DataDirect®
Connect ODBC™ UNIX Help

March 1999

CONTENTS

INDEX

Table of Contents

Preface

This help file provides information about MERANT™ DataDirect® Connect ODBC™ drivers on UNIX platforms.

Conventions Used in This Help File

This Help file uses the following typographical conventions:

Convention	Explanation
italics	Introduces new terms with which you may not be familiar, and is used occasionally for emphasis.
bold	Emphasizes important information. Also indicates button, menu, and icon names on which you can act. For example, click **Next**.
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.
	Also indicates keys or key combinations that you can use. For example, press the ENTER key.
`monospace`	Indicates syntax examples, values that you specify, or results that you receive.
`monospaced italics`	Indicates names that are placeholders for values that you specify. For example, *`filename`*.
forward slash /	Separates menus and their associated commands. For example, Select File / Copy means that you should select Copy from the File menu.
	The slash also separates directory levels when specifying locations under UNIX.
vertical rule |	Indicates an "OR" separator used to delineate items.

DataDirect Connect ODBC UNIX Help

Convention	Explanation
brackets [ ]	Indicates optional items. For example, in the following statement: SELECT [DISTINCT], DISTINCT is an optional keyword.
	Also indicates sections of the Windows Registry.
braces { }	Indicates that you must select one item. For example, {yes | no} means that you must specify either yes or no.
ellipsis . . .	Indicates that the immediately preceding item can be repeated any number of times in succession. An ellipsis following a closing bracket indicates that all information in that unit can be repeated.

Mouse Conventions

This action…	Means to…
Click	Point to an object with the mouse pointer and momentarily press the left mouse button.
Double-click	Press the left mouse button twice.
Right-click	Momentarily press the right mouse button.
Drag	Press and hold the left mouse button while dragging item(s) to another part of the screen.
SHIFT+Click	Click an object to select it; then, press and hold the SHIFT key. Click another object to select the intervening series of objects.
CTRL+Click	Press and hold the CTRL key; then, click a selection. This lets you select or deselect any combination of objects.

Keyboard Conventions

Select menu items by using the mouse or pressing ALT+ the key letter of the menu name or item.

Contacting Technical Support

MERANT provides technical support for all registered users of Connect ODBC, including limited installation support, for the first 30 days. For support after that time, contact us using one of the following methods or purchase further support by enrolling in the SupportNet program. For more information about SupportNet, contact your sales representative.

World Wide Web

http://www.merant.com/datadirect/support

The MERANT Web site provides the latest support information through SupportNet Online, our global service network that provides access to valuable tools and information. Our SupportNet users access information using the Web, automatic email notification, newsgroups, and regional user groups. SupportNet Online includes a knowledge base that allows you to search on keywords for technical bulletins and other information. You also can download product fixes for your DataDirect products.

Internet

Australia and New Zealand	australia.answerline@merant.com
EMEA	int.datadirect.answerline@merant.com
Japan	jpn.answerline@merant.co.jp
USA and Canada	datadirect.answerline@merant.com

Telephone

Australia	1 800 335 664 or 9816 9977 for Melbourne Metro	8:30-5:30 p.m. Local Melbourne Time (LMT)
Belgium	0800 724 61	9:00-6:30 p.m. CET
France	0800 91 56 07	9:00-6:30 p.m. CET

Germany	0130 822 496 or +44 1727 812898	9:00-6:30 p.m. CET
Japan	81-3-5401-9660	9:00-12:00, 1:00-5:00 p.m. JST
The Netherlands	0800 022 1609	9:00-6:30 p.m. CET
New Zealand	1 800 335 664	8:30-5:30 p.m. LMT
United Kingdom and Ireland	+44 1727 811881	8:00-5:30 p.m. GMT
USA and Canada	1 800 443 1601	8:30-8:00 p.m. EST

Fax and Mail Information

Fax US	1 919 461 4527
Fax International	+32-15-320919
Mail	1500 Perimeter Park Drive, Suite 100, Morrisville, NC 27560 USA

When you contact us, make sure that you can provide the following information:

■ The **product serial number** located on the Product Registration Information card or on a product serial number card in your package. The number will be checked to verify your support eligibility. If you do not have a SupportNet contract, we will ask you to speak with a sales representative.

■ Your **name and organization**. For a first-time call, you may be asked for full customer information, including location and contact details.

■ The **version number** of your DataDirect product.

■ The type and version of your **operating system**.

■ Any **third-party software or other environment information** required to understand the problem.

- A **brief description of the problem,** including any error messages that you have received, **and the steps preceding the occurrence of the problem**. Depending on the complexity of the problem, you may be asked to submit an example so that we can recreate the problem.

- An assessment of the **severity level** of the problem.

1   The UNIX Environment

The following are system requirements for using Connect ODBC drivers on AIX, HP-UX, and Solaris. See the *DataDirect Connect ODBC Reference* for further details.

AIX

- 40 MB of available hard disk space
- AIX 4.21 or 4.3 operating system

HP-UX aCC Enabled

- 25 MB of available hard disk space
- HP-UX 10.20 or higher operating system
- HP aCC (1.12 or higher) compiled application

HP-UX 11 aCC Enabled

- 25 MB of available hard disk space
- HP-UX 11.0 or higher operating system
- HP aCC (3.05 or higher) compiled application

HP-UX cFront Enabled

- 25 MB of available hard disk space
- HP-UX 10.10.36 or higher operating system
- HP cFront compiled application

Solaris

- 25 MB of available hard disk space
- Sun SPARCstation
- Solaris 2.51, 2.6, or 7 operating system
  (SunOS 5.51, 5.6, or 5.7)

The System Information File (.odbc.ini)

In the UNIX environment, there is no ODBC Administrator. To configure a data source, you must edit the system information file, a plain text file that is normally located in the user's $HOME directory and is usually called *.odbc.ini*. This file is maintained 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. A sample file (odbc.ini) is located in the driver installation directory.

UNIX support of the database drivers also permits the use of a centralized system information file that a system administrator can control. This is accomplished by setting the environment variable ODBCINI 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 ODBCINI /opt/odbc/system_odbc.ini
```

In the Bourne or Korn shell, you would set it as:

```
ODBCINI=/opt/odbc/system_odbc.ini;export ODBCINI
```

The search order for the location of the system information file file is as follows:

1   Check ODBCINI

2   Check $HOME for .odbc.ini

There must be an [ODBC] section in the system information file that includes the InstallDir keyword. The value of this keyword must be the path to the directory under which the /lib and /messages directories are contained. For example, if you choose the default install directory, then the following line must be in the [ODBC] section:

```
InstallDir=/opt/odbc
```

Sample System Information File for Solaris

In the following sample, *xx* represents the driver number:

```
[ODBC Data Sources]
Oracle7=Sample Oracle dsn
dBase=Sample dBASE dsn
Sybase=Sample Sybase dsn
Informix=Sample Infofrmix dsn
OpenIngres=Sample OpenIngres dsn
DB2=Sample DB2 dsn
Text=Sample Text file dsn

[dBase]
Driver=/opt/odbc/lib/ivdbfxx.so
Description=dBase
Database=/opt/odbc/demo

[Sybase]
Driver=/opt/odbc/lib/ivsybxx.so
Description=Sybase
Database=odbc
ServerName=SYBASE
LogonID=odbc01
Password=odbc01
OptimizePrepare=2
SelectMethod=1

[Oracle7]
Driver=/opt/odbc/lib/ivor7xx.so
Description=Oracle7
ServerName=oraclehost
LogonID=odbc01
Password=odbc01
```

```
[Informix]
Driver=/opt/odbc/lib/ivinfxx.so
Description=Informix7
Database=odbc
HostName=informixhost
LogonID=odbc01
Password=odbc01

[DB2]
Driver=/opt/odbc/lib/ivdb2xx.so
Description=DB2
Database=ODBC

[OpenIngres]
Driver=/opt/odbc/lib/ivoingxx.so
ServerName=ingreshost
Database=odbc
LogonID=odbc01
Password=odbc01

[Text]
Driver=/opt/odbc/lib/ivtxtxx.so
Description=Text driver
Database=/opt/odbc/demo
[ODBC]
Trace=0
TraceFile=odbctrace.out
TraceDll=/opt/odbc/lib/odbctrac.so
InstallDir=/opt/odbc
```

Environment Variables

Connect ODBC drivers require several environment variables to be set.

Required Environment Variables

Most of the varibles can be set by executing the appropriate shell script located in the ODBC home directory.

For example, C shell (and related shell) users should execute the following command before attempting to use ODBC-enabled applications:

```
% source odbc.csh
```

Bourne shell (and related shell) users should initialize their environment as follows:

```
$ . odbc.sh
```

Executing these scripts will set the appropriate library search path environment variable (LD_LIBRARY_PATH on Solaris, SHLIB_PATH on HP/UX, or LIBPATH on AIX).

The library search path environment variables are required to be set so that the ODBC core components and drivers can be located at the time of execution.

Optional Environment Variables

Many of the Connect ODBC drivers must have environment variables set as required by the database client components used by the drivers. Consult the system requirements in each of the individual driver sections for additional information pertaining to individual driver requirements.

ODBCINI is an optional environment variable that all Connect ODBC drivers will recognize. ODBCINI is used to locate a system information file other than the default file and is described in detail under "The System Information File (.odbc.ini)" on page 14.

Using Double-Byte Character Sets

Connect ODBC drivers are capable of using double-byte character sets. The drivers normally use the character set defined by the default locale "C" unless explicitly pointed to another character set. The default locale "C" corresponds to the 7-bit ASCII character set in which only characters from ISO 8859-1 are valid. Use the following procedure to set the locale to a different character set.

1 Add the following line at the very beginning of applications that use double-byte character sets:

```
setlocale (LC_ALL, "");
```

This is a standard UNIX function. It selects the character set indicated by the environment variable LANG as the one to be used by X/Open compliant character handling functions. If this line is not present, or if LANG is either not set or is set to NULL, the default locale "C" is used.

2 Set the LANG environment variable to the appropriate character set. The UNIX command `locale -a` can be used to display all supported character sets on your system.

For more information, see the man pages for "locale" and "setlocale."

The ivtestlib Tool

The ivtestlib tool is provided to help diagnose configuration problems (such as environment variables not correctly set or missing DBMS client components) in the UNIX environment. This command will attempt to load a specified ODBC driver and will print out all available error information if the load fails.

On HP-UX, for example, if a driver is installed in /opt/odbc, the command:

```
ivtestlib /opt/odbc/lib/ivinfxx.sl
```

(where *xx* represents the driver number) will attempt to load the Informix driver. If the driver cannot be loaded, ivtestlib will return an error message explaining why.

Note: On Solaris and AIX, the full path to the driver does not have to be specified for ivtestlib. The HP-UX version of ivtestlib, however, requires the full path.

Translators

DataDirect provides a sample translator named "OEM to ANSI" that provides a framework for coding a translation library.

You must add the TranslationSharedLibrary keyword to the data source section of the system information file to perform a translation. Adding the TranslationOption keyword is optional.

Keyword	Definition
TranslationSharedLibrary	Full path of translation library
TranslationOption	ASCII representation of the 32-bit integer translation option

2 Connect ODBC for DB2

System Requirements

The server requirement for all platforms is the same: The DB2 Server must be installed as the Server Version (*not* the Local Version).

To access the DB2 family of databases, you must have one of the following software packages:

AIX

■ IBM DB2 Client Application Enabler (CAE) for AIX, version 5.0 or higher
■ IBM DB2 Software Development Kit (DB2 SDK) for AIX, version 5.0 or higher

HP-UX

■ IBM DB2 Client Application Enabler (CAE) for HP-UX, version 2.12 or higher
■ IBM DB2 Software Development Kit (DB2 SDK) for HP-UX, version 2.12 or higher

Solaris

■ IBM DB2 Client Application Enabler (CAE) for Solaris, version 2.12 or higher
■ IBM DB2 Software Development Kit (DB2 SDK) for Solaris, version 2.12 or higher

DB2 Binding and Privileges

Access to DB2 requires that you bind and grant privileges to the DataDirect bind files, a process described in this section.

Enter the DB2 command line processor by typing db2 from a shell prompt. For example, from Windows NT, type:

```
c:\> db2
```

Note: The DB2 command processor prompt is **db2=>**.

Once inside the DB2 command line processor, the first step is to connect your DB2 database using the following syntax:

```
db2=> CONNECT TO <database_name> USER <userid> USING <password>
```

Binding

The next step is to bind the DataDirect SQL files to the database. You may choose to use special options on the BIND command, based on your installation. Consult the manual "Command Reference" in the DB2 manual set for a detailed list of BIND options. To bind the files, enter the commands listed in the following two sections. To exit the DB2 command processor, enter the verb quit.

AIX

```
db2=> BIND iscsax.bnd blocking all grant public
db2=> BIND isrrax.bnd blocking all grant public
db2=> BIND isurax.bnd blocking all grant public
db2=> BIND iscswhax.bnd blocking all grant public
db2=> BIND isrrwhax.bnd blocking all grant public
db2=> BIND isurwhax.bnd blocking all grant public
```

HP-UX

```
db2=> BIND iscshp.bnd blocking all grant public
db2=> BIND isrrhp.bnd blocking all grant public
db2=> BIND isurhp.bnd blocking all grant public
db2=> BIND iscswhhp.bnd blocking all grant public
db2=> BIND isrrwhhp.bnd blocking all grant public
db2=> BIND isurwhhp.bnd blocking all grant public
```

Solaris

```
db2=> BIND iscsso.bnd blocking all grant public
db2=> BIND isrrso.bnd blocking all grant public
db2=> BIND isurso.bnd blocking all grant public
db2=> BIND iscswhso.bnd blocking all grant public
db2=> BIND isrrwhso.bnd blocking all grant public
db2=> BIND isurwhso.bnd blocking all grant public
```

Configuring and Connecting to Data Sources

To configure a data source, you must use the following data source entries listed in Table 2-1. See the help chapter "The UNIX Environment" for details on configuring the system information file.

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 in the system information file 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 the system information. These values are not written to the system information.

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 is:

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

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 the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

Table 2-1. DB2 Connection String Attributes

Attribute	Description
ApplicationUsing Threads (AUT)	ApplicationUsingThreads={0 | 1}. Ensures that the driver works with multi-threaded applications. The default is 1, which makes the driver thread-safe. When using the driver with single-threaded applications, you may set this option to 0 to avoid additional processing required for ODBC thread safety standards.
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.
Database (DB)	The name of the database to which you want to connect.

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

Attribute	Description
DataSourceName (DSN)	A string that identifies a DB2 data source configuration in the system information. Examples include "Accounting" or "DB2-Serv1."
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 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 Common Server on UNIX, normal UNIX security is used. The LogonID value is your UNIX user ID.
Password (PWD)	Password.
Sysibm (SI)	On most DB2 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.

Data Types

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

Table 2-2. DB2 Data Types

DB2	ODBC
Char	SQL_CHAR
Char() for Bit Data	SQL_BINARY
Date	SQL_TYPE_DATE
Decimal	SQL_DECIMAL
Float	SQL_DOUBLE
Integer	SQL_INTEGER
Long Varchar	SQL_LONGVARCHAR
Long Varchar for Bit Data	SQL_LONGVARBINARY
Smallint	SQL_SMALLINT
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Varchar	SQL_VARCHAR
Varchar() for Bit Data	SQL_VARBINARY

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

Isolation and Lock Levels Supported

DB2 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," in the *Connect ODBC Reference* for a discussion of these topics.

ODBC Conformance Level

The DB2 driver supports the API functions listed in Appendix C, "ODBC API and Scalar Functions," in the *Connect ODBC Reference*. In addition, the following X/Open functions are supported:

- SQLProcedures
- SQLProcedureColumns

The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The DB2 database system supports a single connection and multiple statements per connection.

3 Connect ODBC for dBASE

System Requirements

The dBASE driver supports dBASE IV and V, and FoxPro 3.0 in the UNIX environment.

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

See the README file shipped with your MERANT DataDirect product for the file name of the dBASE driver.

Defining Index Attributes

Index files for dBASE contain index tags for each index that exists for a database file. These index tags can be marked as unique, that is, the driver will ensure that no duplicate values exist for the columns that define the index tag. The unique attribute is not natively supported by the dBASE or FoxPro products. The enforcement and recognition of the unique attribute is an extension of the MERANT dBASE driver. The driver must be notified that index tags are unique. No configuration is needed for unique indexes that were created using the MERANT dBASE driver. When using files that were not created with the MERANT dBASE driver, you must define unique index tags as outlined in the following procedure.

In the directory where the database and index files are located, use any plain text editor, such as vi, to define or edit the QEDBF.INI as follows:

1   Create a [`filename`] section where `filename` is the name of the database file. This entry is case sensitive and the file extension should be included.

2   In the [`filename`] section, specify the number of unique indexes on the file (`NUMUNIQUE=`) and the index specifications (`UNIQUE#=index_filename,index_tag`). The `index_tag` can be determined by calling the ODBC function SQLStatistics and examining the INDEX_NAME result column.

For example, to define two unique indexes on the accts.dbf table, the QEDBF.INI would be defined as:

```
[accts.dbf]
NUMUNIQUE=2
UNIQUE0=accts.mdx,ACCT_NAME
UNIQUE1=accts.mdx,ACCT_ID
```

Configuring and Connecting to Data Sources

To configure a data source, you must use the following data source entries listed in Table 3-1. See the help chapter "The UNIX Environment" for details on configuring the system information file.

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 in the system information file 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 the system information. These values are not written to the system information.

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 3-1 gives the long and short names for each attribute, as well as a description. The system information 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 the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

Table 3-1. dBASE Connection String Attributes

Attribute	Description
ApplicationUsing Threads (AUT)	ApplicationUsingThreads={0	1}. Ensures that the driver works with multi-threaded applications. The default is 1, which makes the driver thread-safe. When using the driver with single-threaded applications, you may set this option to 0 to avoid additional processing required for ODBC thread safety standards.
CacheSize (CSZ)	The number of 64KB blocks the driver uses to cache database records. The greater 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 run the Select statement again. The initial default is 4.

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

Attribute	Description
CreateType (CT)	CreateType={dBASE3 | dBASE4 | dBASE5 | Clipper | FoxPro25 | FoxPro30}. The type of table or index to be created on a Create Table or Create Index statement. The initial default is dBASE5.
Database (DB)	The directory in which the dBASE files are stored.
DataFileExtension (DFE)	String of three or fewer characters that specifies the file extension to use for data files. The initial default value is DBF. This value cannot be an extension the driver already uses, such as MDX or CDX. This value is used for all Create Table statements. Sending a Create Table using an extension other than the value specified for this attribute causes an error.
	In other SQL statements, such as Select or Insert, users can specify an extension other than the one specified for this attribute. The DataFileExtension value is used when no extension is specified.
DataSourceName (DSN)	A string that identifies a dBASE data source configuration in the system information. Examples include "Accounting" or "dBASE Files."
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.

Table 3-1. dBASE Connection String Attributes _(cont.)

Attribute	Description
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.
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=0 (the initial default), the driver uses the ASCII sort order. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."
	If IntlSort=1, the driver uses the international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.

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

Attribute	Description
LockCompatibility (LCOMP)	LockCompatibility={Clipper	dBASE	Fox	Q+E	Q+EVirtual}. The locking scheme to be used in your dBASE tables. 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=Clipper specifies Clipper-compatible locking.

- LockCompatibility=dBASE specifies Borland-compatible locking. This is the initial default for all platforms.

- LockCompatibility=Fox specifies FoxPro-compatible locking.

- 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.

Table 3-1. 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. If you are accessing a table with two applications, however, 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.
	Locking=RECORD locks only the records affected by the statement. This is the initial default for all platforms.
	Locking=FILE locks all of the records in the table.

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

Attribute	Description
ModifySQL (MS)	ModifySQL={0	1}. This attribute is provided for backward compatibility with earlier versions of MERANT 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.
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.

Data Types

Table 3-2 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," in the *Connect ODBC Reference*.

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 3-2. dBASE Data Types

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

[1] dBASE V only

[2] 254 characters maximum

[3] dBASE IV and V only

[4]FoxPro and dBASE V only

Column Names

Column names in SQL statements (such as, Select, Insert, etc.) can be up to ten 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," in the *Connect ODBC Reference*. This section describes the information that is specific to dBASE, which is Rowid.

Rowid Pseudo-Column

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
```

Alter Table Statement

The dBASE driver supports the Alter Table statement to add one or more columns to a table or to delete (drop) a single column.

The Alter Table statement has the form:

```
ALTER TABLE table_name {ADD column_name data_type
| ADD
(column_name data_type [, column_name data_type]
... ) |
DROP [COLUMN] column_name}
```

table_name is the name of the table to which you are adding or dropping columns.

column_name assigns a name to the column you are adding or specifies the column you are dropping.

data_type specifies the native data type of each column you add.

For example, to add two columns to the emp table,

```
ALTER TABLE emp (ADD startdate date, dept char
(10))
```

You cannot add columns and drop columns in a single statement, and you can drop only one column at a time. For example, to drop a column,

```
ALTER TABLE emp DROP startdate
```

The Alter Table statement fails if you attempt to drop a column upon which other objects, such as indexes or views, are dependent.

Create Index Statement

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

■ dBASE (.MDX)
■ 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 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 index files are named *base_table_name.*MDX and FoxPro 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 3-3 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 3-3. dBASE-Compatible Index Summary

Create Type .Extension	dBASE UNIQUE	DESC	Max Size of Key Column	Max Size of Column Specification	Production/ Structural Indexes	Supports FOR Expressions
dBASE IV, V .MDX	Yes	Yes	100	220	Yes	Yes
FoxPro .IDX*	Yes	Yes	240	255	No	Yes
FoxPro .CDX	Yes	Yes	240	255	Yes	Yes

*   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.

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=).

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.

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).

How Transactions Affect Record Locks

When an Update or Delete statement is run, 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 run.

When a Select...For Update statement is run, 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," in the *Connect ODBC Reference* for a discussion of these topics.

ODBC Conformance Level

The dBASE driver supports the API functions listed in Appendix C, "ODBC API and Scalar Functions," in the *Connect ODBC Reference*. In addition, the following function is supported: SQLSetPos.

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

Number of Connections and Statements Supported

dBASE supports multiple connections and multiple statements per connection.

4 Connect ODBC for Informix

System Requirements

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

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

```
setenv INFORMIXDIR /databases/informix
```

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

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

In addition, the INFORMIXSERVER variable must be set to the name of the Informix server (as defined in your $INFORMIXDIR/ext/sqlhosts file). For further details, refer to the Informix documentation.

Informix

To access remote Informix 5.*x*, 6.*x*, or 7.*x* databases through the Informix driver, you need INFORMIX-Client Software Developer's Kit for UNIX platforms, version 2.*x*, from Informix.

Note: The DataDirect Informix driver does not work with versions earlier than 7.23 of the previous products, INFORMIX-Connect and ESQL/C .

Informix 9

To access remote Informix 7.*x* or 9 databases through the Informix 9 driver, you need INFORMIX-Client Software Developer's Kit for UNIX platforms, version 2.*x*, from Informix.

Note: The DataDirect Informix 9 driver for Solaris and HP-UX does not work with versions earlier than 9.1.3 of the previous products, INFORMIX-Connect and ESQL/C. The driver for AIX does not work with versions earlier than 9.1.4 of INFORMIX-Connect and ESQL/C.

Configuring and Connecting to Data Sources

To configure a data source, you must use the following data source entries listed in Table 4-1. See the help chapter "The UNIX Environment" for details on configuring the system information file.

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 in the system information file 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 the system information. These values are not written to the system information.

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 4-1 gives the long and short names for each attribute, as well as a description. The system information 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 the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

Table 4-1. Informix Connection String Attributes

Attribute	Description
ApplicationUsing Threads (AUT)	ApplicationUsingThreads={0 | 1}. Ensures that the driver works with multi-threaded applications. The default is 1, which makes the driver thread-safe. When using the driver with single-threaded applications, you may set this option to 0 to avoid additional processing required for ODBC thread safety standards.
CancelDetect Interval (CDI)	(Valid for all platforms except OS/2.) Lets you cancel long-running queries in threaded applications. Select a value to determine how often the driver checks whether a request has been canceled using SQLCancel. For example, if CDI=5, then for every pending request, the driver checks every five seconds to see whether the user has canceled execution of the query using SQLCancel. The default is 0, which means that requests will not be canceled until a request has completed execution.
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.
Database (DB)	The name of the database to which you want to connect.

Table 4-1. Informix Connection String Attributes (cont.)

Attribute	Description
DataSourceName (DSN)	A string that identifies an Informix data source configuration in the system information. Examples include "Accounting" or "INFORMIX-Serv1."
EnableInsert Cursors (EIC)	EnableInsertCursors={0	1}. Determines whether the driver can use Insert cursors during inserts governed by parameters. The initial default value is 1 (driver uses Insert cursors). Using Insert cursors improves performance during multiple Insert operations using the same statement. This option enables insert data to be buffered in memory before being written to disk. When EnableInsertCursors=0, the driver does not use Insert cursors.
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. |
LogonID (UID)	Your user name as specified on the Informix server.
Password (PWD)	A password.
ServerName (SRVR)	The name of the server running the Informix database.
TrimBlankFrom IndexName (TBFIN)	TrimBlankFromIndexName={0	1}. Specifies whether or not the leading space should be trimmed from a system-generated index name. This option is provided to address problems with applications that cannot process a leading space in index names. When set to 1 (the default), the driver trims the leading space. When set to 0, the driver does not trim the space.

Data Types

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

Table 4-2. Informix Data Types

Informix 5 and 7	ODBC
Byte[1]	SQL_LONGVARBINARY
Char	SQL_CHAR
Date	SQL_TYPE_DATE
Datetime year to fraction(5)	SQL_TYPE_TIMESTAMP
Datetime year to fraction(f)[2]	SQL_TYPE_TIMESTAMP
Datetime year to second	SQL_TYPE_TIMESTAMP
Datetime year to day	SQL_TYPE_DATE
Datetime hour to second	SQL_TYPE_TIME
Datetime hour to fraction(f)[2]	SQL_TYPE_TIME
Decimal	SQL_DECIMAL
Float	SQL_DOUBLE
Integer	SQL_INTEGER
Interval year(p) to year	SQL_INTERVAL_YEAR
Interval year(p) to month	SQL_INTERVAL_YEAR_TO_MONTH
Interval month(p) to month	SQL_INTERVAL_MONTH
Interval day(p) to day	SQL_INTERVAL_DAY
Interval day(p) to hour	SQL_INTERVAL_DAY_TO_HOUR
Interval day(p) to minute	SQL_INTERVAL_DAY_TO_MINUTE
Interval day(p) to second	SQL_INTERVAL_DAY_TO_SECOND
Interval day(p) to fraction(f)[2]	SQL_INTERVAL_DAY_TO_SECOND
Interval hour(p) to hour	SQL_INTERVAL_HOUR
Interval hour(p) to minute	SQL_INTERVAL_HOUR_TO_MINUTE

[1]Not supported for Standard Engine Databases

[2]Fraction(f) types are mapped to fraction(5) in the driver. The precision is type dependent and the scale is 5.

Table 4-2. Informix Data Types *(cont.)*

Informix 5 and 7	ODBC
Interval hour(p) to second	SQL_INTERVAL_HOUR_TO_SECOND
Interval hour(p) to fraction(f)[2]	SQL_INTERVAL_HOUR_TO_SECOND
Interval minute(p) to minute	SQL_INTERVAL_MINUTE
Interval minute(p) to second	SQL_INTERVAL_MINUTE_TO_SECOND
Interval minute(p) to fraction(f)[1]	SQL_INTERVAL_MINUTE_TO_SECOND
Interval second(p) to second	SQL_INTERVAL_SECOND
Interval second(p) to fraction(f)[1]	SQL_INTERVAL_SECOND
Interval fraction to fraction(f)[2]	SQL_VARCHAR
Money	SQL_DECIMAL
Serial	SQL_INTEGER
Smallfloat	SQL_REAL
Smallint	SQL_SMALLINT
Text[1]	SQL_LONGVARCHAR
Varchar[1]	SQL_VARCHAR

[1]Not supported for Standard Engine Databases

[2]Fraction(f) types are mapped to fraction(5) in the driver. The precision is type dependent and the scale is 5.

Informix 9

Table 4-3 shows how the Informix 9 data types map to the standard ODBC data types. These types are in addition to the Informix data types described in Table 4-2.

Table 4-3. Informix 9 Data Types

Informix 9	ODBC
Blob	SQL_LONGVARBINARY
Boolean	SQL_BIT
Clob	SQL_LONGVARCHAR
Int8	SQL_BIGINT
Lvarchar	SQL_VARCHAR
Serial8	SQL_BIGINT

The Informix 9 driver does not support any complex data types (for example, set, multiset, list, and named/unnamed abstract types). When the driver encounters a complex type it will return an Unknown Data Type error (SQL State HY000).

Isolation and Lock Levels Supported

If connected to an Online Server, Informix 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 also supports an alternative isolation level 1, called "cursor stability." Your ODBC application can use this isolation level by calling SQLSetConnectAttr (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 supports page-level and row-level locking.

See Appendix D, "Locking and Isolation Levels," in the *Connect ODBC Reference* for a discussion of these topics.

ODBC Conformance Level

The Informix driver supports the API functions listed in Appendix C, "ODBC API and Scalar Functions," in the *Connect ODBC Reference*. In addition, the following functions are supported:

- SQLProcedures
- SQLColumnPrivileges
- SQLTablePrivileges
- SQLPrimaryKeys
- SQLForeignKeys
- SQLProcedureColumns

The driver also supports scrollable cursors with SQLFetchScroll or SQLExtendedFetch. The driver supports the core SQL grammar.

Number of Connections and Statements Supported

The Informix drivers support multiple connections and multiple statements per connection to the Informix database system.

5   Connect ODBC for OpenIngres

System Requirements

The following section lists requirements for both OpenIngres and OpenIngres 2.

OpenIngres

To access OpenIngres databases from your client workstation you must have the OpenIngres/Net Release 1.2 product (int.wnt/03 or higher) installed on your client node.

OpenIngres 2

To access OpenIngres 2 databases from your client workstation you must have CA OpenIngres Net version 2.0 or greater installed on your client node.

OpenIngres and OpenIngres 2

The remote or host OpenIngres databases must be INGRES 1.2 or later.

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 /databases
```

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

```
II_SYSTEM=/databases;export II_SYSTEM
```

Configuring and Connecting to Data Sources

To configure a data source, you must use the following data source entries listed in Table 5-1. See the help chapter "The UNIX Environment" for details on configuring the system information file.

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 in the system information file 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 the system information. These values are not written to the system information.

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 OpenIngres is:

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

Table 5-1 gives the long and short names for each attribute, as well as a description. The system information 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 the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

Table 5-1. OpenIngres Connection String Attributes

Attribute	Description
ApplicationUsing Threads (AUT)	ApplicationUsingThreads={0 | 1}. Ensures that the driver works with multi-threaded applications. The default is 1, which makes the driver thread-safe. When using the driver with single-threaded applications, you may set this option to 0 to avoid additional processing required for ODBC thread safety standards.
CenturyBoundary (CB) *OpenIngres 1.2 only*	CenturyBoundary=20. Century Boundary specifies the cutoff year for century inference when converting two-digit dates to four-digit dates. Two-digit dates that are less than the specified year number will be converted to 20xx. Two-digit dates greater than or equal to the number will be converted to 19xx. The default value is 20. For example, using the default value, a date of 19 will be interpreted as 2019 and a date of 21 will be interpreted as 1921. This option is necessary only when the OpenIngres 1.2 environment variable, II_DATE_FORMAT, which specifies date format, is set to convert years to two-digit numbers.
Database (DB)	The name of the database to which you want to connect.
DataSourceName (DSN)	A string that identifies an OpenIngres data source configuration in the system information. Examples include "Accounting" or "INGRES-Serv1."

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

Attribute	Description
DefaultLongData BuffLen (DLDBL)	An integer value that specifies, in 1024-byte multiples, the maximum amount of data that will be transferred to the client for unbound long data result columns. The default is 1024 (DefaultLongDataBuffLen=1024); that is, 1024 * 1024 = 1 MB.
EnableSelectLoop	EnableSelectLoop={0 | 1}. This attribute enables the retrieval of multiple rows using the select loop model instead of cursors. The default is 0; that is, use cursors. Specify 1 to use select loops.
LockMode (LM) *OpenIngres 2.0 only*	LockMode={0 | 1 | 2}. Allows you to select row, page, or table locking. Options are Row (0), Page (1), or Table (2). The default is page locking (1).
LogonID (UID)	The default logon ID (user name) used to connect to your OpenIngres 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 OpenIngres SQL command line. Some examples are ■ -l (locks the database exclusively) ■ -u (logs on as username) ■ +w or -w (waits/does-not-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 5-1. OpenIngres Connection String Attributes (cont.)

Attribute	Description
RepeatedCache Size (RCS)	An integer value that determines whether all Update and Insert statements are to be run 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 OpenIngres Repeated syntax.
RepeatedSelects (RS)	RepeatedSelects={0 | 1 | 2}. This attribute determines whether the driver optimizes Select statements or runs them as repeated queries. When set to 0, the initial default, the driver runs 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 runs all Select statements as repeated queries. If this attribute is set to 1 or 2, the RepeatedCacheSize attribute must be set to greater than 0.
	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
ServerName (SRVR)	The name of the virtual node that you defined using the OpenIngres NETU utility. This virtual node tells OpenIngres which system to call, how to call it, and the user's name and password.
SQLGrammar (SG)	SQLGrammar={0 | 1}. Provides the ability to access data sources using OpenSQL. The default is 0; that is, INGRES SQL.

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

Attribute	Description
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 OpenIngres 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 5-2 shows how OpenIngres data types map to the standard ODBC data types.

Table 5-2. OpenIngres Data Types

OpenIngres	ODBC
Byte*	SQL_BINARY
Byte varying*	SQL_VARBINARY
Char	SQL_CHAR
Date	SQL_TYPE_TIMESTAMP
Float	SQL_DOUBLE
Float4	SQL_REAL
Integer	SQL_INTEGER
Integer1	SQL_TINYINT
Long byte*	SQL_LONGVARBINARY
Long varchar*	SQL_LONGVARCHAR
Money*	SQL_DECIMAL
Smallint	SQL_SMALLINT
Varchar	SQL_VARCHAR

* Not supported by OpenSQL.

Note: OpenIngres Date values do not directly map to the ODBC type SQL_TYPE_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

OpenIngres supports isolation level 1 (read committed, the default). OpenIngres supports page-level locking. See Appendix D, "Locking and Isolation Levels," in the *Connect ODBC Reference* for a discussion of these topics

ODBC Conformance Level

The OpenIngres drivers support the API functions listed in Appendix C, "ODBC API and Scalar Functions," in the *Connect ODBC Reference*. SQLProcedures and SQLProcedureColumns are supported unless SQLGrammar=1.

The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The OpenIngres 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.

6 Connect ODBC for Oracle

System Requirements

Both Oracle and Oracle8 client information for UNIX is listed below.

Important: You must have *all* components of the Oracle client software installed; otherwise, the driver will not operate properly. See the DataDirect README for a component list.

Oracle and Oracle8

Before you can use the Oracle data source, you must have the Oracle SQL*Net or Net8 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, SQL*Net, or Net8 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 Oracle7 SQL*Net Shared Library

The Oracle driver requires a one-time site linking to build an Oracle7 SQL*Net driver on AIX and, for Oracle7.1 only, on Solaris and HP-UX. This site linking binds your unique Oracle7 SQL*Net configuration into the file, which is used by the Oracle driver to access local and remote Oracle databases.

Before you build the Oracle7 SQL*Net shared library, install Oracle and set the environment variable ORACLE_HOME to the directory where you installed Oracle. Connect ODBC provides a script, genclntsh, that builds the Oracle7 SQL*Net driver. This script is in the scr/oracle directory.

The following command builds the Oracle7 SQL*Net shared library:

```
genclntsh
```

Building the Required Oracle Net8 Shared Library on Solaris

Under Oracle 8.0.3 or 8.0.4 for Solaris, the Oracle8 driver requires a one-time site linking to build a replacement Oracle Net8 driver. This site linking binds your unique Oracle Net8 configuration into the file, which is used by the Oracle driver to access local and remote Oracle databases.

The Oracle8 driver requires the shared library libclntsh.so, which is built by the Oracle script genclntsh. The genclntsh script provided by Oracle causes an error resulting from the undefined symbol slpmprodstab. Oracle8 users must therefore use the genclntsh8 script provided with Connect ODBC to build a replacement libclntsh.so. This script, in the scr/oracle directory, places the new libclntsh.so in ../../lib, which is your $ODBC_HOME/lib directory; it does not overwrite the original libclntsh.so in the $ORACLE_HOME/lib directory.

Before you build the Oracle Net8 shared library, install Oracle and set the environment variable ORACLE_HOME to the directory where you installed Oracle.

The following command builds the Oracle Net8 shared library:

```
genclntsh8
```

Warning: Oracle8 users will have the original libclntsh.so library in the $ORACLE_HOME/lib directory. Therefore, the $ODBC_HOME/lib directory, containing the correct library, *must* be on the LD_LIBRARY_PATH *before* $ORACLE_HOME/lib. Otherwise, the original Oracle library will be loaded, resulting in the unresolved symbol error.

Configuring and Connecting to Data Sources

To configure a data source, you must use the following data source entries listed in Table 6-1. See the help chapter "The UNIX Environment" for details on configuring the system information file.

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 in the system information file 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 the system information. These values are not written to the system information.

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 Oracle is:

```
DSN=Accounting;SRVR=X:QESRVR;UID=JOHN;PWD=XYZZY
```

If the server name contains a semicolon, enclose it in quotation marks:

```
DSN=Accounting;SRVR="X:QE;SRVR";UID=JOHN;PWD=XYZZY
```

Table 6-1 gives the long and short names for each attribute, as well as a description. The system information 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 the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

Connecting via Operating System Parameters

Oracle has a feature that allows you to connect to Oracle via the operating system user name and password. To connect, use a slash (/) for the user name and leave the password blank. To configure the Oracle server/client, refer to the Oracle server documentation.

Table 6-1. Oracle Connection String Attributes

Attribute	Description
ApplicationUsing Threads (AUT)	ApplicationUsingThreads={0 | 1}. Ensures that the driver works with multi-threaded applications. The default is 1, which makes the driver thread-safe. When using the driver with single-threaded applications, you may set this option to 0 to avoid additional processing required for ODBC thread-safety standards. When you specify ApplicationUsingThreads=1, SQLGetInfo(SQL_ASYNC_MODE) returns SQL_AM_NONE, SQLSetConnectAttr(SQL_ATTR_ASYNC_ENABLE) returns "optional feature not implemented," and SQLSet/GetStmtAttr(SQL_ATTR_ASYNC_ENABLE) returns "optional feature not implemented." Asynchronous execution is not supported by the Oracle client in a multi-threaded environment.
ArraySize (AS)	The number of bytes the driver uses for fetching multiple rows. Values can be an integer from 1 up to 4 GB. The initial default is 60,000. 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.
CatalogOptions (CO)	CatalogOptions={0 | 1}. Specifies whether the result column REMARKS for the catalog functions SQLTables and SQLColumns and COLUMN_DEF for the catalog function SQLColumns have meaning for Oracle. If you want to obtain the actual default value, set CO=1. The default is 0.
DataSourceName (DSN)	A string that identifies an Oracle data source configuration in the system information. Examples include "Accounting" or "Oracle-Serv1."

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

Attribute	Description
EnableDescribe Param (EDP)	EnableDescribeParam={0 | 1}. Enables the ODBC API function SQLDescribeParam, which results in all parameters being described with a data type of SQL_VARCHAR. This option should be set to 1 when using Microsoft Remote Data Objects (RDO) to access data. The default is 0.
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 (do not wait).
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. To use your operating system user name, see "Connecting via Operating System Parameters" on page 66.
PacketSize (PS) **Oracle7 Only**	PacketSize={1024 | 2048 | 4096 | 8192}. A value that controls the packet size for TCP/IP connections. Any values other than 1024, 2048, 4096, or 8192 are ignored. This value is used only when the ServerName attribute (described above) is set to T for TCP/IP.
Password (PWD)	The password that the application uses to connect to your Oracle database. To use your operating system password, see "Connecting via Operating System Parameters" on page 66.
ProcedureRet Results (PRR)	ProcedureRetResults={0 | 1}. Values are Off (0) and On (1). The default is 0. When the option is on, the driver will return result sets from stored procedures/functions. If this option is on and you execute a stored procedure that does not return result sets, you will incur a small performance penalty. See "Stored Procedure Results" on page 71.

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

Attribute	Description
ServerName (SRVR)	The client connection string designating the server and database to be accessed. The information required varies depending on the client driver that you are using.

For Oracle7Oracle8 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.

Oracle8

For Oracle8 remote servers, the Net8 Client connection string has the following form:

`TNSNAME`

TNSNAME is the alias name of the Oracle Listener on your network.

If the Net8 Client connection string contains semicolons, enclose it in quotation marks. See your Net8 Client documentation for more information.

Oracle Data Types

Table 6-2 shows how the Oracle data types are mapped to the standard ODBC data types.

Table 6-2. Oracle Data Types

Oracle	ODBC
Char	SQL_CHAR
Date	SQL_TYPE_TIMESTAMP
Long	SQL_LONGVARCHAR
Long Raw	SQL_LONGVARBINARY
Number	SQL_DOUBLE
Number(p,s)	SQL_DECIMAL
Raw	SQL_VARBINARY
Varchar2	SQL_VARCHAR

Oracle8

Table 6-3 shows how the Oracle8 data types are mapped to the standard ODBC data types. These are in addition to the Oracle data types described above.

Table 6-3. Oracle8 Data Types

Oracle8	ODBC
Bfile	SQL_LONGVARBINARY*
Blob	SQL_LONGVARBINARY
Clob	SQL_LONGVARCHAR

* Read-Only

The Oracle8 driver does not support any Abstract Data Types. When the driver encounters an Abstract Data Type during data retrieval, it will return an Unknown Data Type error (SQL State HY000). It also does not support asynchronous operations, due to constraints in the current Oracle8 client.

Stored Procedure Results

When the option Procedure Returns Results is active, the driver returns result sets from stored procedures/functions. In addition, SQLGetInfo(SQL_MULT_RESULTS_SETS) will return "Y" and SQLGetInfo(SQL_BATCH_SUPPORT) will return SQL_BS_SELECT_PROC. If this option is on and you execute a stored procedure that does not return result sets, you will incur a small performance penalty.

This feature requires that stored procedures be in a certain format. First, a package must be created to define all of the cursors used in the procedure, then the procedure can be created using the new cursor. For example:

```
Create or replace package GEN_PACKAGE as
  CURSOR G1 is select CHARCOL from GTABLE2;
  type GTABLE2CHARCOL is ref cursor return G1%rowtype;
end GEN_PACKAGE;

Create or replace procedure GEN_PROCEDURE1 (
  rset IN OUT GEN_PACKAGE.GTABLE2
  CHARCOL, icol INTEGER) as
begin
  open rset for select CHARCOL from GTABLE2
    where INTEGERCOL <= icol order by INTEGERCOL;
end;
```

When executing the stored procedures with resultsets, do not include the resultset arguments in the list of procedure arguments. The previously described example would be executed as:

```
{call GEN_PROCEDURE1 (?)}
```

where ? is the parameter for the icol argument.

For more information consult your Oracle SQL manual.

Isolation and Lock Levels Supported

Oracle supports isolation level 1 (read committed) and isolation level 3 (serializable isolation—if the server version is Oracle7.3 or greater or Oracle8.*x*). Oracle supports record-level locking.

See Appendix D, "Locking and Isolation Levels," in the *Connect ODBC Reference* for a discussion of these topics.

ODBC Conformance Level

The Oracle drivers support the API functions listed in Appendix C, "ODBC API and Scalar Functions," in the *Connect ODBC Reference*. They support SQLSetPos as well as scrollable cursors with SQLFetchScroll and SQLExtendedFetch. The drivers also support SQLDescribeParam if EnableDescribeParam=1.

The Oracle driver supports the following functions:

■ SQLProcedures
■ SQLProcedureColumns
■ SQLPrimaryKeys
■ SQLForeignKeys
■ SQLTablePrivileges
■ SQLColumnPrivileges

The driver supports the core SQL grammar.

Number of Connections and Statements Supported

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

7 Connect ODBC for SQL Server

System Requirements

The SQL Server driver supports only SQL Server 6.5 functionality. To use the SQL Server driver, you must have TCP/IP configured on both the UNIX client and the Windows NT server on which the SQL Server database resides. The UNIX SQL Server TCP-IP network client library is built into the UNIX SQL Server ODBC driver. Nothing else is required beyond the normal ODBC environment.

The SQL Server Client configuration has been merged with the ODBC driver configuration and is set in the system information file. Two additional system information settings file settings are required for the SQL Server Client: ServerIPAddress and ServerPortNumber. See "Configuring and Connecting to Data Sources" on page 75 for details on these settings.

Configuring and Connecting to Data Sources

To configure a data source, you must use the following data source entries listed in Table 7-1. See the help chapter "The UNIX Environment" for details on configuring the system information file.

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 in the system information file 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 the system information. These values are not written to the system information.

Note: The UNIX SQL Server driver supports only SQL Server 6.5 functionality.

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;DATABASE=PAYROLL;UID=JOHN;PWD=XYZZY
```

Table 7-1 gives the name 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 the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

Table 7-1. SQL Server Connection String Attributes

Attribute	Description
Address	Network address of the server running SQL Server. Used only if the Server keyword does not specify the network name of a server running SQL Server. Address is usually the network name of the server, but can be other names such as a pipe, or a TCP/IP port and socket address. **Note:** Address is not valid for the system information file.
AnsiNPW	AnsiNPW={yes | no}. When yes, the driver uses ANSI-defined behaviors for handling NULL comparisons, character data padding, warnings, and NULL concatenation. When no, ANSI defined behaviors are not exposed.

Table 7-1. SQL Server Connection String Attributes (cont.)

Attribute	Description
DATABASE	Name of the default SQL Server database for the connection. If Database is not specified, the default database defined for the login is used. The default database from the ODBC data source overrides the default database defined for the login. The database must be an existing database unless AttachDBFileName is also specified. If AttachDBFileName is specified, the primary file it points to is attached and given the database name specified by DATABASE.
PWD	The password for the SQL Server login account specified in the UID parameter. PWD need not be specified if the login has a NULL password or when using Windows NT authentication (Trusted_Connection = yes).
QuotedID	QuotedID={yes | no}. When yes, QUOTED_IDENTIFIERS is set ON for the connection, and SQL Server uses the SQL-92 rules regarding the use of quotation marks in SQL statements.
	When no, QUOTED_IDENTIFIERS is set OFF for the connection, and SQL Server follows the legacy Transact-SQL rules regarding the use of quotation marks in SQL statements.
ServerIPAddress	The Network TCP/IP address for the Windows NT Server on which the SQL Server database is running.
ServerPortNumber	The Network TCP/IP port on which the SQL Server database is listening for connections.
UID	A valid SQL Server login account. UID need not be specified when using Windows NT authentication.
	Note: UID is not valid for the system information file; you must use LogonID.

Table 7-1. SQL Server Connection String Attributes (cont.)

Attribute	Description
UseProcForPrepare	UseProcForPrepare={0 | 1 | 2}. When 0, the SQL Server ODBC driver does not create temporary stored procedures for SQLPrepare.
	When 1, instructs the SQL Server ODBC driver to create temporary stored procedures when statements are prepared with SQLPrepare. The temporary stored procedures are not dropped until the connection is broken.
	When 2, the SQL Server ODBC driver creates temporary stored procedures for SQLPrepare, but only one procedure is created per statement handle and the procedure is dropped when the statement handle becomes invalid or a new SQL statement is prepared.
UID	A valid SQL Server login account. UID need not be specified when using Windows NT authentication.
	Note: UID is not valid for the system information file; you must use LogonID.
UseProcForPrepare	UseProcForPrepare={0 | 1 | 2}. When 0, the SQL Server ODBC driver does not create temporary stored procedures for SQLPrepare.
	When 1, instructs the SQL Server ODBC driver to create temporary stored procedures when statements are prepared with SQLPrepare. The temporary stored procedures are not dropped until the connection is broken.
	When 2, the SQL Server ODBC driver creates temporary stored procedures for SQLPrepare, but only one procedure is created per statement handle and the procedure is dropped when the statement handle becomes invalid or a new SQL statement is prepared.

Data Types

Table 7-2 shows how the SQL Server data types are mapped to the standard ODBC data types.

Table 7-2. SQL Server Data Types

SQL Server	ODBC
Binary	SQL_BINARY
Bit	SQL_BIT
Char	SQL_CHAR
Datetime	SQL_TYPE_TIMESTAMP
Decimal	SQL_DECIMAL
Decimal() identity	SQL_DECIMAL
Float	SQL_FLOAT
Image	SQL_LONGVARBINARY
Int	SQL_INTEGER
Int identify	SQL_INTEGER
Money	SQL_DECIMAL
Nchar*	SQL_WCHAR
Ntext*	SQL_WLONGVARCHAR
Numeric	SQL_NUMERIC
Numeric() identity	SQL_NUMERIC
Nvarchar*	SQL_WVARCHAR
Real	SQL_REAL
Smalldatetime	SQL_TYPE_TIMESTAMP
Smallint	SQL_SMALLINT
Smallint identity	SQL_SMALLINT
Smallmoney	SQL_DECIMAL
Sysname	SQL_WVARCHAR
Text	SQL_LONGVARCHAR
Timestamp	SQL_VARBINARY

Table 7-2. SQL Server Data Types (cont.)

SQL Server	ODBC
Tinyint	SQL_TINYINT
Tinyint identity	SQL_TINYINT
Uniqueidentifier*	SQL_GUID
Varbinary	SQL_VARBINARY
Varchar	SQL_VARCHAR

* Supported by SQL Server 7 only.

Isolation and Lock Levels Supported

SQL Server supports isolation levels 0 (read uncommitted), 1 (read committed), 2 (repeatable read), and 3 (serializable). SQL Server supports row- and table-level locking.

See Appendix D, "Locking and Isolation Levels," in the *Connect ODBC Reference* for a discussion of these topics.

ODBC Conformance Level

The SQL driver supports ODBC conformance level 2.

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.

8   Connect ODBC for Sybase

System Requirements

Before you can use the System 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 client. 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 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_PATH
```

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

```
LD_LIBRARY_PATH=/databases/sybase/lib:$LD_LIBRARY_
PATH;export LD_LIBRARY_PATH
```

In non-DCE environments, users should use the ivsyb*xx* Sybase driver that requires the library libct. For DCE environments, users should use the ivsyb11*xx* Sybase driver that requires the Sybase 11.1 client library libct_r.

Configuring and Connecting to Data Sources

To configure a data source, you must use the following data source entries listed in Table 8-1. See the help chapter "The UNIX Environment" for details on configuring the system information file.

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 in the system information file 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 the system information. These values are not written to the system information.

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 is:

```
DSN=SYS10 TABLES;SRVR=QESRVR;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

Table 8-1 gives the long and short names for each attribute, as well as a description. The system information 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 the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

Table 8-1. Sybase Connection String Attributes

Attribute	Description
ApplicationName (APP)	The name used by Sybase to identify your application.
ApplicationUsing Threads (AUT)	ApplicationUsingThreads={0	1}. Ensures that the driver works with multi-threaded applications. The default is 1, which makes the driver thread-safe. When using the driver with single-threaded applications, you may set this option to 0 to avoid additional processing required for ODBC thread safety standards.
ArraySize (AS)	The number of rows the driver retrieves from the server for a fetch. This is not the number of rows given to the user. This increases performance by reducing network traffic. The initial default is 50 rows.
Charset (CS)	The name of a character set corresponding to a subdirectory in $SYBASE/charsets.
CursorCacheSize (CCS)	The number of connections that the connection cache can hold. The initial default value for CursorCacheSize is 1 (one cursor). To set the connection cache, you must set the SelectMethod attribute to 1. Increasing the connection cache may increase performance of some applications but requires additional database resources.
Database (DB)	The name of the database to which you want to connect.
DataSourceName (DSN)	A string that identifies a single connection to a Sybase database. Examples include "Accounting" or "Sys10-Serv1."
DefaultLongData BuffLen (DLDBL)	An integer value that specifies, in 1024-byte multiples, the maximum length of data fetched from a TEXT or IMAGE column. The default is DefaultLongDataBuffLen=1024. You will need to increase this value if the total size of any long data exceeds 1 megabyte.

Table 8-1. Sybase Connection String Attributes (cont.)

Attribute	Description
DirectoryService Provider (DSP)	A string that indicates which Directory Service Provider the Sybase Open Client uses when connecting with this data source. The available Directory Service Providers can be found using the OpenClient/OpenServer Configuration Utility that is installed with Sybase Open Client version 11.1 or higher. If the client is not using Open Client version 11.1 or higher, this option is ignored.
EnableQuoted Identifiers (EQI)	EnableQuotedIdentifiers={0 | 1}. Specify 1 to allow support of quoted identifiers. The default is 0.
InitializationString (IS)	InitializationString={<Sybase set commands>;...}. Supports the execution of Sybase commands at connect time. Multiple commands must be separated by semicolons.
InterfacesFile (IFILE)	The path name to the interfaces file.
Language (LANG)	The national language corresponding to a subdirectory in $SYBASE/locales.
LogonID (UID)	The default logon ID used to connect to your Sybase 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 8-1. Sybase Connection String Attributes (cont.)

Attribute	Description
OptimizePrepare (OP)	OptimizePrepare={0 | 1 | 2 | 3}. 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 processing static statements.
	When set to 1, the initial default, the driver creates stored procedures only if the statement contains parameters. Otherwise, the statement is cached and run directly at SQLExecute time.
	When set to 2, the driver never creates stored procedures.
	When set to 3, the Sybase data provider never creates stored procedures. Any syntax errors, or similar errors, will be returned at SQLPrepare time instead of at SQLExecute time.
	This attribute is ignored for Sybase 4.9.2 servers.

Table 8-1. Sybase 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 Sybase 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 system information.
	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 Sybase 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 Sybase 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. |
| Password (PWD) | A case-sensitive password. |

Table 8-1. Sybase Connection String Attributes (cont.)

Attribute	Description
PasswordEncryption (PE)	PasswordEncryption={0 | 1}. A value that determines whether password encryption can be performed from the Open Client Library to the server. When set to 0, the initial default, this cannot be done. When set to 1, password encryption is enabled.
RaiseErrorPosition Behavior (REPB)	RaiseErrorPositionBehavior={0 | 1}. A value that specifies when the error is returned and where the cursor is positioned when raiserror is encountered.
	When set to 0 (the default), raiserror is handled separately from surrounding statements. The error is returned when raiserror is processed via SQLExecute, SQLExecDirect, or SQLMoreResults. The result set is empty.
	When set to 1 (MS compatible), raiserror is handled with the next statement. The error is returned when the next statement is processed; the cursor is positioned on the first row of subsequent result set. This could result in multiple raiserrors being returned on a single execute.
SecurityService Provider (SSP)	A string that indicates which Security Service Provider the Sybase Open Client uses when connecting with this data source. The available Security Service Providers can be found using the OpenClient/OpenServer Configuration Utility that is installed with Sybase Open Client version 11.1 or higher. If the client is not using Open Client version 11.1 or higher, this option is ignored.

Table 8-1. Sybase Connection String Attributes (cont.)

Attribute	Description
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 run directly without using database cursors. When set to 1, the data source is limited to one active statement.
ServerName (SRVR)	The name of the server containing the Sybase 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.
WorkstationID (WKID)	The workstation ID used by the client.

Data Types

Table 8-2 shows how the Sybase data types are mapped to the standard ODBC data types.

Table 8-2. Sybase Data Types

Sybase	ODBC
binary	SQL_BINARY
bit	SQL_BIT
char	SQL_CHAR
datetime	SQL_TYPE_TIMESTAMP
decimal*	SQL_DECIMAL
float	SQL_FLOAT
image	SQL_LONGVARBINARY
int	SQL_INTEGER
money	SQL_DECIMAL
numeric*	SQL_NUMERIC
real	SQL_REAL
smalldatetime	SQL_TYPE_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

* Not supported with Sybase 4.9.2 servers.

Isolation and Lock Levels Supported

Sybase supports isolation levels 0 (if the server version is 11 or higher), 1 (read committed, the default), and 3 (serializable). It supports page-level locking. See Appendix D, "Locking and Isolation Levels," in the *Connect ODBC Reference* for a discussion of these topics.

ODBC Conformance Level

The Sybase driver supports the API functions listed in Appendix C, "ODBC API and Scalar Functions," in the *Connect ODBC Reference*. In addition, the following functions are supported:

- SQLColumnPrivileges
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedureColumns
- SQLProcedures
- SQLTablePrivileges

The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

The Sybase database system supports multiple connections and multiple statements per connection. If SelectMethod=1, Sybase data sources are limited to one active statement in manual commit mode.

9   Connect ODBC for Text

The Text driver supports ASCII text files. 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 MERANT DataDirect product for the file name 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. You can execute Update and Delete statements conditionally.

Formats for Text Files

Some common formats for text files are listed in Table 9-1.

Table 9-1. 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.

Table 9-1. Common Text File Formats *(cont.)*

Format	Description
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.
Stream	No character separates column values nor records. The table is one long stream of bytes.

Comma-, tab-, and character-separated files are called character-delimited files because values are separated by a special character.

Defining Table Structure

Because text files do not all have the same structure, the driver provides the option to define 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 the system information file 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 two fields,

```
FIELD1=EMP_ID,VARCHAR,6,0,6,0,
FIELD2=HIRE_DATE,DATE,10,0,10,0,m/d/yy
```

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.INI

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=EMP

[EMP]
FILE=emptext.txt
FLN=1
TT=Comma
Charset=ANSI
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/yy
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 9-2 lists the symbols to use when specifying the date mask.

Table 9-2. 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 Ms (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 Ds (that is, mon, Mon, MON).
dddd, Dddd, DDDD	Output the day depending on the case of the Ds (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.

Table 9-2. Date Masks for Text Driver *(cont.)*

Symbol	Description
\ - . : , (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 9-3 shows some example date values, masks, and how the date appears in the text file.

Table 9-3. 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

Configuring and Connecting to Data Sources

To configure a data source, you must use the following data source entries listed in Table 9-4. See the help chapter "The UNIX Environment" for details on configuring the system information file.

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 in the system information file 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 the system information. These values are not written to the system information.

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 9-4 gives the long and short names for each attribute, as well as a description. The system information 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 the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

Table 9-4. Text Connection String Attributes

Attribute	Description
AllowUpdateAnd Delete (AUD)	AllowUpdateAndDelete={0 | 1}. Specifies whether a data source allows Update and Delete statements. The default is 0. Because Update and Delete statements cause immediate changes to a table, only one connection at a time can operate on a table. When this option is set, tables are opened exclusively by the current connection. Each update and delete on a text file can cause significant changes to the file, and performance may be poor. Consider a more appropriate database form if performance is a significant factor.
ApplicationUsing Threads (AUT)	ApplicationUsingThreads={0 | 1}. A setting that ensures that the driver works with multi-threaded applications. You can clear this check box when using the driver with single-threaded applications. Turning off this setting avoids additional processing required for ODBC thread safety standards.

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.

Table 9-4. Text 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 run the Select statement again. The initial default is 4.
CenturyBoundary (CB)	CenturyBoundary=20. Century Boundary specifies the cutoff year for century inference when converting two-digit dates to four-digit dates. Two-digit dates that are less than the specified year number will be converted to 20xx. Two-digit dates greater than or equal to the number will be converted to 19xx. The default value is 20. For example, using the default value, a date of 19 will be interpreted as 2019 and a date of 21 will be interpreted as 1921.
Database (DB)	The directory in which the text files are stored.
DataFileExtension (DFE)	Specifies the file extension to use for data files. The default Data File Extension setting is TXT. The Data File Extension setting cannot be greater than three characters. The Data File Extension setting is used for all Create Table statements. Sending a Create Table using an extension other than the Data File Extension setting causes an error.
	In other SQL statements, such as Select or Insert, users can specify an extension other than the Data File Extension setting. The Data File Extension setting is used when no extension is specified.

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.

Table 9-4. Text Connection String Attributes (cont.)

Attribute	Description
DataSourceName (DSN)	A string that identifies a Text data source configuration in the system information. Examples include "Accounting" or "Text Files."
DecimalSymbol (DS)	DecimalSymbol={.	,}. Specifies the decimal separator used when data is stored. The international decimal symbol (.) must be used in DML statements and parameter buffers.
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 (,).
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.
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.

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.

Table 9-4. Text Connection String Attributes (cont.)

Attribute	Description
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=0 (the initial default), the driver uses the ASCII sort order. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."
	If IntlSort=1, the driver uses the international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.
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.

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.

Table 9-4. Text Connection String Attributes (cont.)

Attribute	Description
TableType (TT)	TableType={Comma | Tab | Character | Fixed | Stream}. The Text driver supports five types: comma-separated, tab-separated, character-separated, fixed length, and stream. Setting this value tells the driver the default type, which is used when creating a new table and opening an undefined table.
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.

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 9-5 shows how the text file data types are mapped to the standard ODBC data types.

Table 9-5. Text Data Types

Text	ODBC
Numeric	SQL_NUMERIC
Date	SQL_TYPE_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," in the *Connect ODBC Reference*.

Alter Table Statement

The Text driver supports the Alter Table statement to add one or more columns to a table or to delete (drop) a single column.

The Alter Table statement has the form:

```
ALTER TABLE table_name
{ADD column_name data_type
| ADD (column_name data_type
[,column_name data_type] ...)
| DROP [COLUMN] column_name
}
```

table_name is the name of the table to which you are adding or dropping columns.

column_name assigns a name to the column you are adding or specifies the column you are dropping.

data_type specifies the native data type of each column you add.

For example, to add two columns to the emp table,

```
ALTER TABLE emp (ADD startdate date, dept
varchar(10))
```

You cannot add columns and drop columns in a single statement, and you can drop only one column at a time. For example, to drop a column,

```
ALTER TABLE emp DROP startdate
```

The Alter Table statement fails when you attempt to drop a column upon which other objects, such as indexes or views, are dependent.

ODBC Conformance Level

The Text driver supports the API functions listed in Appendix C, "ODBC API and Scalar Functions," in the *Connect ODBC Reference*. In addition, the following function is supported: SQLSetPos.

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

Number of Connections and Statements Supported

Text files support multiple connections and multiple statements per connection.

Index

A

C

D

F

I

T

U

W