



 **INTERSOLV**®

**DataDirect**®  
**Connect ODBC**™

**March 1998**



*Reference*

(c) 1998 INTERSOLV, Inc. All rights reserved. Printed in the U.S.A.

INTERSOLV, DataDirect, APS, Maintenance Workbench, PVCs, TechGnosis, SequeLink, and middleware are registered trademarks of INTERSOLV, Inc. SiteSync, INTERSOLV Messaging, VM Server, Connect ODBC, Connect OLE DB, Reflector, WebDBLink, and Client/Server MiddleWare are trademarks of INTERSOLV, Inc. Other company or product names mentioned herein may be trademarks or registered trademarks of their respective companies.

No part of this publication, with the exception of the software product user documentation contained on a CD-ROM, may be copied, photocopied, reproduced, transmitted, transcribed, or reduced to any electronic medium or machine-readable form without prior written consent of INTERSOLV, Inc.

Licensees may duplicate the software product user documentation contained on a CD-ROM, but only to the extent necessary to support the users authorized access to the software under the license agreement. Any reproduction of the documentation, regardless of whether the documentation is reproduced in whole or in part, must be accompanied by this copyright statement in its entirety, without modification.

U.S. GOVERNMENT RESTRICTED RIGHTS. It is acknowledged that the Software and the Documentation were developed at private expense, that no part is in the public domain, and that the Software and Documentation are Commercial Computer Software provided with RESTRICTED RIGHTS under Federal Acquisition Regulations and agency supplements to them. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFAR 252.227-7013 et. seq. or subparagraphs (c)(1) and (2) of the Commercial Computer Software Restricted Rights at FAR 52.227-19, as applicable. Contractor is INTERSOLV, Inc., 9420 Key West Avenue, Rockville, Maryland, 20850. Rights are reserved under copyright laws of the United States with respect to unpublished portions of the Software.

INTERSOLV, Inc.  
9420 Key West Avenue  
Rockville, Maryland 20850

# Table of Contents

<b>Preface</b> .....	<b>17</b>
Using this Manual .....	17
Additional Documentation .....	19
Conventions Used in This Manual .....	19
Typography .....	19
Mouse Conventions .....	20
Keyboard Conventions .....	21
Environment-Specific Information .....	21
Contacting Technical Support .....	22
<b>1 Getting Started</b> .....	<b>25</b>
About DataDirect Connect ODBC Drivers .....	25
Support for Multiple Environments .....	26
Installing the ODBC Drivers .....	26
Environment-Specific Information .....	27
For Windows 95 and Windows NT Users .....	27
For Macintosh Users .....	28
For UNIX Users .....	29
Error Messages .....	32
UNIX Error Handling .....	33
<b>2 Connect ODBC for Btrieve</b> .....	<b>35</b>
System Requirements .....	35
Managing Databases .....	36
Transactions .....	37
Configuring Data Sources .....	37

Defining Table Structure . . . . .	42
Connecting to a Data Source Using a Connection String . . .	44
Data Types. . . . .	49
Indexes . . . . .	50
Column Names . . . . .	50
Select Statement. . . . .	51
Rowid Pseudo-Column . . . . .	51
Alter Table Statement . . . . .	52
Create and Drop Index Statements. . . . .	53
Create Index . . . . .	53
Drop Index . . . . .	53
Isolation and Lock Levels Supported. . . . .	54
ODBC Conformance Level . . . . .	54
Number of Connections and Statements Supported. . . . .	54
<b>3 Connect ODBC for DB2 . . . . .</b>	<b>55</b>
System Requirements. . . . .	55
Windows 95 and Windows NT. . . . .	55
UNIX . . . . .	56
Configuring Data Sources . . . . .	56
DB2 . . . . .	62
Binding . . . . .	62
Connecting to a Data Source Using a Logon Dialog Box. . . .	63
Connecting to a Data Source Using a Connection String . . .	64
Data Types. . . . .	67
Isolation and Lock Levels Supported. . . . .	68
ODBC Conformance Level . . . . .	68
Number of Connections and Statements Supported. . . . .	68

<b>4</b>	<b>Connect ODBC for dBASE</b> . . . . .	<b>69</b>
	System Requirements . . . . .	69
	Macintosh . . . . .	69
	Configuring Data Sources . . . . .	70
	Configuring FoxPro 3.0 DBC Data Sources . . . . .	76
	Defining Index Attributes . . . . .	82
	Defining Index Attributes on UNIX Platforms . . . . .	84
	Connecting to a Data Source Using a Connection String . . . . .	85
	Data Types . . . . .	92
	Column Names . . . . .	93
	Select Statement . . . . .	94
	Rowid Pseudo-Column . . . . .	94
	Alter Table Statement . . . . .	95
	Create Index Statement . . . . .	96
	Drop Index Statement . . . . .	98
	Pack Statement . . . . .	99
	SQL Statements for FoxPro 3.0 Database Containers . . . . .	100
	Locking . . . . .	101
	Levels of Database Locking . . . . .	101
	Using Locks on Local Files . . . . .	102
	Limit on Number of Locks . . . . .	102
	How Transactions Affect Record Locks . . . . .	102
	Isolation and Lock Levels Supported . . . . .	103
	ODBC Conformance Level . . . . .	103
	Number of Connections and Statements Supported . . . . .	103

<b>5</b>	<b>Connect ODBC for Excel Workbook . . . . .</b>	<b>105</b>
	System Requirements . . . . .	105
	Using an Excel Database . . . . .	105
	Configuring Data Sources . . . . .	106
	Connecting to a Data Source Using a Connection String . . .	110
	Data Types . . . . .	113
	Table and Column Names . . . . .	114
	Select Statement . . . . .	114
	Create Table Statement . . . . .	114
	ODBC Conformance Level . . . . .	115
	Number of Connections and Statements Supported . . . . .	115
<b>6</b>	<b>Connect ODBC for INFORMIX . . . . .</b>	<b>117</b>
	System Requirements . . . . .	117
	Windows 95 and Windows NT . . . . .	117
	UNIX (AIX, HP-UX, and Solaris for SPARC) . . . . .	119
	Configuring Data Sources . . . . .	120
	Connecting to a Data Source Using a Logon Dialog Box . . .	124
	Connecting to a Data Source Using a Connection String . . .	126
	Data Types . . . . .	130
	INFORMIX 9 . . . . .	132
	Isolation and Lock Levels Supported . . . . .	132
	ODBC Conformance Level . . . . .	133
	Number of Connections and Statements Supported . . . . .	133

<b>7</b>	<b>Connect ODBC for OpenIngres</b> . . . . .	<b>135</b>
	System Requirements . . . . .	135
	Windows 95 and Windows NT . . . . .	135
	UNIX . . . . .	136
	Configuring Data Sources . . . . .	137
	Connecting to a Data Source Using a Logon Dialog Box . . . .	143
	Connecting to a Data Source Using a Connection String . . . .	144
	Data Types . . . . .	149
	Isolation and Lock Levels Supported . . . . .	150
	ODBC Conformance Level . . . . .	150
	Number of Connections and Statements Supported. . . . .	150
<b>8</b>	<b>Connect ODBC for Oracle</b> . . . . .	<b>151</b>
	System Requirements . . . . .	151
	Windows 95 and Windows NT . . . . .	151
	UNIX . . . . .	152
	Macintosh . . . . .	154
	Configuring Data Sources . . . . .	154
	Connecting to a Data Source Using a Logon Dialog Box . . . .	159
	Connecting to a Data Source Using a Connection String . . . .	160
	Oracle Data Types . . . . .	164
	Oracle 8 . . . . .	165
	Stored Procedure Results . . . . .	165
	Isolation and Lock Levels Supported . . . . .	166
	ODBC Conformance Level . . . . .	167
	Number of Connections and Statements Supported. . . . .	167

<b>9</b>	<b>Connect ODBC for Paradox.....</b>	<b>169</b>
	System Requirements.....	169
	Multiuser Access to Tables.....	170
	Locking.....	170
	Configuring Data Sources.....	171
	Connecting to a Data Source Using a Connection String ...	175
	Data Types.....	180
	Select Statement.....	181
	Column Names.....	181
	Alter Table Statement.....	181
	Dropping Columns.....	182
	Create Table Statement.....	183
	Password Protection.....	184
	Encrypting a Paradox Table.....	185
	Accessing an Encrypted Paradox Table.....	185
	Decrypting a Paradox Table.....	186
	Removing a Password from Paradox.....	186
	Removing All Passwords from Paradox.....	186
	Index Files.....	186
	Primary Index.....	187
	Non-Primary Index.....	187
	Create and Drop Index Statements.....	188
	Create Index Primary Statement.....	189
	Create Index Statement.....	189
	Drop Index Statement.....	191
	Transactions.....	191
	Isolation and Lock Levels Supported.....	192
	ODBC Conformance Level.....	192
	Number of Connections and Statements Supported.....	193



<b>10</b>	<b>Connect ODBC for PROGRESS</b>	<b>195</b>
	System Requirements	195
	Configuring Data Sources	196
	Remote OID with Direct Access	196
	Remote OID with Database Access via Server	203
	Connecting to a Data Source Using a Logon Dialog Box	208
	Connecting to a Data Source Using a Connection String	211
	Data Types	214
	Isolation and Lock Levels Supported	215
	ODBC Conformance Level	215
	Connections and Statements Supported	215
<b>11</b>	<b>Connect ODBC for SQL Server</b>	<b>217</b>
	System Requirements	217
	Configuring Data Sources	218
	Connecting to a Data Source Using a Logon Dialog Box	224
	Connecting to a Data Source Using a Connection String	225
	Data Types	230
	Isolation and Lock Levels Supported	231
	ODBC Conformance Level	231
	Number of Connections and Statements Supported	232
<b>12</b>	<b>Connect ODBC for SQLBase</b>	<b>233</b>
	System Requirements	233
	Configuring Data Sources	234
	Connecting to a Data Source Using a Logon Dialog Box	237
	Connecting to a Data Source Using a Connection String	238
	Data Types	241

Isolation and Lock Levels Supported . . . . .	241
ODBC Conformance Level . . . . .	242
Number of Connections and Statements Supported . . . . .	242
<b>13 Connect ODBC for Sybase . . . . .</b>	<b>243</b>
System Requirements . . . . .	243
Windows 95 and Windows NT . . . . .	243
UNIX . . . . .	244
Macintosh . . . . .	245
Configuring Data Sources . . . . .	245
Connecting to a Data Source Using a Logon Dialog Box . . . . .	253
Connecting to a Data Source Using a Connection String . . . . .	254
Data Types . . . . .	260
Isolation and Lock Levels Supported . . . . .	261
ODBC Conformance Level . . . . .	261
Number of Connections and Statements Supported . . . . .	261
<b>14 Connect ODBC for Text . . . . .</b>	<b>263</b>
System Requirements . . . . .	263
Formats for Text Files . . . . .	264
Configuring Data Sources . . . . .	265
Defining Table Structure . . . . .	271
Defining Table Structure on UNIX Platforms . . . . .	276
Example of QETXT.INI . . . . .	278
Date Masks . . . . .	279
Connecting to a Data Source Using a Connection String . . . . .	281
Data Types . . . . .	286
Select Statement . . . . .	287
Alter Table Statement . . . . .	287

ODBC Conformance Level . . . . .	288
Number of Connections and Statements Supported. . . . .	288
<b>A SQL for Flat-File Drivers . . . . .</b>	<b>289</b>
Select Statement . . . . .	289
Select Clause. . . . .	290
From Clause . . . . .	292
Where Clause . . . . .	292
Group By Clause. . . . .	293
Having Clause. . . . .	294
Union Operator . . . . .	294
Order By Clause . . . . .	295
For Update Clause . . . . .	296
SQL Expressions . . . . .	296
Create and Drop Table Statements. . . . .	308
Create Table . . . . .	308
Drop Table . . . . .	309
Insert Statement . . . . .	310
Update Statement . . . . .	312
Delete Statement. . . . .	313
Reserved Keywords . . . . .	314
<b>B Using Indexes. . . . .</b>	<b>315</b>
Introduction . . . . .	315
Improving Record Selection Performance . . . . .	317
Indexing Multiple Fields . . . . .	317
Deciding Which Indexes to Create . . . . .	319
Improving Join Performance. . . . .	321

<b>C</b>	<b>ODBC API and Scalar Functions</b>	<b>323</b>
	API Functions	323
	Scalar Functions	326
	String Functions	326
	Numeric Functions	329
	Date and Time Functions	331
	System Functions	333
<b>D</b>	<b>Locking and Isolation Levels</b>	<b>335</b>
	Locking	335
	Isolation Levels	336
	Locking Modes and Levels	339
<b>E</b>	<b>Designing Performance-Oriented ODBC Applications</b>	<b>341</b>
	Optimizing Performance	341
	Catalog Functions	342
	Catalog Functions Are Relatively Slow	343
	Passing Null Arguments	343
	SQLColumns	345
	Retrieving Data	347
	Retrieving Long Data	347
	Reducing the Size of Data Retrieved	347
	Using Bound Columns	349
	Using SQLExtendedFetch instead of SQLFetch	350
	ODBC Function Selection	351
	Using SQLPrepare/SQLExecute and SQLExecDirect	351
	Using SQLPrepare and Multiple SQLExecute Calls	352
	Using the Cursor Library	354

Design Options . . . . .	355
Managing Connections . . . . .	355
Committing Data . . . . .	356
Asynchronous Execution . . . . .	356
Updating Data . . . . .	358
Using Positional Updates and Deletes . . . . .	358
Using SQLSpecialColumns . . . . .	358
<b>F Threading . . . . .</b>	<b>361</b>
<b>G Microsoft Query '97 . . . . .</b>	<b>365</b>
Creating a Flat-File Data Source for Use with Microsoft Query '97 365	
Using Microsoft Query '97 with Single-Connect Data Sources	366
<b>H The UNIX Environment . . . . .</b>	<b>369</b>
The System Information File (.odbc.ini) . . . . .	369
Sample Solaris System Information File . . . . .	370
Environment Variables . . . . .	372
Required Environment Variables . . . . .	372
Optional Environment Variables . . . . .	372
Using Double-Byte Character Sets . . . . .	373
The ivtestlib Tool . . . . .	374
Translators . . . . .	374
<b>Index . . . . .</b>	<b>375</b>



# List of Tables

Table 2-1.	Btrieve Connection String Attributes . . . . .	45
Table 2-2.	Btrieve Data Types . . . . .	49
Table 3-1.	DB2 Connection String Attributes . . . . .	65
Table 3-2.	DB2 Data Types . . . . .	67
Table 4-1.	dBASE Connection String Attributes . . . . .	86
Table 4-2.	dBASE Data Types . . . . .	92
Table 4-3.	Additional FoxPro 3.0 Data Types . . . . .	93
Table 4-4.	dBASE-Compatible Index Summary . . . . .	98
Table 5-1.	Excel Connection String Attributes . . . . .	111
Table 5-2.	Excel Data Types . . . . .	113
Table 6-1.	INFORMIX Connection String Attributes . . . . .	127
Table 6-2.	INFORMIX Data Types . . . . .	130
Table 6-3.	INFORMIX 9 Data Types . . . . .	132
Table 7-1.	OpenIngres Connection String Attributes . . . . .	145
Table 7-2.	OpenIngres Data Types . . . . .	149
Table 8-1.	Oracle Connection String Attributes . . . . .	161
Table 8-2.	Oracle Data Types . . . . .	164
Table 8-3.	Oracle 8 Data Types . . . . .	165
Table 9-1.	Paradox Connection String Attributes . . . . .	176
Table 9-2.	Paradox Data Types . . . . .	180
Table 10-1.	PROGRESS Connection String Attributes . . . . .	212
Table 10-2.	PROGRESS Data Types . . . . .	214
Table 11-1.	SQL Server Connection String Attributes . . . . .	226

Table 11-2.	SQL Server Data Types . . . . .	230
Table 12-1.	SQLBase Connection String Attributes . . . . .	239
Table 12-2.	SQLBase Data Types . . . . .	241
Table 13-1.	Sybase Connection String Attributes . . . . .	255
Table 13-2.	Sybase Data Types . . . . .	260
Table 14-1.	Common Text File Formats . . . . .	264
Table 14-2.	Date Masks for Text Driver . . . . .	279
Table 14-3.	Date Mask Examples . . . . .	280
Table 14-4.	Text Connection String Attributes . . . . .	282
Table 14-5.	Text Data Types . . . . .	286
Table A-1.	Aggregate Functions . . . . .	291
Table A-2.	Relational Operators . . . . .	300
Table A-3.	Operator Precedence . . . . .	302
Table A-4.	Functions that Return Character Strings . . . . .	303
Table A-5.	Functions that Return Numbers . . . . .	306
Table A-6.	Functions that Return Dates . . . . .	307
Table C-1.	Supported 2.x ODBC API Functions . . . . .	324
Table C-2.	Supported 3.x ODBC API Functions . . . . .	325
Table C-3.	Scalar String Functions . . . . .	327
Table C-4.	Scalar Numeric Functions . . . . .	329
Table C-5.	Scalar Time and Date Functions . . . . .	331
Table C-6.	Scalar System Functions . . . . .	333
Table D-1.	Isolation Levels and Data Consistency . . . . .	338
Table E-1.	Common ODBC System Performance Problems and Solutions . . . . .	341
Table F-1.	Threading Information . . . . .	363



# Preface

This book is your reference to INTERSOLV® DataDirect® Connect ODBC™. The DataDirect Connect ODBC product consists of a number of database *drivers* that are compliant with the Open Database Connectivity (ODBC) specification.

---

## Using this Manual

The content of this manual is based on the assumption that you are familiar with your operating system and its commands. It contains the following chapters:

- An introductory chapter ([Chapter 1, “Getting Started,” on page 25](#)) that explains the DataDirect Connect ODBC drivers and ODBC, discusses environment-specific subjects, and explains the error messages returned by the drivers.
- A chapter for each database driver. Each driver’s chapter is structured in the same way. First, it lists system requirements for your operating environment. Next, it explains how to configure a data source. Finally, it explains how to connect to that data source.

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

- [Appendix A, “SQL for Flat-File Drivers,” on page 289](#) explains the SQL statements that you can use with Btrieve, dBASE, Excel, Paradox, and text files.
- [Appendix B, “Using Indexes,” on page 315](#) provides general guidelines on how to improve performance when querying a database system.

- [Appendix C, “ODBC API and Scalar Functions,” on page 323](#) lists the ODBC API functions that each driver supports. Any exceptions are listed in each driver chapter, in the section “ODBC Conformance Level.” This appendix also lists the ODBC scalar functions.
- [Appendix D, “Locking and Isolation Levels,” on page 335](#) provides a general discussion of isolation levels and locking.
- [Appendix E, “Designing Performance- Oriented ODBC Applications,” on page 341](#) provides guidelines for designing performance-oriented ODBC applications.
- [Appendix F, “Threading,” on page 361](#) discusses how ODBC ensures thread safety.
- [Appendix G, “Microsoft Query ‘97,” on page 365](#) discusses how to use ODBC with Microsoft Query ‘97.
- [Appendix H, “The UNIX Environment,” on page 369](#) explains the structure of the *system information file* (used in the UNIX environment), provides a sample system information file, and discusses UNIX environment variables.

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

---

## Additional Documentation

In addition to this Reference, an online Installation Guide is provided on your DataDirect CD-ROM.

You can view this online documentation using the Adobe Acrobat Reader, which is also available on your CD-ROM. For instructions about installing the Acrobat Reader and viewing ODBC driver online documentation, refer to the insert in your CD-ROM case.

---

## Conventions Used in This Manual

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

The suffix .DLL is used to refer to a dynamic link library file. Your operating system may use the term shared object or shared library files instead.

### Typography

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

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

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

## Mouse Conventions.





<b>This action...</b>	<b>Means to...</b>
Click	Point to an object with the mouse pointer and press the left mouse button.
Double-click	Click the left mouse button twice.
Right-click	Click the right mouse button.
Drag	Press and hold the left mouse button while dragging item(s) to another part of the screen.
SHIFT+Click	Press and hold the SHIFT key; then, click a selection. This lets you select a series of objects.
CTRL+Click	Press and hold the CTRL key; then, click a selection. This lets you select objects randomly.

## Keyboard Conventions

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

## Environment-Specific Information

Wherever this manual provides information that is not applicable to all supported environments, the following symbols are used to identify that information:

Symbol	Environment
	<b>Windows 95.</b> Information specific to the Microsoft Windows 95 environment is identified by the Windows symbol and the numbers “95.”
	<b>Windows NT.</b> Information specific to the Microsoft Windows NT environment is identified by the Windows symbol and the letters “NT.”
	<b>Macintosh Power PC.</b> Information specific to the Macintosh Power PC environment is identified by the MacOS symbol, which is a registered trademark of Apple Computer, Inc., and the words “Power PC.”
	<b>UNIX.</b> Information specific to UNIX environments is identified by this symbol, which applies to all supported UNIX environments. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

This manual shows dialog boxes that are specific to Windows 95. If you are using the drivers on Windows NT or Macintosh, the dialog box that you see may differ slightly from the Windows 95 version. If you are using a graphical user interface in the UNIX environment, you will see a dialog box for logon, but not for driver configuration.

---

## Contacting Technical Support

INTERSOLV 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.

**WorldWide Web** <http://www.intersolv.com>

The INTERSOLV 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.

<b>Internet</b>	Australia and New Zealand	<a href="mailto:australia_answerline@intersolv.com">australia_answerline@intersolv.com</a>
	EMEA	<a href="mailto:int_datadirect_answerline@intersolv.com">int_datadirect_answerline@intersolv.com</a>
	Japan	<a href="mailto:jpn_answerline@intersolv.co.jp">jpn_answerline@intersolv.co.jp</a>
	USA and Canada	<a href="mailto:datadirect_answerline@intersolv.com">datadirect_answerline@intersolv.com</a>

### 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
<b>Fax US</b>	1 919 461 4527	
<b>Fax International</b>	+32-15-320919	
<b>Mail</b>	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 Getting Started

This chapter contains the following sections:

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

---

## About DataDirect Connect ODBC Drivers

DataDirect  
database drivers  
and ODBC

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

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

INTERSOLV provides ODBC drivers for both relational and flat-file database systems. The flat-file drivers provide full SQL support as discussed in [Appendix A, “SQL for Flat-File Drivers,”](#) on page 289.

## Support for Multiple Environments

DataDirect drivers offer multi-platform support

INTERSOLV provides ODBC-compliant database drivers on the Windows 95, Windows NT, Windows 3.1, and Macintosh platforms, as well as the following UNIX platforms: Solaris for SPARC, HP-UX, and AIX.

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

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

["Environment-Specific Information" on page 27](#) explains the environment-specific differences that you should be aware of when using the database drivers in your operating environment.

## Installing the ODBC Drivers

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

---

# Environment-Specific Information

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

## For Windows 95 and Windows NT Users



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

### *Starting the ODBC Administrator*

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

### *Driver Names*

The prefix for all Connect ODBC driver filenames on Windows 95 and Windows NT is “IV.” The file extension is .DLL. This indicates that they are dynamic link libraries. For example, the Oracle 7 driver filename is *IVOR7nn.DLL*, where *nn* is the revision number of the driver.

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

## ***Disk Space and Memory Requirements***

Disk space requirements are 15 MB of free disk space on the disk drive where Windows 95 or Windows NT is installed.

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

## **For Macintosh Users**



On Macintosh systems, the ODBC drivers are built using the Code Fragment Manager and are stored in the Extensions folder. For the Macintosh Power PC, no additional software is required to use the Code Fragment Manager.

The operating system required is System 7.1 or greater.

## ***Starting the ODBC Administrator***

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

## ***Driver Names***

The database drivers are shared libraries. An example of a driver name would be INTERSOLV 3.*nn* Sybase Driver, where *nn* is the revision number of the driver.

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

## ***Disk Space and Memory Requirements***

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

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

## **For UNIX Users**



Consult the “System Requirements” section of each database driver chapter. The UNIX platforms supported are:

### **Solaris**

- Sun SPARCstation
- Solaris 2.51 or 2.6 operating system

### **AIX**

- AIX 4.1.4 or 4.2 operating system

### **HP-UX cFront enabled**

- HP-UX 10.10 or 10.20 operating system
- cFront compiled application

### HP-UX aCC enabled

- HP-UX 10.20 operating system
- aCC compiled application

## *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.

[Appendix H, "The UNIX Environment," on page 369](#) explains the structure of the system information file, provides a sample file, and discusses UNIX environment variables.

## *Driver Names*

The Connect ODBC drivers are ODBC API-compliant dynamic link libraries, referred to in UNIX as *shared objects*. The prefix for all ODBC driver filenames on UNIX is "iv." On UNIX the driver filenames are lowercase and the extension is *.so* or *.sl*. This is the standard form for a shared object. For example, the Oracle 7 driver filename is *ivor7nn.so*, where *nn* is the revision number of the driver.

**Note:** The convention in this manual is to list the driver names in uppercase with the extension *.DLL*.

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

## ***Setting the Library Path Environment Variable***

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

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

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

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

```
. /opt/odbc/odbc.sh
```

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

## ***Disk Space and Memory Requirements***

Disk space requirements are 25 MB of free disk space on the disk where the UNIX system is installed.

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

---

## Error Messages

Error messages can come from:

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

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

```
[ vendor ] [ ODBC_component ] message
```

*ODBC\_component* is the component in which the error occurred. For example, an error message from INTERSOLV's SQL Server driver would look like this:

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

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

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

```
[ vendor ] [ ODBC_component ] [ data_source ] message
```

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

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

If you get this type of error, you did something incorrectly with the database system. Check your database system documentation



for more information or consult your database administrator. In this example, you would check your Oracle documentation.

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

```
[vendor] [ODBC XXX] message
```

For example, an error from the Microsoft driver manager might look like this:

```
[Microsoft] [ODBC Driver Manager] Driver does not support this function
```

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

## UNIX Error Handling



UNIX error handling follows the X/Open XPG3 messaging catalog system. Localized error messages are stored in the subdirectory `locale/localized_territory_directory/LC_MESSAGES`, where `localized_territory_directory` depends on your language.

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

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

```
[INTERSOLV] [ODBC 20101 driver] 30040
```



## 2 Connect ODBC for Btrieve



Connect ODBC for Btrieve (the “Btrieve driver”) supports Btrieve version 6.x and higher under the Windows 95 and Windows NT. The driver executes SQL statements directly on Btrieve databases.

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

---

### System Requirements

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



For Windows 95:

- Btrieve Developer's Kit for Windows 95
- Btrieve WorkStation Database Engine for Windows NT or Windows 95



For Windows NT:

- Btrieve Developer's Kit for Windows NT
- Btrieve WorkStation Database Engine for Windows NT or Windows 95
- Btrieve Client/Server Database Engine

Before you attempt to access Btrieve files, you must incorporate existing Btrieve files into a Scalable SQL database. See [“Defining Table Structure” on page 42](#) for more information.

If you attempt to configure a data source and you do not have the client loaded, a message similar to the following one appears:

```
The setup routines for the INTERSOLV 3.00 32-BIT
BTRIEVE ODBC driver could not be loaded due to
system error code 126.
```

When you click **OK**, the following message appears:

```
Could not load the setup or translator library.
```

**Note:** The Btrieve driver may experience problems if the Btrieve Microkernel Engine's communication buffer size is smaller than the Btrieve driver's Array Size attribute. You can increase the communication buffer size with the BTI Database Setup Utility. You can decrease the array size option when you configure a data source using the ODBC Btrieve Driver Setup dialog box, or when passing a connection string.

---

## Managing Databases

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

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

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

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

---

## Transactions

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

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

To use the Btrieve driver's transaction processing capabilities, consult the *Btrieve Installation and Operations Manual*.

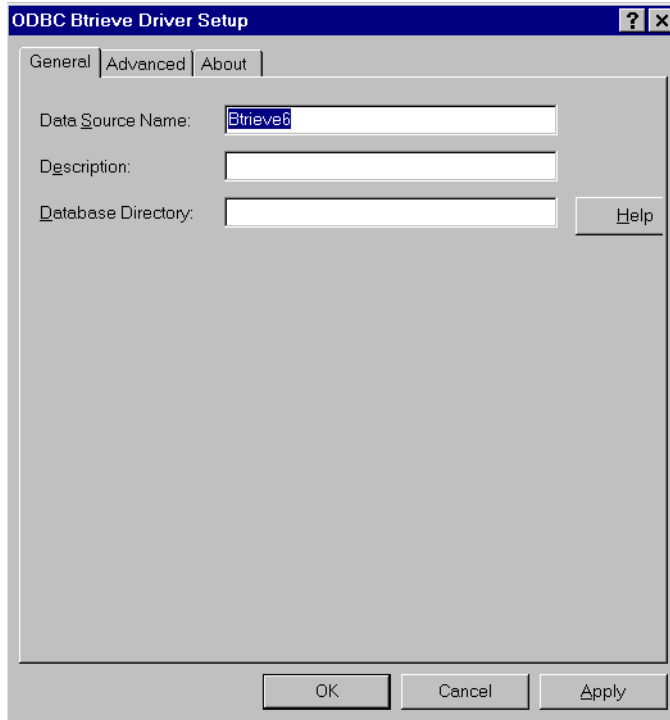
---

## Configuring Data Sources

To configure a Btrieve data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Btrieve Driver Setup dialog box.

If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Btrieve driver and click **Finish** to display the ODBC Btrieve Driver Setup dialog box.



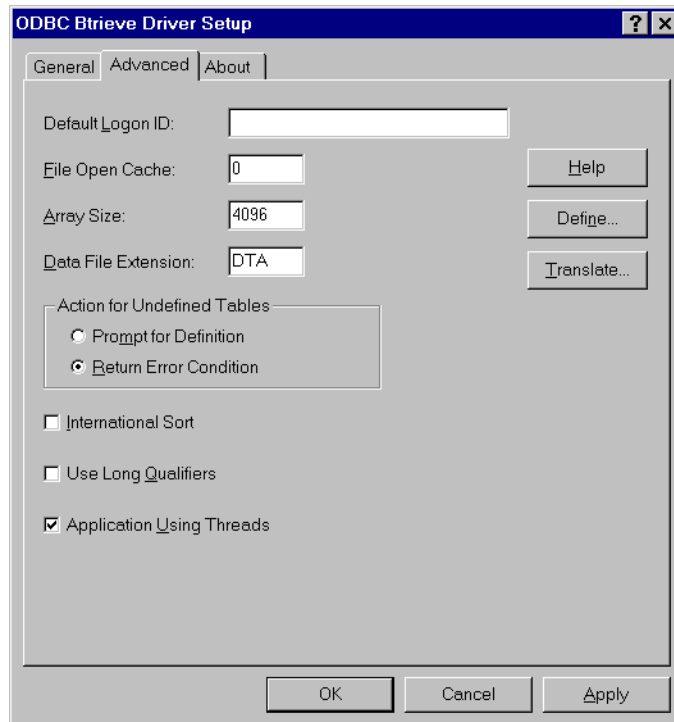
**3** Specify values as follows; then, click **Apply**:

**Data Source Name:** A string that identifies this Btrieve data source configuration in the system information. Examples include "Accounting" or "Btrieve Files."

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

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

- 4 Click the **Advanced** tab to configure additional, optional settings for the data source.



- 5 Specify values as follows; then, click **Apply**:

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

**File Open Cache:** A numeric value to specify the maximum number of used file handles to cache. For example, the value 4 specifies that when a user opens and closes four tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is increased performance. The

disadvantage is that a user who specifies file locking on open may get a locking conflict even though no one appears to have the file open. The default is 0, which means no file open caching.

**Array Size:** A value that enables the driver to retrieve an array of records from the Btrieve engine and in most cases results in better performance for the application. The Array Size value is the number of bytes in the array. The default value is 4,096 bytes, and the maximum is 65,535 bytes.

**Data File Extension:** A string of three or fewer characters that specifies the file extension to use for data files. The default value is DTA. This value is used for all Create Table statements. Sending a Create Table statement that uses an extension other than the one specified as the DataFileExtension value causes an error.

In other SQL statements, such as Select or Insert, you can specify an extension other than the DataFileExtension value. If you do not specify an extension value in these cases, the DataFileExtension value is used.

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

**International Sort:** A setting to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this box to use 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.

**Use Long Qualifiers:** A setting that specifies whether the driver uses long path names. If you select the Use Long Qualifiers check box, path names can be up to 255 characters. If the check box is cleared (the default), the maximum path name length is 128 characters.

**Application Using Threads:** 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.

**Define:** Click **Define** to define table structure. See [“Defining Table Structure”](#) for step-by-step instructions.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 6 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

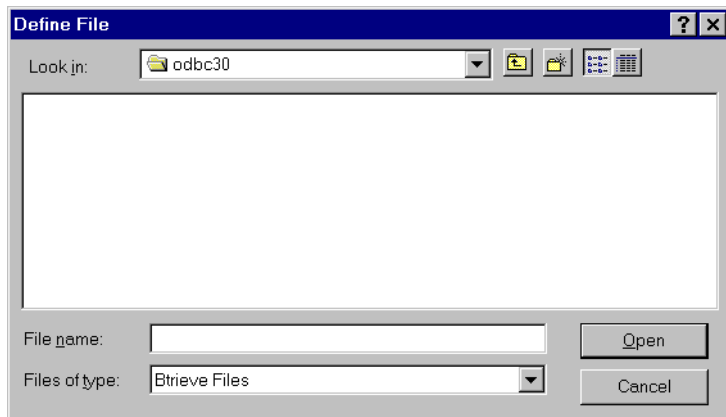
---

## Defining Table Structure

Because Btrieve does not store any column information in the data file, you must define its structure.

To define the structure of a file:

- 1 Display the ODBC Btrieve Driver Setup dialog box through the ODBC Administrator. Click the **Advanced** tab; then, click **Define** to display the Define File dialog box.



- 2 Select the file you want to define and click **Open** to display the Define Table dialog box.

The screenshot shows a 'Define Table' dialog box with the following fields and controls:

- Database Name:** C:
- Table Information:**
  - File:** Suhdlog.dat
  - Table:** [Empty text box]
- Column Information:**
  - [Empty list box for column names]
  - Name:** [Empty text box]
  - Type:** AUTOINCREMENT(2) [Dropdown menu]
  - Length:** 2 [Text box]
  - Scale:** 0 [Text box]
  - Buttons:** Add, Modify, Remove
- Right-side Buttons:** OK, Cancel, Help

**Database Name:** The name of the Scalable SQL data dictionary directory that you selected in the Define File dialog box.

**File:** The name of the file that you selected in the Define File dialog box.

**Table:** Type the name of the table to be returned by SQLTables. The name can be up to 20 characters and cannot be the same as another defined table in the database. This field is required.

- 3 Enter values in the following fields to define each column. Click **Add** to add the column name to the list box.

**Name:** Type the name of the column.

**Type:** Select the data type of the column.

**Length:** Type the length of the column, if applicable.

**Scale:** Type the scale of the column, if applicable.

- 4 To modify an existing column definition, select the column name in the list box. Modify the values for that column name; then, click **Modify**.
- 5 To delete an existing column definition, select a column name in the list box and click **Remove**.
- 6 Click **OK** to define the table.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 Btrieve is:

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

[Table 2-1](#) lists the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you

specified a value for the attribute when configuring the data source, that value is your default.

---

**Table 2-1. Btrieve Connection String Attributes**

---

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

**Table 2-1. Btrieve Connection String Attributes** (cont.)

Attribute	Description
Password (PWD)	The password that you must enter if your Scalable SQL data dictionary files have security restrictions imposed.
IntlSort (IS)	<p data-bbox="686 421 1258 673">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."</p> <p data-bbox="686 690 1258 907">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.</p>
ModifySQL (MS)	ModifySQL={0   1}. This attribute is provided for backward compatibility. Specify ModifySQL=0 to have the driver understand SQL dialects found in earlier drivers. Specify ModifySQL=1 (the default) to have the driver modify the SQL statement to conform to ODBC specifications.

**Table 2-1. Btrieve Connection String Attributes** (cont.)

Attribute	Description
DeferQuery Evaluation (DQ)	<p data-bbox="725 317 1296 435">DeferQueryEvaluation={0   1}. This attribute determines when a query is evaluated—after all records are read or each time a record is fetched.</p> <p data-bbox="725 456 1296 864">If DeferQueryEvaluation=0, the driver generates a result set when the first record is fetched. The driver reads all records, evaluates each one against the Where clause, and compiles a result set containing the records that satisfy the search criteria. This process slows performance when the first record is fetched, but activity performed on the result set after this point is much faster because the result set has already been created. You do not see any additions, deletions, or changes in the database that occur while working with this result set.</p> <p data-bbox="725 885 1296 1295">If DeferQueryEvaluation=1 (the default), the driver evaluates the query each time another record is fetched and stops reading through the records when it finds one that matches the search criteria. This setting avoids the slowdown while fetching the first record, but each fetch takes longer because of the evaluation taking place. The data you retrieve reflect the latest changes to the database; however, a result set is still generated if the query is a Union of multiple Select statements, if it contains the Distinct keyword, or if it has an Order By or Group By clause.</p>
UseLongQualifiers (ULQ)	<p data-bbox="725 1315 1296 1498">UseLongQualifiers={0   1}. This attribute specifies whether the driver uses long path names as table qualifiers. With UseLongQualifiers set to 1, path names can be up to 255 characters. The default is 0; maximum path name length is 128 characters.</p>

**Table 2-1. Btrieve Connection String Attributes** (cont.)

Attribute	Description
DataFileExtension (DFE)	<p>A string of three or fewer characters that specifies the file extension to use for data files. The default value is DTA. This value is used for all Create Table statements. If you execute a Create Table statement that uses an extension other than the one specified as the DataFileExtension value, an error occurs.</p> <p>In other SQL statements, such as Select or Insert, you can specify an extension other than the DataFileExtension value. If you do not specify an extension value in these cases, the DataFileExtension value is used.</p>
ArraySize (AS)	<p>An integer value that enables the driver to retrieve an array of records from the Btrieve engine and in most cases results in better performance for the application. The value of ArraySize is the number of bytes in the array. The default ArraySize is 4,096 bytes and the maximum is 65,535 bytes.</p>
ApplicationUsing Threads (AUT)	<p>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.</p>



---

## Data Types

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

---

**Table 2-2. Btrieve Data Types**

---

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

---

## Indexes

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

For query optimization, the Btrieve driver does not use

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

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

---

## Column Names

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

```
SELECT `name` FROM emp
```

---

# Select Statement

You use the SQL Select statement to specify the columns and records to be read. Btrieve Select statements support all the Select statement clauses as described in [Appendix A, "SQL for Flat-File Drivers," on page 289](#). This section describes the information that is specific to Btrieve.

## Rowid Pseudo-Column

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

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

```
SELECT last_name, first_name, salary, rowid FROM 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 Btrieve 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 when you attempt to drop a column upon which other objects, such as indexes or views, are dependent.

---

# Create and Drop Index Statements

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

## Create Index

The Create Index statement for Btrieve files has the form:

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

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

*index\_name* is the name of the index.

*table\_name* is the name of the table on which the index is to be created.

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

For example:

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

## Drop Index

The form of the Drop Index statement is

```
DROP INDEX table_name.index_name
```

*table\_name* is the name of the table from which the index is to be dropped.

*index\_name* is the name of the index.

For example:

```
DROP INDEX emp.lname
```

---

## Isolation and Lock Levels Supported

Btrieve supports isolation level 1 (read committed) only. Btrieve supports page-level locking. See [Appendix D, “Locking and Isolation Levels,”](#) on page 335 for details.

---

## ODBC Conformance Level

The Btrieve driver supports the functions listed in [Appendix C, “ODBC API and Scalar Functions,”](#) on page 323. In addition, the following function is supported: SQLSetPos.

The Btrieve driver also supports backward and random fetching in SQLExtendedFetch and SQLFetchScroll. The driver supports the minimum SQL grammar with several core extensions.

---

## Number of Connections and Statements Supported

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

## 3 Connect ODBC for DB2



Connect ODBC for DB2 (the “DB2 driver”) supports the following database systems in the Windows 95, Windows NT Solaris, HP-UX, and AIX environments:

- DB2 for Windows NT
- DB2/6000

See the README file shipped with your INTERSOLV DataDirect product for the file name of the DB2 driver.

---

### 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:

#### Windows 95 and Windows NT



- IBM DB2 Client Application Enabler (CAE) for Windows 95 and Windows NT, version 2.1 or higher
- IBM DB2 Software Development Kit (DB2 SDK) for Windows 95 and Windows NT, version 2.1 or higher

## UNIX



### Solaris

- IBM DB2 Client Application Enabler (CAE) for Solaris, version 2.1 or higher
- IBM DB2 Software Development Kit (DB2 SDK) for Solaris, version 2.1 or higher

### HP-UX

- IBM DB2 Client Application Enabler (CAE) for HP-UX, version 2.1 or higher
- IBM DB2 Software Development Kit (DB2 SDK) for HP-UX, version 2.1 or higher

### AIX

- IBM DB2 Client Application Enabler (CAE) for AIX, version 2.1 or higher
- IBM DB2 Software Development Kit (DB2 SDK) for AIX, version 2.1 or higher

---

## Configuring Data Sources



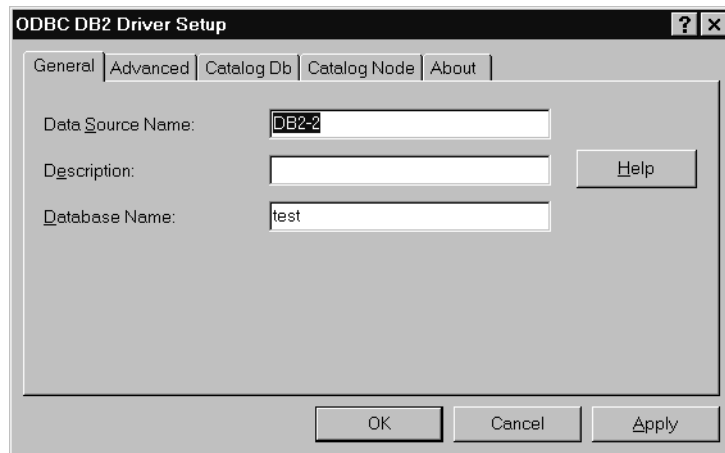
In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 3-1](#). You must also edit this file to perform a translation. For information about this file, see [Appendix H, “The UNIX Environment,” on page 369](#).



To configure a DB2 data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC DB2 Setup dialog box.

If you are configuring a new data source, click **Add** to display a list of installed drivers. Select the DB2 driver and click **Finish** to display the ODBC DB2 Setup dialog box.



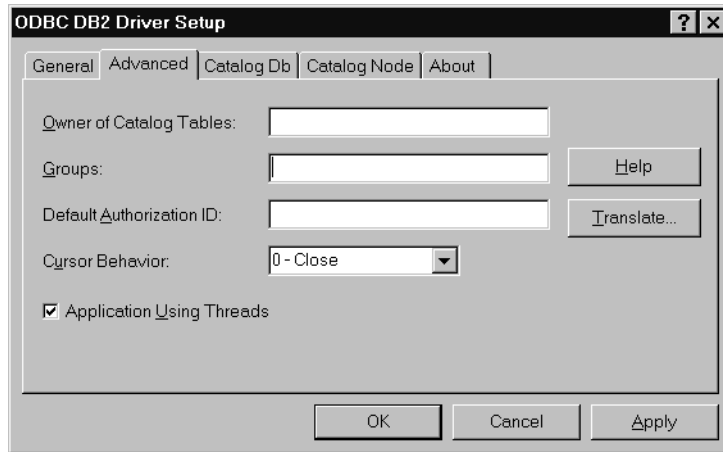
- 3 Specify values as follows; then, click **Apply**:

**Data Source Name:** A string that identifies this DB2 data source configuration in the system information. Examples include "Accounting" or "DB2-Serv1."

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

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

- 4 Click the **Advanced** tab to configure additional, optional settings for the data source.



5 Specify values as follows; then, click **Apply**:

**Owner of Catalog Tables:** 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 system administrator must create a view of the system tables in another account and give you permission to use that view. In this case, specify the Authorization ID for the account that owns the views of the system tables.

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

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

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

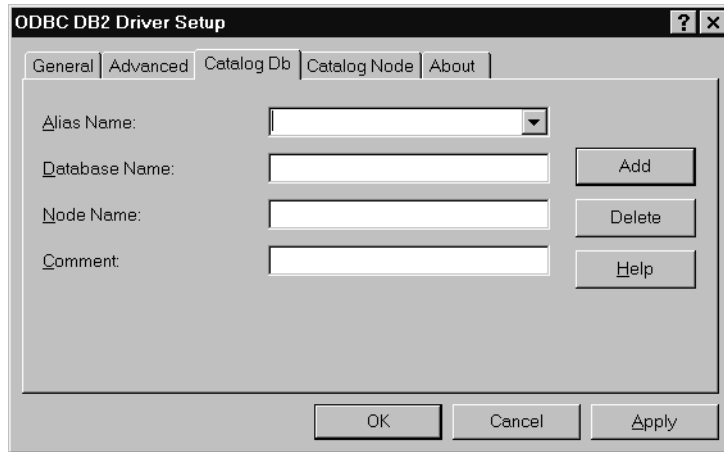
When you select Preserve, the driver returns SQL\_CB\_PRESERVE from SQLGetInfo (SQL\_CURSOR\_COMMIT\_BEHAVIOR). But only Select statements and prepared Update or Delete...Where Current of Cursor statements are preserved when the transaction ends. All other prepared statements are closed and deleted.

**Application Using Threads:** 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.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box and perform the translation.

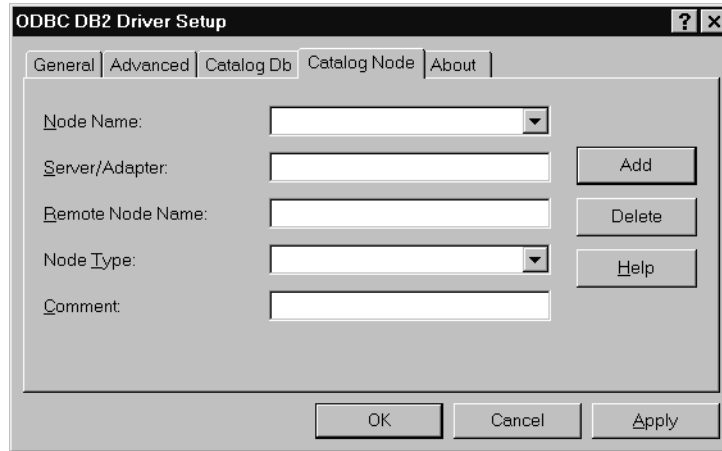
- 6 You must catalog any remote database that you want to access. To do so, click the **Catalog Db** tab.



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

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

- 7 To provide information about the remote site for the DB2 databases, click the **Catalog Node** tab.



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

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

- 8 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

## DB2

Access to DB2 requires that you bind and grant privileges to the INTERSOLV 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
```

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>
```

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

## Binding

The next step is to bind the INTERSOLV SQL files to the database. To do this, enter the following commands:



### Windows 95 and Windows NT

```
db2=> BIND QE.QECSV1.BND blocking all grant public
db2=> BIND QE.QERRV1.BND blocking all grant public
db2=> BIND QE.QERVV1.BND blocking all grant public
db2=> BIND QE.QECSWHV1.BND blocking all grant public
db2=> BIND QE.QERRWHV1.BND blocking all grant public
db2=> BIND QE.QEURWHV1.BND blocking all grant public
```



### Solaris

```
db2=> BIND odbcv1sol.bnd blocking all grant public
```

### HP-UX

```
db2=> BIND odbcv1hp.bnd blocking all grant public
```

### AIX

```
db2=> BIND odbcv1aix.bnd blocking all grant public
```

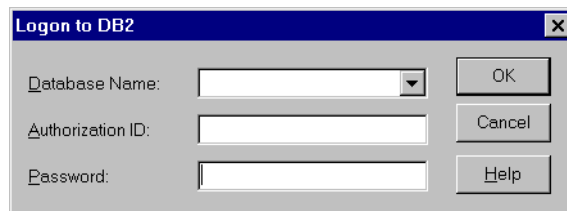
You may choose to give special options on the BIND command based on your installation. Consult the manual “Command Reference” in the DB2 manual set for a detailed list of BIND options.

To exit the DB2 command processor, enter the verb `quit`.

---

## Connecting to a Data Source Using a Logon Dialog Box

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



In this dialog box, do the following:

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

You must have cataloged any database you want to access from the client. (See the section [“Configuring Data Sources” on page 56](#) for information on how to do this.)

- 2 If required, type your user name (authorization ID).
- 3 If required, type your password.
- 4 Click **OK** to complete the logon and to update the values in the system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 3-1](#) gives the long and short names for each attribute, as well as a description.



To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. For information about this file, see [Appendix H, “The UNIX Environment,” on page 369](#).

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 your default.

---

**Table 3-1. DB2 Connection String Attributes**

---

<b>Attribute</b>	<b>Description</b>
DataSourceName (DSN)	A string that identifies a DB2 data source configuration in the system information. Examples include “Accounting” or “DB2-Serv1.”
Database (DB)	The name of the database to which you want to connect.
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.

**Table 3-1. DB2 Connection String Attributes** (cont.)

Attribute	Description
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 DB2/6000 on UNIX, normal UNIX security is used. The LogonID value is your UNIX user ID.
Password (PWD)	Password.
CursorBehavior (CB)	CursorBehavior={0   1}. This attribute determines whether cursors are preserved or closed at the end of each transaction. The initial default is 0 (close). Set this attribute to 1 if you want cursors to be held at the current position when the transaction ends. Doing so may impact the performance of your database operations.  This attribute is not valid for SQL/DS.  When CursorBehavior=1, the driver returns SQL_CB_PRESERVE from SQLGetInfo (SQL_CURSOR_COMMIT_BEHAVIOR). But only Select statements and prepared Update or Delete...Where Current of Cursor statements are preserved when the transaction ends. All other prepared statements are closed and deleted.
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.

---

## Data Types

Table 3-2 shows how the DB2 data types map to the standard ODBC data types.

---

**Table 3-2. DB2 Data Types**

---

<b>DB2</b>	<b>ODBC</b>
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,”](#) on page 335 for a discussion of these topics.

---

## ODBC Conformance Level

The DB2 driver supports the functions listed in [Appendix C, “ODBC API and Scalar Functions,”](#) on page 323. 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.

# 4 Connect ODBC for dBASE



Connect ODBC for dBASE (the “dBASE driver”) supports the following files in the Windows 95, Windows NT, UNIX, and Macintosh environments:

File Type	Operating Environments
dBASE III	Windows 95 and Window NT
dBASE IV and V	Windows 95 and Windows NT, UNIX, and Macintosh Power PC
Clipper	Windows 95 and Windows NT
FoxPro 2.5	Windows 95 and Windows NT, and Macintosh Power PC
FoxPro 3.0 database containers (DBC)	All platforms except Alpha NT and UNIX

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

---

## System Requirements

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

### Macintosh



Macintosh users who are accessing the same file must have file sharing enabled.

## Configuring Data Sources



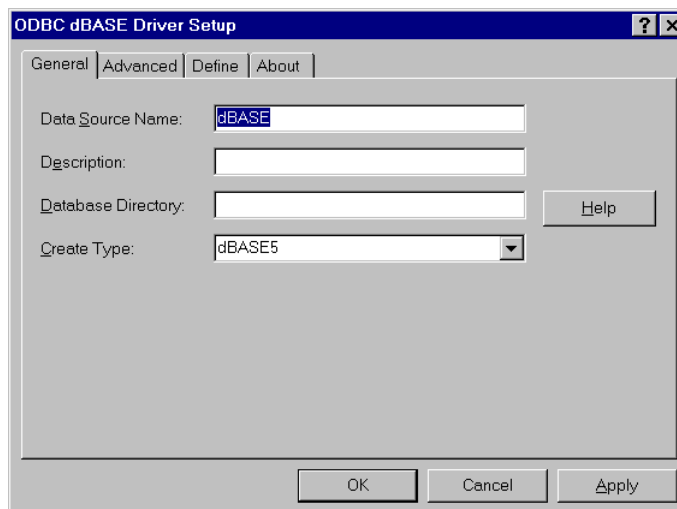
In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 4-1](#). You must also edit this file to perform a translation. For information about this file, see [Appendix H, “The UNIX Environment,” on page 369](#).

**Note:** To configure a data source for FoxPro 3.0 database containers (DBC), see [“Configuring FoxPro 3.0 DBC Data Sources” on page 76](#).

To configure a dBASE data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC dBASE Driver Setup dialog box.

If you are configuring a new data source, click **Add** to display a list of installed drivers. Select the dBASE driver and click **Finish** to display the ODBC dBASE Driver Setup dialog box:



3 Specify values as follows; then, click **Apply**:



**Apply** is not available on the Macintosh. Clicking **OK** saves the values.

**Data Source Name:** A string that identifies this dBASE data source configuration in the system information. Examples include "Accounting" or "dBASE Files."

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

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



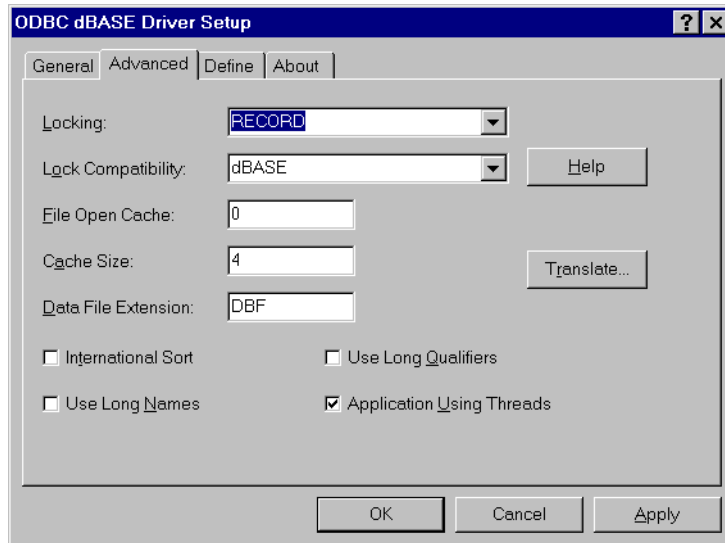
On the Macintosh, click **Select Directory**.

**Create Type:** The type of table or index to be created on a Create Table or Create Index statement. Select dBASE III, dBASE IV, dBASE V, Clipper, FoxPro1, FoxPro25, or FoxPro30. The default is dBASE V for all environments except the Macintosh.



On the Macintosh, the default create type is FoxPro25.

4 Click the **Advanced** tab to configure additional, optional settings for the data source.



5 Specify values as follows; then, click **Apply**:

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



On the Macintosh, the default locking level is None.

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

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



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

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

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

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



On the Macintosh, the default lock compatibility is Fox.

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

**Cache Size:** The number of 64 KB 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 default is 4.

**Data File Extension:** A setting to specify the file extension to use for data files. The default setting is DBF. The setting cannot be greater than three characters, and it cannot be one the driver already uses, such as MDX or CDX. The Data File Extension setting is used for all Create Table statements. Sending a Create Table using an extension other than the value specified for this option 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.

**International Sort:** A setting to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this box to use 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.

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

**Use Long Qualifiers:** Select this check box to use long path names as table qualifiers. When you select this check box, path names can be up to 255 characters. The default length for path names is 128 characters.

**Applications Using Threads:** 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.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box and perform the translation.



- 6 On the Macintosh, click the **Mac File Types** tab to specify the creator and file types. Specify 4-character, case-sensitive values for the following:

Database creator type (default is FOXX)

- Database file type (default is F+DB)
- Memo file type (default is F+DT)
- Single Entry Index file type (default is F+IX)
- Multiple Entry Index file type (default is FCDX).

You can also search for additional files by file type. Enter the file types in the search field as a comma-separated list. The default types are F+DB and BINA.

- 7 If you use index files that have different names than their corresponding data files and you have not defined this association, click the **Define** tab. See [“Defining Index Attributes” on page 82](#) for step-by-step instructions.

- 8 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Configuring FoxPro 3.0 DBC Data Sources

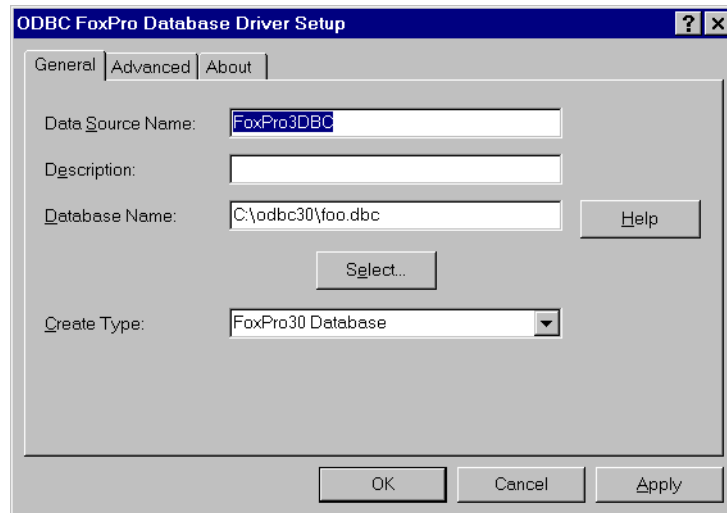
To configure a data source for FoxPro 3.0 database containers:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC FoxPro Driver Setup dialog box.

If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the dBASE driver and click **Finish** to display the ODBC FoxPro Driver Setup dialog box.



On the Macintosh, select the FoxProDB driver from the list of installed drivers.



3 Specify values as follows; then, click **Apply**:



**Apply** is not available on the Macintosh. Clicking **OK** saves the values.

**Data Source Name:** A string that identifies this dBASE data source configuration in the system information, for example, "Accounting."

**Description:** An optional long description of a data source name, for example, "My Accounting Database."

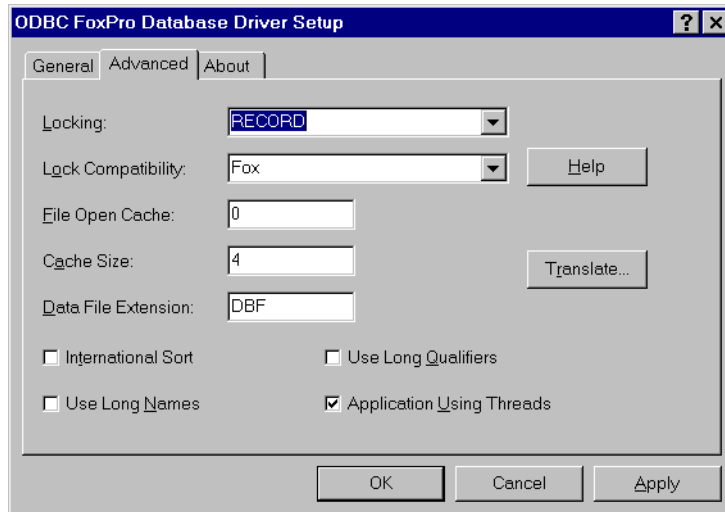
**Database Name:** A path specification to the directory that contains the database files. If none is specified, the current working directory is used. Click **Select** to browse the available FoxPro 3.0 database containers.



On the Macintosh, click **Select Database**.

**Create Type:** You cannot change the Create Type for the FoxPro 3.0 DBC driver.

4 Click the **Advanced** tab to configure additional, optional settings for the data source.



5 Specify values as follows; then, click **Apply**:

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



On the Macintosh, the default locking level is None.

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

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

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

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

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

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



On the Macintosh, the default lock compatibility is Fox.

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

**Cache Size:** The number of 64 KB 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 default is 4.

**Data File Extension:** A setting to specify the file extension to use for data files. The default setting is DBF. The setting cannot be greater than three characters, and it cannot be one the driver already uses, such as MDX or CDX. The Data File Extension setting is used for all Create Table statements. Sending a Create Table using an extension other than the value specified for this option 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.

**International Sort:** A setting to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this box to use 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.

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



**Use Long Qualifiers:** Select this check box to use long path names as table qualifiers. When you select this check box, path names can be up to 255 characters. The default length for path names is 128 characters.

**Application Using Threads:** 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.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box and perform the translation.



- 6 On the Macintosh, click the **Mac File Types** tab to specify the creator and file types. Specify 4-character, case-sensitive values for the following:

Database creator type (default is FOXX)

- Database file type (default is F+DB)
- Memo file type (default is F+DT)
- Single Entry Index file type (default is F+IX)
- Multiple Entry Index file type (default is FCDX).
- Database Container file type (default is FDBC)

- 7 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Defining Index Attributes

**Note:** This section does not apply to UNIX platforms. See [“Defining Index Attributes on UNIX Platforms” on page 84](#) for information on how to set index attributes on the UNIX platforms.

The Define tab of the ODBC dBASE Driver Setup dialog box allows you to define the attributes of index files. With the exception of Clipper and dBASE III, the family of databases that includes dBASE and FoxPro uses a multiple index file associated with a particular table (database file). This index file has a .MDX or .CDX extension and is automatically maintained by the driver. Tags within this index can be marked as unique.

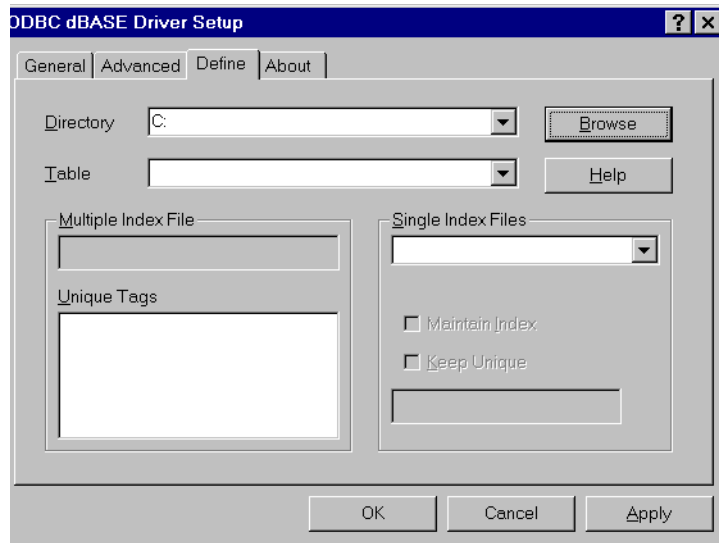
Clipper and dBASE III use single index files that are not automatically associated with a particular table. You can choose to have the driver maintain an index and choose whether or not the index is unique.

To define index file attributes, follow these steps:

- 1 Display the ODBC dBASE Driver Setup dialog box.
- 2 Click the **Define** tab.
- 3 Select a directory and table (database file). To display a directory listing, click **Browse**.



On the Macintosh, click the pop-up menu to select a directory and table.



The upper section of the Define tab displays the directory name and the table containing the database information. The lower section of the tab displays the index information for that table.

**4** Specify values as follows; then, click **Apply**.



On the Macintosh, click **Save Changes**.

**Multiple Index File:** Any multiple index file (with a .CDX extension or .MDX extension) associated with the table will be displayed in this field. This index file cannot be marked as unique, but tags within it can be. This field is not displayed on the Macintosh.

**Unique Tags:** Tags associated with the multiple index file will appear in the list. To mark a tag as unique, single-click it; it will remain selected until you single-click it again. You can mark multiple tags in this manner.

**Note:** The Single Index Files pane is active only if you have selected a dBASE III or Clipper table. It is not available on the Macintosh.

**Single Index Files:** To define the attributes of a single index file, select the file from the drop-down file list.

**Maintain Index:** select this check box to associate the specified single index file with the selected table.

**Keep Unique:** select this check box to specify that the single index file is unique.

- 5 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Defining Index Attributes on UNIX Platforms



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 INTERSOLV dBASE driver. The driver must be notified that index tags are unique. No configuration is needed for unique indexes that were created using the INTERSOLV dBASE driver. When using files that were not created with the INTERSOLV 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
```

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 4-1** gives the long and short names for each attribute, as well as a description.



To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. For information about this file, see [Appendix H, “The UNIX Environment,”](#) on page 369.

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 your default.

---

**Table 4-1. dBASE Connection String Attributes**

---

Attribute	Description
DataSourceName (DSN)	A string that identifies a dBASE data source configuration in the system information. Examples include “Accounting” or “dBASE Files.”
Database (DB)	The directory in which the dBASE files are stored.
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 for all environments except the Macintosh; on the Macintosh, the initial default is FoxPro25. See <a href="#">“Connect ODBC for dBASE”</a> on page 69 for the Create Types that are valid for your platform.

**Table 4-1. dBASE Connection String Attributes**

Attribute	Description
LockCompatibility (LCOMP)	<p data-bbox="679 314 1329 404">LockCompatibility={Q+E   Q+EVirtual   dBASE   Clipper   Fox}. The locking scheme to be used in your dBASE tables.</p> <ul style="list-style-type: none"> <li data-bbox="679 430 1329 699">■ 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.</li> <li data-bbox="679 725 1329 890">■ 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.</li> </ul> <p data-bbox="679 916 1329 1034">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.</p> <ul style="list-style-type: none"> <li data-bbox="679 1060 1329 1150">■ LockCompatibility=dBASE specifies Borland-compatible locking. This is the initial default for all platforms except Macintosh.</li> <li data-bbox="679 1177 1329 1237">■ LockCompatibility=Clipper specifies Clipper-compatible locking.</li> <li data-bbox="679 1263 1329 1366">■ LockCompatibility=Fox specifies FoxPro-compatible locking. This is the initial default on the Macintosh.</li> </ul>

---

**Table 4-1. dBASE Connection String Attributes**


---



Attribute	Description
LockCompatibility (LCOMP) ( <i>cont.</i> )	<p>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.</p>
Locking (LCK)	<p>Locking={NONE   RECORD   FILE}. This attribute determines the level of locking for the database tables.</p> <p>Locking=None offers the best performance but is intended only for single-user environments. This is the initial default on the Macintosh.</p> <p>Locking=Record locks only the records affected by the statement. This is the initial default for all platforms except Macintosh.</p> <p>Locking=File locks all of the records in the table.</p>
FileOpenCache (FOC)	<p>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.</p>



**Table 4-1. dBASE Connection String Attributes**

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.
IntlSort (IS)	<p data-bbox="679 586 1329 803">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."</p> <p data-bbox="679 821 1329 1003">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.</p>
ModifySQL (MS)	ModifySQL={0   1}. This attribute is provided for backward compatibility with earlier versions of INTERSOLV 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.

**Table 4-1. dBASE Connection String Attributes**

Attribute	Description
UltraSafeCommit (USF)	UltraSafeCommit={0   1}. This attribute specifies when the driver flushes the file cache. If UltraSafeCommit=1, the driver updates directory entries after each Commit. This decreases performance. If UltraSafeCommit=0 (the default) the driver updates the directory entry when the file is closed. In this case, a machine "crash" before closing the file causes newly inserted records to be lost.
ExtensionCase (EC)	ExtensionCase={LOWER   UPPER}. This attribute specifies whether upper- or lowercase file extensions are accepted. If ExtensionCase=Lower, lowercase extensions are accepted. If ExtensionCase=Upper (the default), uppercase extensions are accepted.
 MacFileInfo (MFI)	On Macintosh systems, four-character, case-sensitive values that specify the following in the order shown: <ul style="list-style-type: none"> <li>■ Database Creator (initial default is FOXX)</li> <li>■ Database file (initial default is F+DB)</li> <li>■ Memo file (initial default is F+DT)</li> <li>■ Single Entry Index file (initial default is F+IX)</li> <li>■ Multiple Entry Index file (initial default is FCDX)</li> <li>■ Database Container file (default it FDBC).</li> </ul> The values are specified in a comma-separated list. For example, MacFileInfo=ABCD,EFGH,IJKL,MNOP,QRST.
 MacFileTypesList (MFTL)	On Macintosh systems, allows the user to specify which file types are included. The default types are F+DB and BINA. The values are specified in a comma-separated list. For example, MacFileTypesList=ABCD,EFGH,IJKL,MNOP,QRST.

**Table 4-1. dBASE Connection String Attributes**

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

## Data Types

[Table 4-2](#) shows how dBASE data types map to the standard ODBC data types. These dBASE data types can be used in a Create Table statement. For the syntax of the Create Table statement, see [Appendix A, “SQL for Flat-File Drivers,” on page 289](#).

[Table 4-3](#) shows how the additional FoxPro 3.0 tables and database containers map to the ODBC data types.

**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 4-2. dBASE Data Types**

---

dBASE	ODBC
Binary <sup>1</sup>	SQL_LONGVARBINARY
Char <sup>2</sup>	SQL_CHAR
Date	SQL_TYPE_DATE
Float <sup>3</sup>	SQL_DECIMAL

---

<sup>1</sup> dBASE V only

<sup>2</sup> 254 characters maximum (1024 for Clipper)

<sup>3</sup> dBASE IV and V only

<sup>4</sup> FoxPro and dBASE V only

<sup>5</sup> FoxPro and Clipper

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

General <sup>4</sup>	SQL_LONGVARIABLE
Logical	SQL_BIT
Memo	SQL_LONGVARCHAR
Numeric	SQL_DECIMAL
Picture <sup>5</sup>	SQL_LONGVARIABLE

<sup>1</sup> dBASE V only<sup>2</sup> 254 characters maximum (1024 for Clipper)<sup>3</sup> dBASE IV and V only<sup>4</sup> FoxPro and dBASE V only<sup>5</sup> FoxPro and Clipper**Table 4-3. Additional FoxPro 3.0 Data Types**

<b>FoxPro 3.0</b>	<b>ODBC</b>
Character (binary)	SQL_CHAR
Currency	SQL_DOUBLE
Datetime	SQL_TYPE_TIMESTAMP
Double	SQL_DOUBLE
Integer	SQL_INTEGER
Memo (binary)	SQL_LONGVARIABLE

---

## Column Names

Column names in SQL statements (such as Select and Insert, for example) 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," on page 289](#). 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 setup dialog box (for UNIX, edit the system information file) or as a connection string option. The index can be:

- dBASE III (.NDX)
- dBASE IV or V (.MDX)
- Clipper (.NTX)
- FoxPro (.CDX)

The syntax for creating an index is:

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

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

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

*base\_table\_name* is the name of the database file whose index is to be created. The .DBF extension is not required; the driver



automatically adds it if it is not present. By default, dBASE IV or V 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 4-4](#) 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 4-4. dBASE-Compatible Index Summary**

Create Type/Ext.	dBASE UNIQUE	DESC	Max Size of Key Column	Max Size of Column Spec.	Production/ Structural Indexes	Supports FOR Expressions
dBASE III .NDX	Yes	No	100	219	No	No
Clipper .NTX	Yes	Yes	250	255	No	Yes
dBASE IV and 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.

For FoxPro and dBASE IV or V, *index\_name* is the tag. Otherwise, *index\_name* is the name of the index file without the extension.

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

```
DROP INDEX emp.emphire
```

---

## Pack Statement

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

```
PACK filename
```

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

```
PACK emp
```

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

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

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

---

## SQL Statements for FoxPro 3.0 Database Containers

The FoxPro DBC driver supports four additional SQL statements:

- Create Database
- Add Table
- Remove Table
- Use

To create a new FoxPro 3.0 database container, use

```
CREATE DATABASE database_name
```

To add an existing table to the database container, use

```
ADD TABLE table_name
```

To remove a table from the database container (not delete the table, but unlink it from the database container), use

```
REMOVE TABLE table_name
```

To set the current database container to an existing database container, use

```
USE database_name
```

To add or delete columns from a table in a database container, use the Alter Table statement (see ["Alter Table Statement" on page 95](#)).

---

# Locking

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

## Levels of Database Locking

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

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

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

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

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

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

## Using Locks on Local Files



If you use database locking and are accessing files locally (not on a network), run the DOS utility SHARE.EXE before running Windows 95. If you add SHARE.EXE to your AUTOEXEC.BAT file, it runs automatically each time you boot your computer.



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

## Limit on Number of Locks

There is a limit on the number of locks that can be placed on a file. If you are accessing a dBASE file from a server, the limit depends on the server (see your server documentation).



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

## How Transactions Affect Record Locks

When an Update or Delete statement is 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,” on page 335](#) for a discussion of these topics.

---

## ODBC Conformance Level

The dBASE driver supports the functions listed in [Appendix C, “ODBC API and Scalar Functions,” on page 323](#). 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.





# 5 Connect ODBC for Excel Workbook



Connect ODBC for Excel Workbook (the “Excel Workbook driver”) is supported in the Windows 95 and Windows NT environments.

See the README file shipped with your INTERSOLV DataDirect product for the file name of the Excel Workbook driver.

---

## System Requirements

The Excel Workbook driver requires the OLE32.DLL. This DLL is typically provided by the Windows System. It is also provided with Excel version 5 or 7. The Excel Workbook driver does not require Excel 5 or 7 to read .XLS files. To modify the files, however, you must use the Excel application.

---

## Using an Excel Database

An Excel workbook database is any Excel 5 or 7 or .XLS workbook file that contains one or more named lists containing record data. Each named list is treated as a table within the workbook database. The lists must be set up as follows:

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

To name a table from within Excel, highlight all the columns that are part of the table. After the columns are highlighted, from the menu bar select **Insert**, then **Name**, then **Define**. Type the table name in the dialog box to describe the highlighted table.

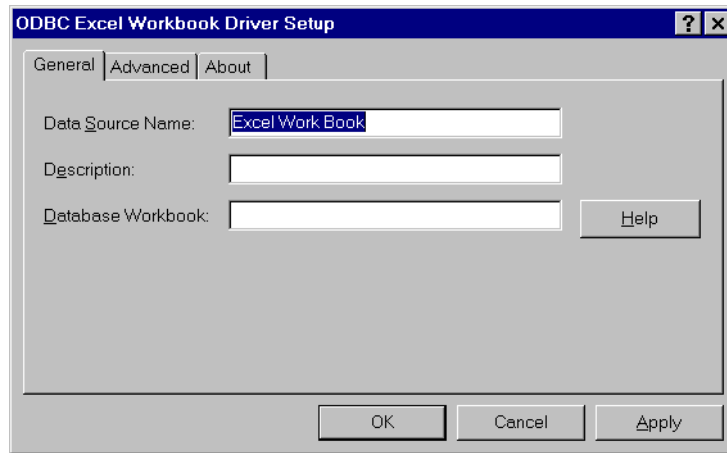
---

## Configuring Data Sources

To configure an Excel data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Excel Workbook Driver Setup dialog box.

If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Excel Workbook driver and click **Finish** to display the ODBC Excel Workbook Driver Setup dialog box.



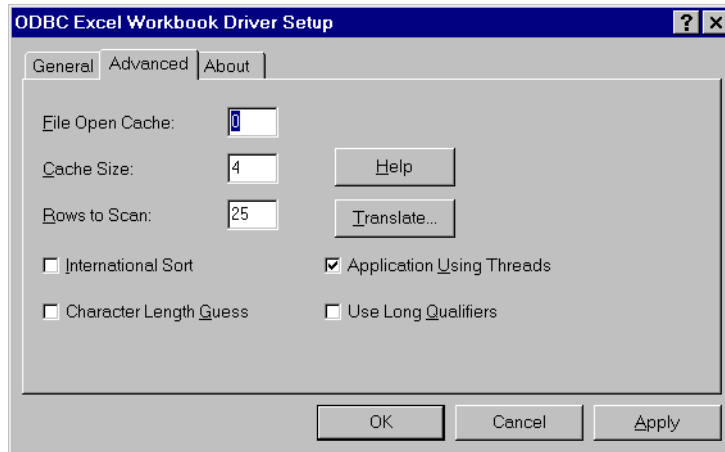
**3** Specify values as follows; then, click **Apply**:

**Data Source Name:** A string that identifies this Excel data source configuration in the system information. Examples include "Accounting" or "Excel Database."

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

**Database Workbook:** Identifies the workbook file containing the Excel database.

**4** Click the **Advanced** tab to configure additional, optional settings for the data source.



5 Specify values as follows; then, click **Apply**:

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

**Cache Size:** The number of 64 KB 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. The value must be a multiple of 64. The default is 256. 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.

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

**International Sort:** A setting to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this box to use 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.

**Character Length Guess:** Determines whether the driver tries to guess the length of character columns. If the check box is cleared (the default), the driver does not try to guess the length of character columns; instead, it assumes that all character columns have a length of 255. If set to 1, the driver tries to guess the length.

**Application Using Threads:** 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.



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

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 6 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these default values by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 Excel is:

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

[Table 5-1](#) 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 your default.

---

**Table 5-1. Excel Connection String Attributes**

---


<b>Attribute</b>	<b>Description</b>
DataSourceName (DSN)	A string that identifies an Excel data source configuration in the system information. Examples include "Accounting" or "Excel Files."
Database (DB)	The name of the .XLS workbook file containing the Excel database.
ScanRows (SR)	The number of rows to scan to determine the column data types. A value of 0 means to scan the whole table. The initial default is 25.
FileOpenCache (FOC)	The maximum number of unused file opens to cache. For example, when FileOpenCache=4, and a user opens and closes four files, the files are not actually closed. The driver keeps them open so that if another query uses one of these files, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The initial default is 0.
CacheSize (CSZ)	The number of 64 KB 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.

**Table 5-1. Excel Connection String Attributes** (cont.)

Attribute	Description
IntlSort (IS)	<p data-bbox="648 314 1260 565">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."</p> <p data-bbox="648 583 1260 800">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.</p>
FieldGuessing (FG)	<p data-bbox="648 817 1260 1003">FieldGuessing={0   1}. This attribute determines whether the driver will try to guess column data types. If FieldGuessing=1 (the default), the driver attempts to guess the data types. If FieldGuessing=0, the driver assumes that all data types are SQL_CHAR.</p>
CharacterLength Guess (CLG)	<p data-bbox="648 1020 1260 1239">CharacterLengthGuess={0   1}. This attribute determines whether the driver tries to guess the length of character columns. If set to 0 (the initial default), the driver does not try to guess the length of character columns; instead, it assumes that all character columns have a length of 255. If set to 1, the driver tries to guess the length.</p>
ModifySQL (MS)	<p data-bbox="648 1256 1260 1569">ModifySQL={0   1}. This attribute is provided for backward compatibility with earlier versions of Q+E products. It determines whether the driver modifies SQL statements to conform to ODBC specifications or passes the SQL statement directly to Excel. Specify ModifySQL=1 to have the driver modify the SQL statement to conform to ODBC specifications. Specify ModifySQL=0 to have the driver understand SQL dialects found in earlier drivers. The default is 1.</p>



**Table 5-1. Excel Connection String Attributes** (cont.)

Attribute	Description
UseLongQualifiers (ULQ) 	UseLongQualifiers={0   1}. It specifies whether the driver uses long pathnames as table qualifiers. The default is 0, do not use long pathnames (the default length of pathnames is 128 characters). If UseLongQualifiers=1, the driver uses long pathnames (up to 255 characters).
ApplicationUsing Threads	ApplicationUsingThreads={0   1}. Ensures that the drivers works with multi-threaded applications. The default is 1, which makes the driver thread-safe. When using the driver with single-threaded applications, you can set this option to 0 to avoid additional processing required for ODBC thread safety standards.

## Data Types

[Table 5-2](#) shows how Excel data types map to the standard ODBC data types.

**Table 5-2. Excel Data Types**

Excel	ODBC
Char	SQL_VARCHAR
Date	SQL_TYPE_DATE
Logical	SQL_BIT
Number	SQL_DOUBLE
Float	
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP

---

## Table and Column Names

Table names are case-sensitive when using the Excel Workbook driver.

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

For example:

```
SELECT `name` FROM emp
```

---

## Select Statement

You use a SQL Select statement to specify the columns and records to be read. Excel Select statements support all of the Select statement clauses as described in [Appendix A, "SQL for Flat-File Drivers," on page 289](#).

---

## Create Table Statement

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

---

## ODBC Conformance Level

The Excel Workbook driver supports the functions listed in [Appendix C, “ODBC API and Scalar Functions,” on page 323](#). In addition, the following function is supported: SQLSetPos.

The Excel Workbook driver supports backward and random fetching in SQLExtendedFetch or SQLFetchScroll. The drivers support the minimum SQL grammar.

---

## Number of Connections and Statements Supported

The Excel Workbook driver supports multiple connections and multiple statements per connection.



## 6 Connect ODBC for INFORMIX



Connect ODBC for INFORMIX supports two separate drivers. Connect ODBC for INFORMIX (the “INFORMIX driver”) supports multiple connections to the INFORMIX database system versions 5.x, 6.x, or 7.x in the Windows 95, Windows NT, and UNIX environments.



Connect ODBC for INFORMIX 9 (the “INFORMIX 9 driver”) supports multiple connections to the INFORMIX database system versions 7.x and 9.x in the Windows 95 and Windows NT environments.

See the README file shipped with your INTERSOLV DataDirect product for the file names of the INFORMIX drivers.

---

### System Requirements

The following section lists requirements for all supported platforms.

#### Windows 95 and Windows NT



Both INFORMIX and INFORMIX 9 are supported on Windows 95 and Windows NT.

## ***INFORMIX***

To access remote INFORMIX 5.x, 6.x, or 7.x databases through the INFORMIX driver, you need INFORMIX-Connect 7.2 for Windows 95 and Windows NT from INFORMIX.

**Note:** The DataDirect INFORMIX driver for Windows 95 and Windows NT does not work with versions of INFORMIX-Connect earlier than 7.2.

Use the SETNET32.EXE utility supplied with INFORMIX-Connect 7.2 to define servers and the location of the INFORMIX directory. Use ILOGIN.EXE to test your connection to the INFORMIX server.

The INFORMIX-Connect 7.2 package includes ISQLT07C.DLL. The path to this DLL must be in your PATH environment variable. If it is not and you attempt to configure a data source, a message similar to the following appears:

```
The setup routines for the INTERSOLV 3.00 32-BIT
INFORMIX ODBC driver could not be loaded due to
system error code 126.
```

When you click **OK**, the following message appears:

```
Could not load the setup or translator library.
```

## ***INFORMIX 9***

To access remote INFORMIX 7.x or 9 databases through the INFORMIX 9 driver, you need INFORMIX-Connect 9.1.3 or greater for Windows 95 and Windows NT from INFORMIX.

Use the SETNET32.EXE utility supplied with INFORMIX-Connect 9.1.3 to define servers and the location of the INFORMIX directory. Use ILOGIN.EXE to test your connection to the INFORMIX server.

The INFORMIX-Connect 9.1.3 package includes ISQLT09A.DLL. The path to this DLL must be in your PATH environment variable. If it is not and you attempt to configure a data source, a message similar to the following appears:

```
The setup routines for the INTERSOLV 3.00 32-BIT
INFORMIX ODBC driver could not be loaded due to
system error code 126.
```

When you click **OK**, the following message appears:

```
Could not load the setup or translator library.
```

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



The INFORMIX driver under UNIX requires INFORMIX-Connect or ESQ-L-C 7.23.

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 Online Dynamic Server Administrator's Guide, Volume 2* or the *INFORMIX UNIX Installation Guide*.

## Configuring Data Sources

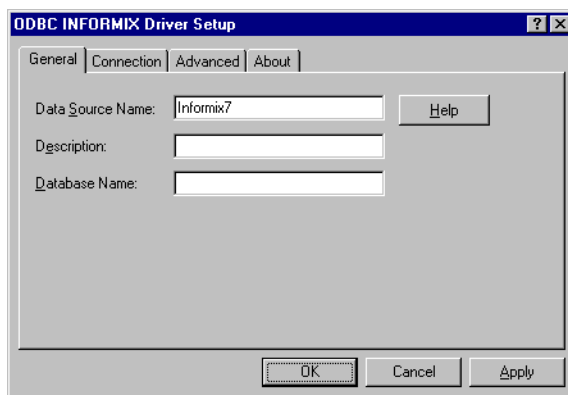


In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 6-1](#). You must also edit this file to perform a translation. For information about this file, see [Appendix H, “The UNIX Environment,” on page 369](#).

To configure an INFORMIX data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC INFORMIX Driver Setup dialog box.

If you are configuring a new data source, click **Add** to display a list of installed drivers. Select the INFORMIX driver and click **Finish** to display the ODBC INFORMIX Driver Setup dialog box.



- 3 Specify values as follows; then, click **Apply**:

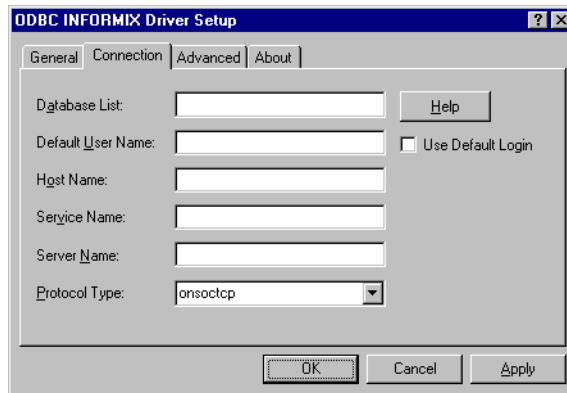
**Data Source Name:** A string that identifies this INFORMIX data source configuration in the system information. Examples include “Accounting” or “INFORMIX-Serv1.”



**Description:** An optional long description of a data source name. For example, “My Accounting Database” or “INFORMIX 7 files on Server number 1.”

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

- 4 Click the **Connection** tab to configure additional, optional settings for the data source.



- 5 Specify values as follows; then, click **Apply**.

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

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

**Use Default Login:** Select this check box to read the Logon ID and Password entries directly from the INFORMIX registry. The check box is cleared by default; that is, logon information is read from the system information, the connection string, or the Logon to INFORMIX dialog box.

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

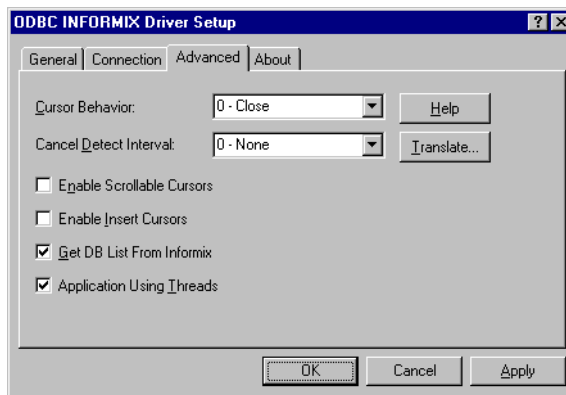
**Service Name:** The name of the service as it appears on the host machine. This service is assigned by the system administrator. The name you specify is displayed in the INFORMIX Server Options dialog box.

**Server Name:** The name of the INFORMIX server as it appears in the sqlhosts file.



**Protocol Type:** The protocol used to communicate with the server. Specify one or more values; separate the names with commas. Values can be olsocsp, olsocsp, onsocsp, onsoctcp, seipcp, sesocsp, and/or sesocsp.

- Click the **Advanced** tab to configure additional, optional settings for the data source.



- Specify values as follows; then, click **Apply**:

**Cursor Behavior:** Holds cursor at the current position when the transaction ends if you select Preserve. Otherwise, leave this set to Close. Selecting Preserve may impact the performance of your database operations.

**Cancel Detect Interval:** 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 the request has completed execution.

**Enable Scrollable Cursors:** Determines whether the driver provides scrollable cursors. The check box is cleared by default (no use of scrollable cursors). The INFORMIX driver can use scrollable cursors only if there are no long columns (SQL\_LONGVARCHAR or SQL\_LONGVARBINARY) in a Select list. If you select this check box, you must not include long columns in the Select list.

**Enable Insert Cursors:** Determines whether the driver can use Insert cursors during parameterized inserts. 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 this check box is cleared (the default), the driver does not use Insert cursors.

**Get DB List From Informix:** 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 the check box is selected, the default, the driver requests the database list from the INFORMIX server. When the check box is cleared, the driver uses the list that was entered by the user at driver setup.

**Application Using Threads:** 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. Clearing this check box avoids the additional processing required for ODBC thread-safety standards.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that

translates your data from the IBM PC character set to the ANSI character set.

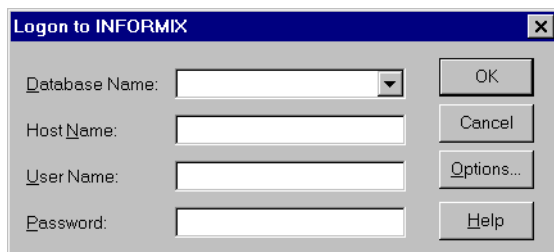
Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 8 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Connecting to a Data Source Using a Logon Dialog Box

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

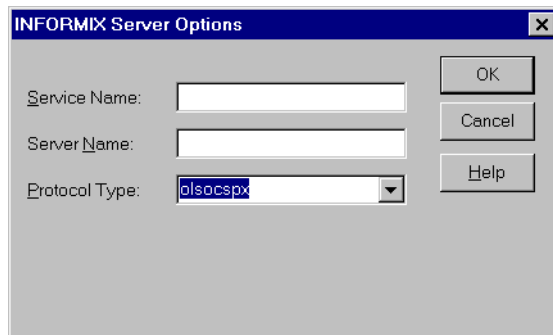


In this dialog box, do the following:

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

displayed in this drop-down list are returned from the INFORMIX server.

- 2 Type the name of the server (host name) on which INFORMIX resides.
- 3 If required, type your user name as specified on the INFORMIX server.
- 4 If required, type your password.
- 5 Optionally, click **Options** to display the INFORMIX Server Options dialog box, where you can change the Service Name, Server Name, and Protocol Type that you specified in the ODBC INFORMIX Driver Setup dialog box. Click **OK** to save your changes.



- 6 Click **OK** to complete the logon and to update these values in the system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 6-1](#) gives the long and short names for each attribute, as well as a description.



To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. For information about this file, see [Appendix H, “The UNIX Environment,”](#) on page 369.


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 your default.

---

**Table 6-1. INFORMIX Connection String Attributes**

---

<b>Attribute</b>	<b>Description</b>
DataSourceName (DSN)	A string that identifies an INFORMIX data source configuration in the system information. Examples include "Accounting" or "INFORMIX-Serv1."
Database (DB)	The name of the database to which you want to connect.
HostName (HOST)	The name of the machine on which the INFORMIX server resides.
UseDefaultLogin (UDL)	UseDefaultLogin={0   1}. Specify 1 to read the Logon ID and Password directly from the INFORMIX registry. The default is 0; that is, logon information is read from the system information, the connection string, or the Logon to INFORMIX dialog box.
LogonID (UID)	Your user name as specified on the INFORMIX server.
Password (PWD)	A password.
Service (SERV)	The name of the service as it appears on the host machine. This service is assigned by the system administrator.
ServerName (SRVR)	The name of the server running the INFORMIX database.
Protocol (PRO) 	Protocol={olsocspix   olsoctcp   onsocspix   onsoctcp   seipcpip   sesocspix   sesoctcp}. The protocol used to communicate with the server. You can specify one or more values; separate the names with commas.
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.

**Table 6-1. INFORMIX Connection String Attributes** (cont.)

Attribute	Description
EnableScrollable Cursors (ESC)	EnableScrollableCursors={0   1}. This attribute determines whether the driver provides scrollable cursors. The initial default value is 0 (no use of scrollable cursors). The INFORMIX driver can use scrollable cursors only if there are no long columns (SQL_LONGVARCHAR or SQL_LONGVARBINARY) in a Select list. If you set this option to use scrollable cursors (EnableScrollableCursors=1), you must not include long columns in the Select list.
EnableInsert Cursors (EIC)	EnableInsertCursors={0   1}. Determines whether the driver can use Insert cursors during parametrized inserts. 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.



---

**Table 6-1. INFORMIX Connection String Attributes** (cont.)

---

<b>Attribute</b>	<b>Description</b>
ApplicationUsingThreads (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.
CancelDetectInterval (CDI)	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.

---

## Data Types

Table 6-2 shows how the INFORMIX data types map to the standard ODBC data types.

**Table 6-2. INFORMIX Data Types**

<b>INFORMIX</b>	<b>ODBC</b>
Byte <sup>1</sup>	SQL_LONGVARBINARY
Char	SQL_CHAR
Date	SQL_TYPE_DATE
Datetime year to fraction(5)	SQL_TYPE_TIMESTAMP
Datetime year to fraction(f) <sup>2</sup>	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) <sup>2</sup>	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) <sup>2</sup>	SQL_INTERVAL_DAY_TO_SECOND

<sup>1</sup> Not supported for Standard Engine Databases

<sup>2</sup> Fraction(f) types are mapped to fraction(5) in the driver. The precision is type dependent and the scale as 5.

---

**Table 6-2. INFORMIX Data Types** (cont.)
 

---

<b>INFORMIX</b>	<b>ODBC</b>
Interval hour(p) to hour	SQL_INTERVAL_HOUR
Interval hour(p) to minute	SQL_INTERVAL_HOUR_TO_MINUTE
Interval hour(p) to second	SQL_INTERVAL_HOUR_TO_SECOND
Interval hour(p) to fraction(f) <sup>2</sup>	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) <sup>2</sup>	SQL_INTERVAL_MINUTE_TO_SECOND
Interval second(p) to second	SQL_INTERVAL_SECOND
Interval second(p) to fraction(f) <sup>2</sup>	SQL_INTERVAL_SECOND
Interval fraction to fraction(f) <sup>2</sup>	SQL_VARCHAR
Money	SQL_DECIMAL
Serial	SQL_INTEGER
Smallfloat	SQL_REAL
Smallint	SQL_SMALLINT
Text <sup>1</sup>	SQL_LONGVARCHAR
Varchar <sup>1</sup>	SQL_VARCHAR

---

<sup>1</sup> Not supported for Standard Engine Databases

<sup>2</sup> Fraction(f) types are mapped to fraction(5) in the driver. The precision is type dependent and the scale as 5.

## INFORMIX 9

[Table 6-3](#) 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 6-2](#).

---

**Table 6-3. INFORMIX 9 Data Types**

---

INFORMIX	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,"](#) on page 335 for a discussion of these topics.

---

## ODBC Conformance Level

The INFORMIX driver supports the functions listed in [Appendix C, "ODBC API and Scalar Functions,"](#) on page 323. In addition, the following X/Open functions are supported:

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

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

---

## Number of Connections and Statements Supported

The INFORMIX driver supports multiple connections and multiple statements per connection to the INFORMIX database system.



# 7 Connect ODBC for OpenIngres



Connect ODBC for OpenIngres supports two separate drivers. Connect ODBC for OpenIngres (the “OpenIngres driver”) supports OpenIngres 1.2 in the Windows 95, Windows NT, HP-UX, AIX, and Solaris environments.



Connect ODBC for OpenIngres 2 (the “OpenIngres 2 driver”) supports OpenIngres 1.2 and 2.0 in the Windows 95 and Windows NT environments.

See the README file shipped with your INTERSOLV DataDirect product for the file names of the OpenIngres drivers.

---

## System Requirements

The following section lists requirements for all supported platforms.

### Windows 95 and Windows NT



Both OpenIngres and OpenIngres 2 are supported on Windows 95 and Windows NT.

### *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***

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

```
SET II_SYSTEM=C:\OPING
```

Two directories in `II_SYSTEM`, `INGRES\BIN` and `INGRES\UTILITY`, must be on your path. If you attempt to configure a data source without these directories on your path, a message similar to the following appears:

```
The setup routines for the INTERSOLV 3.00 32-BIT
OpenIngres driver could not be loaded due to
system error code 126.
```

When you click **OK**, the following message appears:

```
Could not load the setup or translator library.
```

## **UNIX**



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 Data Sources

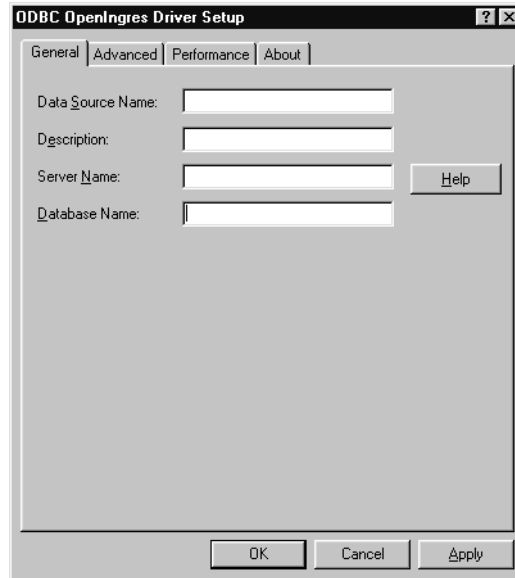


In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 7-1](#). You must also edit this file to perform a translation. For information about this file, see [Appendix H, “The UNIX Environment,” on page 369](#).

To configure an OpenIngres data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC OpenIngres Driver Setup dialog box.

If you are configuring a new data source, click **Add** to display list of installed drivers. Select the OpenIngres driver and click **Finish** to display the ODBC OpenIngres Driver Setup dialog box.



- 3 Specify values as follows; then, click **Apply**:

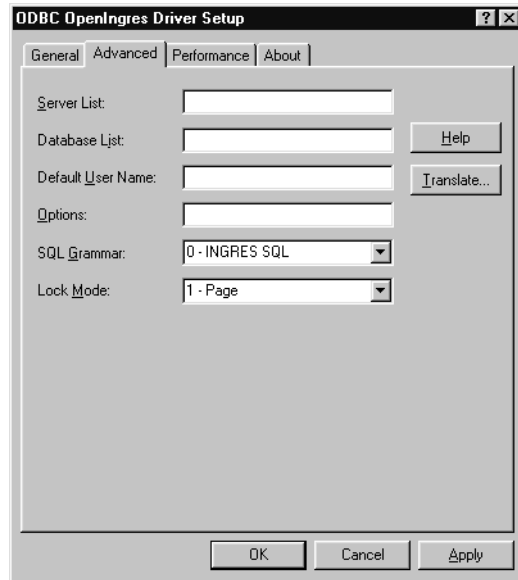
**Data Source Name:** A string that identifies this OpenIngres data source configuration in the system information. Examples include "Accounting" or "OINGRES-Serv1."

**Description:** An optional long description of a data source name. For example, "My Accounting Database" or "OINGRES on Server number 1."

**Server Name:** 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.

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

- 4 Click the **Advanced** tab to configure additional, optional settings for the data source.



5 Specify values as follows. then click **Apply**:

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

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

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

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

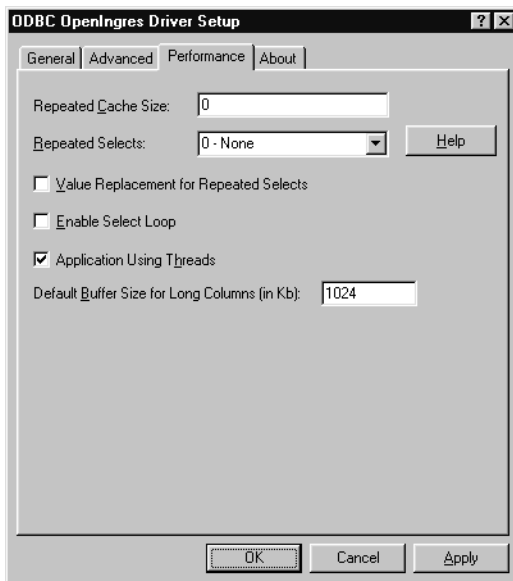
- -l (locks the database exclusively)
- -u (logs on as username)

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

**SQL Grammar:** Provides the ability to access data sources through INGRES\*Star using OpenSQL. The default is 0; that is, INGRES SQL. Specify 1 to use OpenSQL to connect to data sources through INGRES\*Star.

**Lock Mode (OpenIngres 2.0 only):** Enables locking by row, page, or table.

- 6 Click the **Performance** tab to configure additional, optional settings for the data source.



- 7 Specify values as follows. then click **Apply**:

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

To repeat a single statement rather than all statements, use the OpenIngres Repeated syntax.

**Repeated Selects:** A value of 0, 1, or 2 that determines whether the driver optimizes Select statements or runs them as repeated queries. When set to 0, the 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 Repeated Cache Size option must be set to greater than zero.

Setting this option to 1 or 2:

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

**Value Replacement for Repeated Selects:** A check box that determines whether the driver substitutes parameters for hardcoded values in repeated statements. This option is convenient in applications that do not use dynamic parameters.

When cleared (the default), the driver does not substitute parameters. When checked, 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 to optimize 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.

**Enable Select Loop:** 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.

**Application Using Threads:** 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.

**Default Buffer Size for Long Columns (in Kb):** 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; that is,  $1024 * 1024 = 1 \text{ MB}$ .

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

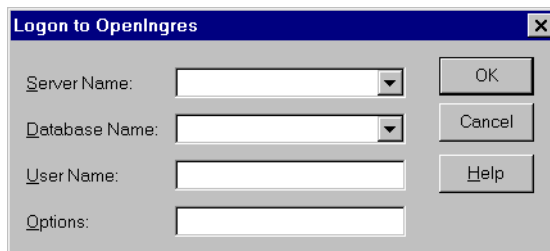
Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 8 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Connecting to a Data Source Using a Logon Dialog Box

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



In this dialog box, do the following:

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

- 2 Type the name of the database to which you want to connect or select the name from the Database Name drop-down list box, which displays the names you specified in the Setup dialog box.
- 3 If required, type your user name.
- 4 Type any options for the connection. The options can be any flags allowed on the OpenIngres SQL command line. See [Table 7-1](#) for examples of the flags allowed.
- 5 Click **OK** to log on to OpenIngres and to update the values in the system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 7-1](#) gives the long and short names for each attribute, as well as a description.





To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. For information about this file, see [Appendix H, “The UNIX Environment,” on page 369](#).

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 your default.

---

**Table 7-1. OpenIngres Connection String Attributes**

---

<b>Attribute</b>	<b>Description</b>
DataSourceName (DSN)	A string that identifies an OpenIngres data source configuration in the system information. Examples include “Accounting” or “INGRES-Serv1.”
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.
Database (DB)	The name of the database to which you want to connect.
LogonID (UID)	The default logon ID (user name) used to connect to your 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.

**Table 7-1. OpenIngres Connection String Attributes** (cont.)

Attribute	Description
Options (OPTS)	<p>The flags allowed on the OpenIngres SQL command line. Some examples are</p> <ul style="list-style-type: none"> <li>■ -l (locks the database exclusively)</li> <li>■ -u (logs on as username)</li> <li>■ +w or -w (waits/doesn't wait for the database if someone has already opened it exclusively)</li> <li>■ +U or -U (enables/disables user updating system tables and locks the database exclusively)</li> <li>■ +Y or -Y (enables/disables user updating system tables but does not lock the database exclusively)</li> </ul>
LockMode (LM) <b>OpenIngres 2.0 only</b>	<p>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).</p>
RepeatedCache Size (RCS)	<p>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).</p> <p>To repeat a single statement rather than all statements, use the OpenIngres Repeated syntax.</p>

**Table 7-1. OpenIngres Connection String Attributes** (cont.)

Attribute	Description
RepeatedSelects (RS)	<p data-bbox="694 317 1293 499">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.</p> <p data-bbox="694 522 1293 579">When set to 1, the driver optimizes Select statements that return only one result row.</p> <p data-bbox="694 586 1293 708">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 zero.</p> <p data-bbox="694 725 1039 751">Setting this option to 1 or 2:</p> <ul data-bbox="694 769 1293 916" style="list-style-type: none"> <li data-bbox="694 769 1293 826">■ limits the driver to one active statement and one active connection</li> <li data-bbox="694 855 1293 913">■ has no effect on Select statements containing a For Update clause</li> </ul>
EnableSelectLoop	<p data-bbox="694 933 1279 1055">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.</p>

**Table 7-1. OpenIngres Connection String Attributes** (cont.)

Attribute	Description
ValueReplacement (VR)	<p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>
ApplicationUsingThreads (AUT)	<p>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.</p>
SQLGrammar (SG)	<p>SQLGrammar={0   1}. Provides the ability to access data sources using OpenSQL. The default is 0; that is, INGRES SQL.</p>
DefaultLongDataBuffLen (DLDBL)	<p>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.</p>

---

## Data Types

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

---

**Table 7-2. 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 levels 1 (read committed, the default) and 3 (serializable). OpenIngres supports page-level locking. See [Appendix D, “Locking and Isolation Levels,”](#) on page 335 for a discussion of these topics.

---

## ODBC Conformance Level

The OpenIngres driver supports the functions listed in [Appendix C, “ODBC API and Scalar Functions,”](#) on page 323. 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.

## 8 Connect ODBC for Oracle



Connect ODBC for Oracle supports two separate drivers. Connect ODBC for Oracle (the “Oracle driver”) supports the Oracle 7 database system. The Oracle driver is supported in the Windows 95 and Windows NT, Macintosh Power PC, and UNIX environments.



Connect ODBC for Oracle 8 (the “Oracle 8 driver”) supports the Oracle 8 database system. The Oracle 8 driver is supported in the Windows 95 and Windows NT environments.

See the README file shipped with your INTERSOLV DataDirect product for the file names of the Oracle drivers.

---

## System Requirements

The following section lists requirements for all supported platforms.

### Windows 95 and Windows NT



Both Oracle and Oracle 8 client information for Windows 95 and Windows NT is listed below.

### *Oracle*

The Oracle SQL\*Net product is required to access remote Oracle databases. The appropriate DLLs for the current version of SQL\*Net and OCIW32.DLL must be on your path. For example,

SQL\*Net 2.3 requires ORA73.DLL, CORE35.DLL, NLSRTL32.DLL, and CORE350.DLL, as well as OCIW32.DLL. If you attempt to configure an Oracle 7 data source and you do not have these DLLs on your path, a message similar to the following appears:

```
The setup routines for the INTERSOLV 3.00 32-BIT
Oracle driver could not be loaded due to system
error code 126.
```

When you click **OK**, the following message appears:

```
Could not load the setup or translator library.
```

## **Oracle 8**

The Oracle Net8 Client version 8.0.4, or greater, is required to access remote Oracle 8 databases. For Alpha NT systems, version 8.0.3 is required. On Intel systems, the appropriate DLLs for the Oracle Net8 Client must be on your path, for example, ORA804.DLL, CORE40.DLL, NLSRTL33.DLL, and OCI.DLL. If you attempt to configure an Oracle 8 data source and you do not have these DLLs on your path, a message similar to the following appears:

```
The setup routines for the INTERSOLV 3.00 32-BIT
Oracle driver could not be loaded due to system
error code 126.
```

When you click **OK**, the following message appears:

```
Could not load the setup or translator library.
```

## **UNIX**



Before you can use the Oracle data source, you must have the Oracle SQL\*Net drivers you plan to use installed on your workstation in the \$ORACLE\_HOME source tree. ORACLE\_HOME is an environment variable created by the Oracle installation



process that identifies the location of your Oracle client components.

Oracle refers to the runtime Oracle component as "Oracle RDBMS." From the Oracle RDBMS product, the Oracle driver depends on the executables in \$ORACLE\_HOME/bin and the interface libraries in \$ORACLE\_HOME/rdbms/lib.

Set the environment variable ORACLE\_HOME to the directory where you installed the Oracle RDBMS or SQL\*Net product. For example, for C-shell users, the following syntax is valid:

```
setenv ORACLE_HOME /databases/oracle
```

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

```
ORACLE_HOME=/databases/oracle;export ORACLE_HOME
```

## ***Building the Required Oracle SQL\*Net Shared Library***

The Oracle 7 driver requires a one-time site linking to build an Oracle SQL\*Net driver on AIX and, for Oracle 7.1 only, on Solaris and HP-UX. This site linking binds your unique Oracle SQL\*Net configuration into the file, which is used by the Oracle driver to access local and remote Oracle databases.

Before you build the Oracle SQL\*Net shared library, install Oracle and set the environment variable ORACLE\_HOME to the directory where you installed Oracle.

A script is provided that builds the Oracle SQL\*Net driver. This script is in the scr/oracle directory.

The following command builds the Oracle SQL\*Net shared library:

```
%genclntsh.sh
```

## Macintosh



The Oracle SQL\*Net 2.3.2.0.3 product is required to access remote Oracle databases. Other system requirements are:

- 8 MB of memory (16 MB recommended)
- MacTCP version 1.1 or greater, or, for System 7.5, OpenTransport 1.1 or later if using TCP-IP as the transport protocol

---

## Configuring Data Sources

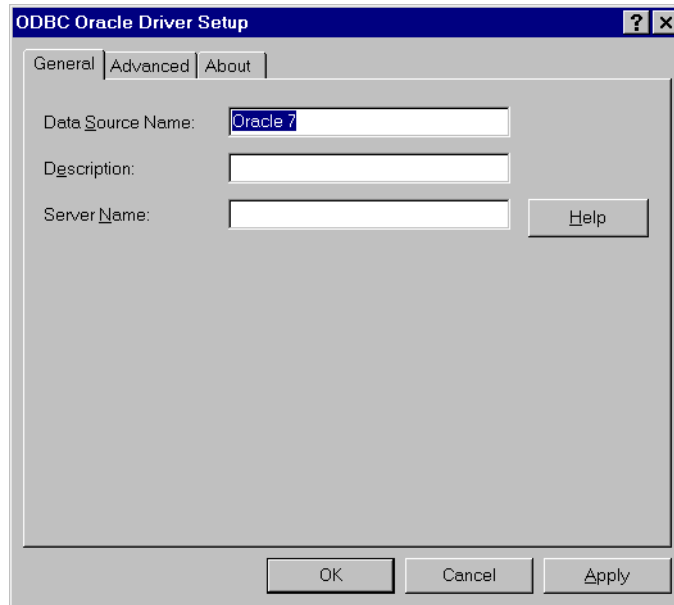


In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 8-1](#). You must also edit this file to perform a translation. For information about this file, see [Appendix H, “The UNIX Environment,” on page 369](#).

To configure an Oracle data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Oracle Driver Setup dialog box (if you are using Apple’s ODBC Driver Manager on the Macintosh, this button is labeled **Modify**).

If you are configuring a new data source, click **Add** to display a list of installed drivers. Select the Oracle driver of your choice and click **Finish** to display the ODBC Oracle Driver Setup dialog box.



3 Specify values as follows; then, click **Apply**:



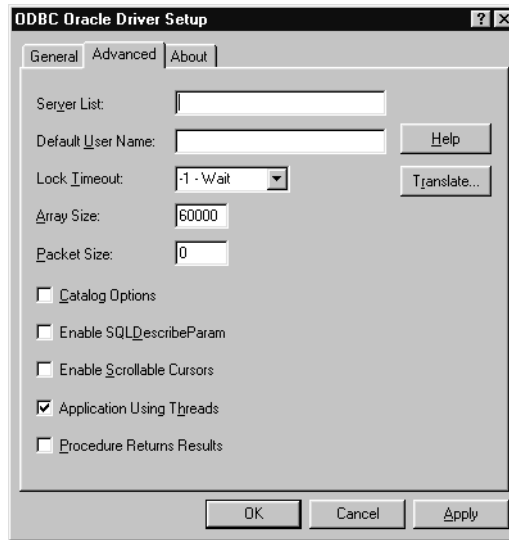
**Apply** is not available on the Macintosh. Clicking **OK** saves the values.

**Data Source Name:** A string that identifies this Oracle data source configuration in the system information. Examples include "Accounting" or "Oracle-Serv1."

**Description:** An optional long description of a data source name. For example, "My Accounting Database" or "Oracle on Server number 1."

**Server Name:** The client connection string designating the server and database to be accessed. The information required varies depending on the client driver you are using. The format of the connection string is described in ["Connecting to a Data Source Using a Connection String"](#) on page 160.

4 Click the **Advanced** tab to configure additional, optional settings for the data source.



5 Specify values as follows; then, click **Apply**:

**Server List:** The list of client connection strings that will appear in the logon dialog box. Separate the strings with commas. If the client connection string contains a comma, enclose it in quotation marks; for example, "Serv,1", "Serv,2", "Serv,3."

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

**Lock Timeout (Oracle 7 Only):** A value of 0 or -1 that specifies whether Oracle should wait for a lock to be freed before raising an error when processing a Select...For Update statement. Values can be -1 (wait forever) or 0 (do not wait). The default is -1.

**Array Size:** The number of bytes the driver uses for fetching multiple rows. Values can be an integer from 0 to 65536; the default is 60000. Larger values increase throughput by

reducing the number of times the driver fetches data across the network. Smaller values increase response time, as there is less of a delay waiting for the server to transmit data.

**Packet Size (Oracle 7 Only):** A value that controls the packet size for TCP/IP connections. Specify one of the following packet sizes: 1024, 2048, 4096, or 8192. Any other value is ignored.

The Packet Size option is used only when the connection string specified in the Server Name option is T for TCP/IP as the driver prefix. See the ServerName option described in [Table 8-1, "Oracle Connection String Attributes," on page 161](#).

**Catalog Options:** Check this box if you want the result column REMARKS for the catalog functions SQLTables and SQLColumns, and COLUMN\_DEF for the catalog function SQLColumns to have meaning for Oracle. The default is unchecked, which returns SQL\_NULL\_DATA for the result column COLUMN\_DEF and REMARKS columns. Checking this box reduces the performance of your queries.

**Enable SQLDescribeParam:** Check this box to enable the SQLDescribeParam function, which results in all parameters being described with a data type of SQL\_VARCHAR. This option should be checked when using Microsoft Remote Data Objects (RDO) to access data.

**Enable Scrollable Cursors:** Check this box to enable scrollable cursors for the data source. Both Keyset and Static cursors are enabled. This option may need to be checked when using Microsoft Foundation Classes for database access.

**Application Using Threads:** 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.

When Application Using Threads is enabled, 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.



**Procedure Returns Results:** Check this box to enable the driver to 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 165](#).

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

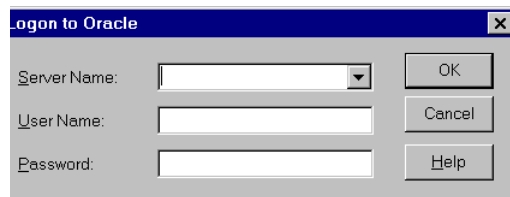
Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 6 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Connecting to a Data Source Using a Logon Dialog Box

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



In this dialog box, do the following:

- 1 Type the client connection string of the computer containing the Oracle database tables you want to access or select the string from the Server Name drop-down list box, which displays the names you specified in the setup dialog box.
- 2 If required, type your Oracle user name.
- 3 If required, type your Oracle password.
- 4 Click **OK** to log on to the Oracle database installed on the server you specified and to update the values in the system information.



**Note:** 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.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 8-1](#) gives the long and short names for each attribute, as well as a description.



To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. For information about this file, see [Appendix H, "The UNIX Environment,"](#) on page 369.



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 your default.

---

**Table 8-1. Oracle Connection String Attributes**

---

<b>Attribute</b>	<b>Description</b>
DataSourceName (DSN)	A string that identifies an Oracle data source configuration in the system information. Examples include "Accounting" or "Oracle-Serv1."
LogonID (UID)	The logon ID (user name) that the application uses to connect to your Oracle database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID. To use your operating system user name, see <a href="#">"Connecting to a Data Source Using a Logon Dialog Box"</a> on page 159.
Password (PWD)	The password that the application uses to connect to your Oracle database. To use your operating system password, see <a href="#">"Connecting to a Data Source Using a Logon Dialog Box"</a> on page 159.
LockTimeOut (LTO) <b>Oracle 7 Only</b>	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).
ArraySize (AS)	The number of bytes the driver uses for fetching multiple rows. Values can be an integer from 0 to 65,536. 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.

**Table 8-1. Oracle Connection String Attributes** (cont.)

Attribute	Description
ServerName (SRVR)	<p>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.</p> <p>For Oracle 7 remote servers, the SQL*Net connection string has the following form:</p> <pre><i>driver_prefix:computer_name[:sid]</i></pre> <p><i>driver_prefix</i> 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.</p> <p><i>computer_name</i> is the name of the Oracle Listener on your network.</p> <p><i>sid</i> 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.</p> <p>For local servers, the SQL*Net connection string has the form:</p> <pre><i>database_name</i></pre> <p><i>database_name</i> identifies your Oracle database.</p> <p>If the SQL*Net connection string contains semicolons, enclose it in quotation marks. See your SQL*Net documentation for more information.</p> <p><b>Oracle 8</b></p> <p>For Oracle 8 remote servers, the Net8 Client connection string has the following form:</p> <pre>TNSNAME</pre> <p><i>TNSNAME</i> is the alias name of the Oracle Listener on your network.</p> <p>If the Net8 Client connection string contains semicolons, enclose it in quotation marks. See your Net8 Client documentation for more information.</p>

**Table 8-1. Oracle Connection String Attributes** (cont.)

Attribute	Description
PacketSize (PS) Oracle 7 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.
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.
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 <a href="#">"Stored Procedure Results" on page 165</a> .
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.



**Table 8-1. Oracle Connection String Attributes** (cont.)

Attribute	Description
EnableDescribeParam (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.
EnableScrollableCursors (ESC)	EnableScrollableCursors={0   1}. Enables scrollable cursors for the data source. Both Keyset and Static cursors are enabled. This option may need to be set to 1 when using Microsoft Foundation Classes for database access. The default is 0.

---

## Oracle Data Types

[Table 8-2](#) shows how the Oracle data types are mapped to the standard ODBC data types.

**Table 8-2. 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

---

## Oracle 8

Table 8-3 shows how the Oracle 8 data types are mapped to the standard ODBC data types. These are in addition to the Oracle data types described above.

**Table 8-3. Oracle 8 Data Types**

Oracle 8	ODBC
Bfile	SQL_LONGVARBINARY*
Blob	SQL_LONGVARBINARY
Clob	SQL_LONGVARCHAR

\* Read-Only

The Oracle 8 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 Oracle 8 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.GTABLE2CHARCOL, icol INTEGER) as begin
open rset for select CHARCOL from GTABLE2 where INTEGERCOL <= icol
order by INTEGERCOL;
end;
```

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 Oracle 7.3 or greater or Oracle 8.x). Oracle supports record-level locking.

See [Appendix D, “Locking and Isolation Levels,”](#) on page 335 for a discussion of these topics.

---

## ODBC Conformance Level

The Oracle drivers support the functions listed in [Appendix C, “ODBC API and Scalar Functions,”](#) on page 323. The drivers also support SQLDescribeParam if EnableDescribeParam=1. If EnableScrollableCursors=1, they support SQLSetPos as well as scrollable cursors with SQLFetchScroll and SQLExtendedFetch.

The Oracle drivers support the following X/Open level functions:

- SQLProcedures
- SQLProcedureColumns
- SQLPrimaryKeys
- SQLForeignKeys
- SQLTablePrivileges
- SQLColumnPrivileges
- SQLSetPos (SQL\_ADD)

The drivers support the core SQL grammar.

---

## Number of Connections and Statements Supported

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





# 9 Connect ODBC for Paradox



Connect ODBC for Paradox (the “Paradox driver”) supports Paradox 3.0, 3.5, 4.0, 4.5, 5.0, 7.0 and 8.0 tables in the Windows 95 and Windows NT environments.

See the README file shipped with your INTERSOLV DataDirect product for the file name of the Paradox driver.

---

## System Requirements

To use the Paradox driver, you must install the Borland Database Engine, which conforms to the IDAPI programming interface. The Borland Database Engine can be found in any of the following software packages:

- Borland C++ for Windows NT or Windows 95
- Delphi for Windows NT or Windows 95
- Paradox 7 or 8 for Windows NT or Windows 95

If you attempt to configure a data source and you do not have IDAPI32.DLL on your path or in your Windows 95 SYSTEM directory or Windows NT SYSTEM32 directory, a message similar to the following appears:

```
The setup routines for the INTERSOLV 3.00 32-BIT
ParadoxFile (*.db) ODBC driver could not be loaded
due to system error code 126.
```

When you click **OK**, the following message appears:

```
Could not load the setup or translator library.
```

## Multiuser Access to Tables

You can query Paradox tables that reside in a shared directory on a network or that are to be shared among applications running on a local workstation. If the tables are on a network server, multiple users can query these tables simultaneously. To share Paradox tables among multiple users, the tables must be located in a shared directory on your network server.

Two connection attributes identify the Paradox database you are accessing: Database (database directory) and NetDir (network directory). The Database setting specifies the directory of Paradox tables that is the database. If the Database setting you specify is a shared network directory, then Paradox requires a NetDir specification as well. This value identifies the directory containing the PARADOX.NET file that corresponds to the Database setting you have specified.

Every user who accesses the same shared directory of tables must set the NetDir value to point to the same PARADOX.NET file. If your connection string does not specify a NetDir value, then Paradox uses the NetDir value specified in the Paradox section of the IDAPI configuration file. This makes it important that the NetDir specification in each user's IDAPI configuration file be set correctly.

Whenever you open a Paradox table that another user opens at the same time, the consistency of the data becomes an issue if both individuals are updating the table.

## Locking

The Paradox driver has two locking levels: record locking and table locking. Tables that have no primary key always have a prevent write lock placed on the table when the table is opened.

The table lock is escalated to a write lock when an operation that changes the table is attempted.

Tables that have primary keys use record locking. The locking level is escalated from record locking to a table write lock if the transaction runs out of record locks.

Primary keys provide the greatest concurrency for tables that are accessed and modified by multiple users.

**Note:** If a table lock is placed on a Paradox table, the Paradox driver prevents users from updating and deleting records but does not prevent them from inserting records into the locked table.

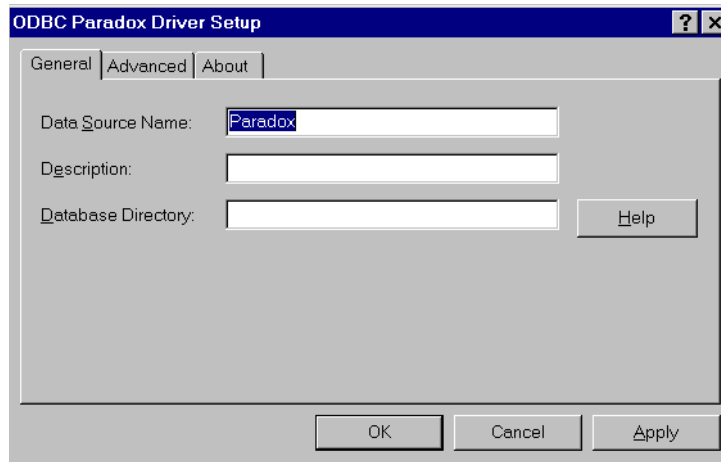
---

## Configuring Data Sources

To configure a Paradox data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Paradox Driver Setup dialog box.

If you are configuring a new data source, click **Add** to display a list of installed drivers. Select the Paradox driver and click **Finish** to display the ODBC Paradox Driver Setup dialog box.



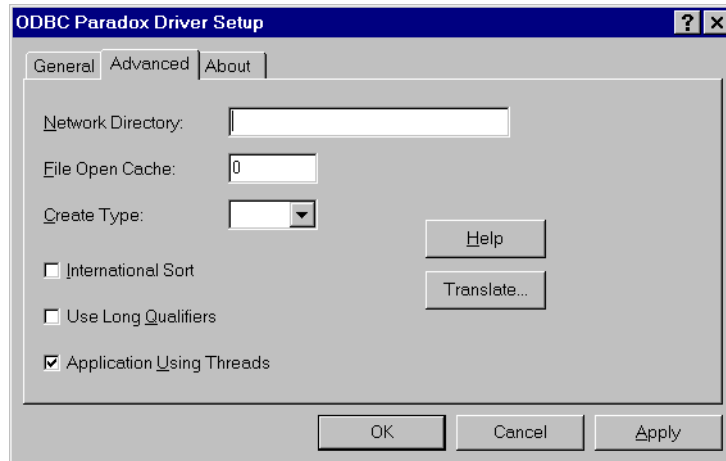
3 Specify values as follows; then, click **Apply**:

**Data Source Name:** A string that identifies this Paradox data source configuration in the system information. Examples include "Accounting" or "Paradox-Serv1."

**Description:** An optional long description of a data source name. For example, "My Accounting Database" or "Paradox Files on Server number 1."

**Database Directory:** The directory in which the Paradox files are stored. If a directory is not specified, the current working directory is used. You can also specify aliases that are defined in your IDAPI configuration file, if you have one. To do this, enclose the alias name in colons. For example, to use the alias MYDATA, specify ":MYDATA:"

4 Click the **Advanced** tab to configure additional, optional settings for the data source.



5 Specify values as follows; then, click **Apply**:

**Network Directory:** The directory containing the PARADOX.NET file that corresponds to the database you have specified. If the Paradox database you are using is shared on a network, then every user who accesses it must set this value to point to the same PARADOX.NET file. If not set here, this value is determined by the NetDir setting in the Paradox section of the IDAPI configuration file. If you are not sure how to set this value, contact your network administrator.

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

**Create Type:** The Paradox table version desired for any Create Table statements that you execute. You can specify 3, 4, 5, 7, or leave the box blank and use the default, which is determined by the Level setting in the Paradox section of the IDAPI configuration file. The numeric values map to the major revision numbers of the Paradox family of products.

**Note:** If you select Create Type 7, the Paradox driver supports table names up to 128 characters long. For all other Create Type settings, the driver supports table names up to 8 characters long.

**International Sort:** A setting to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this box to use 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.

**Use Long Qualifiers:** Specifies whether the driver uses long path names as table qualifiers. If the Use Long Qualifiers check box is selected, path names can be up to 255 characters. The default is unselected (path names can be up to 128 characters).

**Application Using Threads:** 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.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 6 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 Paradox is:

```
DSN=PARADOX TABLES ; DB=C:\ODBC\EMP ; PW=ABC , DEF , GHI
```

**Table 9-1** 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 your default.

---

**Table 9-1. Paradox Connection String Attributes**

---

Attribute	Description
DataSourceName (DSN)	A string that identifies a Paradox data source configuration in the system information. Examples include "Accounting" or "Paradox-Serv1."
Database (DB)	The directory in which the Paradox files are stored.  For this attribute, you can also specify aliases that are defined in your IDAPI configuration file, if you have one. To do this, enclose the alias name in colons. For example, to use the alias MYDATA, specify "Database=:MYDATA:"
NetDir (ND)	The directory containing the PARADOX.NET file that corresponds to the database you have specified. If the Paradox database you are using is shared on a network, then every user who accesses it must set this value to point to the same PARADOX.NET file. If not specified, this value is determined by the NetDir setting in the Paradox section of the IDAPI configuration file. If you are not sure how to set this value, contact your network administrator.



**Table 9-1. Paradox Connection String Attributes** (cont.)

Attribute	Description
FileOpenCache (FOC)	<p>The maximum number of unused table opens to cache. For example, when FileOpenCache=4, and a user opens and closes four tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The disadvantage is that a user who tries to open the table exclusively may get a locking conflict even though no one appears to have the table open. The initial default is 0.</p>
CreateType (CT)	<p>CreateType={3   4   5   7   null (blank)}. This attribute specifies the table version for Create Table statements. There are four valid values for this connection string: 3, 4, 5, and null (blank). The numeric values map to the major revision numbers of the Paradox family of products. To override another CreateType setting chosen during data source configuration with the default create type determined by the Level setting in the Paradox section of the IDAPI configuration file, set CreateType= (null).</p> <p><b>Note:</b> When CreateType is set to 7, the Paradox driver supports table names up to 128 characters long. For all other CreateType settings, the driver supports table names up to 8 characters long.</p>
IntlSort (IS)	<p>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."</p> <p>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.</p>

---

**Table 9-1. Paradox Connection String Attributes** (cont.)

---

<b>Attribute</b>	<b>Description</b>
UltraSafeCommit (USF)	UltraSafeCommit={0   1}. This attribute determines when the driver flushes its changes to disk. If UltraSafeCommit=1, the driver does this at each COMMIT. This decreases performance. The default is 0. This means that the driver flushes its changes to disk when the table is closed or when internal buffers are full. In this case, a machine “crash” before closing a table may cause recent changes to be lost.
Passwords (PW)	A password or list of passwords. You can add 1 to 50 passwords into the system using a comma-separated list of passwords. Passwords are case-sensitive. For example, Passwords=psw1, psw2, psw3.
UseLongQualifiers (ULQ)	UseLongQualifiers={0   1}. This attribute specifies whether the driver uses long path names as table qualifiers. With UseLongQualifiers set to 1 path names can be up to 255 characters. The default is 0; maximum length is 128 characters.

**Table 9-1. Paradox Connection String Attributes** (cont.)

Attribute	Description
DeferQueryEvaluation (DQ)	<p data-bbox="525 314 1275 407">DeferQueryEvaluations={0   1}. This attribute determines when a query is evaluated—after all records are read or each time a record is fetched.</p> <p data-bbox="525 421 1275 708">If DeferQueryEvaluation=0, the driver generates a result set when the first record is fetched. The driver reads all records, evaluates each one against the Where clause, and compiles a result set containing the records that satisfy the search criteria. This process slows performance when the first record is fetched, but activity performed on the result set after this point is much faster, because the result set has already been created. You do not see any additions, deletions, or changes in the database that occur while working from this result set.</p> <p data-bbox="525 722 1275 1041">If DeferQueryEvaluation=1 (the default), the driver evaluates the query each time a record is fetched, and stops reading through the records when it finds one that matches the search criteria. This setting avoids the slowdown while fetching the first record, but each fetch takes longer because of the evaluation taking place. The data you retrieve reflect the latest changes to the database; however, a result set is still generated if the query is a Union of multiple Select statements, if it contains the Distinct keyword, or if it has an Order By or Group By clause.</p>
ApplicationUsingThreads (AUT)	<p data-bbox="525 1055 1275 1241">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.</p>

## Data Types

[Table 9-2](#) shows how the Paradox data types are mapped to the standard ODBC data types. It also identifies the types supported by Paradox versions 3.x and 4.x. These Paradox data types can be used in a Create Table statement. For the syntax of the Create Table statement, see [Appendix A, "SQL for Flat-File Drivers," on page 289](#).

**Table 9-2. Paradox Data Types**

Paradox	ODBC	3.x Support	4.x Support
Alpha	SQL_VARCHAR	Yes	Yes
AutoIncrement	SQL_INTEGER	No	No
BCD	SQL_DECIMAL	No	No
Binary*	SQL_LONGVARBINARY	No	Yes
Bytes*	SQL_BINARY	No	No
Date	SQL_TYPE_DATE	Yes	Yes
Formatted Memo*	SQL_LONGVARBINARY	No	Yes
Graphic*	SQL_LONGVARBINARY	No	Yes
Logical*	SQL_BIT	No	No
Long Integer	SQL_INTEGER	No	No
Memo*	SQL_LONGVARCHAR	No	Yes
Money	SQL_DOUBLE	Yes	Yes
Number	SQL_DOUBLE	Yes	Yes
OLE*	SQL_LONGVARBINARY	No	Yes
Short	SQL_SMALLINT	Yes	No
Time	SQL_TYPE_TIME	No	No
TimeStamp	SQL_TYPE_TIMESTAMP	No	No

\* Cannot be used in an index. Of the starred types, only Logical can be used in a Where clause.

---

## Select Statement

You use a SQL Select statement to specify the columns and records to be read. The Paradox driver supports all of the Select statement clauses as described in [Appendix A, "SQL for Flat-File Drivers," on page 289](#). This section describes the information that is specific to the Paradox driver.

### Column Names

Paradox column names are case-insensitive and their maximum length is 25 characters. If a column name contains a special character, does not begin with a letter, or is a reserved word, surround it with the grave character ( ` ) (ASCII 96). For example:

```
SELECT `last name` FROM emp
```

---

## Alter Table Statement

The Paradox 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 [DEFAULT default_value]
| ADD (column_name data_type [DEFAULT default_value]
| , column_name data_type [DEFAULT default_value] , , , )
| DROP [COLUMN] column_name [CASCADE | RESTRICT]}
```

*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  
alphanumeric (10))
```

## Dropping Columns

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

When dropping a column, use the Cascade keyword to drop the column while removing references from any dependent objects, such as indexes or views. Use Restrict to cause the Alter Table statement to fail if other objects are dependent upon the column you are dropping. For example, to drop a column and remove its references from dependent objects:

```
ALTER TABLE emp DROP startdate CASCADE
```

If the Alter Table statement contains neither Cascade nor Restrict, it fails when you attempt to drop a column upon which other objects are dependent.

---

# Create Table Statement

The Create Table statement is used to create database files. The form of the Create Table statement is:

```
CREATE TABLE filename (col_definition  
[,col_definition,. . .])
```

*filename* can be a simple name or a full name. A simple file name is preferred for portability to other SQL data sources. If it is a simple file name, the file is created in the directory you specified as the database directory in the connection string. If you did not specify a database directory in the connection string, the file is created in the directory you specified as the database directory in the system information. If you did not specify a database directory in either place, the file is created in the current working directory at the time you connected to the driver.

*col\_definition* is the column name, followed by the data type, Default clause, followed by an optional column constraint definition. Values for column names are database specific. The data type specifies a column's data type.

The only column constraint definition currently supported by some flat-file drivers is "not null." Not all flat-file tables support "not null" columns. In the cases where "not null" is not supported, this restriction is ignored and the driver returns a warning if "not null" is specified for a column. The "not null" column constraint definition is allowed in the driver so that you can write a database-independent application (and not be concerned about the driver raising an error on a Create Table statement with a "not null" restriction).

A sample Create Table statement to create an employee database table is:

```
CREATE TABLE emp (last_name CHAR(20)
DEFAULT default_value] NOT NULL DEFAULT 'JOHNSON',
first_name CHAR(12) NOT NULL,
salary NUMERIC (10,2) NOT NULL,
hire_date DATE NOT NULL)
```

---

## Password Protection

Paradox supports two types of passwords: master and auxiliary. The Paradox driver supports master passwords only and can manage up to 50 passwords.

Paradox tables can be encrypted to provide limited access to users who do not know the password. The driver maintains a list of passwords for each connection. The driver can access only encrypted tables for which a password appears in this list. You can supply a password in three ways: by typing it in the Password dialog box (which appears when the driver does not have the password to open an encrypted table), by including it in a connection string (with the Passwords attribute), or by using the Add Password statement.

Paradox provides five statements that manage passwords for Paradox tables. These statements are specific to the Paradox driver:

```
ENCRYPT filename USING PASSWORD password
ADD PASSWORD password
DECRYPT filename USING PASSWORD password
REMOVE PASSWORD password
REMOVE ALL PASSWORDS
```



*filename* can be a simple filename or a full pathname. If a simple filename is given, the file must be in the directory specified with the Database connection string attribute. The .DB extension is not required.

*password* is a case-sensitive text string up to 15 characters in length, including blanks. If your password includes lower-case letters or nonalphanumeric characters, enclose it in single quotation marks (').

## Encrypting a Paradox Table

The Encrypt statement associates a password with a table. For example:

```
ENCRYPT emp USING PASSWORD test
```

## Accessing an Encrypted Paradox Table

To access an encrypted Paradox table, add the password to the list of passwords Paradox maintains for that connection. To do so, you can

- Issue an Add Password statement before you access the table. For example:

```
ADD PASSWORD test
SELECT * FROM emp
```

- Specify the passwords using the Passwords attribute at connection time.

If you do not add the password, the driver prompts you for it when you access the table.

## Decrypting a Paradox Table

The Decrypt statement disassociates a password from a table. You no longer need to enter the password to open the table. For example:

```
DECRYPT emp USING PASSWORD test
```

## Removing a Password from Paradox

The Remove Password statement removes a password from the list Paradox maintains for the connection. For example:

```
REMOVE PASSWORD test
```

## Removing All Passwords from Paradox

The Remove All Passwords statement removes the list of passwords Paradox maintains.

If you remove a password from Paradox and do not decrypt the table, you must continue entering the password to open the table.

---

## Index Files

An index is used to read records in sorted order and to improve performance when selecting records and joining tables. Paradox indexes are stored in separate files and are either *primary* or *non-primary*.

## Primary Index

A primary index is made up of one or more fields from the Paradox table. The primary key fields of a primary index consist of one or more consecutive fields in the table, beginning with the first field in the table. A table can have only one primary index.

Collectively, the primary key fields uniquely identify each record in the Paradox table. Thus, no two records in a Paradox table can share the same values in their primary key fields. Once a primary index is created for a Paradox table, the table's records are re-ordered based on the primary key fields. At the time a primary index is created, if any records have matching primary key field values, Paradox deletes all but the first record. Paradox creates this index as maintained; that is, if you modify, add, or delete records in the table, the primary index is updated automatically to reflect these changes.

A primary index is a single file with the same name as the table on which it is based but with a .PX extension.

To lock records, you must have a primary index.

## Non-Primary Index

Paradox 7 tables support UNIQUE secondary indexes. Refer to ["Create and Drop Index Statements" on page 188](#) for more information.

A non-primary index is defined by specifying one or more fields in the Paradox table that constitute the non-primary key field. It allows Paradox to sort each record in the table according to the values of the non-primary key fields.

There are two kinds of non-primary indexes: maintained and non-maintained. A maintained index is automatically updated when the table is changed, whereas a non-maintained index is

not. Instead, a non-maintained index is tagged out-of-date and is updated when the index is used again.

You must have a primary index on a table before you create a maintained, non-primary index.

The Paradox driver uses non-maintained indexes only for read-only queries on locked tables. A primary index is not required for the non-maintained index to be used.

For Paradox 3.x, a single non-primary index consists of a pair of files with the same name as the table on which the non-primary index is based; one of these files has an *.Xnn* extension while the other has a *.Ynn* extension (where the hexadecimal number *nn* corresponds to the field number of the non-primary key field for the non-primary index).

For Paradox 4.x, 5, and 7, single-field non-primary indexes that are case-sensitive have the same name as their associated table and are assigned file extensions *.X01* through *.XFF*, depending on the number of the field on which the index is based. Single-field non-primary indexes that are case insensitive and composite indexes have the same name as the table on which they are based. They are assigned file extensions sequentially starting with *.XG0* (with hexadecimal increments).

---

## Create and Drop Index Statements

The Paradox driver supports SQL statements to create and delete indexes. The Create Index Primary statement is used to create primary indexes. The Create Index statement is used to create non-primary indexes. The Drop Index statement is used to delete indexes.

## Create Index Primary Statement

The syntax for creating a primary index is:

```
CREATE [UNIQUE] INDEX PRIMARY
    ON table_name (column [,column...])
```

The UNIQUE keyword is optional; the index is unique whether or not you include this keyword.

*table\_name* is the name of the table on which the index is to be based.

*column* is the name of a column that is included as a the key field for the index. The column list must contain one or more consecutive fields in the table, beginning with the first field in the table.

For example:

```
CREATE UNIQUE INDEX PRIMARY ON emp (emp_id)
```

Be careful when you create a primary key because any rows that have a primary key duplication are deleted when you execute the Create Index Primary statement.

## Create Index Statement

For Paradox 3.0, 3.5, 4.0, 4.5, and 5.0 tables, the syntax for creating a non-primary index is:

```
CREATE INDEX index_name [/NON_MAINTAINED]
[/CASE_INSENSITIVE] ON table_name (column
[, column...])
```

For Paradox 7.0 tables, the syntax for creating a non-primary index is:

```
CREATE [UNIQUE] INDEX index_name [/NON_MAINTAINED]
[/CASE_INSENSITIVE] ON table_name [column [DESC]
[, column...
```

For Paradox 7 tables only (when the Create Type is 7), the optional UNIQUE keyword prevents duplicate values in the non-primary index.

*index\_name* identifies the index. If the name contains blanks or special characters, or does not begin with a letter, surround it with the grave character ( ` ) (ASCII 96).

The NON\_MAINTAINED switch makes the index non-maintained. The default is to create a maintained index.

The CASE\_INSENSITIVE switch makes the index case-insensitive. The default is to create a case-sensitive index.

*table\_name* is the name of the table on which the index is to be based.

*column* is the name of a column that is included as a the key field for the index.

For Paradox 7 tables only (when the Create Type is 7), the DESC keyword creates a non-primary index that uses descending keys.

Paradox 3.0 and 3.5 tables cannot have composite or case-insensitive indexes. When you create a non-primary index for Paradox 3.0 and 3.5 tables, follow these rules:

- Specify only one column name.
- Do not use the CASE\_INSENSITIVE switch.
- Use the column name as the index name.

For example:

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

## Drop Index Statement

The syntax for dropping a primary index is

```
DROP INDEX path_name.PRIMARY
```

For example:

```
DROP INDEX emp.PRIMARY
```

The syntax for dropping a non-primary index is

```
DROP INDEX path_name.index_name
```

*path\_name* is the name of the table from which the index is being dropped. The pathname can be either the fully qualified pathname or, if the table is specified with the Database attribute of the connection string, a simple table name.

*index\_name* is the name that was given to the index when it was created. If the name contains blanks or special characters, or does not begin with a letter, surround it with the grave character ( ` ) (ASCII 96). Use the column name as the index name when dropping indexes from Paradox 3.5 or 3.0 tables.

For example:

```
DROP INDEX emp.last_name
```

---

## Transactions

The Paradox driver supports transactions. A transaction is a series of database changes that is treated as a single unit. In applications that don't use transactions, the Paradox driver immediately executes Insert, Update, and Delete statements on the database tables and the changes are automatically committed when the SQL statement is executed. There is no way

to undo such changes. In applications that use transactions, inserts, updates, and deletes are held until a Commit or Rollback is specified. A Commit saves the changes to the database file; a Rollback discards the changes.

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

---

## Isolation and Lock Levels Supported

Paradox supports isolation levels 1 (read committed) and 3, (serializable). It supports record- and table-level locking. See [Appendix D, “Locking and Isolation Levels,” on page 335](#) for a discussion of these topics.

---

## ODBC Conformance Level

The Paradox driver supports the functions listed in [Appendix C, “ODBC API and Scalar Functions,” on page 323](#). In addition, the following functions are supported:

- SQLSetPos
- SQLPrimaryKeys

When used with Paradox 5 or Paradox 7 tables, the Paradox driver supports the SQLForeignKeys function. The Paradox driver also supports backward and random fetching in SQLExtendedFetch and SQLFetchScroll. The Paradox driver supports the minimum SQL grammar.



---

## Number of Connections and Statements Supported

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



# 10 Connect ODBC for PROGRESS



Connect ODBC for PROGRESS (the “PROGRESS driver”) supports database version 7.3 and version 8.x of the PROGRESS database system in the Windows 95 and Windows NT environments.

See the README file shipped with your INTERSOLV DataDirect product for the file name of the PROGRESS driver.

---

## System Requirements

To access a PROGRESS database with a PROGRESS driver, your system must include the following items.

On the client computer:

- PROGRESS Version 8.2A installation Client Networking for Windows NT.

On the Server computer:

- PROGRESS Version 7.3C (or later) Server Networking and Database Server installed on the platform you choose as your database server machine.
- PROGRESS Version 7.3C (or later) Client Networking installed on the platform you choose for your OID, if you choose to connect via server rather than directly.

**Note:** The INTERSOLV PROGRESS driver accesses PROGRESS databases that are created with Version 7.3C or later. This driver cannot access Version 6 PROGRESS databases.

Before using the driver, you must set the IDLC environment variable to; your PROGRESS DLC directory. For example:

```
set IDLC=C:\DLC
```

---

## Configuring Data Sources

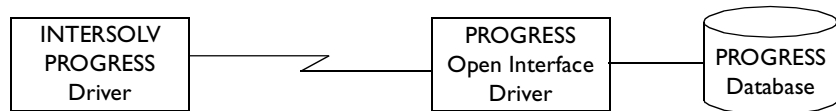
The DataDirect ODBC PROGRESS driver supports the following data source configurations:

- Remote OID with direct database access
- Remote OID with database access via server

The PROGRESS driver does not support remote, single-user configurations.

### Remote OID with Direct Access

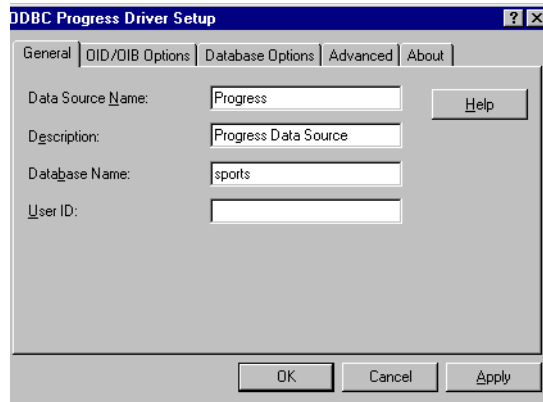
The following figure shows a Direct Access configuration.



To configure a PROGRESS data source for a remote OID with direct database access (no server process) configuration:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC PROGRESS Driver Setup dialog box.

If you are configuring a new data source, click **Add** to display a list of installed drivers. Select the PROGRESS driver and click **Finish** to display the ODBC PROGRESS Driver Setup dialog box.



3 Specify values as follows; then, click **Apply**:

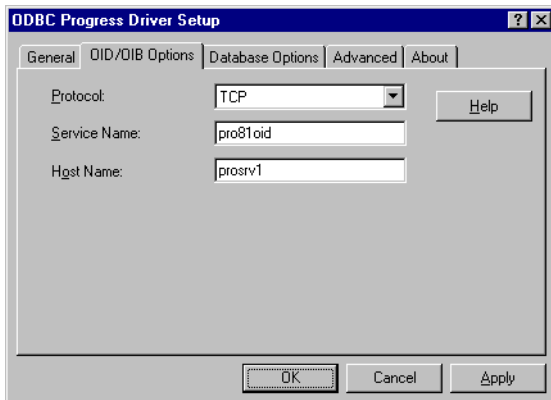
**Data Source Name:** Identifies this PROGRESS data source configuration in the system information. Examples include "Accounting" or "PROG-Serv1."

**Description:** An optional long description of a data source name. For example, "My Accounting Database" or "PROGRESS on Server number 1."

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

**User ID:** The default logon ID used to connect to your PROGRESS database. Your ODBC application may override this value or you may override this value in the Logon dialog box or connection string.

4 Click the **OID/OIB Options** tab to specify OID options.



5 Specify values as follows; then, click **Apply**:

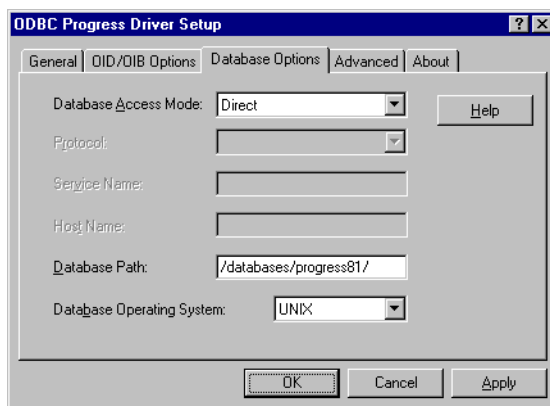
**Protocol:** TCP, NETBIOS, or SPX.

**Service Name:** Specify the service name of the OID as listed in the services file.

**Host Name:** Specify the host name of the OID/OIB machine.

An Open Interface Broker or Open Interface Driver must be running on the specified host to connect to the database.

6 Click the **Database Options** tab to specify database options.



- 7 Specify values as follows; then, click **Apply**:

**Database Access Mode:** Select Direct.

**Protocol:** Not applicable for this configuration.

**Service Name:** Not applicable for this configuration.

**Host Name:** Not applicable for this configuration.

**Database Path:** The fully qualified directory path on the server containing the database (not including the database name or extension).

**Database Operating System:** Select the type of server that the database resides on.

If you select Ignore, you must type a qualified separator as the last character in the Database Path field describe above.

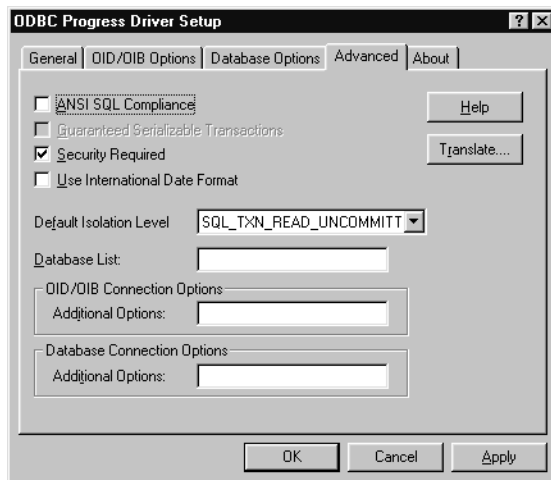
For example, for Windows NT:

```
c:\progress\
```

and for UNIX:

```
/databases/progress
```

- 8 Click the **Advanced** tab to specify optional data source settings.



9 Specify values as follows; then, click **Apply**:

**ANSI SQL Compliance:** Allows case-sensitive character data.

**Guaranteed Serializable Transactions:** Allows serializable transactions. To select this option, the ANSI SQL Compliance check box must be selected.

**Security Required:** Determines whether the logon ID (UID) is required in the connection string.

**Use International Date Format:** Specifies whether PROGRESSPROGRESS uses international date formats (d/m/y) rather than United States formats (m/d/y).

**Default Isolation Level:** Specifies the default isolation level for concurrent transactions. SQL\_TXN\_READ\_COMMITTED and SQL\_TXN\_READ\_UNCOMMITTED are the values. You should select SQL\_TXN\_READ\_UNCOMMITTED to work around specific locking problems.

**Database List:** Specify a list of databases that are available to the application. In a session, users can access only those databases specified in this list.



**OID/OIB Connection Options/Additional Options:** Specify additional PROGRESS parameters for Open Interface Driver/ Open Interface Broker installation. The format for specifying these options is:

*-syntax parameter\_definition...*

For more information about these parameters, refer to the system administration documentation for PROGRESS.

**Database Connection Options/Additional Options:** Specify additional PROGRESS parameters for the database connection. The format for specifying these options is:

*-syntax parameter\_definition...*

For more information about these parameters, refer to the system administration documentation for PROGRESS.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 10 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

## ***Local Access on a Windows 95 or Windows NT System***

Local access on a Windows 95 or Windows NT system can be accomplished by using Remote OID (located on the client) and Direct Access mode (to a database located on the client). Both OID and the database server must be started on the client from the directory containing the database.

### **1 Set the environment:**

```
SET IDLC=C:\DLC82
SET DLC=C:\DLC82
SET PROOIBRK=%DLC%\BIN\oibrkr32.exe
SET PROOIDRV=%DLC%\BIN\oidrvr32.exe
SET PATH=%PATH%;%DLC%;%DLC%\BIN
```

where C:\DLC82 is the directory to which PROGRESS was installed.

### **2 You must start Open Interface Broker in the directory containing the database to ensure a proper self-serving OID client:**

```
%DLC%\BIN\oibrkr32.exe -SV -S pdro82oid
```

where "pdro82oid" is the service name as specified in the SERVICES file.

### **3 Start the database server:**

```
%DLC%\BIN\_mprosrv test
```

where "test" is the database name.

#### 4 The following is sample connection information:

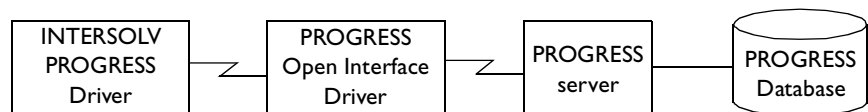
```

Database=test
UID=<blank>
PWD=<blank>
OID/OIB Protocol=TCP
OID/OIB Service Name=pdro82oid
OID/OIB Host Name=localhost
Database Access Mode=Direct
Database Path=c:\protmp82
Database Operating System=Windows

```

## Remote OID with Database Access via Server

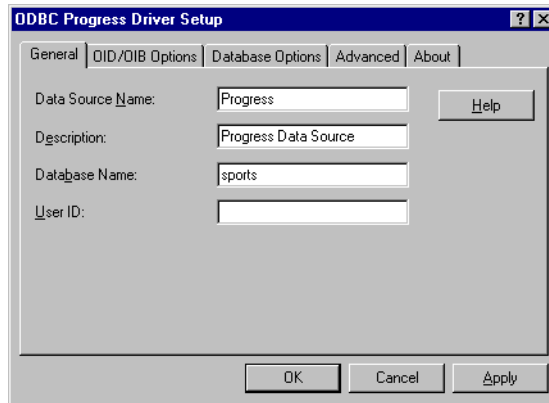
The following figure shows a database access via server configuration.



To configure a PROGRESS data source for a remote OID with database access via a server:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC PROGRESS Driver Setup dialog box.

If you are configuring a new data source, click **Add** to display a list of installed drivers. Select the PROGRESS driver and click **Finish** to display the ODBC PROGRESS Driver Setup dialog box.



3 Specify values as follows; then, click **Apply**:

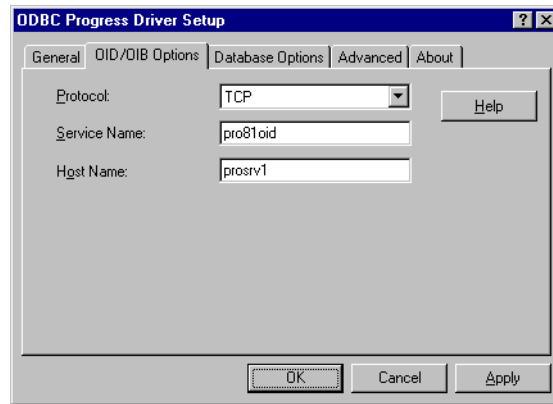
**Data Source Name:** Identifies this PROGRESS data source configuration in the system information. Examples include “Accounting” or “PROG-Serv1.”

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

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

**User ID:** The default logon ID used to connect to your PROGRESS database. Your ODBC application may override this value or you may override this value in the Logon dialog box or connection string.

4 Click the **OID/OIB Options** tab to specify OID options.



5 Specify values as follows; then, click **Apply**:

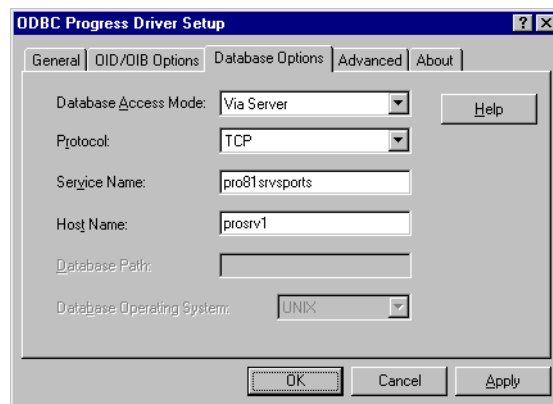
**Protocol:** TCP, NETBIOS, or SPX.

**Service Name:** Specify the service name of the OID as listed in the services file.

**Host Name:** Specify the host name of the OID/OIB machine.

An Open Interface Broker or Open Interface Driver must be running on the specified host to connect to the database.

6 Click the **Database Options** tab to specify database options.



7 Specify values as follows; then, click **Apply**:

**Database Access Mode:** Select Via Server.

**Protocol:** Select the type of protocol your configuration uses: NETBIOS, SPX, or TCP.

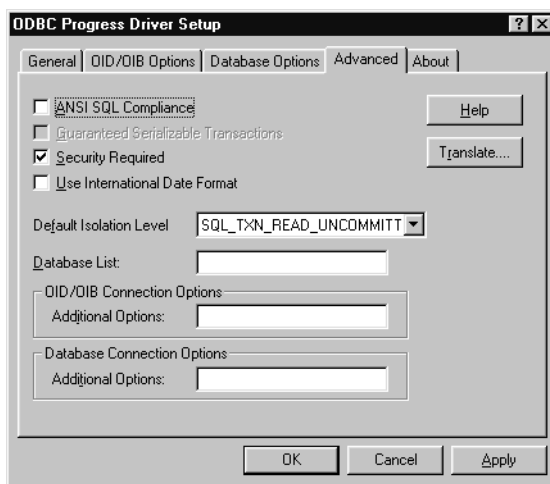
**Service Name:** Specify the service name of the server process machine.

**Host Name:** Specify the host name of the server process machine.

**Database Path:** Not applicable for this configuration.

**Database Operating System:** Not applicable for this configuration.

- 8 Click the **Advanced** tab to specify optional data source settings.



- 9 Specify values as follows; then, click **Apply**:

**ANSI SQL Compliance:** Allows case-sensitive character data.

**Guaranteed Serializable Transactions:** Allows serializable transactions. To select this option, the ANSI SQL Compliance check box must be selected.

**Security Required:** Determines whether the logon ID (UID) is required in the connection string.

**Use International Date Format:** Specifies whether PROGRESS uses international date formats (d/m/y) rather than United States formats (m/d/y).

**Default Isolation Level:** Specifies the default isolation level for concurrent transactions. `SQL_TXN_READ_COMMITTED` and `SQL_TXN_READ_UNCOMMITTED` are the values. You should select `SQL_TXN_READ_UNCOMMITTED` to work around specific locking problems.

**Database List:** Specify a list of databases that are available to the application. In a session, users can access only those databases specified in this list.

**OID/OIB Connection Options/Additional Options:** Specify additional PROGRESS parameters for Open Interface Driver/Open Interface Broker installation. The format for specifying these options is:

*-syntax parameter\_definition...*

For more information about these parameters, refer to the system administration documentation for PROGRESS.

**Database Connection Options/Additional Options:** Specify additional PROGRESS parameters for the database connection. The format for specifying these options is:

*-syntax parameter\_definition...*

For more information about these parameters, refer to the system administration documentation for PROGRESS.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. `INTERSOLV` provides a translator named `INTERSOLV OEM ANSI` that translates your data from the IBM PC character set to the ANSI character set.

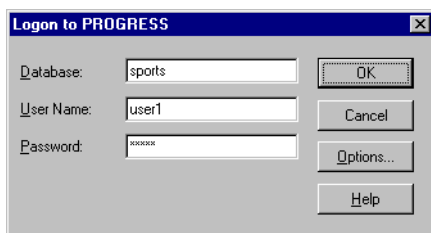
Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 10 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Connecting to a Data Source Using a Logon Dialog Box

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



In this dialog box, do the following:

- 1 Type the name of the database to which you want to connect (without a path or an extension).
- 2 If required, type your user name.



3 If required, type your password.

User name and password are optional parameters. You can log on to the database without these parameters, but you may not be able to create, delete, or manipulate the data.

4 Click **Options** to display the PROGRESS Logon Options dialog box.

5 Specify values as follows; then, click **OK** to complete the logon and to update the values in the system information:

**Protocol:** Specifies the protocol used for communication with the OID; TCP, NETBIOS, or SPX.

**Service Name:** Specifies the service name of the OID as listed in the services file.

**Host Name:** Specifies the host name of the OID/OIB machine.

An Open Interface Broker or Open Interface Driver must be running on the specified host to connect to the database.

**Database Access Mode:** Specifies how the OID accesses the database. Select Direct if the OID access the database file directly. Select Via Server if the OID accesses the database through a server process.

**Protocol:** Specifies the type of protocol your configuration uses: NETBIOS, SPX, or TCP. This option is available only when the Database Access option is set to Via Server.

**Service Name:** For Via Server configurations, specifies the service name of the server process. Not applicable for Direct configurations.

**Host Name:** For Via Server configurations, specifies the host name of the server process machine. Not applicable for Direct configurations.

**Database Path:** Specifies the path name of the database when the OID accesses the database directly (not via a server). This setting is necessary only when the Database Connection Options/Database Access option is set to Direct.

**Database Operating System:** Specifies the operating system the database is stored under. The system uses the specified operating system to determine which qualifier separator to use when building the database path: backslash (\) for Windows or slash (/) for UNIX. This setting is necessary only if the Database Connection Options/Database Access is set to Direct.

**Note:** If you select Ignore, you must type a qualifier separator as the last character in the Database Path field described above. For example, for Windows NT:

```
c:\progress\  
and for UNIX:
```

```
/databases/progress/
```

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 PROGRESS is:

```
DSN=PROGRESS;DB=PAYROLL;UID=JOHN;PWD=XYZZY;  
OIDP=TCP;OIDS=PRO81OID;OIDH=PROSRV1;  
DBAM=VIASERVER;DBPR=TCP;DBS=PRO81SRVPAYROLL;  
DBH=PROSRV1;ASC=0
```

[Table 10-1](#) gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you

specified a value for the attribute when configuring the data source, that value is your default.

---

**Table 10-1. PROGRESS Connection String Attributes**

---

<b>Attribute</b>	<b>Description</b>
DataSourceName (DSN)	A string that identifies a PROGRESS data source configuration in the system information. Examples include "Accounting" or "PROG-Serv1."
Database (DB)	The name of the database to which you want to connect.
DBPath (DBPA)	If DBAccessMode (DBAM) is Via Server, the path name of the database, (without the database name).
LogonID (UID)	The default logon ID (user name) used to connect to your PROGRESS database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID. This ID is case-sensitive.
OIDProtocol (OIDP)	OIDProtocol={TCP   NETBIOS   SPX}. The protocol to connect the PROGRESS driver to the Open Interface Driver (OID).
OIDService (OIDS)	The service name of the Open Interface Driver (OID).
OIDHost (OIDH)	The host name of the Open Interface Driver (OID). (If the OID is local, enter the local machine's host name in this field).
	An Open Interface Broker or Open Interface Driver must be running on the specified host to connect to the database.
DBAccessMode (DBAM)	DBAccessMode={Direct   Via Server}. Specifies how the OID accesses the database. Select Direct if the OID access the database file directly. Select Via Server if the OID accesses the database through a server process.
DBProtocol (DBPR)	DBProtocol={NETBIOS   SPX   TCP}. If DBAccessMode=Via Server, specify the protocol to connect the Open Interface Driver (OID) to the PROGRESS database server.

**Table 10-1. PROGRESS Connection String Attributes** (cont.)

DBService (DBS)	If DBAccessMode=Via Server, specify the service name of the server machine.
DatabaseOS (DBOS)	DatabaseOS={Windows   Unix   Ignore   VMS}. This attribute specifies the type of path separator (forward slash or backward slash) to used when building the database path.
DBHost (DBH)	If DBAccessMode=Via Server, specify the host name of the server machine.
ANSISQL Compliance (ASC)	ANSISQLCompliance={0   1}. This attribute determines whether the values in a character column, and comparisons made to them, must be case-sensitive. The default is 0; that is, not case-sensitive.
GSTransaction (GST)	GSTransaction={0   1}. Set this attribute to 1 to allow serializable transactions. This attribute can be set to 1 only if the ANSISQLCompliance attribute is set to 1. The initial default is 0.
SecurityRequired (SR)	SecurityRequired={0   1}. Determines whether the logon ID (UID) is required in the connection string. The default is 1, which is Security Required.
UseInternational Date (UIND)	UseInternationalDate={0   1}. Specifies whether PROGRESS uses international date formats (d/m/y) rather than United States formats (m/d/y). Values are Off (0) and On (1). The default is Off.
DefaultIsolation Level (DIL)	DefaultIsolationLevel={0   1}. Specifies the default isolation level for concurrent transactions. The values are SQL_TXN_READ_COMMITTED (0) and SQL_TXN_READ_UNCOMMITTED (1). The default is SQL_TXN_READ_UNCOMMITTED. You should select SQL_TXN_READ_UNCOMMITTED to work around specific locking problems.

**Table 10-1. PROGRESS Connection String Attributes** (cont.)

---

OIDOptions (OIDO)	A string containing parameters to be passed to the OID when it is auto-started. The format for specifying OID connection options is: <i>OIDOptions=-syntax parameter_definition. . .</i>
DBOptions (DBO)	A string containing parameters to be used when the OID connects to the specified database(s). The format for specifying OID connections options is: <i>ODBOptions=-syntax parameter_definition. . .</i>

---



---

## Data Types

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

**Table 10-2. PROGRESS Data Types**


---

PROGRESS	ODBC
Character	SQL_VARCHAR
Date	SQL_TYPE_DATE
Decimal	SQL_DECIMAL
Integer	SQL_INTEGER
Logical	SQL_BIT
Float	SQL_FLOAT

---

---

## Isolation and Lock Levels Supported

PROGRESS supports isolation level 1 (read committed) if the data source was defined without Guaranteed Serializable Transactions; otherwise, PROGRESS supports isolation level 3 (serializable). PROGRESS supports record-level locking. See [Appendix D, “Locking and Isolation Levels,” on page 335](#) for a discussion of these topics.

---

## ODBC Conformance Level

The PROGRESS driver supports the functions listed in [Appendix C, “ODBC API and Scalar Functions,” on page 323](#). The driver supports the core SQL grammar.

The driver also supports the function `SQLSetPos` and forward and backward scrolling with `SQLExtendedFetch` and `SQLFetchScroll`.

---

## Connections and Statements Supported

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





# 11 Connect ODBC for SQL Server



Connect ODBC for SQL Server (the “SQL Server driver”) supports the SQL Server 6.x database system available from Microsoft in the Windows 95 and Windows NT environments.

See the README file shipped with your INTERSOLV DataDirect product for the file name of the SQL Server driver.

---

## System Requirements

To use the SQL Server driver, you must have the appropriate Microsoft SQL Server DB-Library and Net-Library version installed (version 6.0 for access to 6.0 DBMS; version 6.5 for access to 6.5 DBMS).

Your database must support catalog stored procedures.

The DataDirect SQL Server driver for Windows NT and Windows 95 requires Microsoft client software; it does not work with Sybase System 10 or 11 software.

The DB-Library is NTWDBLIB.DLL. The Net-Library you need depends on the network protocol used to connect to the SQL Server. For example, Named Pipes requires DBNMPNTW.DLL, and TCP/IP requires DBMSSOCN.DLL. Contact your Microsoft SQL Server vendor to obtain the appropriate DB-Library and Net-Library.

If you attempt to configure a data source and you do not have NTWDBLIB.DLL on your path or in your Windows 95 SYSTEM or Windows NT SYSTEM32 directory, a message similar to the following appears:

```
The setup routines for the INTERSOLV 3.00 32-BIT
SQLServer6 ODBC driver could not be loaded due to
system error code 126.
```

When you click **OK**, the following message appears:

```
Could not load the setup or translator library.
```

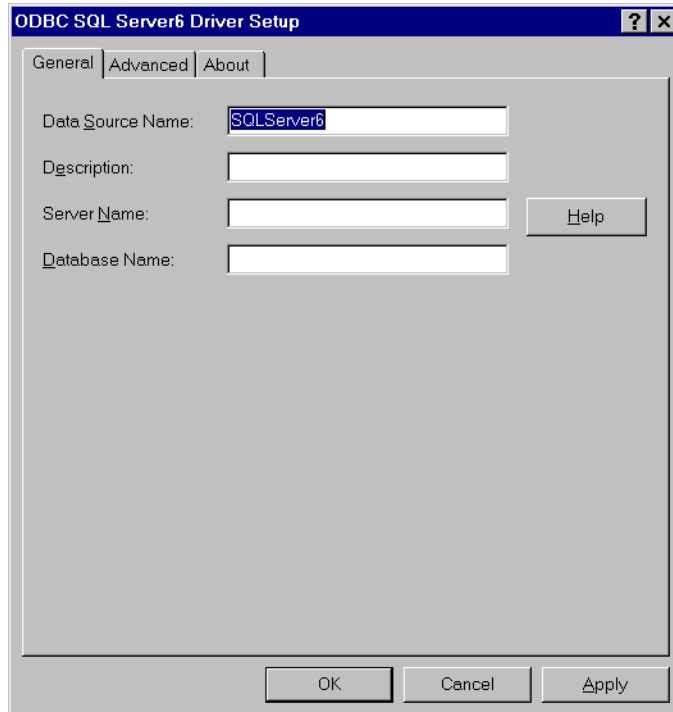
---

## Configuring Data Sources

To configure a SQL Server data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select INTERSOLV SQLServer6 and click **Finish** to display the ODBC SQL Server Driver Setup dialog box.

If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC SQL Server Driver Setup dialog box.



**3** Specify values as follows; then, click **Apply**:

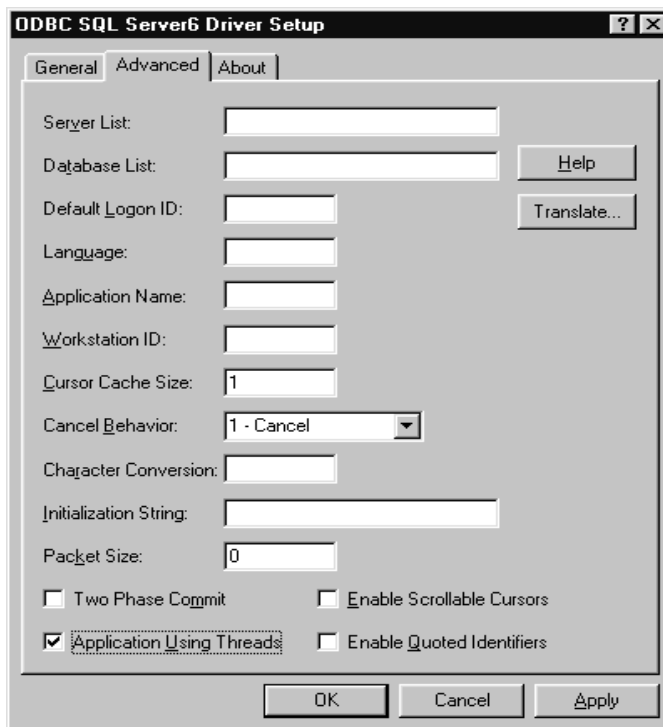
**Data Source Name:** A string that identifies this SQL Server data source configuration in the system information. Examples include "Accounting" or "SQL Server-Serv1."

**Description:** An optional long description of a data source name. For example, "My Accounting Database" or "SQL Server on Server number 1."

**Server Name:** The name of the server that contains the database you want.

**Database Name:** The name of the database to which you want to connect by default. If you do not specify a value, the default database defined by SQL Server is used.

- 4 Click the **Advanced** tab to configure additional, optional settings for the data source.



- 5 Specify values as follows; then, click **Apply**:

**Server List:** A comma-separated list of servers that will appear in the Logon dialog box.

**Database List:** The databases that will be available in the SQL Server Logon Options dialog box. Separate the names with commas.

**Default Logon ID:** The default logon ID used to connect to your SQL Server database. This ID is case-sensitive. A logon ID is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the Logon dialog box or connection string.

**Language:** The national language to be used by the client. The default is English.

**Application Name:** The name SQL Server uses to identify your application.

**Workstation ID:** The workstation ID used by the client.

**Cursor Cache Size:** The number of cursors the cursor cache can hold. The driver creates a cache of statements; each statement represents an open connection to SQL Server. The cursor cache increases performance but uses database resources. The default is 1 (one cursor).

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

- 0 fetches all of the remaining records if the statement was a Select.
- 1 cancels the statement by calling `dbcancel`. This is the default and should be used if `dbcancel` is supported in your client/server configuration.
- 2 closes the connection to the server for the statement. Use this value only if `dbcancel` is not supported for your configuration and the performance of fetching all remaining records is unacceptable.

**Character Conversion:** This value controls the character set conversion between SQL Server and a client application. If you omit this value, no character conversion takes place on your server.

Common values include `iso_1` for ISO-8859-1, `cp850` for Code Page 850, `roman8` for Roman8 character set, and `SJIS` for a Japanese character set. See your SQL Server documentation for a complete list of values.

**Initialization String:** The value of this option is a string containing one or more SQL Server commands that you want to run when the data source connection is initialized. Multiple commands must be separated by a semicolon (;).

**Packet Size:** A value of -1, 0, or  $x$  that determines the number of bytes per network packet transferred from the database server to the client. The correct setting of this attribute can improve performance.

When set to 0, the default, the driver uses the default packet size as specified in the 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, 6 means to set the packet size to  $6 * 512 = 3072$  bytes).

Note that the ODBC specification identifies a connect option, `SQL_PACKET_SIZE`, that offers this same functionality. To avoid conflicts with applications that may set both the connection string attribute and the ODBC connect option, the ODBC connect option will take precedence.

**Two-Phase Commit:** This check box, when selected, enables you to have two active statements within a transaction, using the SQL Server two-phase commit services. The active statements may deadlock if they reference the same SQL Server table.

**Application Using Threads:** 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.

**Enable Scrollable Cursors:** A setting that provides access to scrollable cursors. If selected, the driver will support Keyset Driven cursors and will support SQLSetPos with Update and Delete functionality. The default is unselected. Scrollable cursors require the following attributes:

- A unique index must be available on all Selected tables.
- The Select statement cannot contain any of the following: Into, For Browse, Compute, Union, Compute By, Aggregate functions, Table aliases.
- If the Select statement includes a view, the From clause must include only a single view (no other tables or views).
- The select list must include all unique index columns of the base tables.

**Enable Quoted Identifiers:** Enables quoted identifiers; that is, identifiers in SQL Server can be quoted using a quoting character. The default is not selected.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

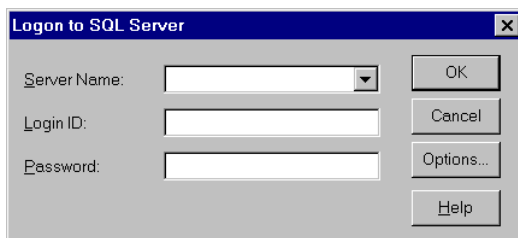
Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 6 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

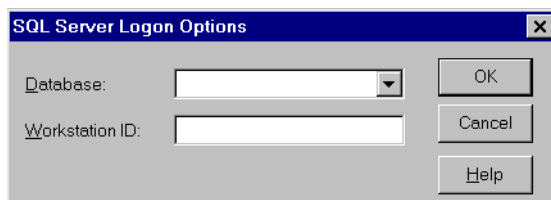
## Connecting to a Data Source Using a Logon Dialog Box

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



In this dialog box, do the following:

- 1 Type the name of the server containing the SQL Server database tables you want to access (case-sensitive) or select the name from the Server Name drop-down list box, which displays the server names you specified in the ODBC SQL Server Driver Setup dialog box.
- 2 If required, type your case-sensitive login ID.
- 3 If required, type your case-sensitive password for the system.
- 4 Optionally, click **Options** to display the SQL Server Logon Options dialog box and specify the initial SQL Server database to connect to and the name of your workstation.





- 5 Click **OK** to log on to the SQL Server database installed on the server you specified and to update the values in the system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 SQL Server is:

```
DSN=Accounting;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

[Table 11-1](#) gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you

specified a value for the attribute when configuring the data source, that value is your default.

---

**Table 11-1. SQL Server Connection String Attributes**

---

<b>Attribute</b>	<b>Description</b>
DataSourceName (DSN)	A string that identifies a SQL Server data source configuration in the system information. Examples include "Accounting" or "SQL Server-Serv1."
ServerName (SRVR)	The name of the server containing the SQL Server tables you want to access.
Database (DB)	The name of the database to which you want to connect.
LogonID (UID)	The case-sensitive logon ID used to connect to your SQL Server database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.
Password (PWD)	A case-sensitive password.
Language (LANG)	The national language to be used by the client. The initial default is English.
ApplicationName (APP)	The name SQL Server uses to identify your application.
WorkstationID (WKID)	The workstation ID used by the client.
CursorCacheSize (CCS)	The number of cursors the cursor cache can hold. The driver creates a cache of statements; each statement represents an open connection to SQL Server. The cursor cache increases performance but uses database resources. The initial default is 1.
CharConv (CC)	A value that controls the character set conversion between SQL Server (version 4.8 or later) and a client application. Common values include iso_1 for ISO-8859-1, cp850 for Code Page 850, roman8 for the Roman8 character set, and SJIS for a Japanese character set. See your SQL Server documentation for a complete list of values.

**Table 11-1. SQL Server Connection String Attributes** (cont.)

Attribute	Description
Cancel (CAN)	<p data-bbox="682 317 1300 404">Cancel={0   1   2}. This attribute specifies how a previously executed statement should be canceled. Valid values are</p> <ul data-bbox="682 430 1300 843" style="list-style-type: none"> <li data-bbox="682 430 1300 491">■ Cancel=0 fetches all remaining records if the statement was a Select.</li> <li data-bbox="682 517 1300 647">■ Cancel=1 cancels the statement by calling dbcancel. Set Cancel=1 if dbcancel is supported in your client/server configuration. This is the initial default.</li> <li data-bbox="682 673 1300 843">■ Cancel=2 closes the connection to the server for the statement. Set Cancel=2 only if dbcancel is not supported for your configuration and the performance of fetching all remaining records is unacceptable.</li> </ul>
TwoPhaseCommit (TPC)	<p data-bbox="682 861 1272 1112">TwoPhaseCommit={0   1}. This attribute lets you have two or more active statements within a transaction, using the SQL Server two-phase commit services. Set TwoPhaseCommit=1 to use two-phase commit. The active statements may deadlock if they reference the same SQL Server table. Otherwise, set TwoPhaseCommit=0 (the initial default).</p>
ApplicationUsing Threads (AUT)	<p data-bbox="682 1130 1300 1345">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.</p>

**Table 11-1. SQL Server Connection String Attributes** (cont.)

Attribute	Description
EnableScrollable Cursors (ESC)	<p data-bbox="644 314 1265 465">EnableScrollableCursors={0   1}. Provides access to scrollable cursors. If ESC=1, the driver will support Keyset Driver cursors and will support SQLSetPos with Update and Delete functionality. The default is 0.</p> <p data-bbox="644 482 1265 512">Scrollable cursors require the following attributes:</p> <ul data-bbox="644 534 1265 965" style="list-style-type: none"> <li data-bbox="644 534 1265 591">■ A unique index must be available on all Selected tables.</li> <li data-bbox="644 621 1265 753">■ The Select statement cannot contain any of the following: Into, For Browse, Compute, Union, Compute By, Aggregate functions, Table aliases.</li> <li data-bbox="644 782 1265 874">■ If the Select statement includes a view, the From clause must include only a single view (no other tables or views).</li> <li data-bbox="644 904 1265 965">■ The select list must include all unique index columns of the base tables.</li> </ul>
InitializationString (IS)	<p data-bbox="644 982 1265 1138">The value of this option is a string containing one or more SQL Server commands that you want to run when the data source connection is initialized. Multiple commands must be separated by a semicolon (;).</p>

**Table 11-1. SQL Server Connection String Attributes** (cont.)

Attribute	Description
PacketSize (PS)	<p data-bbox="682 314 1300 470">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.</p> <p data-bbox="682 487 1300 574">When set to 0, the initial default, the driver uses the default packet size as specified in the server configuration.</p> <p data-bbox="682 591 1300 713">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.</p> <p data-bbox="682 730 1300 852">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 <math>6 * 512 = 3072</math> bytes).</p> <p data-bbox="682 869 1300 1060">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, the ODBC connect option will take precedence.</p>
EnableQuoted Identifiers (EQI)	<p data-bbox="682 1078 1300 1164">EnableQuotedIdentifiers={0   1}. Enables quoted identifiers; that is, identifiers in SQL Server can be quoted using a quoting character. The default is 0.</p>
ModifySQL (MS)	<p data-bbox="682 1182 1300 1466">ModifySQL={0   1}. This attribute is provided for backward compatibility. It determines whether the driver modifies SQL statements to conform to ODBC specifications or passes the SQL statement directly to SQL Server. Specify ModifySQL=1 to have the driver modify the SQL statement to conform to ODBC specifications. Specify ModifySQL=0 to have the driver understand SQL dialects found in earlier drivers. The default is 1.</p>

---

## Data Types

Table 11-2 shows how the SQL Server data types are mapped to the standard ODBC data types.

---

**Table 11-2. SQL Server Data Types**

---

SQL Server	ODBC Data Type
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 identity	SQL_INTEGER
money	SQL_DECIMAL
numeric	SQL_NUMERIC
numeric() identity	SQL_NUMERIC
real	SQL_REAL
smalldatetime	SQL_TYPE_TIMESTAMP
smallint	SQL_SMALLINT
smallint identity	SQL_SMALLINT
smallmoney	SQL_DECIMAL
sysname	SQL_VARCHAR
text	SQL_LONGVARCHAR
timestamp	SQL_VARBINARY
tinyint	SQL_TINYINT
tinyint identity	SQL_TINYINT

---

**Table 11-2. SQL Server Data Types** (cont.)

---

varbinary	SQL_VARBINARY
varchar	SQL_VARCHAR

---

---

## Isolation and Lock Levels Supported

SQL Server supports isolation levels 1 (read committed) and 3 (serializable). SQL Server supports page-level locking. See [Appendix D, “Locking and Isolation Levels,”](#) on page 335 for a discussion of these topics.

---

## ODBC Conformance Level

The SQL Server driver supports the functions listed in [Appendix C, “ODBC API and Scalar Functions,”](#) on page 323.

In addition, the following functions are supported:

- SQLColumnPrivileges
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedureColumns
- SQLProcedures
- SQLTablePrivileges
- If EnableScrollableCursors=1, then scrollable cursors are supported with SQLExtendedFetch and SQLFetchScroll and SQLSetPos is supported.

The driver supports the minimum SQL grammar.

## Number of Connections and Statements Supported

The SQL Server database system supports multiple connections. With two-phased commit, SQL Server supports multiple statements per connection. Otherwise, SQL Server supports a single statement per connection if `SQL_AUTOCOMMIT` is 0 and multiple statements per connection if `SQL_AUTOCOMMIT` is 1.



# 12 Connect ODBC for SQLBase



Connect ODBC for SQLBase (the “SQLBase driver”) supports the Centura Software SQLBase database system in the Windows 95 and Windows NT environments.

See the README file shipped with your INTERSOLV DataDirect product for the file name of the SQLBase driver.

---

## System Requirements

To communicate with the server, the SQLBase driver requires SQLWNTM.DLL, SQLNGCI.DLL, and a communication DLL (for example, SQLNPIPE.DLL for Named Pipes). The directory containing this file must be on your path.

If you attempt to configure a data source and you do not have SQLWNTM.DLL and SQLNGCI.DLL on your path or in your Windows 95 SYSTEM or Windows NT SYSTEM32 directory, a message similar to the following appears:

```
The setup routines for the INTERSOLV 3.00 32-BIT
SQLBase ODBC driver could not be loaded due to
system error code 126.
```

When you click **OK**, the following message appears:

```
Could not load the setup or translator library.
```

You must have version 6.x or higher of SQLBase.

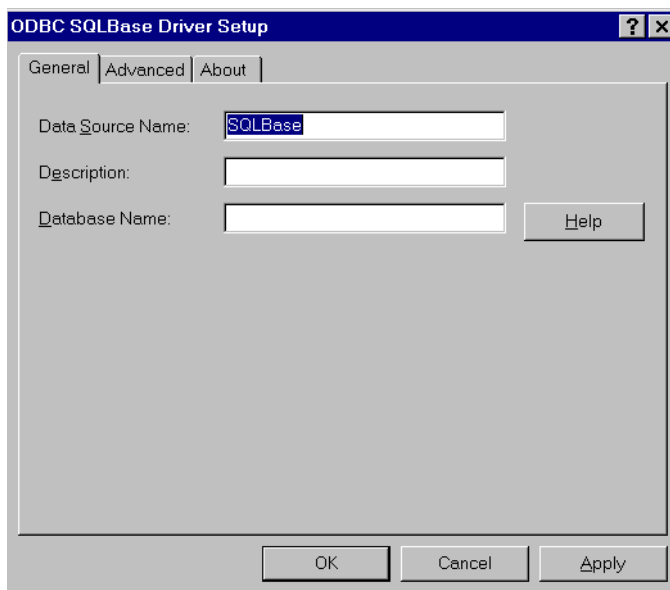
---

## Configuring Data Sources

To configure a SQLBase data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC SQLBase Driver Setup dialog box.

If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the SQLBase driver and click **Finish** to display the ODBC SQLBase Driver Setup dialog box.



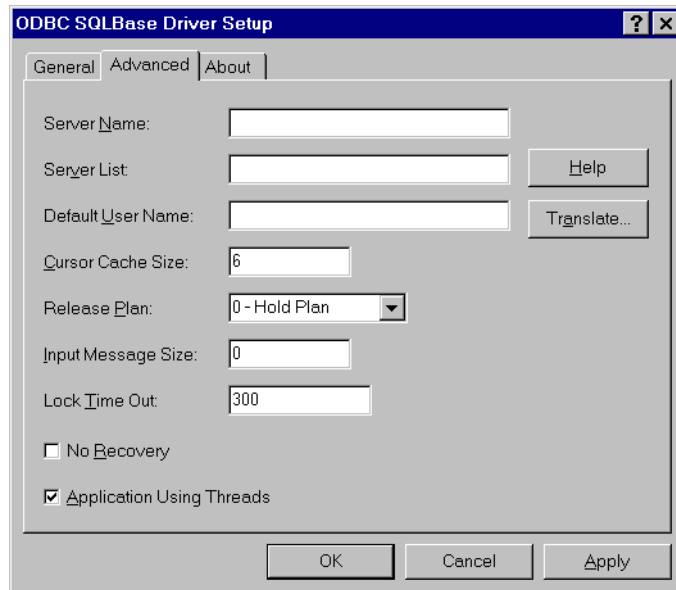
- 3 Specify values as follows; then, click **Apply**:

**Data Source Name:** A string that identifies this SQLBase data source configuration in the system information. Examples include "Accounting" or "SQLBase-Serv1."

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

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

- 4 Click the **Advanced** tab to configure additional, optional settings for the data source.



- 5 Specify values as follows; then, click **Apply**:

**Server Name:** The name of the server that contains the desired database. Specify the word Local if you are using the local server.

The server name is not required to log on. The dialog box appears more quickly if you do not specify a server name. If you do specify a server name, the Database Name drop-down list in the Logon dialog box is populated with the names of the databases available on that server.

**Server List:** A comma-separated list of servers that will appear in the Logon dialog box. Specify Local to add the local server to the list.

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

**Cursor Cache Size:** The number of cursors the cursor cache can hold. The default is 6.

**Release Plan:** A value of 0 or 1 that determines whether a lock is maintained on a table when the cursors accessing the table are freed. Freeing the lock on the table results in a request to the server, which can decrease performance but will always allow you to Drop or Alter the table.

- 0. Hold Plan, the default—no locks on tables are freed.
- 1. Free Plan—locks are freed.

**Input Message Size:** The number of bytes in the input message buffer. The default is determined by SQLBase. Increasing this value retrieves more records across the network in a single fetch.

**Lock Time Out:** The number of seconds SQLBase should wait for a lock to be freed before raising an error. Values can be -1 (wait forever) to 1800; the default is 300.

**No Recovery:** Select this check box to disable transaction recovery. Selecting this box is dangerous because your database can become inconsistent in the event of a system failure.

**Application Using Threads:** 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.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

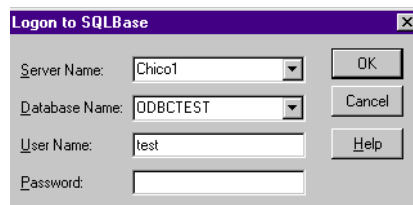
Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 6 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a Logon dialog box when you are connecting to a data source. For SQLBase, the dialog box is as follows:



In this dialog box, do the following:

- 1 Optionally, type the name of the server containing the SQLBase database tables you want to access or select the name from the Server Name drop-down list, which displays the server names you specified in the Setup dialog box. Specify Local to access a local SQLBase database.
- 2 Type the name of the database you want to access. If you specified a server name, you can select the name from the Database Name drop-down list.
- 3 If required, type your user name.
- 4 If required, type your password.
- 5 Click **OK** to complete the logon and to update the values in the system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 SQLBase is:

```
DSN=SQLBASE TABLES;SRVR=QESVR;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

[Table 12-1](#) 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 your default.

---

**Table 12-1. SQLBase Connection String Attributes**

---

Attribute	Description
DataSourceName (DSN)	A string that identifies a SQLBase data source configuration in the system information. Examples include "Accounting" or "SQLBase-Serv1."
Database (DB)	The name of the database to which you want to connect.
LogonID (UID)	The default logon ID (user name) used to connect to your SQLBase database. If so, contact your system administrator to get your logon ID. A logon ID is required only if security is enabled on your database.
ServerName (SRVR)	The name of the server containing the SQLBase database tables you want to access. Specify ServerName=Local if you are using the local server.
Servers (SRVRLIST)	A comma-separated list of servers with which to prompt the user in a Logon dialog box. Specify Local to add the local server to the list.
Password (PWD)	A case-sensitive password.
CursorCacheSize (CCS)	The number of cursors the cursor cache can hold. The cursor cache increases performance and uses few database resources. The initial default is 6.

**Table 12-1. SQLBase Connection String Attributes** (cont.)

Attribute	Description
InputMessageSize (IMS)	An integer value that determines the number of bytes of the input message buffer. Increasing this value retrieves more records across the network in a single fetch. The initial default is determined by SQLBase.
LockTimeOut (LTO)	The number of seconds SQLBase should wait for a lock to be freed before raising an error. Values can be -1 to 1800; -1 means wait forever. The initial default is 300.
ReleasePlan (RP)	ReleasePlan={0   1}. This attribute determines whether a lock is maintained on a table when the cursors accessing the table are freed. Freeing the lock on the table results in a request to the server, which can decrease performance but will always allow you to Drop or Alter the table.  When set to 0, the initial default, no locks on tables are freed. When set to 1, locks are freed.
NoRecovery (NR)	NoRecover={0   1}. This attribute enables or disables transaction recovery. NoRecovery=0 (the initial default) enables recovery; NoRecovery=1 disables recovery. NoRecovery=1 improves performance but is dangerous because your database can become inconsistent in the event of a system crash. See your SQLBase documentation for information on this option.
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.



---

## Data Types

Table 12-2 shows how the SQLBase data types are mapped to the standard ODBC data types.

---

**Table 12-2. SQLBase Data Types**

---

<b>SQLBase</b>	<b>ODBC</b>
Char	SQL_VARCHAR
Date	SQL_TYPE_DATE
Decimal	SQL_DECIMAL
Double Precision	SQL_DOUBLE
Integer	SQL_INTEGER
Long Varchar	SQL_LONGVARCHAR
Number	SQL_DOUBLE
Real	SQL_REAL
Smallint	SQL_SMALLINT
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Varchar	SQL_VARCHAR

---



---

## Isolation and Lock Levels Supported

SQLBase supports isolation levels 1 (read committed, the default) and 3 (serializable). SQLBase also supports an alternative isolation level 1 called cursor stability. Your ODBC application can use this isolation level by calling `SQLSetConnectOption(1040,1)`.

SQLBase supports page-level locking.

See [Appendix D, “Locking and Isolation Levels,”](#) on page 335 for a discussion of these topics.

---

## ODBC Conformance Level

The SQLBase driver supports the functions listed in [Appendix C, “ODBC API and Scalar Functions,”](#) on page 323. In addition, the following functions are supported:

- SQLColumnPrivileges
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedures
- SQLSetPos
- SQLTablePrivileges

The driver also supports backward and random fetching in `SQLExtendedFetch` and `SQLFetchScroll`. It supports the core SQL grammar.

---

## Number of Connections and Statements Supported

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

# 13 Connect ODBC for Sybase



Connect ODBC for Sybase (the “Sybase driver”) supports the SQL Server System 10, System 11, and Adaptive Server 11.5 database systems from Sybase in the Windows 95 and Windows NT, Macintosh, and UNIX environments. The driver supports the SQL Server 4.9.2 database system in the Windows 95 and Windows NT environments.

See the README file shipped with your INTERSOLV DataDirect product for the file name of the Sybase driver.

---

## System Requirements

The following section lists requirements for all supported platforms.

### Windows 95 and Windows NT



You must install the Sybase Open Client-Library (version 10.0.4 or higher for Intel systems, version 11.1.1 for Alpha systems) and the appropriate Sybase Net-Library to gain access to the Sybase server.

SYBPING is a tool that is provided to test connectivity from your client workstation to the database server (servers that are added through SQLEdit). Use this tool to test your connection.

SQLEdit is a tool that allows you to define servers and adds them to SQL.INI.

Set the environment variable SYBASE to the directory where you installed the Sybase Open Client. For example, set SYBASE=C:\SQL10. For Windows, set this environment variable in the Control Panel under System.

## UNIX



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 ivsybxx Sybase driver that requires the library libct. For DCE environments, users should use the ivsyb11xx Sybase driver that requires the Sybase 11.1 client library libct\_r.

## Macintosh



You must install the Sybase Open Client-Library, version 10.0.3 or higher, and the appropriate Sybase Net-Library to gain access to the Sybase server. Other system requirements are:

- 8 MB of memory
- MacTCP version 1.1 or greater or OpenTransport version 1.1 if using TCP

Double-click the Sybase Config Control Panel and select your interface file. See your System documentation for more information.

You can use Sybping to test the connection to the database server.

---

## Configuring Data Sources



**Note:** In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 13-1 on page 255](#). You must also edit this file to perform a translation. For information about this file, see [Appendix H, “The UNIX Environment,” on page 369](#).

To configure a Sybase data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Sybase Driver Setup dialog box.

If you are configuring a new data source, click **Add** to display a list of installed drivers. Select the Sybase driver and click **Finish** to display the ODBC Sybase Driver Setup dialog box.



3 Specify values as follows; then, click **Apply**:



**Apply** is not available on the Macintosh. Clicking **OK** saves the values.

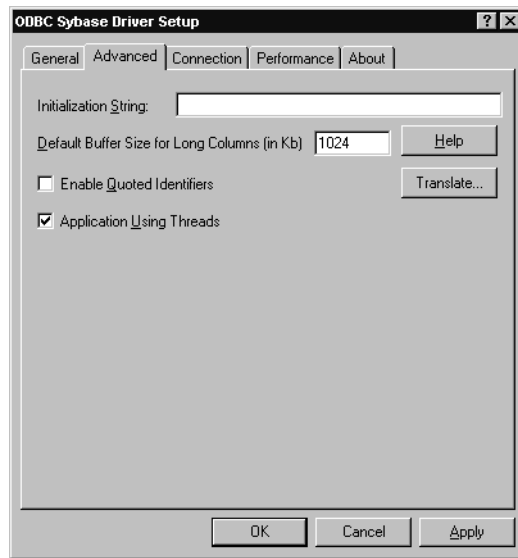
**Data Source Name:** A string that identifies this Sybase data source configuration in the system information. Examples include "Accounting" or "Sys10-Serv1."

**Description:** An optional long description of a data source name. For example, "My Accounting Database" or "System 10 on Server number 1."

**Database Name:** The name of the database to which you want to connect by default. If you do not specify a value, the default is the database defined by the system administrator for each user.

**Server Name:** The name of the server that contains the Sybase tables you want to access. If not supplied, the server name in the DSQUERY environment variable is used. On UNIX, the name of a server from your \$SYBASE/interfaces file.

- 4 Click the **Advanced** tab to configure additional, optional settings for the data source.



- 5 Specify values as follows; then, click **Apply**:

**Initialization String:** Supports the running of Sybase commands at connect time. Multiple commands must be separated by semicolons.

**Default Buffer Size for Long Columns (in Kb):** An integer value that specifies, in 1024-byte multiples, the maximum length of data fetched from a TEXT or IMAGE column. The default is 1,024 kilobytes. You will need to increase this value if the total size of any long data exceeds 1 megabyte.

**Enable Quoted Identifiers:** Allows support of quoted identifiers in System 10 or System 11 servers.

**Application Using Threads:** 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.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 6 Click the **Connection** tab to configure optional data source settings.



- 7 Specify values as follows; then, click **Apply**:



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

**Database List:** The databases that appear in the logon dialog box. Separate the names with commas.

**Default Logon ID:** The default logon ID used to connect to your Sybase database. This ID is case-sensitive. A logon ID is required only if security is enabled for the database you are connecting to. Your ODBC application may override this value or you can override this value in the logon dialog box or connection string.

**Interfaces File:** The path name of the interfaces file. The default is the normal Sybase interfaces file.

**Workstation ID:** The workstation ID used by the client.

**Charset:** The name of a character set corresponding to a subdirectory in \$SYBASE/charsets. The default is the setting on the Sybase server.

**Application Name:** The name used by Sybase to identify your application.

**Language:** The national language corresponding to a subdirectory in \$SYBASE/locales. The default is English.

**Directory Service Provider:** 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.

Directory Service Provider is not available on the Macintosh.



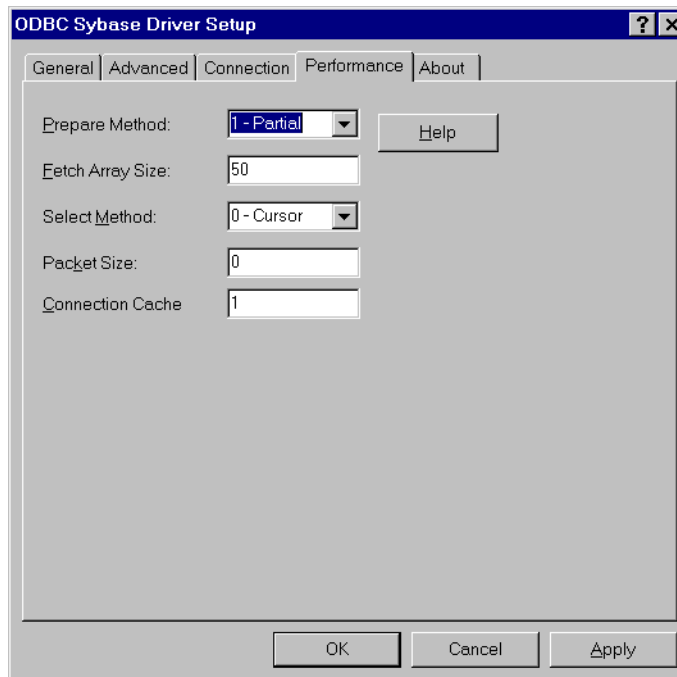
**Security Service Provider:** A string that indicates which Security Service Provider the Sybase Open Client uses when connecting with this data source. The available Security Service Providers can be found using the OpenClient/OpenServer Configuration Utility that is installed with Sybase Open Client version 11.1 or higher. If the client is not using Open Client version 11.1 or higher, this option is ignored.



Security Service Provider is not available on the Macintosh.

**Password Encryption:** A value of 0 or 1 that determines whether password encryption can be performed from the Open Client Library to the server. A value of 1 enables this password encryption; a value of 0, the default, does not.

- 8 Click the **Performance** tab to configure performance settings for this data source.



- 9 Specify values as follows; then, click **Apply**:

**Prepare Method:** A value of 0, 1, or 2 that determines whether stored procedures are created on the server for every call to SQLPrepare.

When set to 0, stored procedures are created for every call to SQLPrepare. This setting can result in bad performance when 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. This setting is ignored when connected to Sybase 4.9.2 servers.

**Fetch Array Size:** The number of rows the driver retrieves when fetching from the server. This is not the number of rows given to the user. The default is 10 rows.

**Select Method:** A value of 0 or 1 that determines whether database cursors are used for Select statements. When set to 0, the default, database cursors are used; when set to 1, Select statements are run directly without using database cursors. A setting of 1 limits the data source to one active statement. This setting is ignored when connected to Sybase 4.9.2 servers.

**Packet Size:** A value of -1, 0, or x that determines the number of bytes per network packet transferred from the database server to the client. The correct setting of this attribute can improve performance.

When set to 0, the default, the driver uses the default packet size as specified in the 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, 6 means to set the packet size to  $6 * 512 = 3072$  bytes).

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 identifies a connect option, `SQL_PACKET_SIZE`, that offers this same functionality. To avoid conflicts with applications that may set both the connection string attribute and the ODBC connect option, they have been defined as mutually exclusive. If `PacketSize` is specified, you will receive a message "Driver Not Capable" if you attempt to call `SQL_PACKET_SIZE`. If you do not set `PacketSize`, then application calls to `SQL_PACKET_SIZE` are accepted by the driver.

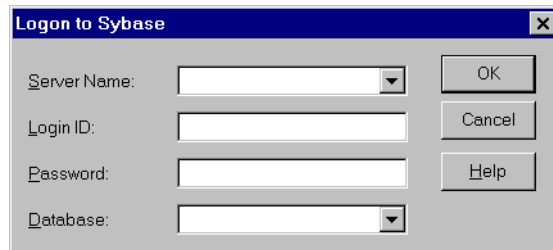
**Connection Cache:** A value that determines the number of connections that the connection cache can hold. The default Connection Cache setting is 1. To set the connection cache, you must set the Select Method option to 1 - Direct. Increasing the connection cache may increase performance of some applications but requires additional database resources.

- 10 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Connecting to a Data Source Using a Logon Dialog Box

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



In this dialog box, do the following:

- 1 Type the case-sensitive name of the server containing the Sybase database tables you want to access or select the name from the Server Name drop-down list, which displays the server names you specified in the ODBC Sybase Driver Setup dialog box.
- 2 If required, type your case-sensitive login ID.
- 3 If required, type your case-sensitive password for the system.
- 4 Type the name of the database you want to access (case-sensitive) or select the name from the Database drop-down list, which displays the names you specified in the ODBC Sybase Driver Setup dialog box.
- 5 Click **OK** to complete the logon and to update the values in the system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 13-1](#) gives the long and short names for each attribute, as well as a description.



To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. For information about this file, see [Appendix H, “The UNIX Environment,”](#) on page 369.

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 your default.

**Table 13-1. Sybase Connection String Attributes**

<b>Attribute</b>	<b>Description</b>
DataSourceName (DSN)	A string that identifies a single connection to a Sybase database. Examples include "Accounting" or "Sys10-Serv1."
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.
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.
Password (PWD)	A case-sensitive password.
Database (DB)	The name of the database to which you want to connect.
Language (LANG)	The national language corresponding to a subdirectory in \$SYBASE/locales.
Charset (CS)	The name of a character set corresponding to a subdirectory in \$SYBASE/charsets.
WorkstationID (WKID)	The workstation ID used by the client.
ApplicationName (APP)	The name used by Sybase to identify your application.
InterfacesFile (IFILE)	The path name to the interfaces file.
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 10 rows.

**Table 13-1. Sybase Connection String Attributes** (cont.)

Attribute	Description
OptimizePrepare (OP)	<p data-bbox="654 314 1182 435">OptimizePrepare={0   1   2}. This attribute determines whether stored procedures are created on the server for every call to SQLPrepare.</p> <p data-bbox="654 453 1262 574">When set to 0, stored procedures are created for every call to SQLPrepare. This setting can result in bad performance when processing static statements.</p> <p data-bbox="654 591 1262 713">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.</p> <p data-bbox="654 730 1219 788">When set to 2, the driver never creates stored procedures.</p> <p data-bbox="654 805 1248 829">This attribute is ignored for Sybase 4.9.2 servers.</p>
SelectMethod (SM)	<p data-bbox="654 847 1239 1100">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.</p> <p data-bbox="654 1117 1262 1211">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.</p> <p data-bbox="654 1229 1248 1253">This attribute is ignored for Sybase 4.9.2 servers.</p>
Password Encryption (PE)	<p data-bbox="654 1270 1262 1420">PasswordEncryption={0   1}. This attribute determines whether password encryption can be performed from the Open Client Library to the server (PasswordEncryption=1). When set to 0, the initial default, this cannot be done.</p>



**Table 13-1. Sybase Connection String Attributes** (cont.)

Attribute	Description
PacketSize (PS)	<p data-bbox="692 314 1300 470">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.</p> <p data-bbox="692 487 1300 574">When set to 0, the initial default, the driver uses the default packet size as specified in the Sybase server configuration.</p> <p data-bbox="692 591 1300 713">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.</p> <p data-bbox="692 730 1300 852">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 <math>6 * 512 = 3072</math> bytes).</p> <p data-bbox="692 869 1300 1025">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:</p> <pre data-bbox="692 1043 1300 1156">sp_configure "maximum network packet size", 5120 reconfigure Restart Sybase Server</pre> <p data-bbox="692 1173 1300 1519">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.</p>

**Table 13-1. Sybase Connection String Attributes** (cont.)

Attribute	Description
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.
ApplicationUsingThreads (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.
EnableQuotedIdentifiers (EQI)	EnableQuotedIdentifiers={0   1}. Specify 1 to allow support of quoted identifiers. The default is 0.
DefaultLongDataBuffLen (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.
InitializationString (IS)	InitializationString={<Sybase set commands>;...}. Supports the execution of Sybase commands at connect time. Multiple commands must be separated by semicolons.

**Table 13-1. 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.
SecurityService Provider (SSP)	<p data-bbox="692 614 1290 675">Directory Service Provider is <i>not</i> available on the Macintosh.</p> <p data-bbox="692 690 1300 942">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.</p>
	<p data-bbox="692 956 1272 1017">Security Service Provider is <i>not</i> available on the Macintosh.</p>

---

## Data Types

Table 13-2 shows how the Sybase data types are mapped to the standard ODBC data types.

---

**Table 13-2. Sybase Data Types**

---

<b>Sybase</b>	<b>ODBC</b>
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,"](#) on page 335 for a discussion of these topics.

---

## ODBC Conformance Level

The Sybase driver supports the functions listed in [Appendix C, "ODBC API and Scalar Functions,"](#) on page 323. 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.



# 14 Connect ODBC for Text



Connect ODBC for Text (the “Text driver”) supports ASCII text files in the Windows 95 and Windows NT, Macintosh, and UNIX environments. These files can be printed directly or edited with text editors or word processors, because none of the data is stored in a binary format.

See the README file shipped with your INTERSOLV DataDirect product for the 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.

---

## System Requirements



Macintosh users who are accessing the same text file must have file sharing enabled.

---

## Formats for Text Files

Some common formats for text files are listed in [Table 14-1](#).

---

**Table 14-1. Common Text File Formats**

---

<b>Format</b>	<b>Description</b>
Comma-separated values	Commas separate column values, and each line is a separate record. Column values can vary in length. These files often have the .CSV extension.
Tab-separated values	Tabs separate column values, and each line is a separate record. Column values can vary in length.
Character-separated values	Any printable character except single or double quotation marks can separate column values, and each line is a separate record. Column values can vary in length.
Fixed	No character separates column values. Instead, values start at the same position and have the same length in each line. The values appear in fixed columns if you display the file. Each line is a separate record.
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.



# Configuring Data Sources



**Note:** In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 14-4 on page 282](#). You must also edit this file to perform a translation. For information about this file, see [Appendix H, “The UNIX Environment,” on page 369](#).

To configure a Text data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Text Driver Setup dialog box.

If you are configuring a new data source, click **Add** to display a list of installed drivers. Select the Text driver and click **Finish** to display the ODBC Text Driver Setup dialog box.

The screenshot shows the 'ODBC Text Driver Setup' dialog box. It has a title bar with a question mark and a close button. Below the title bar are three tabs: 'General', 'Advanced', and 'About'. The 'General' tab is selected. The dialog contains the following fields and controls:

- Data Source Name:** A text box containing the word 'Text'.
- Description:** An empty text box.
- Database Directory:** An empty text box with a 'Help' button to its right.
- Default Table Type:** A dropdown menu currently showing 'Comma'.
- Delimiter Character:** An empty text box.
- Column Names in First Line:** A checkbox that is currently unchecked.

At the bottom of the dialog are three buttons: 'OK', 'Cancel', and 'Apply'.

3 Specify values as follows; then, click **Apply**:



**Apply** is not available on the Macintosh. Clicking **OK** saves the values.

**Data Source Name:** A string that identifies this Text data source configuration in the system information. Examples include “Accounting” or “Text Files.”

**Description:** An optional long description of a data source name. For example, “My Accounting Files” or “My Text Files in the Accounting Directory.”

**Database Directory:** The directory in which the text files are stored. If none is specified, the current working directory is used.



On the Macintosh, click **Select Directory**.

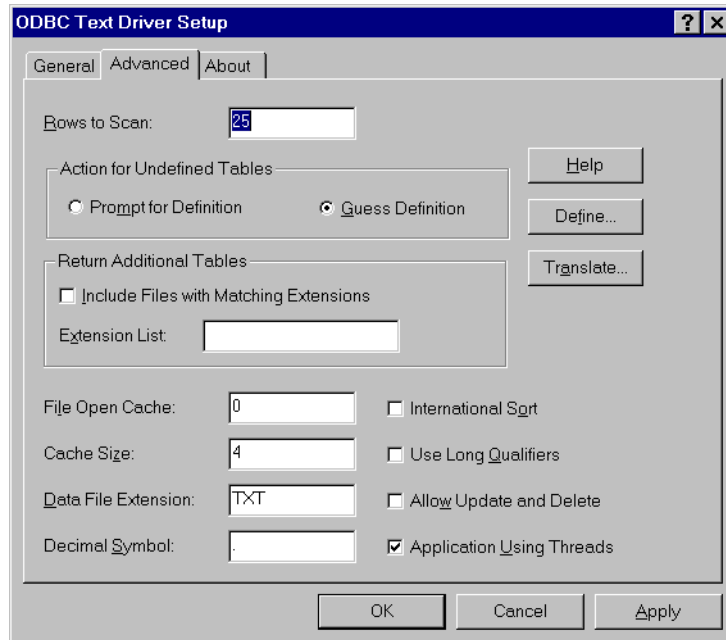
**Default Table Type:** The type of text file: comma-separated, tab-separated, character-separated, fixed length, or stream. This value tells the driver the default type, which is used when creating a new table and opening an undefined table.

**Delimiter Character:** The character used as a delimiter for character-separated files. It can be any printable character. The default is a comma (,).

**Column Names in First Line:** Select this check box to tell the driver to look for column names in the first line of the file.

**Note:** The Default Table Type, Delimiter Character, and Column Names in First Line settings apply only to tables not previously defined. These fields also determine the attributes of new tables created with the Create Table statement.

4 Click the **Advanced** tab to configure additional, optional settings for the data source.



**5** Specify values as follows; then, click **Apply**:

**Rows to Scan:** The number of rows in a text file that the driver scans to determine the data types in the file. If the value is 0, all rows in the file are scanned. The default is 25.

**Note:** The Rows to Scan setting applies only to tables *not* previously defined. This field also determines the attributes of new tables created with the Create Table statement.

**Action for Undefined Tables:** Two radio buttons that indicate what action the driver should take when it encounters a file that has not been defined. Select the Prompt for Definition radio button, if you want the driver to prompt the user when it encounters a file whose format is not defined. Otherwise, select the Guess Definition radio button; in this case, the driver guesses the file's format.



On the Macintosh, select Guess or Prompt for Definition from the pop-up menu.

**Return Additional Tables:** Select this check box to tell the driver to return files with a given extension in addition to the files specified in the Data File Extension field. In Extension List, specify a comma-separated list of the extensions. To have files with no extensions returned, specify NONE. For example, if some of your files have the extensions TXT and CSV and others have no extension, specify TXT,CSV,NONE.

By default, when an application requests a list of tables, only files that have been defined are returned.



On the Macintosh, the Return Additional Tables functionality is found on the Mac File Types tab (see [page 270](#)).

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

**Cache Size:** The number of 64 KB 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 default is 4.

**Data File Extension:** 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.

**Decimal Symbol:** A setting that specifies the decimal separator used when data is stored (may be a comma or period). The international decimal symbol <.> must be used in DML statements and parameter buffers.

**International Sort:** A setting to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this box to use 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.



**Use Long Qualifiers:** Select this check box to use long path names as table qualifiers. When you select this check box, path names can be up to 255 characters. The default length for pathnames is 128 characters.

**Allow Update and Delete:** 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.

**Application Using Threads:** 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.

**Define:** Click **Define** to define the structure of your text files as described in [“Defining Table Structure” on page 271](#).

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box and perform the translation.



**6** On the Macintosh, click the **Mac File Types** tab to specify the creator and file types. Specify 4-character, case-sensitive values for the following:

- Text File Creator (default is ttxt)
- Text File Type (default is TEXT)

**Use Macintosh File Types:** Use this setting to locate tables based on the Macintosh file type.

**Use DOS File Extensions:** Use this setting to locate tables based on the DOS file extension.

**Include file with matching file types:** Select this box to return additional tables based on file type (only if “Macintosh File

Types” is selected). Enter the file types in the field as a comma-separated list.

**Include file with matching extensions:** Select this box to return additional tables based on DOS file extension (only if “DOS File Extensions” is selected). Enter the extensions in the field as a comma-separated list.



- 7 On the Macintosh, click the **Define** tab to define the structure of your text files as described in [“Defining Table Structure” on page 271](#).
- 8 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Defining Table Structure

Note that this section does not apply to the UNIX platforms. See [“Defining Table Structure on UNIX Platforms” on page 276](#) for information on how to define table structure on the UNIX platforms.

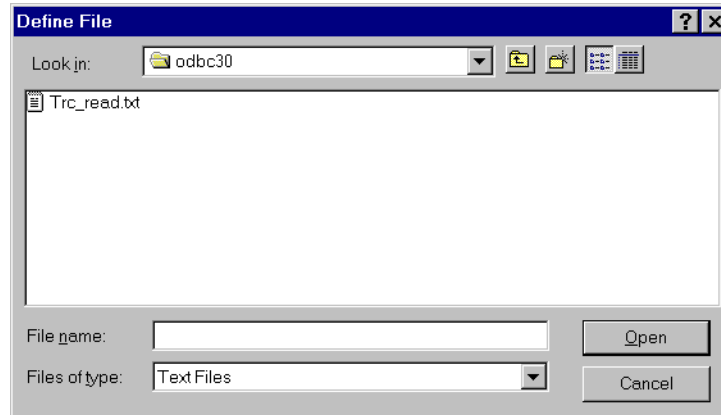
Because text files do not all have the same structure, the driver provides the option of defining the structure of an existing file. Although defining the structure is not mandatory (the driver can attempt to guess the names and types of the columns), this feature is extremely useful.

Define the structure of a file as follows:

- 1 Display the ODBC Text Driver Setup dialog box through the ODBC Administrator. Click the **Advanced** tab; then, click **Define** to display the Define File dialog box.



On the Macintosh, click the **Define** tab instead of the **Advanced** tab. A pop-up menu appears at the top of the **Define** tab. Use this menu to select a database directory and file. The name of the directory and file are displayed on the tab after you have selected them.

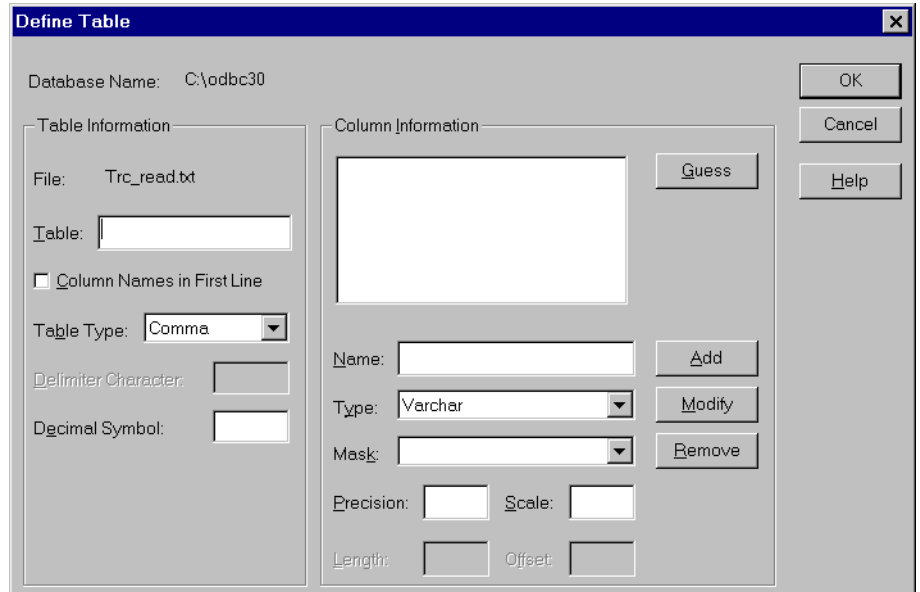


- 2 Select the correct file and click **Open** to display the Define Table dialog box.



On the Macintosh, the **Define** tab is equivalent to the Define Table dialog box.





**Database Name:** The name of the database directory that you selected in the Define File dialog box.

**File:** The name of the file that you selected in the Define File dialog box.

**Table:** Type a table name in the Table box. The name may be up to 32 characters in length and must be unique. This name is returned by SQLTables. By default, it is the file name without its extension.

**Column Names in First Line:** Select this check box if the first line of the file contains column names; otherwise, do not select this box.

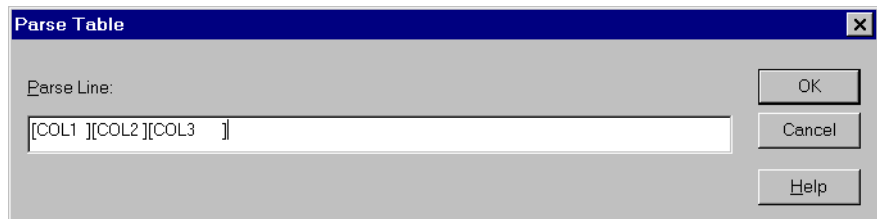
**Table Type:** Select either comma, tab, fixed, character, or stream.

**Delimiter Character:** If the table type is Character, specify the delimiter used in character-separated files.

**Decimal Symbol:** Type a comma to store the data using a comma as the separator for decimal numbers.

- 3 If you specified a *comma-separated*, *tab-separated*, or *character-separated* type in the Table Type field, the **Guess** button is active and you can click it to have the driver guess at the column names and display them in the list box of the Column Information pane.

If you specified a *fixed-length* or *stream* type in the Table Type field, the **Parse** button is active and you can click it to display the Parse Table dialog box and define the table columns.



This dialog box displays the first line of the file. You must mark where *each* field begins and ends by enclosing it in brackets. These brackets indicate the position and length of each field value in the record. Click **OK** to close the Parse Table dialog box. The driver will suggest column names in the list box of the Column Information pane.

- 4 If you do not want the driver to guess or parse, enter values in the following fields to define each column. Click **Add** to add the column name to the list box.



On the Macintosh, the Name, Type, Mask, Precision, Scale, Length, and Offset fields are displayed on a separate dialog. To

define a column, first click **Add**. After setting the values, click **OK** to exit the dialog. The column name will be added to the list box on the **Define** tab.

**Name:** Type the name of the column.

**Type:** Select the data type of the column. If the field type is Date, you must select a date mask for the field or type one in. See ["Date Masks" on page 279](#) for more information.

**Precision:** Type the precision of the column. The precision of numeric data types is defined as the maximum number of digits used by the data type of the column. For character types, this is the length in characters of the data; for binary data types, precision is defined as the length in bytes of the data. For time, timestamp, and all interval data types, precision is the number of characters in the character representation of this data.

**Scale:** Type the scale of the column. The scale of decimal and numeric data types is defined as the maximum number of digits to the right of the decimal point. For approximate floating point number columns, the scale is undefined, since the number of digits to the right of the decimal point is not fixed. For datetime or interval data that contains a seconds component, the scale is defined as the number of digits to the right of the decimal point in the seconds component of the data.

**Note:** The precision and scale values determine how numeric data is to be returned.

**Length:** If you specified a fixed-length table type, length is the number of bytes the data takes up in storage.

**Offset:** If you specified a fixed-length table type, offset is the number of bytes from the start of the table to the start of the field.

- 5 To modify an existing column definition, select the column name in the list box. Modify the values for that column name; then, click **Modify**.



On the Macintosh, select the column name; then, click **Modify** to display the separate dialog. Modify the values for that column name; then, click **OK**.

- 6 To delete an existing column definition, select a column name in the list box and click **Remove**.
- 7 Click **OK** to define the table.

---

## Defining Table Structure on UNIX Platforms



Because text files do not all have the same structure, the driver provides the option 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 system information 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 ["Date Masks" on page 279](#) for information about masks.

Separate the values with commas. For example, to define 2 fields,

```
FIELD1=EMP_ID,VARCHAR,6,0,6,0,
FIELD2=HIRE_DATE,DATE,10,0,10,0,m/d/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 14-2](#) lists the symbols to use when specifying the date mask.

---

**Table 14-2. Date Masks for Text Driver**

---

<b>Symbol</b>	<b>Description</b>
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 Ms (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).

**Table 14-2. Date Masks for Text Driver** (cont.)

Symbol	Description
yy	Output the last two digits of the year.
yyyy	Output the full four digits of the year.
J	Output the Julian value for the date. The Julian value is the number of days since 4712 BC.
\ - . : , (space)	Special characters used to separate the parts of a date.
\	Output the next character. For example, if the mask is mm/dd/yyyy \AD, the value appears as 10/01/1993 AD in the text file.
"string", 'string'	Output the string in the text file.

[Table 14-3](#) shows some example date values, masks, and how the date appears in the text file.

**Table 14-3. Date Mask Examples**

Date	Mask	Value
1993-10-01	yyyy-mm-dd	1993-10-01
	m/d/yy	10/1/93
	Ddd, Mmm dd, yyyy	Fri, Oct 01, 1993



---

# Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information 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 14-4](#) gives the long and short names for each attribute, as well as a description.



To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. For information about this file, see [Appendix H, "The UNIX Environment,"](#) on page 369.

**Table 14-4** also lists the 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 your default.

---

**Table 14-4. Text Connection String Attributes**

---

Attribute	Description
DataSourceName (DSN)	A string that identifies a Text data source configuration in the system information. Examples include "Accounting" or "Text Files."
Database (DB)	The directory in which the text files are stored.
ScanRows (SR)	The number of rows in a text file that the driver scans to determine the column types in the file. If the value is 0, all rows in the file are scanned. The initial default is 25.
TableType (TT)	TableType={Comma   Tab   Character   Fixed   Stream}. The Text driver supports four 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.
Delimiter (DC)	The character used as a delimiter for character-separated files. It can be any printable character except single or double quotes. The initial default is a comma (,).
UndefinedTable (UT)	The Text driver can perform two operations when it encounters a file that has not been defined. UndefinedTable=Prompt tells the driver to display a dialog box that allows the user to describe the file's format. UndefinedTable=Guess tells the driver to guess the file's format. This is the initial default.

---

**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 14-4. Text Connection String Attributes** (cont.)

Attribute	Description
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 <code>None</code> .
FileOpenCache (FOC)	The maximum number of unused file opens to cache. For example, when <code>FileOpenCache=4</code> , and a user opens and closes four files, the files are not actually closed. The driver keeps them open so that if another query uses one of these files, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The disadvantage is that a user who tries to open the file exclusively may get a locking conflict even though no one appears to have the file open. The initial default is 0.
CacheSize (CSZ)	The number of 64 KB 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 <code>Select</code> statement again. The initial default is 4.
FirstLineNames (FLN)	<code>FirstLineNames={0   1}</code> . This attribute determines whether the driver looks for column names in the first line of the file. If <code>FirstLineNames=1</code> , the driver looks for column names in the first line of the file. If <code>FirstLineNames=0</code> (the initial default), the first line is interpreted as the first record in the file.

**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 14-4. Text Connection String Attributes** (cont.)

Attribute	Description
DataFileExtension (DFE)	<p>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.</p> <p>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.</p>
IntlSort (IS)	<p>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."</p> <p>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.</p>
ApplicationUsingThreads (AUT)	<p>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.</p>

---



**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 14-4. Text Connection String Attributes** (cont.)

Attribute	Description
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.
AllowUpdateAndDelete (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.
MacFileInfo (MFI)	On Macintosh systems, four-character, case-sensitive values that specify the following in the order shown:
	<ul style="list-style-type: none"> <li>■ Text File Creator (default is ttxt)</li> <li>■ Text File Type (default is TEXT)</li> </ul>
The values are specified in a comma-separated list. For example, MacFileInfo=ABCD,EFGH.	
MacCompatible (MC)	MacCompatible={0   1}. On Macintosh systems, allows the user to specify whether tables should be accessed by DOS file extension (0) or Macintosh file type (1). The default is 0. If you are accessing tables by extension and want to return additional tables, use ExtraExtensions. If you are accessing tables by file type and want to return additional tables, use MacFileTypesList.
	

**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 14-4. Text Connection String Attributes** (cont.)

Attribute	Description
MacFileTypesList (MFTL) 	On Macintosh systems, specifies which file types will be used when returning additional tables. The default is none. The values are specified in a comma-separated list. For example, MacFileTypesList=ABCD,EFGH,IJKL,MNOP,QRST.
UseLongQualifiers (ULQ) 	UseLongQualifiers={0   1}. It specifies whether the driver uses long pathnames as table qualifiers. The default is 0, do not use long path names (the default length of path names is 128 characters). If UseLongQualifiers=1, the driver uses long path names (up to 255 characters).

**Note:** The ScanRows, TableType, Delimiter, FirstLineNames, and Charset attributes apply to tables that have *not* been defined. These attributes also determine the characteristics of new tables created with the Create Table statement.

## Data Types

Table 14-5 shows how the text file data types are mapped to the standard ODBC data types.

**Table 14-5. 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,"](#) on page 289.

---

## 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 functions listed in [Appendix C, “ODBC API and Scalar Functions,”](#) on page 323. 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.



# A SQL for Flat-File Drivers

This appendix describes the SQL statements that you can use with the flat-file drivers (Btrieve, dBASE, Excel, Paradox, and Text). The database drivers parse SQL statements and translate them into a form that the database can understand. The SQL statements in this appendix let you

- Read, insert, update, and delete records from a database
- Create new tables
- Drop existing tables

These SQL statements allow your application to be portable across other databases.

---

## Select Statement

The form of the Select statement supported by the flat-file drivers is

```
SELECT [DISTINCT] { * | column_expression, ... }
FROM table_names [table_alias] ...
[ WHERE expr1 rel_operator expr2 ]
[ GROUP BY { column_expression, ... } ]
[ HAVING expr1 rel_operator expr2 ]
[ UNION [ALL] (SELECT...) ]
[ ORDER BY { sort_expression [DESC | ASC]}, ... ]
[ FOR UPDATE [OF { column_expression, ... } ] ]
```

## Select Clause

Follow Select with a list of column expressions you want to retrieve or an asterisk (\*) to retrieve all fields.

```
SELECT [DISTINCT] { * | column_expression,
[[AS] column_alias]. . . }
```

*column\_expression* can be simply a field name (for example, LAST\_NAME). More complex expressions may include mathematical operations or string manipulation (for example, SALARY \* 1.05). See [“SQL Expressions” on page 296](#).

*column\_alias* can be used to give the column a more descriptive name. For example, to assign the alias DEPARTMENT to the column DEP,

```
SELECT dep AS department FROM emp
```

Separate multiple column expressions with commas (for example, LAST\_NAME, FIRST\_NAME, HIRE\_DATE).

Field names can be prefixed with the table name or alias. For example, EMP.LAST\_NAME or E.LAST\_NAME, where E is the alias for the table EMP.

The Distinct operator can precede the first column expression. This operator eliminates duplicate rows from the result of a query. For example:

```
SELECT DISTINCT dep FROM emp
```

## Aggregate Functions

Aggregate functions can also be a part of a Select clause. Aggregate functions return a single value from a set of records. An aggregate can be used with a field name (for example, AVG(SALARY)) or in combination with a more complex column expression (for example, AVG(SALARY \* 1.07)). The column

expression can be preceded by the Distinct operator. The Distinct operator eliminates duplicate values from an aggregate expression. For example:

```
COUNT (DISTINCT last_name)
```

In this example, only distinct last name values are counted.

[Table A-1](#) lists valid aggregates.

---

**Table A-1. Aggregate Functions**

---

<b>Aggregate</b>	<b>Returns</b>
SUM	The total of the values in a numeric field expression. For example, SUM(SALARY) returns the sum of all salary field values.
AVG	The average of the values in a numeric field expression. For example, AVG(SALARY) returns the average of all salary field values.
COUNT	The number of values in any field expression. For example, COUNT(NAME) returns the number of name values. When using COUNT with a field name, COUNT returns the number of non-null field values. A special example is COUNT(*), which returns the number of records in the set, including records with null values.
MAX	The maximum value in any field expression. For example, MAX(SALARY) returns the maximum salary field value.
MIN	The minimum value in any field expression. For example, MIN(SALARY) returns the minimum salary field value.

---

## From Clause

The From clause indicates the tables that will be used in the Select statement. The format of the From clause is

```
FROM table_names [table_alias]
```

*table\_names* can be one or more simple table names in the current working directory or complete pathnames.

*table\_alias* is a name used to refer to this table in the rest of the Select statement. Database field names may be prefixed by the table alias. Given the table specification

```
FROM emp E
```

you may refer to the LAST\_NAME field as E.LAST\_NAME. Table aliases must be used if the Select statement joins a table to itself. For example:

```
SELECT * FROM emp E, emp F WHERE E.mgr_id = F.emp_id
```

The equal sign (=) includes only matching rows in the results.

If you are joining more than one table, you can use LEFT OUTER JOIN, which includes nonmatching rows in the first table you name. For example:

```
SELECT * FROM T1 LEFT OUTER JOIN T2 on T1.key = T2.key
```

## Where Clause

The Where clause specifies the conditions that records must meet to be retrieved. The Where clause contains conditions in the form:

```
WHERE expr1 rel_operator expr2
```

*expr1* and *expr2* may be field names, constant values, or expressions.

*rel\_operator* is the relational operator that links the two expressions. See [“SQL Expressions” on page 296](#).

For example, the following Select statement retrieves the names of employees that make at least \$20,000.

```
SELECT last_name,first_name FROM emp WHERE salary >= 20000
```

## Group By Clause

The Group By clause specifies the names of one or more fields by which the returned values should be grouped. This clause is used to return a set of aggregate values.

It has the following form:

```
GROUP BY column_expressions
```

*column\_expressions* must match the column expression used in the Select clause. A column expression can be one or more field names of the database table, separated by a comma (,) or one or more expressions, separated by a comma (,). See [“SQL Expressions” on page 296](#).

The following example sums the salaries in each department.

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id
```

This statement returns one row for each distinct department ID. Each row contains the department ID and the sum of the salaries of the employees in the department.

## Having Clause

The Having clause enables you to specify conditions for groups of records (for example, display only the departments that have salaries totaling more than \$200,000). This clause is valid only if you have already defined a Group By clause.

It has the following form:

```
HAVING expr1 rel_operator expr2
```

*expr1* and *expr2* can be field names, constant values, or expressions. These expressions do not have to match a column expression in the Select clause.

*rel\_operator* is the relational operator that links the two expressions. See the section "SQL Expressions" later in this appendix.

The following example returns only the departments whose sums of salaries are greater than \$200,000:

```
SELECT dept_id, sum(salary) FROM emp
GROUP BY dept_id HAVING sum(salary) > 200000
```

## Union Operator

The Union operator combines the results of two Select statements into a single result. The single result is all of the returned records from both Select statements. By default, duplicate records are not returned. To return duplicate records, use the All keyword (UNION ALL). The form is

```
SELECT statement
UNION [ALL]
SELECT statement
```

When using the Union operator, the select lists for each Select statement must have the same number of column expressions

with the same data types and must be specified in the same order. For example:

```
SELECT last_name, salary, hire_date FROM emp
UNION
SELECT name, pay, birth_date FROM person
```

This example has the same number of column expressions, and each column expression, in order, has the same data type.

The following example is *not* valid because the data types of the column expressions are different (SALARY from EMP has a different data type than LAST\_NAME from RAISES). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
UNION
SELECT salary, last_name FROM raises
```

## Order By Clause

The Order By clause indicates how the records are to be sorted. The form is

```
ORDER BY {sort_expression [DESC | ASC]}, ...
```

*sort\_expression* can be field names, expressions, or the positional number of the column expression to use.

The default is to perform an ascending (ASC) sort.

For example, to sort by LAST\_NAME then by FIRST\_NAME you could use either of the following Select statements:

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY last_name, first_name
```

or

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY 2,3
```

In the second example, LAST\_NAME is the second column expression following Select, so Order By 2 sorts by LAST\_NAME.

## For Update Clause

The For Update clause locks the records of the database table selected by the Select statement. The form is

```
FOR UPDATE [OF column_expressions]
```

*column\_expressions* is a list of field names in the database table that you intend to update, separated by a comma (.). Note that *column\_expressions* is optional.

The following example returns all records in the employee database that have a SALARY field value of more than \$20,000. When each record is fetched, it is locked. If the record is updated or deleted, the lock is held until you commit the change. Otherwise, the lock is released when you fetch the next record.

```
SELECT * FROM emp WHERE salary > 20000
      FOR UPDATE OF last_name, first_name, salary
```

## SQL Expressions

Expressions are used in the Where clauses, Having clauses, and Order By clauses of SQL Select statements.

Expressions enable you to use mathematical operations as well as character string and date manipulation operators to form complex database queries.

The most common expression is a simple field name. You can combine a field name with other expression elements.



Valid expression elements are as follows:

- Field names
- Constants
- Exponential notation
- Numeric operators
- Character operators
- Date operators
- Relational operators
- Logical operators
- Functions

## ***Constants***

Constants are values that do not change. For example, in the expression `PRICE * 1.05`, the value 1.05 is a constant.

You must enclose character constants in pairs of single (') or double quotation marks ("). To include a single quotation mark in a character constant enclosed by single quotation marks, use two single quotation marks together (for example, 'Don't'). Similarly, if the constant is enclosed by double quotation marks, use two double quotation marks to include one.

You must enclose date and time constants in braces ({}), for example, {01/30/89} and {12:35:10}. The form for date constants is MM/DD/YY or MM/DD/YYYY. The form for time constants is HH:MM:SS.

The logical constants are .T. and 1 for True and .F. and 0 for False. For portability, use 1 and 0.

## ***Exponential Notation***

You may include exponential notation. For example:

```
SELECT col1, 3.4E+7 FROM table1 WHERE calc < 3.4E-6 * col2
```

## ***Numeric Operators***

You may include the following operators in numeric expressions

<b>Operator</b>	<b>Meaning</b>
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation
^	Exponentiation

The following table shows examples of numeric expressions. For these examples, assume SALARY is 20000.

<b>Example</b>	<b>Resulting value</b>
salary + 10000	30000
salary * 1.1	22000
2 ** 3	8

You can precede numeric expressions with a unary plus (+) or minus (-). For example, -(salary \* 1.1) is -22000.

## Character Operators

Character expressions may include the following operators:

Operator	Meaning
+	Concatenation keeping trailing blanks.
-	Concatenation moving trailing blanks to the end.

The following chart shows examples of character expressions. In the examples, LAST\_NAME is ' JONES ' and FIRST\_NAME is ' ROBERT '.

Example	Resulting value
<code>first_name + last_name</code>	' ROBERT        JONES '
<code>first_name - last_name</code>	' ROBERTJONES '

**Note:** Some flat-file drivers return character data with trailing blanks as shown in the table; however, you cannot rely on the driver to return blanks. Therefore, if you want an expression that works with drivers that do and do not return trailing blanks, use the TRIM function before concatenating strings to make the expression portable. For example:

```
TRIM(first_name) + ' ' + TRIM(last_name)
```

## Date Operators

You may include the following operators in date expressions:

Operator	Meaning
+	Add a number of days to a date to produce a new date.
-	The number of days between two dates, or subtract a number of days from a date to produce a new date.

The following chart shows examples of date expressions. In these examples, hire\_date is {01/30/90}.

Example	Resulting value
hire_date + 5	{02/04/90}
hire_date - {01/01/90}	29
hire_date - 10	{01/20/90}

## ***Relational Operators***

The relational operators separating the two expressions may be any one of those listed in [Table A-2](#).

---

***Table A-2. Relational Operators***

---

Operator	Meaning
=	Equal.
<>	Not Equal.
>	Greater Than.
>=	Greater Than or Equal.
<	Less Than.
<=	Less Than or Equal.
Like	Matching a pattern.
Not Like	Not matching a pattern.
Is Null	Equal to Null.
Is Not Null	Not Equal to Null.
Between	Range of values between a lower and upper bound.
In	A member of a set of specified values or a member of a subquery.
Exists	True if a subquery returned at least one record.

**Table A-2. Relational Operators** (cont.)

Operator	Meaning
Any	Compares a value to each value returned by a subquery. Any must be prefaced by =, <>, >, >=, <, or <=.  =Any is equivalent to In.
All	Compares a value to each value returned by a subquery. All must be prefaced by =, <>, >, >=, <, or <=.

The following list shows some examples of relational operators:

```
salary <= 40000
dept = 'D101'
hire_date > {01/30/89}
salary + commission >= 50000
last_name LIKE 'Jo%'
salary IS NULL
salary BETWEEN 10000 AND 20000
WHERE salary = ANY (SELECT salary FROM emp WHERE
    dept = 'D101')
WHERE salary > ALL (SELECT salary FROM emp WHERE
    dept = 'D101')
```

## Logical Operators

Two or more conditions may be combined to form more complex criteria. When two or more conditions are present, they must be related by AND or OR. For example:

```
salary = 40000 AND exempt = 1
```

The logical NOT operator is used to reverse the meaning. For example:

```
NOT (salary = 40000 AND exempt = 1)
```

## Operator Precedence

As expressions become more complex, the order in which the expressions are evaluated becomes important. [Table A-3](#) shows the order in which the operators are evaluated. The operators in the first line are evaluated first, then those in the second line, and so on. Operators in the same line are evaluated left to right in the expression.

---

**Table A-3. Operator Precedence**

---

Precedence	Operator
1	Unary -, Unary +
2	**
3	*, /
4	+, -
5	=, <>, <, <=, >, >=, Like, Not Like, Is Null, Is Not Null, Between, In, Exists, Any, All
6	Not
7	AND
8	OR

---

The following example shows the importance of precedence:

```
WHERE salary > 40000 OR
hire_date > {01/30/89} AND
dept = 'D101'
```

Because AND is evaluated first, this query retrieves employees in department D101 hired after January 30, 1989, as well as every employee making more than \$40,000, no matter what department or hire date.

To force the clause to be evaluated in a different order, use parentheses to enclose the conditions to be evaluated first. For example:

```
WHERE (salary > 40000 OR hire_date > {01/30/89})
AND dept = 'D101'
```

retrieves employees in department D101 that either make more than \$40,000 or were hired after January 30, 1989.

## Functions

The flat-file drivers support a number of functions that you may use in expressions. In [Table A-4](#) through [Table A-6](#), the functions are grouped according to the type of result they return.

---

**Table A-4. Functions that Return Character Strings**

---

Function	Description
CHR	Converts an ASCII code into a one-character string. CHR(67) returns C.
RTRIM	Removes trailing blanks from a string. RTRIM('ABC ') returns ABC.
TRIM	Removes trailing blanks from a string. TRIM('ABC ') returns ABC.
LTRIM	Removes leading blanks from a string. LTRIM(' ABC') returns ABC.
UPPER	Changes each letter of a string to uppercase. UPPER('Allen') returns ALLEN.
LOWER	Changes each letter of a string to lowercase. LOWER('Allen') returns allen.
LEFT	Returns leftmost characters of a string. LEFT('Mattson', 3) returns Mat.
RIGHT	Returns rightmost characters of a string. RIGHT('Mattson', 4) returns tson.

**Table A-4. Functions that Return Character Strings** (cont.)

Function	Description
SUBSTR	<p>Returns a substring of a string. Parameters are the string, the first character to extract, and the number of characters to extract (optional).</p> <p>SUBSTR( 'Conrad' , 2 , 3 ) returns onr.</p> <p>SUBSTR( 'Conrad' , 2 ) returns onrad.</p>
SPACE	<p>Generates a string of blanks.</p> <p>SPACE( 5 ) returns '      '.</p>
DTOC	<p>Converts a date to a character string. An optional second parameter determines the format of the result:</p> <p>0 (the default) returns MM/DD/YY</p> <p>1 returns DD/MM/YY</p> <p>2 returns YY/MM/DD</p> <p>10 returns MM/DD/YYYY</p> <p>11 returns DD/MM/YYYY</p> <p>12 returns YYYY/MM/DD</p> <p>An optional third parameter specifies the date separator character. If not specified, a slash (/) is used.</p> <p>DTOC( { 01/30/97 } ) returns 01/30/97</p> <p>DTOC( { 01/30/97 } , 0 ) returns 01/30/97</p> <p>DTOC( { 01/30/97 } , 1 ) returns 30/01/97</p> <p>DTOC( { 01/30/97 } , 2 , '-' ) returns 97-01-30</p>
DTOS	<p>Converts a date to a character string using the format YYYYMMDD.</p> <p>DTOS( { 01/23/90 } ) returns 19900123.</p>
IIF	<p>Returns one of two values. Parameters are a logical expression, the true value, and the false value. If the logical expression evaluates to True, the function returns the true value. Otherwise, it returns the false value.</p> <p>IIF( salary&gt;20000 , 'BIG' , 'SMALL' ) returns BIG if SALARY is greater than 20000. If not, it returns SMALL.</p>



**Table A-4. Functions that Return Character Strings** (cont.)

Function	Description
STR	<p>Converts a number to a character string. Parameters are the number, the total number of output characters (including the decimal point), and optionally the number of digits to the right of the decimal point.</p> <p>STR(12.34567,4) returns 12</p> <p>STR(12.34567,4,1) returns 12.3</p> <p>STR(12.34567,6,3) returns 12.346</p>
STRVAL	<p>Converts a value of any type to a character string.</p> <p>STRVAL('Woltman') returns Woltman</p> <p>STRVAL({12/25/53}) returns 12/25/53</p> <p>STRVAL(5 * 3) returns 15</p> <p>STRVAL(4 = 5) returns 'False'</p>
TIME	<p>Returns the time of day as a string.</p> <p>At 9:49 PM, TIME() returns 21:49:00</p>
TTOC	<p><b>Note:</b> This function is applicable only for those flat-file drivers that support SQL_TIMESTAMP, the Btrieve, Excel 4, Excel 5, FoxPro 3.0, and Paradox 5 drivers.</p> <p>Converts a timestamp to a character string. An optional second parameter determines the format of the result:</p> <ul style="list-style-type: none"> <li>■ 0 or none, the default, returns MM/DD/YY HH:MM:SS AM</li> <li>■ 1 returns YYYYMMDDHHMMSS, which is a suitable format for indexing.</li> </ul> <p>TTOC({1992-04-02 03:27:41}) returns 04/02/92 03:27:41 AM.</p> <p>TTOC({1992-04-02 03:27:41, 1}) returns 19920402032741</p>
USERNAME	<p>For Btrieve, the logon ID specified at connect time is returned. For Paradox and Paradox 5 drivers, the user name specified during configuration is returned. For all other flat file drivers, an empty string is returned.</p>

**Table A-5. Functions that Return Numbers**

<b>Function</b>	<b>Description</b>
MOD	Divides two numbers and returns the remainder of the division. MOD(10,3) returns 1
LEN	Returns the length of a string. LEN('ABC') returns 3
MONTH	Returns the month part of a date. MONTH({01/30/89}) returns 1
DAY	Returns the day part of a date. DAY({01/30/89}) returns 30
YEAR	Returns the year part of a date. YEAR({01/30/89}) returns 1989
MAX	Returns the larger of two numbers. MAX(66,89) returns 89
DAYOFWEEK	Returns the day of week (1-7) of a date expression. DAYOFWEEK({05/01/95}) returns 5.
MIN	Returns the smaller of two numbers. MIN(66,89) returns 66
POW	Raises a number to a power. POW(7,2) returns 49
INT	Returns the integer part of a number. INT(6.4321) returns 6
ROUND	Rounds a number. ROUND(123.456, 0) returns 123 ROUND(123.456, 2) returns 123.46 ROUND(123.456, -2) returns 100

**Table A-5. Functions that Return Numbers** (cont.)

Function	Description
NUMVAL	Converts a character string to a number. If the character string is not a valid number, a zero is returned.  NUMVAL( ' 123 ' ) returns the number 123
VAL	Converts a character string to a number. If the character string is not a valid number, a zero is returned.  VAL( ' 123 ' ) returns the number 123

**Table A-6. Functions that Return Dates**

Function	Description
DATE	Returns today's date. If today is 12/25/79, DATE() returns {12/25/79}
TODAY	Returns today's date. If today is 12/25/79, TODAY() returns {12/25/79}
DATEVAL	Converts a character string to a date. DATEVAL( ' 01/30/89 ' ) returns {01/30/89}
CTOD	Converts a character string to a date. An optional second parameter specifies the format of the character string: 0 (the default) returns MM/DD/YY, 1 returns DD/MM/YY, and 2 returns YY/MM/DD. CTOD( ' 01/30/89 ' ) returns {01/30/89} CTOD( ' 01/30/89 ' , 1) returns {30/01/89}

The following examples use some of the number and date functions.

Retrieve all employees that have been with the company at least 90 days:

```
SELECT first_name, last_name FROM emp
      WHERE DATE() - hire_date >= 90
```

Retrieve all employees hired in January of this year or last year:

```
SELECT first_name, last_name FROM emp
      WHERE MONTH(hire_date) = 1
      AND (YEAR(hire_date) = YEAR(DATE())
      OR YEAR(hire_date) = YEAR(DATE()) - 1)
```

---

## Create and Drop Table Statements

The flat-file drivers support SQL statements to create and delete database files. The Create Table statement is used to create files and the Drop Table statement is used to delete files.

### Create Table

The form of the Create Table statement is

```
CREATE TABLE table_name
      (col_definition[,col_definition, ...])
```

*table\_name* can be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources. If it is a simple table name, the file is created in the directory you specified as the database directory in the connection string. If you did not specify a database directory in the connection string, the file is created in the directory you specified as the database directory in .odbc.ini. If you did not specify a database directory in either place, the file is created in

the current working directory at the time you connected to the driver.

*col\_definition* is the column name, followed by the data type, followed by an optional column constraint definition. Values for column names are database specific. The data type specifies a column's data type.

The only column constraint definition currently supported by some flat-file drivers is "not null." Not all flat-file tables support "not null" columns. In the cases where not null is not supported, this restriction is ignored and the driver returns a warning if "not null" is specified for a column. The "not null" column constraint definition is allowed in the driver so that you can write a database-independent application (and not be concerned about the driver raising an error on a Create Table statement with a "not null" restriction).

A sample Create Table statement to create an employee database table is

```
CREATE TABLE emp (last_name CHAR(20) NOT NULL,
    first_name CHAR(12) NOT NULL,
    salary NUMERIC (10,2) NOT NULL,
    hire_date DATE NOT NULL)
```

## Drop Table

The form of the Drop Table statement is

```
DROP TABLE table_name
```

*table\_name* may be a simple table name (EMP) or a full pathname. A simple table name is preferred for portability to other SQL data sources. If it is a simple table name, the file is dropped from the directory you specified as the database directory in the connection string. If you did not specify a database directory in the connection string, the file is deleted

from the directory you specified as the database directory in `.odbc.ini`. If you did not specify a database directory in either of these places, the file is dropped from the current working directory at the time you connected to the driver.

A sample Drop Table statement to delete the employee database table is

```
DROP TABLE emp
```

---

## Insert Statement

The SQL Insert statement is used to add new records to a database table. With it, you can specify either of the following:

- A list of values to be inserted as a new record
- A Select statement that copies data from another table to be inserted as a set of new records

The form of the Insert statement is

```
INSERT INTO table_name [(col_name, ...)]  
{VALUES (expr, ...) | select_statement}
```

*table\_name* may be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources.

*col\_name* is an optional list of column names giving the name and order of the columns whose values are specified in the Values clause. If you omit *col\_name*, the value expressions (*expr*) must provide values for all columns defined in the file and must be in the same order that the columns are defined for the file.

*expr* is the list of expressions giving the values for the columns of the new record. Usually, the expressions are constant values for the columns. Character string values must be enclosed in single or

double quotation marks, date values must be enclosed in braces {}, and logical values that are letters must be enclosed in periods (for example, .T. or .F.).

An example of an Insert statement that uses a list of expressions is

```
INSERT INTO emp (last_name, first_name, emp_id, salary,
  hire_date)
VALUES ('Smith', 'John', 'E22345', 27500, {4/6/91})
```

Each Insert statement adds one record to the database table. In this case a record has been added to the employee database table, EMP. Values are specified for five columns. The remaining columns in the table are assigned a blank value, meaning Null.

*select\_statement* is a query that returns values for each *col\_name* value specified in the column name list. Using a Select statement instead of a list of value expressions lets you select a set of rows from one table and insert it into another table using a single Insert statement.

An example of an Insert statement that uses a Select statement is:

```
INSERT INTO emp1 (first_name, last_name, emp_id, dept, salary)
SELECT first_name, last_name, emp_id, dept, salary from emp
WHERE dept = 'D050'
```

In this type of Insert statement, the number of columns to be inserted must match the number of columns in the Select statement. The list of columns to be inserted must correspond to the columns in the Select statement just as it would to a list of value expressions in the other type of Insert statement. That is, the first column inserted corresponds to the first column selected; the second inserted to the second, etc.

The size and data type of these corresponding columns must be compatible. Each column in the Select list should have a data type that the ODBC driver accepts on a regular Insert/Update of the corresponding column in the Insert list. Values are truncated

when the size of the value in the Select list column is greater than the size of the corresponding Insert list column.

The *select\_statement* is evaluated before any values are inserted. This query cannot be made on the table into which values are inserted.

---

## Update Statement

The SQL Update statement is used to change records in a database file. The form of the Update statement supported for flat-file drivers is

```
UPDATE table_name SET col_name = expr, ...
[ WHERE { conditions | CURRENT OF cursor_name } ]
```

*table\_name* may be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources.

*col\_name* is the name of a column whose value is to be changed. Several columns can be changed in one statement.

*expr* is the new value for the column. The expression can be a constant value or a subquery. Character string values must be enclosed with single or double quotation marks, date values must be enclosed by braces {}, and logical values that are letters must be enclosed by periods (for example, .T. or .F.). Subqueries must be enclosed in parentheses.

The Where clause is any valid clause as described in [“Select Statement” on page 289](#). It determines which records are to be updated.

The Where Current Of *cursor\_name* clause can be used only by developers coding directly to the ODBC API. It causes the row at which *cursor\_name* is positioned to be updated. This is called a



“positioned update.” You must first execute a Select...For Update statement with a named cursor and fetch the row to be updated.

An example of an Update statement on the employee table is

```
UPDATE emp SET salary=32000, exempt=1
WHERE emp_id = 'E10001'
```

The Update statement changes every record that meets the conditions in the Where clause. In this case the salary and exempt status are changed for all employees having the employee ID E10001. Because employee IDs are unique in the employee table, only one record is updated.

An example using a subquery is

```
UPDATE emp SET salary = (SELECT avg(salary) from emp)
WHERE emp_id = 'E10001'
```

In this case, the salary is changed to the average salary in the company for the employee having employee ID E10001.

---

## Delete Statement

The SQL Delete statement is used to delete records from a database table. The form of the Delete statement supported for flat-file drivers is

```
DELETE FROM table_name
[ WHERE { conditions | CURRENT OF cursor_name } ]
```

*table\_name* may be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources.

The Where clause is any valid clause as described in [“Select Statement” on page 289](#). It determines which records are to be

deleted. If you include only the keyword `Where`, all records in the table are deleted but the file is left intact.

The `Where Current Of cursor_name` clause can be used only by developers coding directly to the ODBC API. It causes the row at which *cursor\_name* is positioned to be deleted. This is called a “positioned delete.” You must first execute a `Select...For Update` statement with a named cursor and fetch the row to be deleted.

An example of a Delete statement on the employee table is

```
DELETE FROM emp WHERE emp_id = 'E10001'
```

Each Delete statement removes every record that meets the conditions in the `Where` clause. In this case every record having the employee ID E10001 is deleted. Because employee IDs are unique in the employee table, at most one record is deleted.

---

## Reserved Keywords

The following words are reserved for use in SQL statements. If they are used for file or column names in a database that you use, you must enclose them in quotation marks in any SQL statement where they appear as file or column names.

- |            |          |           |           |
|------------|----------|-----------|-----------|
| ■ ALL      | ■ FROM   | ■ LEFT    | ■ OPTIONS |
| ■ AND      | ■ FULL   | ■ LIKE    | ■ OR      |
| ■ BETWEEN  | ■ GROUP  | ■ NATURAL | ■ ORDER   |
| ■ COMPUTE  | ■ HAVING | ■ NOT     | ■ RIGHT   |
| ■ CROSS    | ■ INNER  | ■ NULL    | ■ UNION   |
| ■ DISTINCT | ■ INTO   | ■ ON      | ■ WHERE   |
| ■ FOR      |          |           |           |

# B Using Indexes

This appendix discusses the ways in which you can improve the performance of database activity using indexes. It provides general guidelines that apply to most databases. Consult your database vendor's documentation for more detailed information.

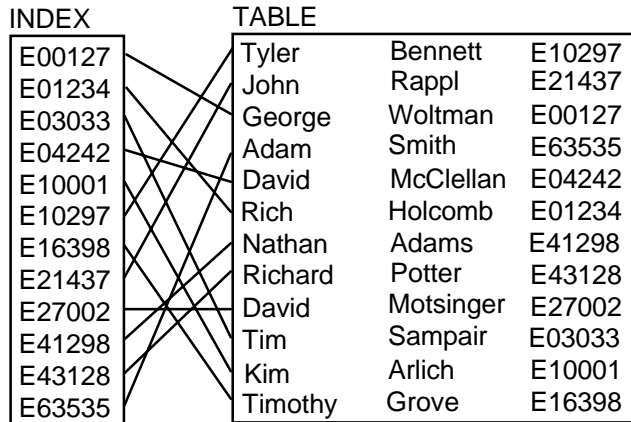
For information regarding how to create and drop indexes, see the appropriate database driver chapter for flat-file drivers or your database system documentation for relational drivers.

---

## Introduction

An index is a database structure that you can use to improve the performance of database activity. A database table can have one or more indexes associated with it.

An index is defined by a field expression that you specify when you create the index. Typically, the field expression is a single field name, like EMP\_ID. An index created on the EMP\_ID field, for example, contains a sorted list of the employee ID values in the table. Each value in the list is accompanied by references to the records that contain that value.



A database driver can use indexes to find records quickly. An index on the EMP\_ID field, for example, greatly reduces the time that the driver spends searching for a particular employee ID value. Consider the following Where clause:

```
WHERE emp_id = 'E10001'
```

Without an index, the driver must search the entire database table to find those records having an employee ID of E10001. By using an index on the EMP\_ID field, however, the driver can quickly find those records.

Indexes may improve the performance of SQL statements. You may not notice this improvement with small tables but it can be significant for large tables; however, there can be disadvantages to having too many indexes. Indexes can slow down the performance of some inserts, updates, and deletes when the driver has to maintain the indexes as well as the database tables. Also, indexes take additional disk space.

---

## Improving Record Selection Performance

For indexes to improve the performance of selections, the index expression must match the selection condition exactly. For example, if you have created an index whose expression is `last_name`, the following Select statement uses the index:

```
SELECT * FROM emp WHERE last_name = 'Smith'
```

This Select statement, however, does not use the index:

```
SELECT * FROM emp WHERE UPPER(last_name) = 'SMITH'
```

The second statement does not use the index because the Where clause contains `UPPER(LAST_NAME)`, which does not match the index expression `LAST_NAME`. If you plan to use the `UPPER` function in all your Select statements and your database supports indexes on expressions, then you should define an index using the expression `UPPER(LAST_NAME)`.

---

## Indexing Multiple Fields

If you often use Where clauses that involve more than one field, you may want to build an index containing multiple fields. Consider the following Where clause:

```
WHERE last_name = 'Smith' and first_name = 'Thomas'
```

For this condition, the optimal index field expression is `LAST_NAME, FIRST_NAME`. This creates a concatenated index.

Concatenated indexes can also be used for Where clauses that contain only the first of two concatenated fields. The LAST\_NAME, FIRST\_NAME index also improves the performance of the following Where clause (even though no first name value is specified):

```
last_name = 'Smith'
```

Consider the following Where clause:

```
WHERE last_name = 'Smith' and middle_name =  
'Edward' and first_name = 'Thomas'
```

If your index fields include all the conditions of the Where clause in that order, the driver can use the entire index. If, however, your index is on two nonconsecutive fields, say, LAST\_NAME and FIRST\_NAME, the driver can use only the LAST\_NAME field of the index.

The driver uses only one index when processing Where clauses. If you have complex Where clauses that involve a number of conditions for different fields and have indexes on more than one field, the driver chooses an index to use. The driver attempts to use indexes on conditions that use the equal sign as the relational operator rather than conditions using other operators (such as greater than). Assume you have an index on the EMP\_ID field as well as the LAST\_NAME field and the following Where clause:

```
WHERE emp_id >= 'E10001' AND last_name = 'Smith'
```

In this case, the driver selects the index on the LAST\_NAME field.

If no conditions have the equal sign, the driver first attempts to use an index on a condition that has a lower *and* upper bound, and then attempts to use an index on a condition that has a lower *or* upper bound. The driver always attempts to use the most restrictive index that satisfies the Where clause.

In most cases, the driver does not use an index if the *Where* clause contains an *OR* comparison operator. For example, the driver does not use an index for the following *Where* clause:

```
WHERE emp_id >= 'E10001' OR last_name = 'Smith'
```

---

## Deciding Which Indexes to Create

Before you create indexes for a database table, consider how you will use the table. The two most common operations on a table are to

- Insert, update, and delete records
- Retrieve records

If you most often insert, update, and delete records, then the fewer indexes associated with the table, the better the performance. This is because the driver must maintain the indexes as well as the database tables, thus slowing down the performance of record inserts, updates, and deletes. It may be more efficient to drop all indexes before modifying a large number of records, and re-create the indexes after the modifications.

If you most often retrieve records, you must look further to define the criteria for retrieving records and create indexes to improve the performance of these retrievals. Assume you have an employee database table and you will retrieve records based on employee name, department, or hire date. You would create three indexes—one on the *DEPT* field, one on the *HIRE\_DATE* field, and

one on the LAST\_NAME field. Or perhaps, for the retrievals based on the name field, you would want an index that concatenates the LAST\_NAME and the FIRST\_NAME fields (see [“Indexing Multiple Fields” on page 317](#)).

Here are a few rules to help you decide which indexes to create.

- If your record retrievals are based on one field at a time (for example, dept='D101'), create an index on these fields.
- If your record retrievals are based on a combination of fields, look at the combinations.
- If the comparison operator for the conditions is AND (for example, CITY = 'Raleigh' AND STATE = 'NC'), then build a concatenated index on the CITY and STATE fields. This index is also useful for retrieving records based on the CITY field.
- If the comparison operator is OR (for example, DEPT = 'D101' OR HIRE\_DATE > {01/30/89}), an index does not help performance. Therefore, you need not create one.
- If the retrieval conditions contain both AND and OR comparison operators, you can use an index if the OR conditions are grouped. For example:

```
dept = 'D101' AND (hire_date > {01/30/89} OR
exempt = 1)
```

In this case, an index on the DEPT field improves performance.

- If the AND conditions are grouped, an index does not improve performance. For example:

```
(dept = 'D101' AND hire_date) > {01/30/89} OR
exempt = 1
```



---

# Improving Join Performance

When joining database tables, index tables can greatly improve performance. Unless the proper indexes are available, queries that use joins can take a long time.

Assume you have the following Select statement:

```
SELECT * FROM dept, emp WHERE dept.dept_id = emp.dept
```

In this example, the DEPT and EMP database tables are being joined using the department ID field. When the driver executes a query that contains a join, it processes the tables from left to right and uses an index on the second table's join field (the DEPT field of the EMP table).

To improve join performance, you need an index on the join field of the second table in the From clause. If there is a third table in the From clause, the driver also uses an index on the field in the third table that joins it to any previous table. For example:

```
SELECT * FROM dept, emp, addr  
WHERE dept.dept_id = emp.dept AND emp.loc = addr.loc
```

In this case, you should have an index on the EMP.DEPT field and the ADDR.LOC field.



# C ODBC API and Scalar Functions

This appendix lists the ODBC API functions that the DataDirect ODBC drivers support and the scalar functions, which you use in SQL statements.

---

## API Functions

All database drivers are ODBC Level 1–compliant—they support all ODBC Core and Level 1 functions. A limited set of Level 2 functions is also supported. The drivers support the functions listed in [Table C-1 and C-2](#). Any additions to these supported functions or differences in the support of specific functions are listed in the “ODBC Conformance Level” section in the individual driver chapters.

---

**Table C-1. Supported 2.x ODBC API Functions**


---

<b>Core Functions</b>	<b>Level 1 Functions</b>
SQLAllocConnect	SQLColumns
SQLAllocEnv	SQLDriverConnect
SQLAllocStmt	SQLGetConnectOption
SQLBindCol	SQLGetData
SQLBindParameter	SQLGetFunctions
SQLCancel	SQLGetInfo
SQLColAttributes	SQLGetStmtOption
SQLConnect	SQLGetTypeInfo
SQLDescribeCol	SQLParamData
SQLDisconnect	SQLPutData
SQLDrivers	SQLSetConnectOption
SQLError	SQLSetStmtOption
SQLExecDirect	SQLSpecialColumns
SQLExecute	SQLStatistics
SQLFetch	SQLTables
SQLFreeConnect	<b>Level 2 Functions</b>
SQLFreeEnv	SQLBrowseConnect (all drivers except PROGRESS)
SQLFreeStmt	SQLDataSources
SQLGetCursorName	SQLExtendedFetch (forward scrolling only)
SQLNumResultCols	SQLMoreResults
SQLPrepare	SQLNativeSql
SQLRowCount	SQLNumParams
SQLSetCursorName	SQLParamOptions
SQLTransact	SQLSetScrollOptions

---

**Table C-2. Supported 3.x ODBC API Functions**


---

SQLAllocHandle	SQLGetData
SQLBindCol	SQLGetDescField
SQLBindParameter	SQLGetDescRec
SQLBrowseConnect (except for PROGRESS)	SQLGetDiagField
SQLBulkOperations	SQLGetDiagRec
SQLCancel	SQLGetEnvAttr
SQLCloseCursor	SQLGetFunctions
SQLColAttribute	SQLGetInfo
SQLColumns	SQLGetStmtAttr
SQLConnect	SQLGetTypeInfo
SQLCopyDesc	SQLMoreResults
SQLDataSources	SQLNativeSql
SQLDescribeCol	SQLNumParens
SQLDisconnect	SQLNumResultCols
SQLDriverConnect	SQLParamData
SQLDrivers	SQLPrepare
SQLEndTran	SQLPutData
SQLError	SQLRowCount
SQLExecDirect	SQLSetConnectAttr
SQLExecute	SQLSetCursorName
SQLExtendedFetch	SQLSetDescField
SQLFetch	SQLSetDescRec
SQLFetchScroll (forward scrolling only)	SQLSetEnvAttr
SQLFreeHandle	SQLSetStmtAttr
SQLFreeStmt	SQLSpecialColumns
SQLGetConnectAttr	SQLStatistics
SQLGetCursorName	SQLTables
	SQLTransact

---

---

## Scalar Functions

The following tables list the scalar functions that ODBC supports; your database system may not support all of these functions. See the documentation for your database system to find out which functions are supported.

You can use these functions in SQL statements using the following syntax:

```
{fn scalar-function}
```

where *scalar-function* is one of the functions listed in the following tables. For example,

```
SELECT {fn UCASE(NAME)} FROM EMP
```

## String Functions

[Table C-3](#) lists the string functions that ODBC supports.

The string functions listed can take the following arguments:

- *string\_exp* can be the name of a column, a string literal, or the result of another scalar function, where the underlying data type is SQL\_CHAR, SQL\_VARCHAR, or SQL\_LONGVARCHAR.
- *start*, *length*, and *count* can be the result of another scalar function or a literal numeric value, where the underlying data type is SQL\_TINYINT, SQL\_SMALLINT, or SQL\_INTEGER.

The string functions are one-based; that is, the first character in the string is character 1.

Character string literals must be surrounded in single quotation marks.

---

**Table C-3. Scalar String Functions**


---

<b>Function</b>	<b>Returns</b>
ASCII( <i>string_exp</i> )	ASCII code value of the leftmost character of <i>string_exp</i> as an integer.
BIT_LENGTH( <i>string_exp</i> ) ODBC 3.0	The length in bits of the string expression.
CHAR( <i>code</i> )	The character with the ASCII code value specified by <i>code</i> . <i>code</i> should be between 0 and 255; otherwise, the return value is data-source dependent.
CHAR_LENGTH( <i>string_exp</i> ) ODBC 3.0	The length in characters of the string expression, if the string expression is of a character data type; otherwise, the length in bytes of the string expression (the smallest integer not less than the number of bits divided by 8). (This function is the same as the CHARACTER_LENGTH function.)
CHARACTER_LENGTH( <i>string_exp</i> ) ODBC 3.0	The length in characters of the string expression, if the string expression is of a character data type; otherwise, the length in bytes of the string expression (the smallest integer not less than the number of bits divided by 8). (This function is the same as the CHAR_LENGTH function.)
CONCAT( <i>string_exp1</i> , <i>string_exp2</i> )	The string resulting from concatenating <i>string_exp2</i> and <i>string_exp1</i> . The string is system dependent.
DIFFERENCE( <i>string_exp1</i> , <i>string_exp2</i> )	An integer value that indicates the difference between the values returned by the SOUNDEX function for <i>string_exp1</i> and <i>string_exp2</i> .
INSERT( <i>string_exp1</i> , <i>start</i> , <i>length</i> , <i>string_exp2</i> )	A string where <i>length</i> characters have been deleted from <i>string_exp1</i> beginning at <i>start</i> and where <i>string_exp2</i> has been inserted into <i>string_exp1</i> , beginning at <i>start</i> .
LCASE( <i>string_exp</i> )	Uppercase characters in <i>string_exp</i> converted to lowercase.
LEFT( <i>string_exp</i> , <i>count</i> )	The <i>count</i> of characters of <i>string_exp</i> .
LENGTH( <i>string_exp</i> )	The number of characters in <i>string_exp</i> , excluding trailing blanks and the string termination character.

**Table C-3. Scalar String Functions** (cont.)

Function	Returns
LOCATE( <i>string_exp1</i> , <i>string_exp2</i> [, <i>start</i> ])	The starting position of the first occurrence of <i>string_exp1</i> within <i>string_exp2</i> . If <i>start</i> is not specified the search begins with the first character position in <i>string_exp2</i> . If <i>start</i> is specified, the search begins with the character position indicated by the value of <i>start</i> . The first character position in <i>string_exp2</i> is indicated by the value 1. If <i>string_exp1</i> is not found, 0 is returned.
LTRIM( <i>string_exp</i> )	The characters of <i>string_exp</i> , with leading blanks removed.
OCTET_LENGTH( <i>string_exp</i> ) ODBC 3.0	The length in bytes of the string expression. The result is the smallest integer not less than the number of bits divided by 8.
POSITION( <i>character_exp</i> IN <i>character_exp</i> )ODBC 3.0	The position of the first character expression in the second character expression. The result is an exact numeric with an implementation-defined precision and a scale of 0.
REPEAT( <i>string_exp</i> , <i>count</i> )	A string composed of <i>string_exp</i> repeated <i>count</i> times.
REPLACE( <i>string_exp1</i> , <i>string_exp2</i> , <i>string_exp3</i> )	Replaces all occurrences of <i>string_exp2</i> in <i>string_exp1</i> with <i>string_exp3</i> .
RIGHT( <i>string_exp</i> , <i>count</i> )	The rightmost <i>count</i> of characters in <i>string_exp</i> .
RTRIM( <i>string_exp</i> )	The characters of <i>string_exp</i> with trailing blanks removed.
SOUNDEX( <i>string_exp</i> )	A data-source-dependent string representing the sound of the words in <i>string_exp</i> .
SPACE( <i>count</i> )	A string consisting of <i>count</i> spaces.
SUBSTRING( <i>string_exp</i> , <i>start</i> , <i>length</i> )	A string derived from <i>string_exp</i> beginning at the character position <i>start</i> for <i>length</i> characters.
UCASE( <i>string_exp</i> )	Lowercase characters in <i>string_exp</i> converted to uppercase.



## Numeric Functions

Table C-4 lists the numeric functions that ODBC supports.

The numeric functions listed can take the following arguments:

- *numeric\_exp* can be a column name, a numeric literal, or the result of another scalar function, where the underlying data type is SQL\_NUMERIC, SQL\_DECIMAL, SQL\_TINYINT, SQL\_SMALLINT, SQL\_INTEGER, SQL\_BIGINT, SQL\_FLOAT, SQL\_REAL, or SQL\_DOUBLE.
- *float\_exp* can be a column name, a numeric literal, or the result of another scalar function, where the underlying data type is SQL\_FLOAT.
- *integer\_exp* can be a column name, a numeric literal, or the result of another scalar function, where the underlying data type is SQL\_TINYINT, SQL\_SMALLINT, SQL\_INTEGER, or SQL\_BIGINT.

---

**Table C-4. Scalar Numeric Functions**

---

Function	Returns
ABS( <i>numeric_exp</i> )	Absolute value of <i>numeric_exp</i> .
ACOS( <i>float_exp</i> )	Arccosine of <i>float_exp</i> as an angle in radians.
ASIN( <i>float_exp</i> )	Arcsine of <i>float_exp</i> as an angle in radians.
ATAN( <i>float_exp</i> )	Arctangent of <i>float_exp</i> as an angle in radians.
ATAN2( <i>float_exp1</i> , <i>float_exp2</i> )	Arctangent of the x and y coordinates, specified by <i>float_exp1</i> and <i>float_exp2</i> as an angle in radians.
CEILING( <i>numeric_exp</i> )	Smallest integer greater than or equal to <i>numeric_exp</i> .
COS( <i>float_exp</i> )	Cosine of <i>float_exp</i> as an angle in radians.
COT( <i>float_exp</i> )	Cotangent of <i>float_exp</i> as an angle in radians.
DEGREES( <i>numeric_exp</i> )	Number if degrees converted from <i>numeric_exp</i> radians.
EXP( <i>float_exp</i> )	Exponential value of <i>float_exp</i> .

**Table C-4. Scalar Numeric Functions** (cont.)

Function	Returns
FLOOR( <i>numeric_exp</i> )	Largest integer less than or equal to <i>numeric_exp</i> .
LOG( <i>float_exp</i> )	Natural log of <i>float_exp</i> .
LOG10( <i>float_exp</i> )	Base 10 log of <i>float_exp</i> .
MOD( <i>integer_exp1</i> , <i>integer_exp2</i> )	Remainder of <i>integer_exp1</i> divided by <i>integer_exp2</i> .
PI()	Constant value of pi as a floating-point number.
POWER( <i>numeric_exp</i> , <i>integer_exp</i> )	Value of <i>numeric_exp</i> to the power of <i>integer_exp</i> .
RADIANS( <i>numeric_exp</i> )	Number of radians converted from <i>numeric_exp</i> degrees.
RAND([ <i>integer_exp</i> ])	Random floating-point value using <i>integer_exp</i> as the optional seed value.
ROUND( <i>numeric_exp</i> , <i>integer_exp</i> )	<i>numeric_exp</i> rounded to <i>integer_exp</i> places right of the decimal (left of the decimal if <i>integer_exp</i> is negative).
SIGN( <i>numeric_exp</i> )	Indicator of the sign of <i>numeric_exp</i> . If <i>numeric_exp</i> < 0, -1 is returned. If <i>numeric_exp</i> = 0, 0 is returned. If <i>numeric_exp</i> > 0, 1 is returned.
SIN( <i>float_exp</i> )	Sine of <i>float_exp</i> , where <i>float_exp</i> is an angle in radians.
SQRT( <i>float_exp</i> )	Square root of <i>float_exp</i> .
TAN( <i>float_exp</i> )	Tangent of <i>float_exp</i> , where <i>float_exp</i> is an angle in radians.
TRUNCATE( <i>numeric_exp</i> , <i>integer_exp</i> )	<i>numeric_exp</i> truncated to <i>integer_exp</i> places right of the decimal. (If <i>integer_exp</i> is negative, truncation is to the left of the decimal.)

## Date and Time Functions

Table C-5 lists the date and time functions that ODBC supports.

The date and time functions listed can take the following arguments:

- *date\_exp* can be a column name, a date or timestamp literal, or the result of another scalar function, where the underlying data type can be represented as SQL\_CHAR, SQL\_VARCHAR, SQL\_DATE, or SQL\_TIMESTAMP.
- *time\_exp* can be a column name, a timestamp or timestamp literal, or the result of another scalar function, where the underlying data type can be represented as SQL\_CHAR, SQL\_VARCHAR, SQL\_TIME, or SQL\_TIMESTAMP.
- *timestamp\_exp* can be a column name; a time, date, or timestamp literal; or the result of another scalar function, where the underlying data type can be represented as SQL\_CHAR, SQL\_VARCHAR, SQL\_TIME, SQL\_DATE, or SQL\_TIMESTAMP.

---

**Table C-5. Scalar Time and Date Functions**

---

Function	Returns
CURRENT_DATE() ODBC 3.0	Current date.
CURRENT_TIME[( <i>time-precision</i> )] ODBC 3.0	Current local time. The <i>time-precision</i> argument determines the seconds precision of the returned value.
CURRENT_TIMESTAMP[( <i>timestamp-precision</i> )] ODBC 3.0	Current local date and local time as a timestamp value. The <i>timestamp-precision</i> argument determines the seconds precision of the returned timestamp.
CURDATE()	Current date as a date value.
CURTIME()	Current local time as a time value.

**Table C-5. Scalar Time and Date Functions** (cont.)

Function	Returns
DAYNAME( <i>date_exp</i> )	Character string containing a data-source-specific name of the day for the day portion of <i>date_exp</i> .
DAYOFMONTH( <i>date_exp</i> )	Day of the month in <i>date_exp</i> as an integer value (1–31).
DAYOFWEEK( <i>date_exp</i> )	Day of the week in <i>date_exp</i> as an integer value (1–7).
DAYOFYEAR( <i>date_exp</i> )	Day of the year in <i>date_exp</i> as an integer value (1–366).
HOUR( <i>time_exp</i> )	Hour in <i>time_exp</i> as an integer value (0–23).
MINUTE( <i>time_exp</i> )	Minute in <i>time_exp</i> as an integer value (0–59).
MONTH( <i>date_exp</i> )	Month in <i>date_exp</i> as an integer value (1–12).
MONTHNAME( <i>date_exp</i> )	Character string containing the data source-specific name of the month.
NOW()	Current date and time as a timestamp value.
QUARTER( <i>date_exp</i> )	Quarter in <i>date_exp</i> as an integer value (1–4).
SECOND( <i>time_exp</i> )	Second in <i>date_exp</i> as an integer value (0–59).
TIMESTAMPADD( <i>interval</i> , <i>integer_exp</i> , <i>time_exp</i> )	Timestamp calculated by adding <i>integer_exp</i> intervals of type <i>interval</i> to <i>time_exp</i> . <i>interval</i> can be SQL_TSI_FRAC_SECOND SQL_TSI_SECOND SQL_TSI_MINUTE SQL_TSI_HOUR SQL_TSI_DAY SQL_TSI_WEEK SQL_TSI_MONTH SQL_TSI_QUARTER SQL_TSI_YEAR  Fractional seconds are expressed in billionths of a second.

**Table C-5. Scalar Time and Date Functions** (cont.)

Function	Returns
TIMESTAMPDIFF( <i>interval</i> , <i>time_exp1</i> , <i>time_exp2</i> )	Integer number of intervals of type <i>interval</i> by which <i>time_exp2</i> is greater than <i>time_exp1</i> . <i>interval</i> has the same values as TIMESTAMPADD. Fractional seconds are expressed in billionths of a second.
WEEK( <i>date_exp</i> )	Week of the year in <i>date_exp</i> as an integer value (1–53).
YEAR( <i>date_exp</i> )	Year in <i>date_exp</i> . The range is data-source dependent.

## System Functions

[Table C-6](#) lists the system functions that ODBC supports.

**Table C-6. Scalar System Functions**

Function	Returns
DATABASE()	Name of the database, corresponding to the connection handle ( <i>hdbc</i> ).
IFNULL( <i>exp</i> , <i>value</i> )	<i>value</i> , if <i>exp</i> is null.
USER()	Authorization name of the user.



# D Locking and Isolation Levels

This appendix discusses locking and isolation levels and how their settings can affect the data you retrieve. Different database systems support different locking and isolation levels. See the section “Isolation and Lock Levels Supported” in the appropriate driver chapter.

---

## Locking

Locking is a database operation that restricts a user from accessing a table or record. Locking is used in situations where more than one user might try to use the same table or record at the same time. By locking the table or record, the system ensures that only one user at a time can affect the data.

Locking is critical in multiuser databases, where different users can try to access or modify the same records concurrently. Although such concurrent database activity is desirable, it can create problems. Without locking, for example, if two users try to modify the same record at the same time, they might encounter problems ranging from retrieving bad data to deleting data that the other user needs. If, however, the first user to access a record can lock that record to temporarily prevent other users from modifying it, such problems can be avoided. Locking provides a way to manage concurrent database access while minimizing the various problems it can cause.

---

## Isolation Levels

An isolation level represents a particular locking strategy employed in the database system to improve data consistency. The higher the isolation level, the more complex the locking strategy behind it. The isolation level provided by the database determines whether a transaction will encounter the following behaviors in data consistency:

Dirty reads	User 1 modifies a row. User 2 reads the same row before User 1 commits. User 1 performs a rollback. User 2 has read a row that has never really existed in the database. User 2 may base decisions on false data.
Non-repeatable reads	User 1 reads a row but does not commit. User 2 modifies or deletes the same row and then commits. User 1 rereads the row and finds it has changed (or has been deleted).
Phantom reads	User 1 uses a search condition to read a set of rows but does not commit. User 2 inserts one or more rows that satisfy this search condition, then commits. User 1 rereads the rows using the search condition and discovers rows that were not present before.

Isolation levels represent the database system's ability to prevent these behaviors. The American National Standards Institute (ANSI) defines four isolation levels:

- Read uncommitted (0)
- Read committed (1)
- Repeatable read (2)
- Serializable (3)



In ascending order (0–3), these isolation levels provide an increasing amount of data consistency to the transaction. At the lowest level, all three behaviors can occur. At the highest level, none can occur. The success of each level in preventing these behaviors is due to the locking strategies that they employ, which are as follows:

- |                      |   |
|----------------------|---|
| Read uncommitted (0) | Locks are obtained on modifications to the database and held until end of transaction (EOT). Reading from the database does not involve any locking.  |
| Read committed (1)   | Locks are acquired for reading and modifying the database. Locks are released after reading but locks on modified objects are held until EOT.   |
| Repeatable read (2)  | Locks are obtained for reading and modifying the database. Locks on all modified objects are held until EOT. Locks obtained for reading data are held until EOT. Locks on non-modified access structures (such as indexes and hashing structures) are released after reading. |
| Serializable (3)     | All data read or modified is locked until EOT. All access structures that are modified are locked until EOT. Access structures used by the query are locked until EOT.  |

Table D-1 shows what data consistency behaviors can occur at each isolation level.

---

**Table D-1. Isolation Levels and Data Consistency**

---

Level	Dirty Read	Nonrepeatable Read	Phantom Read
0, Read uncommitted	Yes	Yes	Yes
1, Read committed	No	Yes	Yes
2, Repeatable read	No	No	Yes
3, Serializable	No	No	No

---

Although higher isolation levels provide better data consistency, this consistency can be costly in terms of the concurrency provided to individual users. Concurrency is the ability of multiple users to access and modify data simultaneously. As isolation levels increase, so does the chance that the locking strategy used will create problems in concurrency.

*Put another way:* The higher the isolation level, the more locking involved, and the more time users may spend waiting for data to be freed by another user. Because of this inverse relationship between isolation levels and concurrency, you must consider how people use the database before choosing an isolation level. You must weigh the trade-offs between data consistency and concurrency, and decide which is more important.

---

## Locking Modes and Levels

Different database systems employ various locking modes, but they have two basic ones in common: shared and exclusive. Shared locks can be held on a single object by multiple users. If one user has a shared lock on a record, then a second user can also get a shared lock on that same record; however, the second user cannot get an exclusive lock on that record. Exclusive locks are exclusive to the user that obtains them. If one user has an exclusive lock on a record, then a second user cannot get either type of lock on the same record.

Performance and concurrency can also be affected by the locking level used in the database system. The locking level determines the size of an object that is locked in a database. For example, many database systems let you lock an entire table, as well as individual records. An intermediate level of locking, page-level locking, is also common. A page contains one or more records and is typically the amount of data read from the disk in a single disk access. The major disadvantage of page-level locking is that if one user locks a record, a second user may not be able to lock other records because they are stored on the same page as the locked record.



# E Designing Performance-Oriented ODBC Applications

This appendix provides information about performance issues and guidelines for developing performance-optimized, ODBC applications for ODBC/OLE DB Adapter and ODBC drivers.

---

## Optimizing Performance

Developing performance-oriented ODBC applications is not easy. Microsoft's *ODBC Programmer's Reference* does not provide information about system performance. In addition, ODBC drivers and the ODBC driver manager do not return warnings when applications run inefficiently.

The following sections contain guidelines compiled by examining the ODBC implementations of numerous shipping ODBC applications.

[Table E-1](#) summarizes some common ODBC system performance problems and possible solutions:

---

**Table E-1. Common ODBC System Performance Problems and Solutions**

---

<b>Problem</b>	<b>Solution</b>
Network communication is slow	Reduce network traffic
The process of evaluating complex SQL queries on the database server is slow and might reduce concurrency	Simplify queries

---

**Table E-1. Common ODBC System Performance Problems and Solutions** (cont.)

---

<b>Problem</b>	<b>Solution</b>
Excessive calls from the application to the driver decreases performance	Optimize application-to-driver interaction
Disk input/output is slow	Limit disk input/output

---

The guidelines are divided into five sections: “[Catalog Functions](#),” “[Retrieving Data](#),” “[ODBC Function Selection](#),” “[Design Options](#),” and “[Updating Data](#).”

---

## Catalog Functions

The following ODBC functions are defined to be catalog functions:

- SQLColumns
- SQLColumnPrivileges
- SQLForeignKeys
- SQLGetTypeInfo
- SQLProcedures
- SQLProcedureColumns
- SQLSpecialColumns
- SQLStatistics
- SQLTables
- SQLTablePrivileges

While some drivers implement `SQLGetTypeInfo` as hard-coded information, many drivers must query the server to obtain accurate information about which types are supported (for example, to find dynamic types such as user defined types, and so on). Therefore, `SQLGetTypeInfo` is included in this list of potentially expensive ODBC functions.

## Catalog Functions Are Relatively Slow

Catalog functions are relatively slow compared to other ODBC functions. Applications should cache information returned from catalog functions so that multiple executions are not needed.

While almost no ODBC application can be written without catalog functions, their use should be minimized. To return all result column information *mandated* by the ODBC specification, a driver may have to perform multiple queries, joins, subqueries, and/or unions in order to return the necessary result set for a single call to a catalog function. These particular elements of the SQL language are performance expenses. Frequent use of catalog functions in an application will likely result in poor performance.

Applications should cache information from catalog functions. For example, call `SQLGetTypeInfo` once in the application and cache away the elements of the result set that your application depends on. It is unlikely that any application uses all elements of the result set generated by a catalog function, so the cache of information should not be difficult to maintain.

## Passing Null Arguments

Passing null arguments to catalog functions results in generating time consuming queries. In addition, network traffic potentially increases due to unwanted result set information. Always supply as many non-null arguments to catalog functions as possible.

Because catalog functions are slow, applications should invoke them as efficiently as possible. Many applications pass the fewest non-null arguments necessary for the function to return success.

For example, consider a call to `SQLTables` where the application requests information about table "Customers." Often, this call is coded in a manner similar to the following example.

```
rc = SQLTables (NULL, NULL, NULL, NULL, "Customers", SQL_NTS,  
              NULL);
```

A driver could turn this `SQLTables` call into SQL similar to:

```
SELECT ... FROM SysTables WHERE TableName = 'Customers' UNION ALL  
SELECT ... FROM SysViews WHERE ViewName = 'Customers' UNION ALL  
SELECT ... FROM SysSynonyms WHERE SynName = 'Customers'  
ORDER BY ...
```

Sometimes, little information is known about the object for which you are requesting information. Any information that the application can send the driver when calling catalog functions can result in improved performance and reliability.

Using the previous example, suppose three "Customers" tables were returned in the result set:

- The first table was owned by the user
- The second table was owned by the sales department
- The third table was a view created by management

It might not be obvious to the user which table to choose. If the application had specified the `OwnerName` argument for the `SQLTables` call, only one table would be returned and performance would increase. This is because less network traffic was required to return only one result row and unwanted rows were filtered by the database.



In addition, if the `TableType` argument can be supplied, then the SQL sent to the server can be optimized from a three query union to a single, `Select` statement as shown:

```
SELECT ... FROM SysTables WHERE TableName =
'Customers' and
    Owner = 'Beth'
```

## SQLColumns

Avoid using `SQLColumns` to determine characteristics about a table. Instead, use a dummy query with `SQLDescribeCol`.

Consider an application that allows the user to choose the columns that will be selected. Should the application use `SQLColumns` to return information about the columns to the user or instead prepare a dummy query and call `SQLDescribeCol`?

### ***Case 1: SQLColumns Method***

```
rc = SQLColumns (... "UnknownTable" ...);
// This call to SQLColumns will generate a query to the system
// catalogs... possibly a join which must be prepared,
// executed, and produce a result set
rc = SQLBindCol (...);
rc = SQLExtendedFetch (...);
// user must retrieve N rows from the server
// N = # result columns of UnknownTable
// result column information has now been obtained
```

## Case 2: *SQLDescribeCol Method*

```
// prepare dummy query
rc = SQLPrepare (... "SELECT * from UnknownTable
    WHERE 1 = 0" ...);
// query is never executed on the server - only prepared
rc = SQLNumResultCols (...);
for (irow = 1; irow <= NumColumns; irow++) {
    rc = SQLDescribeCol (...)
    // + optional calls to SQLColAttributes
}
// result column information has now been obtained
// Note we also know the column ordering within the table!
// This information cannot be
// assumed from the SQLColumns example.
```

In both cases a query is sent to the server, but in Case 1 the query must be evaluated and form a result set that must be sent to the client. Clearly, Case 2 is the better performing model.

To somewhat complicate this discussion, let us consider a DBMS server that does not natively support preparing a SQL statement. The performance of Case 1 does not change but Case 2 increases minutely because the dummy query must be evaluated instead of only prepared. Because the Where clause of the query always evaluates to FALSE, the query generates no result rows and should execute without accessing table data. For even this type of driver, method 2 out performs method 1.

---

## Retrieving Data

This section provides information about retrieving data with ODBC applications.

### Retrieving Long Data

Retrieving long data (SQL\_LONGVARCHAR and SQL\_LONGVARBINARY data) across the network is very resource intensive and thus slow. Applications should avoid requesting long data unless it is absolutely necessary.

How often do users want to see long data? Most users don't want to see such information. If the user does wish to see these result items, then the application can requery the database specifying only the long columns in the select list. This method allows the average user to retrieve the result set without having to pay a high performance penalty for network traffic.

Although the best method is to exclude long data from the select list, some applications do not formulate the select list before sending the query to the ODBC driver (that is, some applications simply `select * from <table name> . . .`). If the select list contains long data then some drivers *must* retrieve that data at fetch time even if the application does not bind the long data in the result set. If possible, the designer should attempt to implement a method that does not retrieve all columns of the table.

### Reducing the Size of Data Retrieved

Reduce the size of any data being retrieved to some manageable limit by calling `SQLSetStmtOption` with the `SQL_MAX_LENGTH` option. This reduces network traffic and improves performance.

While eliminating SQL\_LONGVARCHAR and SQL\_LONGVARBINARY data from the result set is ideal for performance optimization, sometimes, long data must be retrieved. When this is the case, remember that most users do not want to see 100 KB, or more, of text on the screen. What techniques, if any, are available to limit the amount of data retrieved?

Many application developers mistakenly assume that if they call SQLGetData with a container of size  $x$  that the ODBC driver only retrieves  $x$  bytes of information from the server. Because SQLGetData can be called multiple times for any one column, most drivers optimize their network use by retrieving long data in large chunks and then returning it to the user when requested. For example:

```
char CaseContainer[1000];
...
rc = SQLExecDirect (hstmt, "SELECT CaseHistory
FROM Cases WHERE
    CaseNo = 71164", SQL_NTS);
...
rc = SQLFetch (hstmt);
rc = SQLGetData (hstmt, 1, CaseContainer, (SQLWORD)
sizeof(CaseContainer), ...);
```

At this point, it is more likely that an ODBC driver retrieves 64 KB of information from the server instead of 1000 bytes. One 64 KB retrieval is less expensive than sixty-four, 1000-byte retrievals in terms of network access. Unfortunately, the application may not call SQLGetData again; thus, the first and only retrieval of CaseHistory would be slowed by the fact that 64 KB of data had to be sent across the network.

Many ODBC drivers allow limiting the amount of data retrieved across the network by supporting the statement option SQL\_MAX\_LENGTH. This attribute allows the driver to communicate to the database server that only  $Z$  bytes of data are pertinent to the client. The server responds by sending only the

first Z bytes of data for *all* result columns. This optimization greatly reduces network traffic and thus improves performance of the client. Our example returned just one row, but consider the case where 100 rows are returned in the result set. The performance improvement is substantial.

## Using Bound Columns

Retrieving data through bound columns (SQLBindCol) instead of using SQLGetData reduces the ODBC call load and thus improves performance.

Consider the following pseudo-code fragment:

```
rc = SQLExecDirect (hstmt, "SELECT <20 columns>
FROM Employees
    WHERE HireDate >= ?", SQL_NTS);
do {
rc = SQLFetch (hstmt);
// call SQLGetData 20 times
} while ((rc == SQL_SUCCESS) || (rc ==
SQL_SUCCESS_WITH_INFO));
```

Suppose the query returns 90 result rows. More than 1890 ODBC calls are made (20 calls to SQLGetData x 90 result rows + 91 calls to SQLFetch).

Consider the same scenario that uses SQLBindCol instead of SQLGetData:

```
rc = SQLExecDirect (hstmt, "SELECT <20 columns>
FROM Employees
    WHERE HireDate >= ?", SQL_NTS);
// call SQLBindCol 20 times
do {
rc = SQLFetch (hstmt);
} while ((rc == SQL_SUCCESS) || (rc ==
SQL_SUCCESS_WITH_INFO));
```

The number of ODBC calls made is reduced from more than 1890 to about 110 (20 calls to SQLBindCol + 91 calls to SQLFetch). In addition to reducing the call load many drivers optimize use of SQLBindCol by binding result information directly from the database server into the user's buffer. That is, instead of the driver retrieving information into a container then copying that information to the user's buffer, the driver simply requests the information from the server be placed directly into the user's buffer.

## Using SQLExtendedFetch instead of SQLFetch

Use SQLExtendedFetch to retrieve data instead of SQLFetch. The ODBC call load decreases (resulting in better performance,) and the code is less complex (resulting in more maintainable code).

Most ODBC drivers now support SQLExtendedFetch for forward only cursors; yet, most ODBC applications use SQLFetch to retrieve data. Again consider the example above using SQLExtendedFetch instead of SQLFetch:

```
rc = SQLSetStmtOption (hstmt, SQL_ROWSET_SIZE, 100);
// use arrays of 100 elements
rc = SQLExecDirect (hstmt, "SELECT <20 columns> FROM
  Employees WHERE HireDate >= ?", SQL_NTS);
// call SQLBindCol 1 time specifying row-wise binding
do {
rc = SQLExtendedFetch (hstmt, SQL_FETCH_NEXT, 0, &RowsFetched,
  RowStatus);
} while ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO));
```

The number of ODBC calls made by the application is reduced from 110 in the last example to four (1 SQLSetStmtOption + 1 SQLExecDirect + 1 SQLBindCol + 1 SQLExtendedFetch). Note the total savings from an initial call load of more than 1890 ODBC calls in the first presentation of the example to four above. In addition to reducing the call load, many ODBC drivers retrieve

data from the server in arrays that further improves the performance by reducing network traffic.

For ODBC drivers that do not support `SQLExtendedFetch`, the application can enable forward-only cursors using the ODBC cursor library (call `SQLSetConnectOption` using `SQL_ODBC_CURSORS/SQL_CUR_USE_IF_NEEDED`). While using the cursor library does not improve performance, it should not be detrimental to application response time when using forward only cursors (no logging is required). Furthermore, using the cursor library when `SQLExtendedFetch` is not supported natively by the driver simplifies the code because the application can always depend on `SQLExtendedFetch` being available. The application need not code two algorithms (one using `SQLExtendedFetch` and one using `SQLFetch`).

---

## ODBC Function Selection

This section provides guidelines for selecting functions for performance optimization.

### Using `SQLPrepare/SQLExecute` and `SQLExecDirect`

Do not assume that `SQLPrepare/SQLExecute` is always as efficient as `SQLExecDirect`. Use `SQLExecDirect` for queries that will be executed once and `SQLPrepare/SQLExecute` for queries that will be executed more than once.

ODBC drivers are optimized based on the perceived use of the functions that are being executed. `SQLPrepare/SQLExecute` is optimized for multiple executions of a statement that most likely uses parameter markers. `SQLExecDirect` is optimized for a single execution of a SQL statement. Unfortunately, more than

seventy-five percent of all ODBC applications use SQLPrepare/SQLExecute *exclusively*.

The pitfall of always coding SQLPrepare/SQLExecute can be understood better by considering an ODBC driver that implements SQLPrepare by creating a stored procedure on the server that contains the prepared statement. Creating a stored procedure has substantial overhead, but the ODBC driver is assuming is that the statement will be *executed* multiple times. While stored procedure creation is relatively expensive, execution is minimal because the query is parsed and optimization paths are stored at create procedure time. Using SQLPrepare/SQLExecute for a statement that will be executed only once with such an ODBC driver will result in unneeded overhead. Furthermore, applications that use SQLPrepare/SQLExecute for large single execution query batches will almost certainly exhibit poor performance when used with ODBC drivers as previously discussed.

Similar arguments can be used to show applications that always use SQLExecDirect cannot perform as well as those that logically use a combination of SQLPrepare/SQLExecute and SQLExecDirect sequences.

## Using SQLPrepare and Multiple SQLExecute Calls

Applications that use SQLPrepare and multiple SQLExecute calls should use SQLParamOptions if available. Passing arrays of parameter values reduces the ODBC call load and greatly reduces network traffic.



Consider the following example designed to insert data:

```
rc = SQLPrepare (hstmt, "INSERT INTO DailyLedger
(...) VALUES
    (?, ?, ...)", SQL_NTS);
// bind parameters
...
do {
// read ledger values into bound parameter buffers
...
rc = SQLExecute (hstmt);      // insert row
} while ! (eof);
```

If there are 100 rows to insert then `SQLExecute` is called 100 times resulting in 100 network requests to the server. Consider, however, an algorithm that uses parameter arrays by calling `SQLParamOptions`:

```
rc = SQLPrepare (hstmt, "INSERT INTO DailyLedger
(...) VALUES
    (?, ?, ...)", SQL_NTS);
rc = SQLParamOptions (hstmt, (UDWORD) 50,
&CurrentRow);
// pass 50 parameters per execute
// bind parameters
...
do {
// read up to 50 ledger values into bound
parameter buffers
...
rc = SQLExecute (hstmt);      // insert row
```

The call load has been reduced from 100 to just 2 `SQLExecute` calls; furthermore, network traffic is reduced considerably. Some ODBC drivers do not support `SQLParamOptions`. To achieve high performance, applications should contain algorithms for using `SQLParamOptions` if the ODBC driver supports the function. `SQLParamOptions` is ideal for copying data into new tables or bulk loading tables.

## Using the Cursor Library

Do not automatically use the cursor library if scrollable cursors are provided by the driver. The cursor library creates local temporary log files, which are expensive to generate and provide worse performance than using native scrollable cursors.

The cursor library adds support for static cursors, which simplifies the coding of applications that use scrollable cursors. However, the cursor library creates temporary log files on the user's local disk drive to accomplish the task. Disk I/O is typically one of the slowest operations on personal computers. While the benefits of the cursor library are great, applications should not automatically choose to use the cursor library if an ODBC driver supports scrollable cursors natively.

ODBC drivers that support scrollable cursors typically achieve high performance by requesting that the DBMS server produce a scrollable result set instead of emulating the capability by creating log files.

Many applications use

```
rc = SQLSetConnectOption (hdbc, SQL_ODBC_CURSORS,  
    SQL_CUR_USE_ODBC);
```

but should use

```
rc = SQLSetConnectOption (hdbc, SQL_ODBC_CURSORS,  
    SQL_CUR_USE_IF_NEEDED);
```

---

# Design Options

This section provides guidelines for designing ODBC applications.

## Managing Connections

Connection management is important to application performance. Designers should optimize applications by connecting once and using multiple statement handles instead of performing multiple connections. Most ODBC applications contain poorly designed elements for connection management. Connecting to a data source should be avoided after establishing an initial connection.

Some ODBC applications are designed to call informational gathering routines that have no record of already attached connection handles. For example, some applications establish a connection and then call a routine in a separate DLL or shared library that reattaches and gathers information about the driver.

Although gathering driver information at connect time is a good algorithm, it should not be minimized by connecting twice. At least one popular ODBC enabled application connects a second time to gather driver information but *never* disconnects the second connection. Applications that are designed as separate entities should pass the already connected HDBC pointer to the data collection routine instead of establishing a second connection.

Another poor practice is to connect and disconnect several times throughout your application to perform SQL statements. Connection handles can have multiple statement handles associated with them. Statement handles are defined to be memory storage for information about SQL statements. Why then do many applications allocate new connection handles to

perform SQL statements? Applications should use *statement handles* to manage multiple SQL statements.

Connection and statement handling should not be delayed until implementation. Spending time and thoughtfully handling connection management improves application performance and maintainability.

## Committing Data

Committing data is extremely disk I/O intensive and slow. Always turn autocommit off, if the driver can support transactions.

What is actually involved in a commit? The database server must flush back to disk every data page that contains updated or new data. Note that this is not a sequential write but a searched write to replace existing data in the table. By default, autocommit is on when connecting to a data source. Autocommit mode is typically detrimental to performance because of the extreme amount of disk i/o needed to commit every operation.

To further reduce performance some database servers do not provide an "autocommit mode." For this type of server, the ODBC driver must explicitly issue a COMMIT statement and perhaps a BEGIN TRANSACTION for every operation sent to the server. In addition to the large amount of disk I/O required to support autocommit mode, a performance penalty is paid for up to three network requests for every statement issued by an application.

## Asynchronous Execution

Design your application to take advantage of data sources that support asynchronous execution. Asynchronous calls do not perform faster but well designed applications *appear* to run more efficiently.

By default, an application makes calls to an ODBC driver that then executes statements against the DBMS server in a synchronous manner. In this mode of operation, the driver does not return control to the application until its own request to the server is complete. For statements which take more than a few seconds to complete execution, this can result in the perception of poor performance to the end user.

Some data sources support asynchronous execution. When in asynchronous mode, an application makes calls to an ODBC driver and control is returned almost immediately. In this mode the driver returns the status `SQL_STILL_EXECUTING` to the application and then sends the appropriate request to the database backend for execution. The application polls the driver at various intervals at which point the driver itself polls the server to see if the query has completed execution. If the query is still executing, the status `SQL_STILL_EXECUTING` is returned to the application. If it has completed, a status such as `SQL_SUCCESS` is returned and the application can then begin to fetch records.

Turning on asynchronous execution does not by itself improve performance. Well-designed applications, however, can take advantage of asynchronous query execution by allowing the end user to work on other things while the query is being evaluated on the server. Perhaps users will start one or more subsequent queries or choose to work in another application, all while the query is executing on the server. Designing for asynchronous execution makes your application *appear* to run faster by allowing the end user to work concurrently on multiple tasks.

## Updating Data

This section provides guidelines for updating data stored in databases with information supplied by the end user.

### Using Positional Updates and Deletes

Use positioned updates and deletes or `SQLSetPos` to update data.

Designing an efficient method for updating data is difficult. While positioned updates do not apply to all types of applications, developers should attempt to use positioned updates and deletes. Positioned updates (either through “update where current of cursor” or through `SQLSetPos`) allow the developer to update data simply by positioning the database cursor to the appropriate row to be changed and signal the driver to “change the data here.” The designer is not forced to build a complex SQL statement but is simply required to supply the data that is to be changed.

Besides making the application more easily maintainable, positioned updates typically result in improved performance. Because the database server is already positioned on the row (for the Select statement in process), performance-expensive operations to locate the row to be changed are not needed. If the row must be located, the server typically has an internal pointer to the row available (for example, ROWID).

### Using `SQLSpecialColumns`

Use `SQLSpecialColumns` to determine the optimal set of columns to use in the Where clause for updating data. Many times pseudo-columns provide the fastest access to the data, and these columns can only be determined by using `SQLSpecialColumns`.

Some applications cannot be designed to take advantage of positional updates and deletes. These applications typically update data by forming a *Where* clause consisting of some subset of the column values returned in the result set. Some applications might formulate the *Where* clause by using all searchable result columns or by calling `SQLStatistics` to find columns that might be part of a unique index. These methods typically work, but might result in fairly complex queries.

Consider the following:

```
rc = SQLExecDirect (hstmt, "SELECT first_name,
    last_name, ssn,
        address, city, state, zip FROM emp",
SQL_NTS);
// fetchdata
...
rc = SQLExecDirect (hstmt, "UPDATE EMP SET ADDRESS
= ?
    WHERE first_name = ? and last_name = ? and
    ssn = ?
        and address = ? and city = ? and state = ?
and zip = ?",
    SQL_NTS);
// fairly complex query
```

Applications should call `SQLSpecialColumns/SQL_BEST_ROWID` to retrieve the most optimal set of columns (possibly a pseudo-column) that identifies any given record. Many databases support special columns that are not explicitly defined by the user in the table definition but are “hidden” columns of every table (for example, `ROWID` and `TID`). These pseudo-columns almost always provide the fastest access to the data because they typically are pointers to the exact location of the record. Because pseudo-columns are not part of the explicit table definition, they are not returned from `SQLColumns`. The only method of determining if pseudo-columns exist is to call `SQLSpecialColumns`.

Consider the previous example again:

```
...
rc = SQLSpecialColumns (hstmt, ..... 'emp', ...);
...
rc = SQLExecDirect (hstmt, "SELECT first_name,
last_name, ssn,
        address, city, state, zip, ROWID FROM emp",
SQL_NTS);
// fetch data and probably "hide" ROWID from the
user
...
rc = SQLExecDirect (hstmt, "UPDATE emp SET address
= ? WHERE
        ROWID = ?", SQL_NTS);
// fastest access to the data!
```

If your data source does not contain special pseudo-columns, then the result set of `SQLSpecialColumns` consists of the columns of the most optimal unique index on the specified table (if a unique index exists); therefore, your application need not additionally call `SQLStatistics` to find the smallest unique index.



# F Threading

The ODBC specification mandates that all drivers must be thread-safe; that is, drivers must not fail when database requests are made on separate threads. It is a common misperception that issuing requests on separate threads will always result in improved throughput. Because of network transport and database server limitations, some drivers may serialize threaded requests to the server to ensure thread safety.

The ODBC 3.0 specification does not provide a method to find out how a driver will service threaded requests although this information is quite useful to an application. All INTERSOLV drivers provide this information to the user via the SQLGetInfo information type 1028.

The result of calling SQLGetInfo with 1028 is a SQL\_USMALLINT flag which denotes the session's thread model. A return value of 0 denotes that the session is fully thread enabled and that all requests will fully utilize the threaded model. A return value of 1 denotes that the session is restricted at the connection level. Sessions of this type are fully thread enabled when simultaneous threaded requests are made with statement handles that do not share the same connection handle. In this model, if multiple requests are made from the same connection, then the first request received by the driver is processed immediately and all subsequent requests are serialized. A return value of 2 denotes that the session is thread impaired and all requests are serialized by the driver.

Consider the following code fragment:

```
rc = SQLGetInfo (hdbc, 1028, &ThreadModel, NULL, NULL);

If (rc == SQL_SUCCESS) {
    // driver is an INTERSOLV driver which can report
    // threading information

    if (ThreadModel == 0)
        // driver is unconditionally thread enabled
        // application can take advantage of threading

    else if (ThreadModel == 1)
        // driver is thread enabled when thread requests are
        // from different connections
        // some applications can take advantage of threading

    else if (ThreadModel == 2)
        // driver is thread impaired
        // application should only use threads if it reduces
        // program complexity

}
else
    // driver is only guaranteed to be thread-safe
    // use threading at your own risk
```

**Table F-1** summarizes the threading information available at this time for INTERSOLV drivers. Always consult the README file for the most up-to-date information as the threading information is subject to change with new database transport and server revisions.

---

**Table F-1. Threading Information**


---

<b>Driver</b>	<b>Fully Threaded</b>	<b>Thread Per Connect</b>	<b>Thread Impaired</b>
Btrieve	x		
dBASE	x		
DB2		x	
Excel Workbook	x		
INFORMIX		x	
Open INGRES			x
Oracle		x	
Paradox	x		
PROGRESS			x
SQLBase		x	
SQL Server		x	
Sybase (Open Client 10.x)			x
Sybase (Open Client 11.x)		x	
Text	x		

---



# G Microsoft Query '97



This appendix applies only to Windows platforms.

---

## Creating a Flat-File Data Source for Use with Microsoft Query '97

To use a flat-file database driver with Microsoft Query '97, you must alter the data source alias outside of Microsoft Query '97. The data source alias created within Microsoft Query sets a default database, which is the Microsoft Query '97 working directory. Most likely, the Microsoft Query '97 working directory does not contain your data files.

The following steps describe how to update a flat-file data source. In this example, a dBASE data file is used:

- 1 Open WordPad or another text editor and edit the data source file that was created in Microsoft Query '97. The file will have a .dsn extension and will be located in the Microsoft Query '97 working directory.

The information will look similar to the section below:

```
[ODBC]
DRIVER={INTERSOLV 3.10 32-BIT dBASEfile
(*.dbf)}
DB=d:\msoffice\query97
```

- 2 Edit the DB entry to specify the directory that contains your data files. For example, the modified file may look similar to:

```
[ODBC]
DRIVER={INTERSOLV 3.10 32-BIT dBASEFile (*.dbf)}
DB=c:\data\sales
```

- 3 Save the file and exit the editor.

---

## Using Microsoft Query '97 with Single-Connect Data Sources

To use data source that are limited to a single connection per session (Btrieve and DB2 data sources), you must create the data source as a FileDSN through the ODBC Administrator.

Follow these steps to create a Btrieve DB2 data source. A DB2 data source is used in this example:

- 1 Start the ODBC Administrator.
- 2 Click the **File DSN** tab.
- 3 Click **Add**.
- 4 Select the INTERSOLV 3.10 32-BIT DB2 data source. Click **Next**.

You are prompted for the path name of the new file data source name.

- 5 Type the Microsoft Query '97 directory followed by your new data source name. For example,

```
d:\msoffice\msquery\DB2Test.dsn
```

Click **Next**.

- 6 Click **Finish** to validate the new File DSN entry.

The driver's logon box is displayed. Specify the logon options as you normally would for the DB2 database. Click **OK**.

- 7 Exit the ODBC Administrator.
- 8 Open WordPad or another text editor and edit the data source file you created in the ODBC Administrator. The file has a .dsn extension and will be located in the Microsoft Query '97 working directory. In our example, the file name is DB2Test.dsn.

The information will look similar to the following:

```
[ODBC]
DRIVER={INTERSOLV 3.10 32-BIT DB2}
DB=test
UID=TEST1
```

- 9 Add the following entry to specify the applicable driver option:

```
WA=4
```

The modified file will look similar to the following:

```
[ODBC]
DRIVER={INTERSOLV 3.10 32-BIT DB2}
WA=4
DB=test
UID=TEST1
```

- 10 Exit the editor.





# H The UNIX Environment

This appendix contains specific information about using Connect ODBC in the UNIX environment.

---

## 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 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 Solaris System Information File

```
[ODBC Data Sources]
```

```
Oracle7=Sample Oracle dsn
```

```
dBase=Sample dBASE dsn
```

```
Sybase=Sample Sybase dsn
```

```
Informix=Sample Informix dsn
```

```
OpenIngres=Sample OpenIngres dsn
```

```
DB2=Sample DB2 dsn
```

```
Text=Sample Text file dsn
```

```
[dBase]
```

```
Driver=/opt/odbc/lib/ivdbf12.so
```

```
Description=dBase
```

```
Database=/opt/odbc/demo
```

```
[Sybase]
```

```
Driver=/opt/odbc/lib/ivsyb12.so
```

```
Description=Sybase
```

```
Database=odbc
```

```
ServerName=SYBASE
```

```
LogonID=odbc01
```

```
Password=odbc01
```

```
OptimizePrepare=2
```

```
SelectMethod=1
```

## [Oracle7]

Driver=/opt/odbc/lib/ivor712.so  
Description=Oracle7  
ServerName=oraclehost  
LogonID=odbc01  
Password=odbc01

## [Informix]

Driver=/opt/odbc/lib/ivinf12.so  
Description=Informix7  
Database=odbc  
HostName=informixhost  
LogonID=odbc01  
Password=odbc01

## [DB2]

Driver=/opt/odbc/lib/ivdb212.so  
Description=DB2  
Database=ODBC

## [OpenIngres]

Driver=/opt/odbc/lib/ivoing12.so  
ServerName=ingreshost  
Database=odbc  
LogonID=odbc01  
Password=odbc01

## [Text]

Driver=/opt/odbc/lib/ivtxt12.so  
Description=Text driver  
Database=/opt/odbc/demo

## [ODBC]

Trace=0  
TraceFile=odbctrace.out  
TraceDll=/opt/odbc/lib/odbcctrac.so  
InstallDir=/opt/odbc

---

## Environment Variables

The INTERSOLV drivers require several environment variables to be set.

### Required Environment Variables

Most of the variables 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 INTERSOLV 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 INTERSOLV 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 369.

---

## 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.

For example:

```
ivtestlib ivinf12.so
```

will attempt to load the Informix driver on Solaris.

---

## Translators

INTERSOLV provides a sample translator named INTERSOLV OEM ANSI that translates 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.

<b>Keyword</b>	<b>Definition</b>
TranslationSharedLibrary	Full path of translation library.
TranslationOption	ASCII representation of the 32-bit integer translation option.

# Index

## A

Accessing 185  
 aggregate functions, flat-file drivers 290  
 Alter Table statement  
   Btrieve 52  
   dBASE 95  
   Paradox 181  
   Text 287

## B

binding to database, DB2 62  
 Borland Database Engine 169  
 Btrieve driver  
   see *also* flat-file drivers  
   Alter Table statement 52  
   column names 50  
   connection string attributes 45  
   connections supported 54  
   Create Index statement 53  
   data source  
     configuring 37  
     connecting via connection string 44  
   data types 49  
   Drop Index statement 53  
   indexes 50  
   isolation levels 54  
   locking levels 54  
   managing databases 36  
   ODBC conformance 54  
   Rowid pseudo-column 51  
   Select statement 51  
   statements supported 54  
   system requirements 35

table structure 42  
 transactions 37

## C

Clipper files 69  
 column names  
   Btrieve 50  
   Excel 114  
 configuring data source  
   see data source, configuring  
 Connect ODBC  
   Btrieve 35  
   DB2 55  
   dBASE 69  
   Excel Workbook 105  
   INFORMIX 117  
   OpenIngres 135  
   Oracle 151  
   Paradox 169  
   PROGRESS 195  
   SQL Server 217  
   SQLBase 233  
   Sybase 243  
   Text 263  
 connecting to data source  
   see data source, connecting  
 connection string attributes  
   Btrieve 45  
   DB2 65  
   dBASE 86  
   Excel 111  
   INFORMIX 127  
   OpenIngres 145  
   Oracle 161

- Paradox driver 176
- PROGRESS 212
- SQL Server 226
- SQLBase 239
- Sybase 255
- Text 282
- connections supported
  - Btrieve 54
  - DB2 68
  - dBASE 103
  - Excel 115
  - INFORMIX 133
  - OpenIngres 150
  - Oracle 167
  - Paradox 193
  - PROGRESS 215
  - SQL Server 232
  - SQLBase 242
  - Sybase 261
  - Text 288
- conventions 19
- CORE35.DLL 152
- CORE350.DLL 152
- Create Index Primary statement, Paradox 189
- Create Index statement
  - Btrieve driver 53
  - dBASE 96
  - Paradox 189
- Create Table statement
  - Excel 114
  - flat-file drivers 308
  - Paradox 183

## D

- data dictionary file 36
- data source
  - configuring
    - Btrieve 37
    - DB2 56
    - dBASE 70
    - Excel 106

- FoxPro DBC 76
- INFORMIX 120
- OpenIngres 137
- Oracle 154
- Paradox 171
- PROGRESS 196
- SQL Server 218
- SQLBase 234
- Sybase 245
- Text 265
- connecting via connection string
  - Btrieve 44
  - DB2 64
  - dBASE driver 85
  - Excel 110
  - INFORMIX 126
  - OpenIngres 144
  - Oracle
  - Paradox 175
  - PROGRESS 211
  - SQL Server 225
  - SQLBase 238
  - Sybase 254
  - Text 281
- connecting via logon dialog box
  - DB2 63
  - INFORMIX 124
  - OpenIngres 143
  - Oracle 159
  - PROGRESS 208
  - SQL Server 224
  - SQLBase 237
  - Sybase 253
- data types
  - Btrieve driver 49
  - DB2 67
  - dBASE 92
  - Excel 113
  - INFORMIX 130
  - OpenIngres 149
  - Oracle 164
  - Paradox 180
  - PROGRESS 214
  - SQL Server 230



- SQLBase 241
- Sybase 260
- Text 286
- DataDirect drivers 25
- date and time functions 331
- date masks, Text driver 279
- DB2 driver
  - binding to database 62
  - connection string attributes 65
  - connections supported 68
  - data source
    - configuring 56
    - connecting via connection string 64
    - connecting via logon dialog box 63
  - data types 67
  - isolation levels 68
  - locking levels 68
  - ODBC conformance 68
  - statements supported 68
  - system requirements 55
- dBASE driver
  - see *also* flat-file drivers
  - Alter Table statement 95
  - column names 93
  - connection string attributes 86
  - connections supported 103
  - Create Index statement 96
  - data source
    - configuring 70
    - connecting via connection string 85
  - data types 92
  - defining index attributes 82
  - defining index attributes on UNIX 84
  - Drop Index statement 98
  - isolation levels 103
  - locking 101
  - locking levels 103
  - ODBC conformance 103
  - Pack statement 99
  - Rowid pseudo-column 94
  - Select statement 94
  - statements supported 103
  - system requirements 69
- dBASE III, IV, and V files 69

- decrypting table, Paradox 186
- defining table structure
  - Btrieve 42
  - Text 271
  - Text, on UNIX 276
- Delete statement, flat-file drivers 313
- designing ODBC applications for performance 341
- Drop Index statement
  - Btrieve driver 53
  - dBASE driver 98
  - Paradox 191
- Drop Table statement, flat-file drivers 309
- dropping columns, Paradox 182

## E

- encryption, Paradox 184
- environment variable
  - IL\_SYSTEM 136
  - SYBASE 244
- environment-specific information 21, 27
- Error messages
  - UNIX 33
- error messages 32
- Excel driver
  - see *also* flat-file drivers
  - column names 114
  - connection string attributes 111
  - connections supported 115
  - Create Table statement 114
  - data source
    - configuring 106
    - connecting via connection string 110
  - data types 113
  - Excel 5 database 105
  - ODBC conformance 115
  - Select statement 114
  - statements supported 115
  - system requirements 105
  - table names 114

**F**

- FIELD.DDF 36
- FILE.DDF 36
- flat-file drivers
  - aggregate functions 290
  - Create Table statement 308
  - Delete statement 313
  - Drop Table statement 309
  - For Update clause 296
  - From clause 292
  - Group By 293
  - Having clause 294
  - Insert statement 310
  - Order By 295
  - Select clause 290
  - Select statement 289
  - SQL 289
  - SQL expressions 296
  - Union operator 294
  - Update statement 312
  - Where clause 292
- For Update clause, flat-file drivers 296
- formats, for text files 264
- FoxPro
  - configuring DBC data source 76
  - data types 93
  - SQL statements 100
- From clause, flat-file drivers 292

**G**

- getting started 25
- Group By clause, flat-file drivers 293

**H**

- Having clause, flat-file drivers 294

**I**

- IDAPI32.DLL 169
- II\_SYSTEM 136
- improving
  - database performance 315
  - join performance 321
  - record selection performance 317
- index attributes, defining
  - dBASE 82
  - dBASE under UNIX 84
- index file, Paradox 186
- INDEX.DDF 36
- indexes
  - Btrieve driver 50
  - deciding which to create 319
  - improving performance 315
- indexing multiple fields 317
- INFORMIX driver
  - connection string attributes 127
  - connections supported 133
  - data source
    - configuring 120
    - connecting via connection string 126
    - connecting via logon dialog box 124
  - data types 130
  - isolation levels 132
  - locking levels 132
  - ODBC conformance 133
  - statements supported 133
  - system requirements 117
- INFORMIXDIR 119
- Insert statement, flat-file drivers 310
- installing ODBC drivers 26
- interoperability 25
- isolation levels
  - Btrieve 54
  - DB2 68
  - dBASE 103
  - INFORMIX 132
  - OpenIngres 150
  - Oracle 166
  - Paradox 192

PROGRESS 215  
 SQL Server 231  
 SQLBase 241  
 Sybase 261  
 isolation levels and data consistency  
   compared 338  
   dirty reads 336  
   non-repeatable reads 336  
   phantom reads 336  
 isolation levels, general 336  
 isolation levels, specific  
   read committed 337  
   read uncommitted 337  
   repeatable read 337  
   serializable 337  
 ISQLT07C.DLL 118, 119

## J

join performance, improving 321

## L

library path environment variable 31  
 locking levels  
   Btrieve 54  
   DB2 68  
   dBASE 103  
   INFORMIX 132  
   OpenIngres 150  
   Oracle 166  
   Paradox 170, 192  
   PROGRESS 215  
   SQL Server 231  
   SQLBase 241  
   Sybase 261  
 locking modes and levels 339

## M

Macintosh  
   driver names 28  
   starting the ODBC Administrator 28  
   system requirements 29  
 managing  
   databases, Btrieve driver 36  
 multiple fields, indexing 317

## N

NLSRTL32.DLL 152  
 numeric functions 329

## O

OCIW32.DLL 152  
 ODBC  
   API functions 323  
   designing for performance 341  
   drivers, installing 26  
   scalar functions 326  
   specification 25  
 ODBC conformance  
   Btrieve 54  
   DB2 68  
   dBASE 103  
   Excel 115  
   INFORMIX 133  
   OpenIngres 150  
   Oracle 167  
   Paradox 192  
   PROGRESS driver 215  
   SQL Server 231  
   SQLBase 242  
   Sybase 261  
   Text 288  
 OLE32.DLL 105

- Open Database Connectivity
    - see ODBC
  - OpenIngres driver
    - connection string attributes 145
    - connections supported 150
    - data source
      - configuring 137
      - connecting via connection string 144
      - connecting via logon dialog box 143
    - data types 149
    - isolation levels 150
    - locking levels 150
    - ODBC conformance 150
    - statements supported 150
    - system requirements 135
  - operator precedence 302
  - ORA73.DLL 152
  - Oracle driver
    - connection string attributes 161
    - connections supported 167
    - data source
      - connecting via connection string 160
      - configuring 154
      - connecting via logon dialog box 159
    - data types 164
    - isolation levels 166
    - locking levels 166
    - ODBC conformance 167
    - statements supported 167
    - system requirements 151
  - ORACLE\_HOME 153
  - Order By clause, flat-file drivers 295
- P**
- Pack statement, dBASE 99
  - Paradox driver
    - see *also* flat-file drivers
    - Alter Table statement 181
    - connection string attributes 176
    - connections supported 193
    - Create Index Primary statement 189
    - Create Index statement 189
    - Create Table statement 183
    - data source
      - configuring 171
      - connecting via connection string 175
    - data types 180
    - decrypting table 186
    - Drop Index statement 191
    - dropping columns 182
    - encrypting table 185
    - encryption 184
    - index files 186
    - isolation levels 192
    - locking levels 170, 192
    - ODBC conformance 192
    - passwords 184
      - removing 186
    - Select statement 181
    - statements supported 193
    - system requirements 169
    - table access, multiuser 170
    - transactions 191
  - passwords
    - Paradox driver 184
    - removing 186
  - performance
    - improving 315
    - record selection 317
  - performance hints
    - ODBC applications 341
  - PROGRESS driver
    - connection string attributes 212
    - connections supported 215
    - data source
      - configuring 196
      - connecting via connection string 211
      - connecting via logon dialog box 208
    - data types 214
    - isolation levels 215
    - locking levels 215
    - ODBC conformance 215
    - OID with access via server 203

OID with direct access 196  
 statements supported 215  
 system requirements 195

## R

record select, improving performance 317  
 reserved words 314  
 Rowid pseudo-column  
   Btrieve driver 51  
   dBASE 94

## S

Scalable SQL 36  
 scalar functions, ODBC 326  
 Select clause, flat-file drivers 290  
 Select statement  
   dBASE driver 94  
   Excel 114  
   flat-file drivers 289  
   Paradox 181  
   Text 287  
 SQL  
   expressions, flat-file drivers 296  
   flat-file drivers 289  
   reserved words 314  
   Scalable 36  
   Select statement  
     Btrieve driver 51  
 SQL Server driver  
   connection string attributes 226  
   connections supported 232  
   data source  
     configuring 218  
     connecting via connection string 225  
     connecting via logon dialog box 224  
   data types 230  
   isolation levels 231  
   locking levels 231  
   ODBC conformance 231  
   statements supported 232  
   system requirements 217  
 SQL statements, FoxPro 3.0 100  
 SQLBase driver  
   connection string attributes 239  
   connections supported 242  
   data source  
     configuring 234  
     connecting via connection string 238  
     connecting via logon dialog box 237  
   data types 241  
   isolation levels 241  
   locking levels 241  
   ODBC conformance 242  
   statements supported 242  
   system requirements 233  
 SQLEdit 243  
 SQLNGCI.DLL 233  
 SQLWNTM.DLL 233  
 statements supported  
   Btrieve 54  
   DB2 68  
   dBASE 103  
   Excel 115  
   INFORMIX 133  
   OpenIngres 150  
   Oracle 167  
   Paradox 193  
   PROGRESS 215  
   SQL Server 232  
   SQLBase 242  
   Sybase 261  
   Text 288  
 string functions 326  
 Structured Query Language  
   see SQL  
 SupportNet 22  
 Sybase driver  
   connection string attributes 255  
   connections supported 261

- data source
  - configuring 245
  - connecting via connection string 254
  - connecting via logon dialog box 253
- data types 260
- isolation levels 261
- locking levels 261
- ODBC conformance 261
- statements supported 261
- system requirements 243
- SYBPING 243
- system functions 333
- system requirements
  - Btrieve 35
  - DB2 55
  - dBASE 69
  - Excel 105
  - INFORMIX 117
  - OpenIngres 135
  - Oracle driver 151
  - Paradox driver 169
  - PROGRESS 195
  - SQL Server 217
  - SQLBase 233
  - Sybase 243
  - Text 263

## T

- table
  - decrypting 186
  - encrypting 185
- table access, multiuser, Paradox 170
- table names, Excel 114
- table structure, defining
  - Btrieve 42
  - Text 271
  - Text, on UNIX 276
- Technical Support, contacting 22
- Text driver
  - see *also* flat-file drivers
  - Alter Table statement 287

- connection string attributes 282
- connections supported 288
- data source
  - configuring 265
  - connecting via connection string 281
- data types 286
- date masks 279
- defining table structure 271
- defining table structure on UNIX 276
- formats 264
- ODBC conformance 288
- Select statement 287
- statements supported 288
- system requirements 263
- threading 361
- time functions 331
- transactions
  - Btrieve driver 37
  - Paradox 191

## U

- Union operator, flat-file drivers 294
- UNIX
  - driver names 30
  - environment 369
    - double-byte character sets 373
    - ivtestlib tool 374
    - library path 31
    - system information file (.odbc.ini) 30, 369
    - translators 374
    - variables 372
  - error messages 33
  - system requirements 29, 31
- Update statement, flat-file drivers 312

## W

Where clause, flat-file drivers 292

Windows

- driver names 27

- starting the ODBC Administrator 27

- system requirements 28

WWW, using to contact INTERSOLV 22

