AVG(SALARY) returns the average of all salary field values.
COUNT	The number of values in any field expression. For example, COUNT(NAME) returns the number of name values. When using COUNT with a field name, COUNT returns the number of non-null field values. A special example is COUNT(*), which returns the number of records in the set, including records with null values
MAX	The maximum value in any field expression. For example, MAX(SALARY) returns the maximum salary field value
MIN	The minimum value in any field expression. For example, MIN(SALARY) returns the minimum salary field value

From Clause

The From clause indicates the tables that will be used in the Select statement. The format of the From clause is

```
FROM table_names [table_alias]
```

table_names can be one or more simple table names in the current working directory or complete pathnames.

table_alias is a name used to refer to this table in the rest of the Select statement. Database field names may be prefixed by the table alias. Given the table specification

```
FROM emp  E
```

you may refer to the LAST_NAME field as E.LAST_NAME. Table aliases must be used if the Select statement joins a table to itself. For example:

```
SELECT * FROM emp E, emp F WHERE E.mgr_id = F.emp_id
```

The equal sign (=) includes only matching rows in the results.

If you are joining more than one table, you may use LEFT OUTER JOIN, which includes nonmatching rows in the first table you name. For example:

```
SELECT * FROM T1 LEFT OUTER JOIN T2 on T1.key = T2.key
```

Where Clause

The Where clause specifies the conditions that records must meet to be retrieved. The Where clause contains conditions in the form:

WHERE *expr1 rel_operator expr2*

expr1 and *expr2* may be field names, constant values, or expressions.

rel_operator is the relational operator that links the two expressions. See the section “SQL Expressions” later in this appendix.

For example, the following Select statement retrieves the names of employees that make at least $20,000.

```
SELECT last_name,first_name FROM emp WHERE salary >=
20000
```

Group By Clause

The Group By clause specifies the names of one or more fields by which the returned values should be grouped. This clause is used to return a set of aggregate values.

It has the following form:

GROUP BY *column_expressions*

column_expressions must match the column expression used in the Select clause. A column expression can be one or more field names of the database table, separated by a comma (,) or one or more expressions, separated by a comma (,). See the section “SQL Expressions” later in this appendix.

The following example sums the salaries in each department.

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id
```

This statement returns one row for each distinct department ID. Each row contains the department ID and the sum of the salaries of the employees in the department.

Having Clause

The Having clause enables you to specify conditions for groups of records (for example, display only the departments that have salaries totaling more than $200,000). This clause is valid only if you have already defined a Group By clause.

It has the following form:

```
HAVING expr1 rel_operator expr2
```

expr1 and *expr2* can be field names, constant values, or expressions. These expressions do not have to match a column expression in the Select clause.

rel_operator is the relational operator that links the two expressions. See the section “SQL Expressions” later in this appendix.

The following example returns only the departments whose sums of salaries are greater than $200,000:

```
SELECT dept_id, sum(salary) FROM emp
GROUP BY dept_id HAVING sum(salary) > 200000
```

Union Operator

The Union operator combines the results of two Select statements into a single result. The single result is all of the returned records from both Select statements. By default, duplicate records are not returned. To return duplicate records, use the All keyword (UNION ALL). The form is

```
SELECT statement
UNION [ALL]
SELECT statement
```

When using the Union operator, the select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order. For example:

```
SELECT last_name, salary, hire_date FROM emp
UNION
SELECT name, pay, birth_date FROM person
```

This example has the same number of column expressions, and each column expression, in order, has the same data type.

The following example is *not* valid because the data types of the column expressions are different (SALARY from EMP has a different data type than LAST_NAME from RAISES). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
UNION
SELECT salary, last_name FROM raises
```

Order By Clause

The Order By clause indicates how the records are to be sorted. The form is

```
ORDER BY {sort_expression [DESC | ASC]}, ...
```

sort_expression can be field names, expressions, or the positional number of the column expression to use.

The default is to perform an ascending (ASC) sort.

For example, to sort by LAST_NAME you could use either of the following Select statements:

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY last_name
```

or

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY 2
```

In the second example, LAST_NAME is the second column expression following Select, so Order By 2 sorts by LAST_NAME.

For Update Clause

The For Update clause locks the records of the database table selected by the Select statement. The form is

```
FOR UPDATE [OF column_expressions]
```

column_expressions is a list of field names in the database table that you intend to update, separated by a comma (,). Note that *column_expressions* is optional.

The following example returns all records in the employee database that have a SALARY field value of more than $20,000. When each record is fetched, it is locked. If the record is updated or deleted, the lock is held until you commit the change. Otherwise, the lock is released when you fetch the next record.

```
SELECT * FROM emp WHERE salary > 20000
    FOR UPDATE OF last_name, first_name, salary
```

SQL Expressions

Expressions are used in the Where clauses, Having clauses, and Order By clauses of SQL Select statements.

Expressions enable you to use mathematical operations as well as character string and date manipulation operators to form complex database queries.

The most common expression is a simple field name. You can combine a field name with other expression elements.

Valid expression elements are as follows:

- Field names
- Constants
- Exponential notation
- Numeric operators
- Character operators
- Date operators
- Relational operators
- Logical operators
- Functions

Constants

Constants are values that do not change. For example, in the expression PRICE * 1.05, the value 1.05 is a constant.

You must enclose character constants in pairs of single (') or double quotation marks ("). To include a single quotation mark in a character constant enclosed by single quotation marks, use two single quotation marks together (for example, 'Don''t'). Similarly, if the constant is enclosed by double quotation marks, use two double quotation marks to include one.

You must enclose date and time constants in braces ({}), for example, {01/30/89} and {12:35:10}. The form for date constants is MM/DD/YY or MM/DD/YYYY. The form for time constants is HH:MM:SS.

The logical constants are .T. and 1 for True and .F. and 0 for False. For portability, use 1 and 0.

Exponential Notation

You may include exponential notation. For example:

```
SELECT col1, 3.4E+7 FROM table1 WHERE calc < 3.4E-6 *
col2
```

Numeric Operators

You may include the following operators in numeric expressions

Operator	Meaning
+	Addition
–	Subtraction
*	Multiplication
/	Division
**	Exponentiation
^	Exponentiation

The following table shows examples of numeric expressions. For these examples, assume SALARY is 20000.

Example	Resulting value
`salary + 10000`	30000
`salary * 1.1`	22000
`2 ** 3`	8

You can precede numeric expressions with a unary plus (+) or minus (–). For example, `–(salary * 1.1)` is -22000.

Character Operators

Character expressions may include the following operators:

Operator	Meaning
+	Concatenation keeping trailing blanks.
–	Concatenation moving trailing blanks to the end

The following chart shows examples of character expressions. In the examples, LAST_NAME is `'JONES    '` and FIRST_NAME is `'ROBERT      '`.

Example	Resulting value
`first_name + last_name`	`'ROBERT     JONES    '`
`first_name – last_name`	`'ROBERTJONES         '`

Note: Some flat-file drivers return character data with trailing blanks as shown in the table. However, you cannot rely on the driver to return blanks. Therefore, if you want an expression that works with drivers that do and do not return trailing blanks, use the TRIM function before concatenating strings to make the expression portable. For example:

```
TRIM(first_name) + '' + TRIM(last_name)
```

Date Operators

You may include the following operators in date expressions:

Operator	Meaning
+	Add a number of days to a date to produce a new date
–	The number of days between two dates, or subtract a number of days from a date to produce a new date.

The following chart shows examples of date expressions. In these examples, hire_date is {01/30/90}.

Example	Resulting value
`hire_date + 5`	`{02/04/90 }`
`hire_date – {01/01/90 }`	`29`
`hire_date – 1 0`	`{01/20/90 }`

Relational Operators

The relational operators separating the two expressions may be any one of those listed in Table A-2.

Table A-2. Relational Operators

Operator	Meaning
=	Equal
<>	Not Equal
>	Greater Than
>=	Greater Than or Equal
<	Less Than
<=	Less Than or Equal
Like	Matching a pattern
Not Like	Not matching a pattern
Is Null	Equal to Null
Is Not Null	Not Equal to Null
Between	Range of values between a lower and upper bound
In	A member of a set of specified values or a member of a subquery.
Exists	True if a subquery returned at least one record

Table A-2. Relational Operators (cont.)

Operator	Meaning
Any	Compares a value to each value returned by a subquery. Any must be prefaced by =, <>, >, >=, <, or <=. =Any is equivalent to In.
All	Compares a value to each value returned by a subquery. All must be prefaced by =, <>, >, >=, <, or <=.

The following list shows some examples of relational operators:

```
salary <= 4000 0
dept = 'D101 '
hire_date > {01/30/89  }
salary + commission >= 5000   0
last_name LIKE 'Jo%  '
salary IS NUL L
salary BETWEEN 10000 AND 2000   0
WHERE salary = ANY (SELECT salary FROM emp WHERE
    dept = 'D101' )
WHERE salary > ALL (SELECT salary FROM emp WHERE
    dept = 'D101' )
```

Logical Operators

Two or more conditions may be combined to form more complex criteria. When two or more conditions are present, they must be related by AND or OR. For example:

```
salary = 40000 AND exempt =    1
```

The logical NOT operator is used to reverse the meaning. For example:

```
NOT (salary = 40000 AND exempt = 1    )
```

Operator Precedence

As expressions become more complex, the order in which the expressions are evaluated becomes important. Table A-3 shows the order in which the operators are evaluated. The operators in the first line are evaluated first, then those in the second line, and so on. Operators in the same line are evaluated left to right in the expression.

Table A-3. Operator Precedene

Precedence	Operator
1	Unary -, Unary +
2	**
3	*, /
4	+, -
5	=, <>, <, <=, >, >=, Like, Not Like, Is Null, Is Not Null, Between, In, Exists, Any, All
6	Not
7	AND
8	OR

The following example shows the importance of precedence:

```
WHERE salary > 40000 OR
hire_date > {01/30/89} AND
dept = 'D101 '
```

Because AND is evaluated first, this query retrieves employees in department D101 hired after January 30, 1989, as well as every employee making more than $40,000, no matter what department or hire date.

To force the clause to be evaluated in a different order, use parentheses to enclose the conditions to be evaluated first. For example:

```
WHERE (salary > 40000 OR hire_date > {01/30/89}    )
AND dept = 'D101 '
```

retrieves employees in department D101 that either make more than $40,000 or were hired after January 30, 1989.

Functions

The flat-file drivers support a number of functions that you may use in expressions. In Tables A-4 through A-6, the functions are grouped according to the type of result they return.

Table A-4. Functions that Return Character Strings

Function	Description
CHR	Converts an ASCII code into a one-character string `CHR(67 )` returns `C`.
RTRIM	Removes trailing blanks from a string. `RTRIM('ABC    ' )` returns `ABC`.
TRIM	Removes trailing blanks from a string. `TRIM('ABC    ' )` returns `ABC`.
LTRIM	Removes leading blanks from a string. `LTRIM('  ABC' )` returns `ABC`.
UPPER	Changes each letter of a string to uppercase. `UPPER('Allen' )` returns `ALLEN`.
LOWER	Changes each letter of a string to lowercase. `LOWER('Allen' )` returns `allen`.
LEFT	Returns leftmost characters of a string. `LEFT('Mattson',3 )` returns `Mat`.

Table A-4. Functions that Return Character Strings (cont.)

Function	Description
RIGHT	Returns rightmost characters of a string.
	`RIGHT('Mattson',4)` returns `tson`.
SUBSTR	Returns a substring of a string. Parameters are the string, the first character to extract, and the number of characters to extract (optional).
	`SUBSTR('Conrad',2,3)` returns `onr`.
	`SUBSTR('Conrad',2)` returns `onrad`.
SPACE	Generates a string of blanks.
	`SPACE(5)` returns `'     '`.
DTOC	Converts a date to a character string. An optional second parameter determines the format of the result:
	0 (the default) returns MM/DD/YY
	1 returns DD/MM/YY
	2 returns YY/MM/DD
	10 returns MM/DD/YYYY
	11 returns DD/MM/YYYY
	12 returns YYYY/MM/DD
	An optional third parameter specifies the date separator character. If not specified, a slash (/) is used.
	`DTOC({01/30/89})` returns `01/30/89`
	`DTOC({01/30/89}, 0)` returns `01/30/89`
	`DTOC({01/30/89}, 1)` returns `30/01/89`
	`DTOC({01/30/89}, 2,'-')` returns `89-01-30`
DTOS	Converts a date to a character string using the format YYYYMMDD.
	`DTOS({01/23/90})` returns `19900123`.

Table A-4. Functions that Return Character Strings (cont.)

Function	Description
IIF	Returns one of two values. Parameters are a logical expression, the true value, and the false value. If the logical expression evaluates to True, the function returns the true value. Otherwise, it returns the false value. `IIF(salary>20000,'BIG','SMALL')` returns `BIG` if SALARY is greater than 20000. If not, it returns `SMALL`.

STR	Converts a number to a character string. Parameters are the number, the total number of output characters (including the decimal point), and optionally the number of digits to the right of the decimal point.
	`STR(12.34567,4)` returns `12`
	`STR(12.34567,4,1)` returns `12.3`
	`STR(12.34567,6,3)` returns `12.346`

STRVAL	Converts a value of any type to a character string.
	`STRVAL('Woltman')` returns `Woltman`
	`STRVAL({12/25/53})` returns `12/25/53`
	`STRVAL (5 * 3)` returns `15`
	`STRVAL (4 = 5)` returns 'False'

TIME	Returns the time of day as a string. At 9:49 PM, `TIME()` returns `21:49:00`
USERNAME	For Btrieve, the logon ID specified at connect time is returned. For Paradox and Paradox 5 drivers, the user name specified during configuration is returned. For all other flat file drivers, an empty string is returned.

Table A-5. Functions that Return Numbers

Function	Description
MOD	Divides two numbers and returns the remainder of the division `MOD(10,3)` returns `1`
LEN	Returns the length of a string. `LEN('ABC')` returns `3`
MONTH	Returns the month part of a date. `MONTH({01/30/89})` returns `1`
DAY	Returns the day part of a date. `DAY({01/30/89})` returns `30`
YEAR	Returns the year part of a date. `YEAR({01/30/89})` returns `1989`
MAX	Returns the larger of two numbers. `MAX(66,89)` returns `89`
DAYOFWEEK	Returns the day of week (1-7) of a date expression `DAYOFWEEK({05/01/95})` returns `5`.
MIN	Returns the smaller of two numbers. `MIN(66,89)` returns `66`
POW	Raises a number to a power. `POW(7,2)` returns `49`
INT	Returns the integer part of a number. `INT(6.4321)` returns `6`

ROUND	Rounds a number.
	`ROUND(123.456, 0)` returns `123`
	`ROUND(123.456, 2)` returns `123.46`
	`ROUND(123.456, –2)` returns `100`

Table A-5. Functions that Return Numbers (cont.)

Function	Description
NUMVAL	Converts a character string to a number. If the character string is not a valid number, a zero is returned. `NUMVAL('123')` returns the number `123`
VAL	Converts a character string to a number. If the character string is not a valid number, a zero is returned. `VAL('123')` returns the number `123`

Table A-6. Functions that Return Dates

Function	Description
DATE	Returns today's date. If today is 12/25/79, `DATE()` returns `{12/25/79}`
TODAY	Returns today's date. If today is 12/25/79, `TODAY()` returns `{12/25/79}`
DATEVAL	Converts a character string to a date. `DATEVAL('01/30/89')` returns `{01/30/89}`

CTOD	Converts a character string to a date. An optional second parameter specifies the format of the character string: 0 (the default) returns MM/DD/YY, 1 returns DD/MM/YY, and 2 returns YY/MM/DD.
	`CTOD('01/30/89')` returns `{01/30/89}`
	`CTOD('01/30/89',1)` returns `{30/01/89}`

The following examples use some of the number and date functions.

Retrieve all employees that have been with the company at least 90 days:

```
SELECT first_name, last_name FROM emp
    WHERE DATE() – hire_date >= 90
```

Retrieve all employees hired in January of this year or last year:

```
SELECT first_name, last_name FROM emp
    WHERE MONTH(hire_date) = 1
    AND (YEAR(hire_date) = YEAR(DATE())
    OR YEAR(hire_date) = YEAR(DATE()) – 1)
```

Create and Drop Table Statements

The flat-file drivers support SQL statements to create and delete database files. The Create Table statement is used to create files and the Drop Table statement is used to delete files.

Create Table

The form of the Create Table statement is

CREATE TABLE *table_name* (*col_definition*[,*col_definition*, ...])

table_name can be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources. If it is a simple table name, the file is created in the directory you specified as the database directory in the connection string. If you did not specify a database directory in the connection string, the file is created in the directory you specified as the database directory in ODBC.INI. If you did not specify a database directory in either place, the file is created in the current working directory at the time you connected to the driver.

col_definition is the column name, followed by the data type, followed by an optional column constraint definition. Values for column names are database specific. The data type specifies a column's data type.

The only column constraint definition currently supported by some flat-file drivers is "not null." Not all flat-file tables support "not null" columns. In the cases where not null is not supported, this restriction is ignored and the driver returns a warning if "not null" is specified for a column. The "not null" column constraint definition is allowed in the driver so that you can write a database-independent application (and not be concerned about the driver raising an error on a Create Table statement with a "not null" restriction).

A sample Create Table statement to create an employee database table is

```
CREATE TABLE emp (last_name CHAR(20) NOT NULL,
    first_name CHAR(12) NOT NULL,
    salary NUMERIC (10,2) NOT NULL,
    hire_date DATE NOT NULL  )
```

Drop Table

The form of the Drop Table statement is

```
DROP TABLE  table_name
```

table_name may be a simple table name (EMP) or a full pathname. A simple table name is preferred for portability to other SQL data sources. If it is a simple table name, the file is dropped from the directory you specified as the database directory in the connection string. If you did not specify a database directory in the connection string, the file is deleted from the directory you specified as the database directory in ODBC.INI. If you did not specify a database directory in either of these places, the file is dropped from the current working directory at the time you connected to the driver.

A sample Drop Table statement to delete the employee database table is

```
DROP TABLE emp
```

Insert Statement

The SQL Insert statement is used to add new records to a database table. With it, you can specify either of the following:

- A list of values to be inserted as a new record
- A Select statement that copies data from another table to be inserted as a set of new records

The form of the Insert statement is

```
INSERT INTO table_name [(col_name, ...)]
{VALUES (expr, ...) | select_statement}
```

table_name may be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources.

col_name is an optional list of column names giving the name and order of the columns whose values are specified in the Values clause. If you omit *col_name*, the value expressions (*expr*) must provide values for all columns defined in the file and must be in the same order that the columns are defined for the file.

expr is the list of expressions giving the values for the columns of the new record. Usually, the expressions are constant values for the columns. Character string values must be enclosed in single or double quotation marks, date values must be enclosed in braces {}, and logical values that are letters must be enclosed in periods (for example, .T. or .F.).

An example of an Insert statement that uses a list of expressions is

```
INSERT INTO emp (last_name, first_name, emp_id, salary,
    hire_date )
VALUES ('Smith', 'John', 'E22345', 27500, {4/6/91}     )
```

Each Insert statement adds one record to the database table. In this case a record has been added to the employee database table, EMP. Values are specified for five columns. The remaining columns in the table are assigned a blank value, meaning Null.

select_satement is a query that returns values for each *col_name* value specified in the column name list. Using a Select statement instead of a list of value expressions lets you select a set of rows from one table and insert it into another table using a single Insert statement.

An example of an Insert statement that uses a Select statement is:

```
INSERT INTO emp1 (first_name, last_name, emp_id, dept,
salary)
SELECT first_name, last_name, emp_id, dept, salary from
emp
    WHERE dept = ‘D050  ’
```

In this type of Insert statement, the number of columns to be inserted must match the number of columns in the Select statement. The list of columns to be inserted must corresond to the columns in the Select statement just as it would to a list of value expressions in the other type of Insert statement. That is, the first column inserted corresponds to the first column selected; the second inserted to the second, etc.

The size and data type of these corresponding columns must be compatible. Each column in the Select list should have a data type that the ODBC driver accepts on a regular Insert/Update of the corresponding column in the Insert list. Values are truncated when the size of the value in the Select list column is greater than the size of the corresponding Insert list column.

The *select_statement* is evaluated before any values are inserted. This query cannot be made on the table into which values are inserted.

Update Statement

The SQL Update statement is used to change records in a database file. The form of the Update statement supported for flat-file drivers is

```
UPDATE  table_name SET col_name = expr, ...
[ WHERE {  conditions | CURRENT OF  cursor_name } ]
```

table_name may be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources.

col_name is the name of a column whose value is to be changed. Several columns can be changed in one statement.

expr is the new value for the column. The expression can be a constant value or a subquery. Character string values must be enclosed with single or double quotation marks, date values must be enclosed by braces {}, and logical values that are letters must be enclosed by periods (for example, .T. or .F.). Subqueries must be enclosed in parentheses.

The Where clause is any valid clause as described in the “Select Statement” section earlier in this appendix. It determines which records are to be updated.

The Where Current Of *cursor_name* clause can be used only by developers coding directly to the ODBC API. It causes the row at which *cursor_name* is positioned to be updated. This is called a “positioned update.” You must first execute a Select...For Update statement with a named cursor and fetch the row to be updated.

An example of an Update statement on the employee table is

```
UPDATE emp SET salary=32000, exempt=1
WHERE emp_id = 'E10001  '
```

The Update statement changes every record that meets the conditions in the Where clause. In this case the salary and exempt status are changed for all employees having the employee ID E10001. Because employee IDs are unique in the employee table, only one record is updated.

An example using a subquery is

```
UPDATE emp SET salary = (SELECT avg(salary) from emp)
WHERE emp_id = 'E10001  '
```

In this case, the salary is changed to the average salary in the company for the employee having employee ID E10001.

Delete Statement

The SQL Delete statement is used to delete records from a database table. The form of the Delete statement supported for flat-file drivers is

```
DELETE FROM  table_name
[ WHERE {  conditions | CURRENT OF  cursor_name } ]
```

table_name may be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources.

The Where clause is any valid clause as described in the “Select Statement” section earlier in this appendix. It determines which records are to be deleted. If you include only the keyword Where, all records in the table are deleted but the file is left intact.

The Where Current Of *cursor_name* clause can be used only by developers coding directly to the ODBC API. It causes the row at which *cursor_name* is positioned to be deleted. This is called a “positioned delete.” You must first execute a Select...For Update statement with a named cursor and fetch the row to be deleted.

An example of a Delete statement on the employee table is

```
DELETE FROM emp WHERE emp_id = 'E10001    '
```

Each Delete statement removes every record that meets the conditions in the Where clause. In this case every record having the employee ID E10001 is deleted. Because employee IDs are unique in the employee table, at most one record is deleted.

Reserved Keywords

The following words are reserved for use in SQL statements. If they are used for file or column names in a database that you use, you must enclose them in quotation marks in any SQL statement where they appear as file or column names.

- ALL
- AND
- BETWEEN
- COMPUTE
- CROSS
- DISTINCT
- FOR
- FROM
- FULL
- GROUP
- HAVING
- INNER
- INTO
- LEFT
- LIKE
- NATURAL
- NOT
- NULL
- ON
- OPTIONS
- OR
- ORDER
- RIGHT
- UNION
- WHERE

Go To

B Using Indexes

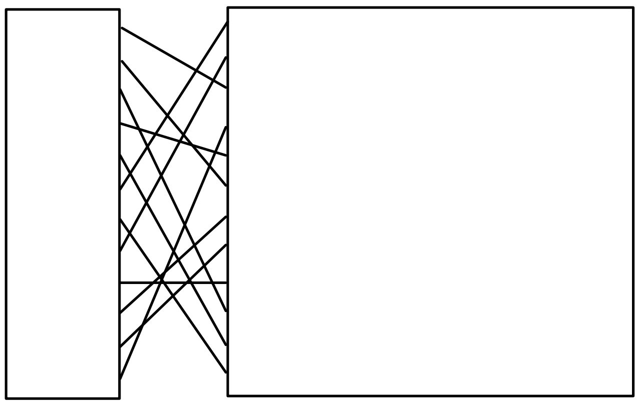
This appendix discusses the ways in which you can improve the performance of database activity using indexes. It provides general guidelines that apply to most databases. Consult your database vendor's documentation for more detailed information.

For information regarding how to create and drop indexes, see the appropriate database driver chapter for flat-file drivers or your database system documentation for relational drivers.

Introduction

An index is a database structure that you can use to improve the performance of database activity. A database table can have one or more indexes associated with it.

An index is defined by a field expression that you specify when you create the index. Typically, the field expression is a single field name, like EMP_ID. An index created on the EMP_ID field, for example, contains a sorted list of the employee ID values in the table. Each value in the list is accompanied by references to the records that contain that value.



A database driver can use indexes to find records quickly. An index on the EMP_ID field, for example, greatly reduces the time that the driver spends searching for a particular employee ID value. Consider the following Where clause:

```
WHERE emp_id = 'E10001  '
```

Without an index, the driver must search the entire database table to find those records having an employee ID of E10001. By using an index on the EMP_ID field, however, the driver can quickly find those records.

Indexes may improve the performance of SQL statements. You may not notice this improvement with small tables but it can be significant for large tables. However, there can be disadvantages to having too many indexes. Indexes can slow down the performance of some inserts, updates, and deletes when the driver has to maintain the indexes as well as the database tables. Also, indexes take additional disk space.

Improving Record Selection Performance

For indexes to improve the performance of selections, the index expression must match the selection condition exactly. For example, if you have created an index whose expression is last_name, the following Select statement uses the index:

```
SELECT * FROM emp WHERE last_name = 'Smith    '
```

This Select statement, however, does not use the index:

```
SELECT * FROM emp WHERE UPPER(last_name) = 'SMITH     '
```

The second statement does not use the index because the Where clause contains UPPER(LAST_NAME), which does not match the index expression LAST_NAME. If you plan to use the UPPER function in all your Select statements and your database supports indexes on expressions, then you should define an index using the expression UPPER(LAST_NAME).

Indexing Multiple Fields

If you often use Where clauses that involve more than one field, you may want to build an index containing multiple fields. Consider the following Where clause:

```
WHERE last_name = 'Smith' and first_name = 'Thomas     '
```

For this condition, the optimal index field expression is LAST_NAME,FIRST_NAME. This creates a concatenated index.

Concatenated indexes can also be used for Where clauses that contain only the first of two concatenated fields. The LAST_NAME, FIRST_NAME index also improves the performance of the following Where clause (even though no first name value is specified):

```
last_name = 'Smith  '
```

Consider the following Where clause:

```
WHERE last_name = 'Smith' and middle_name = 'Edward     '
    and first_name = 'Thomas   '
```

If your index fields include all the conditions of the Where clause in that order, the driver can use the entire index. However, if your index is on two nonconsecutive fields, say, LAST_NAME and FIRST_NAME, the driver can use only the LAST_NAME field of the index.

The driver uses only one index when processing Where clauses. If you have complex Where clauses that involve a number of conditions for different fields and have indexes on more than one field, the driver chooses an index to use. The driver attempts to use indexes on conditions that use the equal sign as the relational operator rather than conditions using other operators (such as greater than). Assume you have an index on the EMP_ID field as well as the LAST_NAME field and the following Where clause:

```
WHERE emp_id >= 'E10001' AND last_name = 'Smith     '
```

In this case, the driver selects the index on the LAST_NAME field.

If no conditions have the equal sign, the driver first attempts to use an index on a condition that has a lower *and* upper bound, and then attempts to use an index on a condition that has a lower *or* upper bound. The driver always attempts to use the most restrictive index that satisfies the Where clause.

In most cases, the driver does not use an index if the Where clause contains an OR comparison operator. For example, the driver does not use an index for the following Where clause:

```
WHERE emp_id >= 'E10001' OR last_name = 'Smith     '
```

Deciding Which Indexes to Create

Before you create indexes for a database table, consider how you will use the table. The two most common operations on a table are to

- Insert, update, and delete records
- Retrieve records

If you most often insert, update, and delete records, then the fewer indexes associated with the table, the better the performance. This is because the driver must maintain the indexes as well as the database tables, thus slowing down the performance of record inserts, updates, and deletes. It may be more efficient to drop all indexes before modifying a large number of records, and re-create the indexes after the modifications.

If you most often retrieve records, you must look further to define the criteria for retrieving records and create indexes to improve the performance of these retrievals. Assume you have an employee database table and you will retrieve records based on employee name, department, or hire date. You would create three indexes—one on the DEPT field, one on the HIRE_DATE field, and one on the LAST_NAME field. Or perhaps, for the retrievals based on the name field, you would want an index that concatenates the LAST_NAME and the FIRST_NAME fields (see the section “Indexing Multiple Fields” earlier in this chapter).

Here are a few rules to help you decide which indexes to create.

- If your record retrievals are based on one field at a time (for example, dept='D101'), create an index on these fields.

- If your record retrievals are based on a combination of fields, look at the combinations.

- If the comparison operator for the conditions is AND (for example, CITY = 'Raleigh' AND STATE = 'NC'), then build a concatenated index on the CITY and STATE fields. This index is also useful for retrieving records based on the CITY field.

- If the comparison operator is OR (for example, DEPT = 'D101' OR HIRE_DATE > {01/30/89}), an index does not help performance. Therefore, you need not create one.
- If the retrieval conditions contain both AND and OR comparison operators, you can use an index if the OR conditions are grouped. For example:

  ```
  dept = 'D101' AND (hire_date > {01/30/89} OR exempt = 1   )
  ```

  In this case, an index on the DEPT field improves performance.
- If the AND conditions are grouped, an index does not improve performance. For example:

  ```
  (dept = 'D101' AND hire_date) > {01/30/89}) OR exempt =    1
  ```

Improving Join Performance

When joining database tables, index tables can greatly improve performance. Unless the proper indexes are available, queries that use joins can take a long time.

Assume you have the following Select statement:

```
SELECT * FROM dept, emp WHERE dept.dept_id = emp.dep     t
```

In this example, the DEPT and EMP database tables are being joined using the department ID field. When the driver executes a query that contains a join, it processes the tables from left to right and uses an index on the second table's join field (the DEPT field of the EMP table).

To improve join performance, you need an index on the join field of the second table in the From clause. If there is a third table in the From clause, the driver also uses an index on the field in the third table that joins it to any previous table. For example:

```
SELECT * FROM dept, emp, add   r
WHERE dept.dept_id = emp.dept AND emp.loc = addr.lo     c
```

In this case, you should have an index on the EMP.DEPT field and the ADDR.LOC field.

Go To

C ODBC API and Scalar Functions

This appendix lists the ODBC API functions that the DataDirect ODBC drivers support and the scalar functions, which you use in SQL statements.

API Functions

All database drivers are ODBC Level 1-compliant—they support all ODBC Core and Level 1 functions. A limited set of Level 2 functions is also supported. The drivers support the functions listed in Table C-1. Any additions to these supported functions or differences in the support of specific functions are listed in the “ODBC Conformance Level” section in the individual driver chapters.

Table C-1. Supported ODBC API Functions

Core Functions	Level 1 Functions
SQLAllocConnect	SQLColumns
SQLAllocEnv	SQLDriverConnect
SQLAllocStmt	SQLGetConnectOption
SQLBindCol	SQLGetData
SQLBindParameter	SQLGetFunctions
SQLCancel	SQLGetInfo
SQLColAttributes	SQLGetStmtOption
SQLConnect	SQLGetTypeInfo
SQLDescribeCol	SQLParamData
SQLDisconnect	SQLPutData
SQLDrivers	SQLSetConnectOption
SQLError	SQLSetStmtOption
SQLExecDirect	SQLSpecialColumns
SQLExecute	SQLStatistics
SQLFetch	SQLTables
SQLFreeConnect	**Level 2 Functions**
SQLFreeEnv	SQLDataSources
SQLFreeStmt	SQLExtendedFetch (forward scrolling only)
SQLGetCursorName	SQLMoreResults
SQLNumResultCols	SQLNativeSql
SQLPrepare	SQLNumParams
SQLRowCount	SQLParamOptions
SQLSetCursorName	SQLSetScrollOptions
SQLTransact	

Scalar Functions

The following tables list the scalar functions that ODBC supports; your database system may not support all of these functions. See the documentation for your database system to find out which functions are supported.

You can use these functions in SQL statements using the following syntax:

```
{fn scalar-function}
```

where *scalar-function* is one of the functions listed in the following tables. For example,

```
SELECT {fn UCASE(NAME)} FROM EMP
```

String Functions

Table C-2 lists the string functions that ODBC supports.

The string functions listed can take the following arguments:

- *string_exp* can be the name of a column, a string literal, or the result of another scalar function, where the underlying data type is SQL_CHAR, SQL_VARCHAR, or SQL_LONGVARCHAR.
- *start*, *length*, and *count* can be the result of another scalar function or a literal numeric value, where the underlying data type is SQL_TINYINT, SQL_SMALLINT, or SQL_INTEGER.

The string functions are one-based; that is, the first character in the string is character 1.

Character string literals must be surrounded in double quotation marks.

Table C-2. Scalar String Functions

Function	Returns
ASCII(*string_exp*)	ASCII code value of the leftmost character of *string_exp* as an integer.
CHAR(*code*)	The character with the ASCII code value specified by *code*. *code* should be between 0 and 255; otherwise, the return value is data-source dependent.
CONCAT(*string_exp1*, *string_exp2*)	The string resulting from concatenating *string_exp2* and *string_exp1*. The string is system dependent.
DIFFERENCE(*string_exp1*, *string_exp2*)	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*)	A string where *length* characters have been deleted from *string_exp1* beginning at *start* and where *string_exp2* has been inserted into *string_exp*, beginning at *start*.
LCASE(*string_exp*)	Uppercase characters in *string_exp* converted to lowercase.
LEFT(*string_exp*,*count*)	The *count* of characters of *string_exp*.
LENGTH(*string_exp*)	The number of characters in *string_exp*, excluding trailing blanks and the string termination character.
LOCATE(*string_exp1*, *string_exp2*[,*start*])	The starting position of the first occurrence of *string_exp1* within *string_exp2*. If *start* is not specified the search begins with the first character position in *string_exp2*. 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, 0 is returned.
LTRIM(*string_exp*)	The characters of *string_exp*, with leading blanks removed.

Table C-2. Scalar String Functions (cont.)

Function	**Returns**
REPEAT(*string_exp*, *count*)	A 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*)	The rightmost *count* of characters in *string_exp*.
RTRIM(*string_exp*)	The characters of *string_exp* with trailing blanks removed.
SOUNDEX(*string_exp*)	A data-source-dependent string representing the sound of the words in *string_exp*.
SPACE(*count*)	A string consisting of *count* spaces.
SUBSTRING(*string_exp*, *start*, *length*)	A string derived from *string_exp* beginning at the character position *start* for *length* characters.
UCASE(*string_exp*)	Lowercase characters in *string_exp* converted to uppercase.

Numeric Functions

Table C-3 lists the numeric functions that ODBC supports.

The numeric functions listed can take the following arguments:

- *numeric_exp* can be a column name, a numeric literal, or the result of another scalar function, where the underlying data type is SQL_NUMERIC, SQL_DECIMAL, SQL_TINYINT, SQL_SMALLINT, SQL_INTEGER, SQL_BIGINT, SQL_FLOAT, SQL_REAL, or SQL_DOUBLE.
- *float_exp* can be a column name, a numeric literal, or the result of another scalar function, where the underlying data type is SQL_FLOAT.

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

Table C-3. Scalar Numeric Functions

Function	Returns
ABS(*numeric_exp*)	Absolute value of *numeric_exp.*
ACOS(*float_exp*)	Arccosine of *float_exp* as an angle in radians.
ASIN(*float_exp*)	Arcsine of *float_exp* as an angle in radians.
ATAN(*float_exp*)	Arctangent of *float_exp* as an angle in radians.
ATAN2(*float_exp1*, *float_exp*2)	Arctangent of the x and y coordinates, specified by *float_exp1* and *float_exp2* as an angle in radians.
CEILING(*numeric_exp*)	Smallest integer greater than or equal to *numeric_exp.*
COS(*float_exp*)	Cosine of *float_exp* as an angle in radians.
COT(*float_exp*)	Cotangent of *float_exp* as an angle in radians.
DEGREES(*numeric_exp*)	Number if degrees converted from *numeric_exp* radians.
EXP(*float_exp*)	Exponential value of *float_exp.*
FLOOR(*numeric_exp*)	Largest integer less than or equal to *numeric_exp*.
LOG(*float_exp*)	Natural log of *float_exp.*
LOG10(*float_exp*)	Base 10 log of *float_exp.*
MOD(*integer_exp1*, *integer_exp2*)	Remainder of *integer_exp1* divided by *integer_exp2*.
PI()	Constant value of pi as a floating-point number.
POWER(*numeric_exp*, *integer_exp*)	Value of *numeric_exp* to the power of *integer_exp*.
RADIANS(*numeric_exp*)	Number of radians converted from *numeric_exp* degrees.
RAND([*integer_exp*])	Random floating-point value using *integer_exp* as the optional seed value.

Table C-3. Scalar Numeric Functions (cont.)

Function	**Returns**
ROUND(*numeric_exp*, *integer_exp*)	*numeric_exp* rounded to *integer_exp* places right of the decimal (left of the decimal if *integer_exp* is negative).
SIGN(*numeric_exp*)	Indicator of the sign of *numeric_exp*. If *numeric_exp* < 0, -1 is returned. If *numeric_exp* = 0, 0 is returned. If *numeric_exp* > 0, 1 is returned.
SIN(*float_exp*)	Sine of *float_exp*, where *float_exp* is an angle in radians.
SQRT(*float_exp*)	Square root of *float_exp*.
TAN(*float_exp*)	Tangent of *float_exp*, where *float_exp* is an angle in radians.
TRUNCATE(*numeric_exp*, *integer_exp*)	*numeric_exp* truncated to *integer_exp* places right of the decimal. (If *integer_exp* is negative, truncation is to the left of the decimal.)

Date and Time Functions

Table C-4 lists the date and time functions that ODBC supports.

The date and time functions listed can take the following arguments:

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

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

Table C-4. Scalar Time and Date Functions

Function	**Returns**
CURDATE()	Current date as a date value.
CURTIME()	Current local time as a time value.
DAYNAME(*date_exp*)	Character string containing a data-source-specific name of the day for the day portion of *date_exp*.
DAYOFMONTH(*date_exp*)	Day of the month in *date_exp* as an integer value (1-31).
DAYOFWEEK(*date_exp*)	Day of the week in *date_exp* as an integer value (1-7).
DAYOFYEAR(*date_exp*)	Day of the year in *date_exp* as an integer value (1-366).
HOUR(*time_exp*)	Hour in *time_exp* as an integer value (0-23).
MINUTE(*time_exp*)	Minute in *time_exp* as an integer value (0-59).
MONTH(*date_exp*)	Month in *date_exp* as an integer value (1-12).
MONTHNAME(*date_exp*)	Character string containing the data source-specific name of the month.
NOW()	Current date and time as a timestamp value.
QUARTER(*date_exp*)	Quarter in *date_exp* as an integer value (1-4).
SECOND(*time_exp*)	Second in *date_exp* as an integer value (0-59).

Table C-4. Scalar Time and Date Functions (cont.)

Function	Returns
TIMESTAMPADD(*interval*, *integer_exp*, *time_exp*)	Timestamp calculated by adding *integer_exp* intervals of type interval to *time_exp*. *interval* can be 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 Fractional seconds are expressed in billionths of a second.
TIMESTAMPDIFF(*interval*, *time_exp1*, *time_exp2*)	Integer number of intervals of type *interval* by which *time_exp2* is greater than *time_exp1*. *interval* has the same values as TIMESTAMPADD. Fractional seconds are expressed in billionths of a second.
WEEK(*date_exp*)	Week of the year in *date_exp* as an integer value (1-53).
YEAR(*date_exp*)	Year in *date_exp*. The range is data-source dependent.

System Functions

Table C-5 lists the system functions that ODBC supports.

Table C-5. Scalar System Functions

Function	Returns
DATABASE()	Name of the database, corresponding to the connection handle (*hdbc*).
IFNULL(*exp*,*value*)	*value*, if *exp* is null.
USER()	Authorization name of the user.

Go To

D Locking and Isolation Levels

This appendix discusses locking and isolation levels and how their settings can affect the data you retrieve. Different database systems support different locking and isolation levels. See the section “Isolation and Lock Levels Supported” in the appropriate driver chapter.

Locking

Locking is a database operation that restricts a user from accessing a table or record. Locking is used in situations where more than one user might try to use the same table or record at the same time. By locking the table or record, the system ensures that only one user at a time can affect the data.

Locking is a vital activity in multiuser databases, where different users can try to access or modify the same records concurrently. While such concurrent database activity is desirable, it can create problems. Without locking, for example, if two users try to modify the same record at the same time, they might encounter problems ranging from retrieving bad data to deleting data that the other user needs. However, if the first user to access a record can lock that record to temporarily prevent other users from modifying it, such problems can be avoided. Locking provides a way to manage concurrent database access while minimizing the various problems it can cause.

Isolation Levels

An isolation level represents a particular locking strategy employed in the database system to improve data consistency. The higher the isolation level, the more complex the locking strategy behind it. The isolation level provided by the database determines whether a transaction will encounter the following behaviors in data consistency:

Dirty reads	User 1 modifies a row. User 2 reads the same row before User 1 commits. User 1 performs a rollback. User 2 has read a row that has never really existed in the database. User 2 may base decisions on false data.
Non-repeatable reads	User 1 reads a row but does not commit. User 2 modifies or deletes the same row and then commits. User 1 rereads the row and finds it has changed (or has been deleted).
Phantom reads	User 1 uses a search condition to read a set of rows but does not commit. User 2 inserts one or more rows that satisfy this search condition, then commits. User 1 rereads the rows using the search condition and discovers rows that were not present before.

Isolation levels represent the database system's ability to prevent these behaviors. The American National Standards Institute (ANSI) defines four isolation levels:

- Read uncommitted (0)
- Read committed (1)
- Repeatable read (2)
- Serializable (3)

In ascending order (0-3), these isolation levels provide an increasing amount of data consistency to the transaction. At the lowest level, all three behaviors can occur. At the highest level, none can occur. The success of each level in preventing these behaviors is due to the locking strategies that they employ, which are as follows:

Read uncommitted (0)	Locks are obtained on modifications to the database and held until end of transaction (EOT). Reading from the database does not involve any locking.
Read committed (1)	Locks are acquired for reading and modifying the database. Locks are released after reading but locks on modified objects are held until EOT.
Repeatable read (2)	Locks are obtained for reading and modifying the database. Locks on all modified objects are held until EOT. Locks obtained for reading data are held until EOT. Locks on non-modified access structures (indexes, hashing structures, etc.) are released after reading.
Serializable (3)	All data read or modified is locked until EOT. All access structures that are modified are locked until EOT. Access structures used by the query are locked until EOT.

Table D-1 shows what data consistency behaviors can occur at each isolation level.

Table D-1. Isolation Levels and Data Consistency

Level	Dirty Read	Nonrepeatable Read	Phantom Read
0, Read uncommitted	Yes	Yes	Yes
1, Read committed	No	Yes	Yes
2, Repeatable read	No	No	Yes
3, Serializable	No	No	No

While higher isolation levels provide better data consistency, this consistency can be costly in terms of the concurrency provided to individual users. Concurrency is the ability of multiple users to access and modify data simultaneously. As isolation levels increase, so does the chance that the locking strategy used will create problems in concurrency.

Put another way: the higher the isolation level, the more locking involved, and the more time users may spend waiting for data to be freed by another user. Because of this inverse relationship between isolation levels and concurrency, you must consider how people use the database before choosing an isolation level. You must weigh the trade-offs between data consistency and concurrency, and decide which is more important.

Locking Modes and Levels

Different database systems employ various locking modes, but they have two basic ones in common: shared and exclusive. Shared locks can be held on a single object by multiple users. If one user has a shared lock on a record, then a second user can also get a shared lock on that same record. However, the second user cannot get an exclusive lock on that record. Exclusive locks are exclusive to the user that obtains them. If one user has an exclusive lock on a record, then a second user cannot get either type of lock on the same record.

Performance and concurrency can also be affected by the locking level used in the database system. The locking level determines the size of an object that is locked in a database. For example, many database systems let you lock an entire table, as well as individual records. An intermediate level of locking, page-level locking, is also common. A page contains one or more records and is typically the amount of data read from the disk in a single disk access. The major disadvantage of page-level locking is that if one user locks a record, a second user may not be able to lock other records because they are stored on the same page as the locked record.

Go To

E ODBC.INI

ODBC.INI is a standard text file on all operating environments except for Windows 95, Windows NT, and OS/2. On Windows 95 and Windows NT, ODBC.INI is a subkey in the Windows 95 or Windows NT registry. On OS/2, it is a binary file.

ODBC.INI stores ODBC configuration and connection information. On the Macintosh, ODBC.INI is called ODBC Preferences. On UNIX, it is called .odbc.ini. This manual refers to the ODBC.INI structure generically as ODBC.INI.

ODBC.INI in the Supported Environments

In all environments, the structure for ODBC.INI is the same as that defined by the Microsoft ODBC specification. However, the methods by which you can access ODBC.INI vary according to your operating environment. The following sections show how ODBC.INI is implemented in all supported environments.

Windows

On Windows, ODBC.INI is a text file. It resides in the Windows directory. You can access the file via either the ODBC Administrator program in the Windows Control Panel or any text file editor such as Windows Notepad.

Windows 95 andWindows NT

On Windows 95 and Windows NT, ODBC.INI is a subkey of the Windows 95 or Windows NT registry. The Windows 95 and Windows NT registry is a binary database that is maintained by WIndows 95 or Windows NT and structured as a set of keys that partition information stored within. The ODBC.INI key is a subkey of the key HKEY_CURRENT_USER. The hierarchy is HKEY_CURRENT_USER, Software, ODBC, ODBC.INI. The discussion in this appendix refers to this level within the registry.

You maintain the ODBC.INI subkey using the ODBC Administrator program, which is located in the Windows 95 or Windows NT Control Panel if you are the user who installed the DataDirect product. All other users (users who are using the product but did not install it) must use ODBCAD32.EXE to run the ODBC Administrator program; an ODBC Administrator icon is then created in the DataDirect product group.

Because Windows 95 and Windows NT can host multiple users, each user has a distinct version of the HKEY_CURRENT_USER database, stored under a unique user key in the registry. Therefore, each user must run the ODBC Administrator to initialize and configure the data sources in the ODBC.INI subkey. To start the ODBC Administrator, double-click the ODBC icon.

During the primary installation of the DataDirect ODBC Pack, another Windows 95 or Windows NT registry subkey, ODBCINST.INI, is initialized and configured. This subkey is stored in the key HKEY_LOCAL_MACHINE and holds the number and types of drivers installed at the machine level. This information is then used by the ODBC Administrator to determine which drivers are to be displayed during the user configuration of the ODBC.INI subkey.

You cannot put comments in ODBC.INI.

OS/2

On OS/2 systems, ODBC.INI resides in the same directory that contains the OS2.INI and OS2SYS.INI files. It is a binary file that you can maintain by using the ODBC Administrator or an INI editor. The format of this file is defined in the section “ODBC.INI Structure.”

You cannot put comments in ODBC.INI.

UNIX

On UNIX systems, ODBC.INI is a plain text file called .odbc.ini. It normally resides in the user’s $HOME directory. This file is maintained manually using any text editor to define data source entries as described in the “Connecting to a Data Source Using a Connection String” section of each driver’s chapter.

UNIX support of the database drivers also allows the use of a centralized ODBC.INI file that a system administrator can control. This is accomplished by setting the environment variable ODBC_INI to point to the fully qualified pathname of the centralized file. For example, in the C shell you could set this variable as follows:

```
setenv ODBC_INI /opt/odbc/system_odbc.ini
```

In the Bourne or Korn shell, you would set it this way:

```
ODBC_INI=/opt/odbc/system_odbc.ini;export ODBC_INI
```

If a user has a private copy of .odbc.ini in his or her $HOME directory, it is used instead of the centralized version.

Macintosh

On the Macintosh, ODBC.INI is a text file called “ODBC Preferences.” It resides in the Preferences folder and can be modified using the TeachText program if needed. On the Macintosh, the ODBC Administrator is called ODBC Setup and is located in the Control Panels folder.

ODBC.INI Structure

An INI file contains a [*section_name*] heading that is followed by optional *attribute*=*value* pairs, called entries. Both the section name and the attributes are case insensitive except on OS/2. Where allowed, comment lines begin with a semicolon (;).

The ODBC.INI format, as specified by the Microsoft® Open Database Connectivity (ODBC) specification, is as follows:

```
[ODBC Data Sources]    ;Lists data sources available to
                       ;  ODBC   .
ds_name1=driver_desc 1  ;Lists each data source name
                        ;  followed by a description    .
ds_name2=driver_desc  2
...

[ds_name 1]             ;Defines the actual ODBC Driver
                       ;  source; for example, Oracle     .
Driver =path/dl l       ;Defines the path to the driver
                       ;  DLL   .
Description =desc       ;Briefly describes the data sourc    e
...

[ds_name 2]
Driver =path/dl l
Description =desc
...
```

Under Windows 95, Windows NT or OS/2, you cannot put comments in ODBC.INI.

For the Macintosh, the format of ODBC.INI is as follows:

```
[ODBC Data Sources]    ;Lists data sources available to
                       ;  ODBC   .
ds_name1=ODBC_resourc  e ;Matches the contents of the ODB   C
                       ;  Resource   .
ds_name2=ODBC_resourc  e
...

[ds_name 1]             ;Defines the actual ODBC Driver
                       ;  source; for example, Oracle    .
Driver =function_set_I D ;Contains the function set ID o   f
                       ;  the ODBC Library    .
...

[ds_name 2]
Driver =function_set_I D
...
```

The [ODBC Data Sources] section is mandatory. It provides the driver manager with a list of data sources that are supported for your connection requests. You can change the names in this list, but each entry must match its corresponding [*ds_name*] section in ODBC.INI.

The [*ds_name*] sections contain a Driver= specification, which points to the location of the installed driver, as well as a Description= specification that describes the driver. If you change the location of a driver, you can change the Driver= specification to match the new location. You can also use just the name of the driver, and the driver manager will attempt to locate the driver based on information obtained from your environment.

You might need to assign other entries depending on the data source you are configuring. The section "Connecting to a Data Source Using a Connection String" in each driver chapter lists the attributes that you can set. Use the ODBC Administrator program to modify ODBC.INI in all environments that provide this interface. This protects the file from becoming corrupted or nonfunctional.

ODBC.INI Examples

The following example shows an ODBC.INI file as defined by the ODBC specification:

```
;---------------------------------------------------    -
; ODBC.INI - INTERSOLV ODBC Driver Manager INI Fil    e
;---------------------------------------------------    -
[ODBC Data Sources  ]
qess=SQL Serve r
qedbf=dBAS E
qeor7=Oracl e
[qess ]
Driver=qess07.dl  l
Description=INTERSOLV SQL Server drive    r
ServerName=alic  e
LogonID=tes t

[qedbf ]
Driver=qedbf07.dl  l
Description=INTERSOLV dBASE drive   r
Database=C:\DBAS  E   ;Windows pat  h
[qeor7 ]
Driver=qeor707.dl  l
Description=INTERSOLV Oracle drive   r
ServerName=t:magna:V  7
LogonID=tes t
```

The following example shows an ODBC.INI file as implemented for the Macintosh:

```
;---------------------------------------------------     -
; ODBC.INI - INTERSOLV ODBC Driver Manager INI Fil     e
;---------------------------------------------------     -

[ODBC Data Sources  ]
qess=INTERSOLV SQLServer Drive   r
qedbf=INTERSOLV dBase Drive   r
qeor7=INTERSOLV Oracle 7 Drive   r

[qess ]
Driver=appl:ODBC$QESQLServerDriverFunctionSe    t
Description=INTERSOLV SQLServer Drive   r
ServerName=magn  a
LogonID=tes t

[qedbf ]
Driver=appl:ODBC$QEdBaseDriverFunctionSe    t
Description=INTERSOLV dBase Drive   r

[qeor7 ]
Driver=appl:ODBC$QEOracle7DriverFunctionSe    t
Description=INTERSOLV Oracle 7 Drive   r
ServerName=t:magna:V  7
LoginID=tes t
```

Go To

F Designing Performance-Oriented ODBC Applications

Designing and coding performance-oriented ODBC applications is not easy. There is no “Performance” chapter in the *ODBC Programmer’s Reference*. There are no warnings returned from ODBC drivers or the ODBC driver manager when applications are coded inefficiently. Furthermore, there are no internals documents describing how ODBC drivers are implemented.

This appendix provides guidelines for designing performance-oriented ODBC applications. These guidelines were accumulated by examining the ODBC implementations of several dozen *shipping* ODBC applications. Perhaps the “ODBC is slow” myth has grown from the release of some of these unoptimized applications.

The guidelines presented are not hard theorems but general advice. While some ODBC drivers may not benefit from one particular guideline presented, we have chosen to present the guidelines that benefit the majority of drivers. The guidelines are based on four general rules:

- Network communication is slow; reduce network traffic as much as possible.
- The process of evaluating complex SQL queries on the DBMS server may be slow and may reduce concurrency; simplify queries as much as possible.
- Excessive calls from the application to the driver decreases performance; optimize the application to driver interaction.

- Disk i/o is slow; limit disk i/o to improve performance.

The guidelines are divided into five sections: “Catalog Functions,” “Retrieving Data,” “ODBC Function Selection,” “Design Options,” and “Updating Data.”

Catalog Functions

The following ODBC functions are defined to be catalog functions:

- SQLColumns
- SQLColumnPrivileges
- SQLForeignKeys
- SQLGetTypeInfo
- SQLProcedures
- SQLProcedureColumns
- SQLSpecialColumns
- SQLStatistics
- SQLTables
- SQLTablePrivileges

While some drivers implement SQLGetTypeInfo as hard coded information, many drivers must query the server to obtain accurate information about which types are supported (such as to find dynamic types such as user defined types, etc.). Therefore, SQLGetTypeInfo is included in this list of potentially expensive ODBC functions.

Catalog Functions Are Relatively Slow

Catalog functions are relatively slow compared to all other ODBC functions. Applications should cache information returned from catalog functions so that multiple executions are not needed.

While almost no ODBC application can be written without catalog functions, their use should be minimized. To return all result column information *mandated* by the ODBC specification, a driver may have to perform multiple queries, joins, subqueries, and/or unions in order to return the necessary result set for a single call to a catalog function. These particular elements of the SQL language are performance hogs. Frequent use of catalog functions in an application will likely result in poor performance.

Applications should attempt to cache information from catalog functions if possible. For example, call SQLGetTypeInfo once in the application and cache away the elements of the result set that your application depends on. It is unlikely that any application uses all elements of the result set generated by a catalog function so the cache of information should not be difficult to maintain.

Passing Null Arguments

Passing null arguments to catalog functions results in time consuming queries being generated by the driver. In addition, network traffic potentially increases due to unwanted result set information. Always supply as many non-null arguments to catalog functions as possible.

Since catalog functions are slow, applications should invoke them as efficiently as possible. Many applications pass the least amount of non-null arguments necessary for the function to return success. Consider a call to SQLTables where the application requests information about table “Customers.” Many times this call is coded similar to the following example.

```
rc = SQLTables (NULL, NULL, NULL, NULL, “Customers”,
SQL_NTS,
     NULL);
```

It is quite possible that a driver could turn this SQLTables call into SQL similar to:

```
SELECT ...  FROM SysTables WHERE TableName = ‘Customers’
    UNION ALL
SELECT ... FROM SysViews WHERE ViewName = ‘Customers’
    UNION ALL
SELECT ... FROM SysSynonyms WHERE SynName = ‘Customers’
    ORDER BY ...
```

In some circumstances not much information is known about the object for which you are requesting information but in many cases at least some information is known. Any additional information that the application can send the driver when calling catalog functions can result in improved performance and reliability.

Suppose in the previous example that three “Customers” tables were returned in the result set: one owned by the user, one owned by sales, and one that was a view created by management. While it may be obvious that the application’s intent is to use the one owned by the user, it may not be obvious to the *user* which table to choose. If the application had specified the OwnerName argument for the SQLTables call, then reliability is improved (only one table returned) and performance increases (less network traffic required to return only one result row and unwanted rows are filtered by the DBMS). In addition, if the TableType argument can be supplied, then the SQL sent to the server can be optimized from a three query union to a singleton Select similar to:

```
SELECT ... FROM SysTables WHERE TableName = ‘Customers’
    and Owner = ‘Beth ’
```

SQLColumns

Avoid using SQLColumns to determine characteristics about a table. Instead, use a dummy query with SQLDescribeCol.

Consider an ad-hoc application that allows the user to choose the columns that will be selected. Should the application use SQLColumns to return information about the columns to the user or instead prepare a dummy query and call SQLDescribeCol?

Case 1: SQLColumns Method

```
rc = SQLColumns (... “UnknownTable” ...)    ;
// This call to SQLColumns will generate a query to the
// system catalogs... possibly a join which must be
// prepared, executed, and produce a result se    t
rc = SQLBindCol (...)  ;
rc = SQLExtendedFetch (...);
// user must retrieve N rows from the server
// N = # result columns of UnknownTabl   e
// result column information has now been obtaine    d
```

Case 2: SQLDescribeCol Method

```
// prepare dummy query
rc = SQLPrepare (... “SELECT * from UnknownTable
    WHERE 1 = 0” ...)  ;
// query is never executed on the server - only prepare     d
rc = SQLNumResultCols (...)   ;
for (irow = 1; irow <= NumColumns; irow++)     {
    rc = SQLDescribeCol (...   )
    // + optional calls to SQLColAttribute   s
    }
// result column information has now been obtaine    d
// Note we also know the column ordering within the
// table!
// This information cannot b   e
// assumed from the SQLColumns example   .
```

In both cases a query is sent to the server, but in Case 1 the query must be evaluated and form a result set that must be sent to the client. Clearly, Case 2 is the better performing model.

To somewhat complicate this discussion, let us consider a DBMS server that does not natively support preparing a SQL statement. The performance of Case 1 does not change but Case 2 increases minutely because the dummy query must be evaluated instead of only prepared. Because the Where clause of the query always evaluates to FALSE, the query generates no result rows and should execute without accessing table data. For even this type of driver, method 2 out performs method 1.

Retrieving Data

Retrieving Long Data

Retrieving long data (SQL_LONGVARCHAR and SQL_LONGVARBINARY data) across the network is very resource intensive and thus slow. Applications should avoid requesting long data unless it is absolutely necessary.

How often do users want to *see* long data? By default, it is generally acceptable not to retrieve long data or binary data because most users don't want to see such information. If the user does wish to see these result items, then the application can requery the database specifying only the long columns in the select list. This method allows the average user to retrieve the result set without having to pay a high performance penalty for intense network traffic.

While the most optimal method is to exclude long data from the select list, some applications do not formulate the select list before sending the query to the ODBC driver (i.e. some applications simply 'select * from <table name> ...'). If the select list contains long data then some drivers *must* retrieve that data at fetch time even if the application does not bind the long data in the result set. If possible, the designer should attempt to implement a method that does not retrieve all columns of the table.

Reducing the Size of Data Retrieved

Reduce the size of any data being retrieved to some manageable limit by calling SQLSetStmtOption with the SQL_MAX_SIZE option. This reduces network traffic and improves performance.

While eliminating SQL_LONGVARCHAR and SQL_LONGVARBINARY data from the result set is ideal in terms of performance, many times long data must be retrieved. Consider, however, that most users do not want to see 100k or more of textual information on the screen. What techniques, if any, are available to limit the amount of data retrieved?

Many application developers mistakenly assume that if they call SQLGetData with a container of size *x* that the ODBC driver only retrieves *x* bytes of information from the server. Because SQLGetData can be called multiple times for any one column, most drivers optimize their network use by retrieving long data in large chunks and then returning it to the user when requested.

Example:

```
char CaseContainer[1000]  ;
...
rc = SQLExecDirect (hstmt, “SELECT CaseHistory FROM Cases
    WHERE CaseNo = 71164”, SQL_NTS)   ;
...
rc = SQLFetch (hstmt)  ;
rc = SQLGetData (hstmt, 1, CaseContainer,(SWORD)
sizeof(CaseContainer), ...)   ;
```

At this point, it is more likely that an ODBC driver retrieves 64k of information from the server instead of 1000 bytes. One 64k retrieval is less expensive than sixty-four 1000 byte retrievals in terms of network access. Unfortunately, the application may not call SQLGetData again; thus, the first and only retrieval of CaseHistory was slowed by the fact that 64k of data had to be sent across the network.

Many ODBC drivers allow limiting the amount of data retrieved across the network by supporting the statement option SQL_MAX_SIZE. This option allows the driver to communicate to the DBMS backend server that only Z bytes of data are pertinent to the client. The server responds by sending only the first Z bytes of data for *all* result columns. This optimization greatly reduces network traffic and thus improves performance of the client. Our example returned just one row, but consider the case where 100 rows are returned in the result set. The performance improvement is substantial.

Using Bound Columns

Retrieving data through bound columns (SQLBindCol) instead of using SQLGetData reduces the ODBC call load and thus improves performance.

Consider the following pseudo-code fragment:

```
rc = SQLExecDirect (hstmt, "SELECT <20 columns> FROM
    Employees WHERE HireDate >= ?", SQL_NTS)    ;
do {
rc  = SQLFetch (hstmt);
// call SQLGetData 20 time  s
} while ((rc == SQL_SUCCESS) || (rc ==
  SQL_SUCCESS_WITH_INFO))  ;
```

Suppose the query returns 90 result rows. The number of ODBC calls made are > 1890 (20 calls to SQLGetData x 90 result rows + 91 calls to SQLFetch).

Consider the same scenario that uses SQLBindCol instead of SQLGetData:

```
rc = SQLExecDirect (hstmt, "SELECT <20 columns> FROM
    Employees WHERE HireDate >= ?", SQL_NTS)    ;
// call SQLBindCol 20 time  s
do {
rc  = SQLFetch (hstmt);
} while ((rc == SQL_SUCCESS) || (rc ==
  SQL_SUCCESS_WITH_INFO))  ;
```

The number of ODBC calls made is reduced from greater than 1890 to about 110 (20 calls to SQLBindCol + 91 calls to SQLFetch). In addition to reducing the call load many drivers optimize use of SQLBindCol by binding result information directly from the DBMS into the user's buffer. That is, instead of the driver retrieving information into a container then copying that information to the user's buffer, the driver simply requests the information from the server be placed directly into the user's buffer.

Using SQLExtendedFetch instead of SQLFetch

Use SQLExtendedFetch to retrieve data instead of SQLFetch. The ODBC call load decreases (resulting in better performance,) and the code is less complex (resulting in more maintainable code).

Most ODBC drivers now support SQLExtendedFetch for forward only cursors; yet, most ODBC applications use SQLFetch to retrieve data. Again consider the example above using SQLExtendedFetch instead of SQLFetch:

```
rc = SQLSetStmtOption (hstmt, SQL_ROWSET_SIZE, 100);
// use arrays of 100 elements
rc = SQLExecDirect (hstmt, “SELECT <20 columns> FROM
    Employees WHERE HireDate >= ?”, SQL_NTS);
// call SQLBindCol 1 time specifying row-wise binding
do {
rc = SQLExtendedFetch (hstmt, SQL_FETCH_NEXT, 0,
    &RowsFetched, RowStatus);
} while ((rc == SQL_SUCCESS) || (rc ==
  SQL_SUCCESS_WITH_INFO));
```

The number of ODBC calls made by the application is reduced from 110 in the last example to four (1 SQLSetStmtOption + 1 SQLExecDirect + 1 SQLBindCol + 1 SQLExtendedFetch). Note the total savings from an initial call load of more than 1890 ODBC calls in the first presentation of the example to four above. In addition to reducing the call load, many ODBC drivers retrieve data from the server in arrays that further improves the performance by reducing network traffic.

For those drivers that do not support SQLExtendedFetch, the application can enable forward only cursors using the ODBC cursor library (call SQLSetConnectOption using SQL_ODBC_CURSORS/ SQL_CUR_USE_IF_NEEDED). While using the cursor library does not improve performance, it should not be detrimental to application response time when using forward only cursors (no logging is required). Furthermore, using the cursor library when SQLExtendedFetch is not supported natively by the driver simplifies the code because the application can always depend on SQLExtendedFetch being available. The application need not code two algorithms (one using SQLExtendedFetch and one using SQLFetch).

ODBC Function Selection

SQLPrepare/SQLExecute Vs. SQLExecDirect

Don’t assume that SQLPrepare/SQLExecute is always as efficient as SQLExecDirect. Use SQLExecDirect for queries that will be executed once and SQLPrepare/SQLExecute for queries that will be executed more than once.

ODBC drivers are optimized based on the perceived use of the functions that are being executed. SQLPrepare/SQLExecute is optimized for multiple executions of a statement that most likely uses parameter markers. SQLExecDirect is optimized for a single execution of a SQL statement. Unfortunately, more than seventy-five percent of all ODBC applications use SQLPrepare/SQLExecute *exclusively*.

The pitfall of always coding SQLPrepare/SQLExecute can be understood better by considering an ODBC driver that implements SQLPrepare by creating a stored procedure on the server that contains the prepared statement. Creating a stored procedure has substantial overhead, but the the ODBC driver is assuming is that the statement will be *executed* multiple times. While stored procedure creation is relatively expensive, execution is minimal because the query is parsed and optimization paths are stored at create procedure time. Using SQLPrepare/SQLExecute for a statement that will be executed only once with such an ODBC driver will result in unneeded overhead. Furthermore, applications that use SQLPrepare/SQLExecute for large single execution query batches will almost certainly exhibit poor performance when used with ODBC drivers as previously discussed.

Similar arguments can be used to show applications that always use SQLExecDirect cannot perform as well as those that logically use a combination of SQLPrepare/SQLExecute and SQLExecDirect sequences.

Using SQLPrepare and Multiple SQLExecute Calls

Applications that use SQLPrepare and multiple SQLExecute calls should use SQLParamOptions if available. Passing arrays of parameter values reduces the ODBC call load and greatly reduces network traffic.

Consider the pseudo-code below for bulk inserting data:

```
rc = SQLPrepare (hstmt, “INSERT INTO DailyLedger (...)
    VALUES (?,?,...)”, SQL_NTS) ;
// bind parameters
...
do {
// read ledger values into bound parameter buffers
...
rc = SQLExecute (hstmt);      // insert row
} while ! (eof) ;
```

If there are 100 rows to insert then SQLExecute is called 100 times resulting in 100 network requests to the server. Consider, however, an algorithm that uses parameter arrays by calling SQLParamOptions:

```
rc = SQLPrepare (hstmt, “INSERT INTO DailyLedger (...)
    VALUES (?,?,...)”, SQL_NTS) ;
rc = SQLParamOptions (hstmt, (UDWORD) 50, &CurrentRow);
// pass 50 parameters per execute
// bind parameters
...
do {
// read up to 50 ledger values into bound parameter
// buffers
...
rc = SQLExecute (hstmt);      // insert row
```

The call load has been reduced from 100 executions to just two SQLExecute calls; furthermore, network traffic is reduced considerably. Some ODBC drivers do not support SQLParamOptions but many drivers do. To achieve high performance, applications should contain algorithms for using SQLParamOptions if the ODBC driver supports the function. SQLParamOptions is ideal for copying data into new tables or bulk loading tables.

Using the Cursor Library

Do not automatically use the cursor library if scrollable cursors are provided by the driver. The cursor library creates local temporary log files, which are expensive to generate and provide worse performance than using native scrollable cursors.

The cursor library adds support for static cursors, which simplifies the coding of applications that use scrollable cursors. However, the cursor library creates temporary log files on the user's local disk drive to accomplish the task. Disk i/o is typically one of the slowest operations on personal computers. While the benefits of the cursor library is great, applications should not automatically choose to use the cursor library if an ODBC driver supports scrollable cursors natively.

ODBC drivers that support scrollable cursors typically achieve high performance by requesting that the DBMS server produce a scrollable result set instead of emulating the capability by creating log files.

Many applications use

```
rc = SQLSetConnectOption (hdbc, SQL_ODBC_CURSORS,
    SQL_CUR_USE_ODBC)  ;
```

but should use

```
rc = SQLSetConnectOption (hdbc, SQL_ODBC_CURSORS,
    SQL_CUR_USE_IF_NEEDED)  ;
```

Design Options

Managing Connections

Connection management is extremely important to application performance. Designers should optimize applications by connecting once and using multiple statement handles instead of performing multiple connections.

Connection management is one of the most poorly designed elements of existing ODBC applications. Connecting to a data source is extremely expensive and should be avoided after establishing an initial connection.

Some ODBC applications are designed such that they call informational gathering routines that have no record of already attached connection handles. For example, some applications establish a connection and then call a routine in a separate DLL or shared library that re-attaches and gathers up front information about the driver to be used later in the application. While gathering driver information at connect time is a good algorithm, it should not be minimized by connecting twice to get this information. At least one popular ODBC enabled application connects a second time to gather driver information but *never* disconnects the second connection. Applications that are designed as separate entities should pass the already connected HDBC pointer to the data collection routine instead of establishing a second connection.

Another similarly poor algorithm is to connect and disconnect several times throughout your application to perform SQL statements. Connection handles can have multiple statement handles associated with them. Statement handles are defined to be memory storage for information about SQL statements. Why then do many applications allocate new connection handles to perform SQL statements? Applications should use *statement* handles to manage multiple SQL statements.

Connection and statement handling should not be delayed until implementation. Spending time in the design phase on connection management makes your application perform better and most certainly makes it more maintainable.

Committing Data

Committing data is extremely disk i/o intensive and thus slow. Always turn auto-commit off if the driver can support transactions.

What is actually involved in a commit? At commit time the DBMS server must flush back to disk every data page that contains updated or new data. Note that this is not a sequential write but a searched write to replace existing data already in the table. By default, auto-commit is on when connecting to a data source. Auto-commit mode is typically detrimental to performance because of the extreme amount of disk i/o needed to commit *every* operation.

To further reduce performance some DBMS servers do not provide an “auto-commit mode.” For this type of server, the ODBC driver must explicitly issue a COMMIT statement and perhaps a BEGIN TRANSACTION for *every* operation sent to the server. In addition to the large amount of disk i/o required to support auto-commit mode, we must also pay a performance penalty for up to three network requests for every statement issued by an application in this scenario.

Asynchronous Execution

Design your application to take advantage of data sources that support asynchronous execution. Asynchronous calls do not perform faster but well designed applications *appear* to run more efficiently.

By default, an application makes calls to an ODBC driver that then executes statements against the DBMS server in a synchronous manner. In this mode of operation the driver does not return control to the application until its own request to the server is complete. For statements which take more than a few seconds to complete execution, this can result in the perception of poor performance to the end user.

Some data sources support asynchronous execution. When in asynchronous mode, an application makes calls to an ODBC driver and control is returned almost immediately. In this mode the driver returns the status SQL_STILL_EXECUTING to the application and then sends the appropriate request to the database backend for execution. The application polls the driver at various intervals at which point the driver itself polls the server to see if the query has completed execution. If the query is still executing, then the status SQL_STILL_EXECUTING is returned to the application. If it has completed, then a status such as SQL_SUCCESS is returned, and the application can then begin to fetch records.

Turning on asynchronous execution does not by itself improve performance. Well designed applications, however, can take advantage of asynchronous query execution by allowing the end user to work on other things while the query is being evaluated on the server. Perhaps users will start one or more subsequent queries or choose to work in another application, all while the query is executing on the server. Designing for asynchronous execution makes your application *appear* to run faster by allowing the end user to work concurrently on multiple tasks.

Updating Data

Using Positional Updates and Deletes

Use positional updates and deletes or SQLSetPos whenever possible to update data.

Designing an efficient method for updating data is difficult. While positioned updates do not apply to all types of applications, developers should attempt to use positioned updates and deletes whenever possible. Positioned updates (either via “update where current of cursor” or via SQLSetPos) allow the developer to update data simply by positioning the database cursor to the appropriate row to be changed and signal the driver to “change the data here.” The designer is not forced to build a complex SQL statement but is simply required to supply the data that is to be changed.

Besides making the code more maintainable, positioned updates typically result in improved performance. Because the database server is already positioned on the row (for the Select statement currently in process), expensive operations to locate the row to be changed are not needed. If the row must be located then the server typically has an internal pointer to the row available (for example, ROWID).

Using SQLSpecialColumns

Use SQLSpecialColumns to determine the most optimal set of columns to use in the Where clause for updating data. Many times pseudo-columns provide the fastest access to the data, and these columns can only be determined by using SQLSpecialColumns.

Many applications cannot be designed to take advantage of positional updates and deletes. These applications typically update data by forming a Where clause consisting of some subset of the column values returned in the result set. Some applications may formulate the Where clause by using all searchable result columns or by calling SQLStatistics to find columns that may be part of a unique index. These methods typically work but may result in fairly complex queries.

Consider the following:

```
rc = SQLExecDirect (hstmt, “SELECT first_name, last_name,
    ssn, address, city, state, zip FROM emp”, SQL_NTS)     ;
// fetchdat a
...
rc = SQLExecDirect (hstmt, “UPDATE EMP SET ADDRESS = ?
     WHERE first_name = ? and last_name = ? and ssn = ?
     and address = ? and city = ? and state = ? and
     zip = ?”, SQL_NTS)  ;
// fairly complex quer  y
```

Applications should call SQLSpecialColumns/SQL_BEST_ROWID to retrieve the most optimal set of columns (possibly a pseudo-column) that identifies any given record. Many databases support special columns that are not explicitly defined by the user in the table definition but are “hidden” columns of every table (for example, ROWID, TID, etc.). These pseudo-columns almost always provide the fastest access to the data because they typically are pointers to the exact location of the record. Since pseudo-columns are not part of the explicit table definition they are not returned from SQLColumns. The only method of determining if pseudo-columns exist is to call SQLSpecialColumns.

Consider the previous example again:

```
...
rc = SQLSpecialColumns (hstmt, ..... ‘emp’, ...)    ;
...
rc = SQLExecDirect (hstmt, “SELECT first_name, last_name,
     ssn, address, city, state, zip, ROWID FROM emp”,
     SQL_NTS) ;
// fetch data and probably “hide” ROWID from the use     r
...
rc = SQLExecDirect (hstmt, “UPDATE emp SET address = ?
     WHERE
     ROWID = ?”, SQL_NTS)  ;
// fastest access to the data   !
```

If your data source does not contain special pseudo-columns, then the result set of SQLSpecialColumns consists of the columns of the most optimal unique index on the specified table (if a unique index exists); therefore, your application need not additionally call SQLStatistics to find the smallest unique index.

Go To

Index

Go To