DataDirect ODBC dBASE Driver

About the dBASE Driver

Configuring dBASE Data Sources

Configuring FoxPro 3.0 DBC Data Sources

Defining Index Attributes

Connecting to dBASE Using a Connection String

Data Types

SQL Statements for dBASE

Locking

Isolation and Lock Levels Supported

ODBC Conformance Level

Number of Connections and Statements Supported

Copyright

About the dBASE Driver

The dBASE driver supports

- dBASE III, dBASE IV and dBASE V files
- Clipper files
- FoxBASE tables and indexes
- FoxPro 1.0, 2.5, and 3.0 tables and indexes, and FoxPro 3.0 database containers

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

The driver filename is IVDBF*nn*.DLL.

Configuring Data Sources

To configure a data source using the dBASE driver, follow these steps. To configure a data source for FoxPro 3.0 DBC tables, see Configuring FoxPro 3.0 DBC Data Sources.

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

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

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

The ODBC dBASE Driver Setup dialog box appears.

3 Specify a data source name and database directory. Specifying a database description and create type is optional.

4 Click the **Advanced** button to configure optional data source settings, such as locking and file caching information. The ODBC Advanced Driver Setup dialog box appears.

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

Configuring FoxPro 3.0 DBC Data Sources

To configure a data source for FoxPro 3.0 database containers, do the following:

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

2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select INTERSOLV FoxPro 3.0 DBC driver and click **OK**.

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

The ODBC FoxPro Database Driver Setup dialog box appears.

3 Specify a data source name and database directory. Specifying a database description is optional. You cannot change the Create Type for the FoxPro 3.0 DBC driver.

4 Click the **Advanced** button to configure optional data source settings, such as locking and file caching information. The ODBC Advanced Driver Setup dialog box appears.

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

Defining Index Attributes

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

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

1. Click **Define** in the dBASE setup dialog box, which you can access through the ODBC Administrator. The Define File dialog box appears.
2. Select a dBASE file and click **OK** to define the special indexes using the Define Table dialog box.
3. The upper section of the dialog box displays the directory name and filename that contains the data file. Under Windows, if this file is stored in the IBM PC character set, select the OEM to ANSI Translation box.
4. The lower section of the dialog box displays the index information for the data file. The Index File drop-down list lets you select any index file in the database directory. If the index file is in a different directory, you must provide the full pathname.

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

   To specify that an index file is unique, select the Unique check box that appears at the right of the index filename.
5. If the selected index has an .MDX or .CDX extension, you cannot mark the index file as unique. Instead, you may mark the tags within the index as unique. To do so, select the tag name in the Tag drop-down list and select the Unique check box that appears at the right of the tag name.
6. Click **OK** to save this information, or press Cancel.

Connecting to dBASE Using a Connection String

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

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

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

An example of a connection string for dBASE is

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

The following table gives the long and short names for each attribute, as well as a description.

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

Attribute	Description
DataSourceName (DSN)	String that identifies a dBASE data source configuration in ODBC.INI. Examples include "Accounting" or "dBASE Files."
Database (DB)	Directory in which the dBASE files are stored.
CreateType (CT)	CreateType={dBASE3 | dBASE4 | dBASE5 | Clipper | FoxBASE | FoxPro1 | FoxPro25 | FoxPro30}. Type of table or index to be created on a Create Table or Create Index statement. dBASE5 is the initial default. **Note:** If you are using the FoxPro 3.0 DBC driver, you cannot change the CreateType attribute.
LockCompatibility (LCOMP)	LockCompatibility={Q+E | Q+EVirtual | dBASE | Clipper | Fox}. Locking schemes to be used in your dBASE tables. ▪ LockCompatibility=Q+E specifies that locks be placed on the actual bytes occupied by the record. Only applications that use the dBASE driver can read and write to the database. Other applications are locked out of the table completely (they cannot even read other records). This locking is compatible with earlier versions of Q+E products. ▪ LockCompatibility=Q+EVirtual specifies that locks be placed on bytes beyond the physical end-of-file. Q+EVirtual is the same as Q+E except that other applications can open the table and read the data. The advantage of using a Q+E locking scheme over dBASE locking is that, on Inserts and Updates, Q+E locks only individual index tags, while dBASE locks the entire index. ▪ LockCompatibility=dBASE specifies Borland-compatible locking. This is the initial default. ▪ LockCompatibility=Clipper specifies Clipper-compatible locking. ▪ LockCompatibility=Fox specifies FoxPro- and FoxBASE-compatible locking.

	If you are accessing a table with an application that uses the dBASE driver, your locking scheme does not have to match the Create Type. However, if you are accessing a table with two applications, and only one uses the dBASE driver, set your locking scheme to match the other application. For example, you don't have to set LCOMP=Fox to work with a FoxPro table. But if you are using a FoxPro application simultaneously with an application using the dBASE driver on the same set of tables, set LCOMP=Fox to ensure that your data does not get corrupted.
Locking (LCK)	Locking={NONE | RECORD | FILE} Level of locking for the database tables. Locking=NONE offers the best performance but is intended only for single-user environments.. Locking=RECORD locks only the records affected by the statement. This is the initial default. Locking=FILE locks all of the records in the table.
FileOpenCache (FOC)	Maximum number of unused file opens to cache. For example, when FileOpenCache=4, and a user opens and closes four files, the files are not actually closed. The driver keeps them open so that if another query uses one of these files, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The disadvantage is that a user who tries to open the file exclusively may get a locking conflict even though no one appears to have the file open. The initial default is 0.
CacheSize (CSZ)	The number of 64K blocks the driver uses to cache database records. The higher the number, the better the performance. The maximum number of blocks you can set depends on the system memory available. If the cache size is greater than 0, when browsing backwards, you will not be able to see updates made by other users until you reexecute the Select statement. The default is 4.
IntlSort (IS)	IntlSort={0 | 1}. A number that specifies the order that records are retrieved when you issue a Select statement with an Order By clause. If IntlSort=1, the driver uses the international sort order as defined by your operating system. The sort is case-insensitive (*a* precedes *B*); the sorting of accented characters is also affected (see your operating system documentation). If IntlSort=0 (the initial default), the driver uses the ASCII sort order, where uppercase letters precede lowercase letters (*B* precedes *a*).
ModifySQL (MS)	ModifySQL={0 | 1}. This attribute is provided for backward compatibility with earlier versions of Q+E products. It determines whether the driver modifies

	SQL statements to conform to ODBC specifications or passes the SQL statement directly to dBASE. Specify ModifySQL=1 to have the driver modify the SQL statement to conform to ODBC specifications. Specify ModifySQL=0 to have the driver understand SQL dialects found in earlier drivers. The default is 1.
Compatibility (COMP)	Compatibility={0 | 1}. This attribute is provided for backward compatibility with earlier versions of Q+E products. Use Compatibility=DBASE for backward compatibility; use Compatibility=ANSI for portability. The default is ANSI.
UltraSafeCommit (USF)	UltraSafeCommit={0 | 1}. A number that 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.
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 enviroment in which you are running (for example, in Windows 95, the maximum table name length is 128.
UseLongQualifiers (ULQ)	UseLongQualifiers={0 | 1}. This attribute specifies whether the driver uses long pathnames as table qualifiers. When you set this check box, pathnames can be up to 255 characters. The default length for pathnames is 128 characters.
AllowNull-Constraints (ANC)	AllowNullConstraints={0 | 1}. (FoxPro 3.0 driver only.) A number that determines how the _NullFlags bits (if present) are interpreted. See Adding Null Support for more information.
AlwaysApply-Translation (AAT)	AlwaysApplyTranslation={0 | 1}. (FoxPro 3.0 driver only.) A number that determines whether the translation DLL is always applied to character columns or if the driver determines whether it is applied.
DataFileExtension (DFE).	String of three or fewer characters that specifies the file extension to use for data files. The default DataFileExtension value is DBF. This value cannot be an extension the driver already uses, such as MDX or CDX. The DataFileExtension value is used for all CREATE TABLE statements. Sending a CREATE TABLE using an extension other than the DataFileExtension value causes an error. In other SQL statements such as SELECT or

INSERT, users can specify an extension other than the DataFileExtension value. The DataFileExtension value is used when no extension is specified.

Adding Null Support

FoxPro 3.0 has true null support. When creating a table in FoxPro, the user has the option to allow columns to contain null values. Internally, FoxPro sets a bit in the oclumn header to specify that the column allows null values. Another bit is allocated in the system column _NullFlags, which is set to True if the column currently contains a null value.

To facilitate null support for FoxPro 3.0 tables in the dBASE driver, the AllowNullConstraints connection string option determines how the _NullFlags bits, if present, are interpreted. This option is valid only when connecting to FoxPro 3.0 and FoxPro 3.0 DBC data sources.

	AllowNullContraints = 1 (On)	**AllowNullConstraints = 0 (Off)**
CREATE TABLE . . . (column type)	Nullable flag is set in column header, and bit is allocated in system column.	
CREATE TABLE . . . (column type NOT NULL)	Nullable flag is not set in column header.	Warning: null constraint is ignored.
UPDATE/INSERT setting column = NULL with nullable flag set in column header	System column null value bit is set for column, alpha columns are set to spaces, and binary columns are set to 0.	System column null value bit is set for column, alpha columns are set to spaces, and binary columns are set to 0.
UPDATE/INSERT setting column=NULL with nullable flag not set in column header	Error: column cannot contain NULL value.	
SELECT column with nullable flag set and null value bit set	Returns NULL.	Returns NULL.
SELECT column with nullable flag set and null value bit not set	Returns column value.	Returns column value.
SELECT column with nullable flag not set	Returns column value.	Interprets all blanks as NULL.

Data Types

The following table shows how dBASE data types map to the standard ODBC data types. These dBASE data types can be used in a Create Table statement.

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.

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

1 dBASE V only

2 254 characters maximum (1024 for Clipper)

3 dBASE III, IV, FoxPro, FoxBASE, and Clipper

4 dBASE IV only

5 FoxPro 2.5 and dBASE V only

6 FoxPro, FoxBASE, and Clipper

FoxPro 3.0 Data Types

Data types in FoxPro 3.0 tables and database containers map to the ODBC data types as follows:

FoxPro 3.0	ODBC
Character (binary)	SQL_CHAR
Currency	SQL_DOUBLE
Datetime	SQL_TIMESTAMP
Double	SQL_DOUBLE
Integer	SQL_INTEGER
Memo (binary)	SQL_LONGVARBINARY

SQL Statements for dBASE

If you are using the dBASE driver, you can use SQL statements as described in SQL for Flat-File Drivers to read, insert, update, and delete records from a database, create new tables, and drop existing tables. In addition, you can use SQL statements and clauses that are specific to the dBASE driver. The dBASE-specific SQL information is described in the following topics.

Select Statement

SQL Statements for FoxPro 3.0 Database Containers

Alter Table Statement

Create Index Statement

Drop Index Statement

Pack Statement

Select Statements

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.

Column Names

The maximum length of a column name is 10 characters. A column name can contain alphanumeric characters and the hyphen character (-). The first character must be a letter (a through z).

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.

SQL Statements for FoxPro 3.0 Database Containers

The FoxPro 3.0 DBC driver supports four additional SQL statements.

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.

Alter Table Statement

The dBASE driver supports the Alter Table statement to add one or more columns to the 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 for 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. See dBASE Data Types for more information.

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

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

You cannot add 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 Index Statement

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

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

The syntax for creating an index is

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

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

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

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

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

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

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

The following table 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

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
FoxBASE .IDX*	Yes	No	100	219	No	Yes
dBASE IV and V .MDX	Yes	Yes	100	220	Yes	Yes
FoxPro 2.5 .IDX**	Yes	Yes	240	255	No	Yes

FoxPro 2.5 and 3.0 .CDX	Yes	Yes	240	255	Yes	Yes

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

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

Drop Index Statement

The syntax for dropping a dBASE index is as follows:

```
DROP INDEX table_name.index_name
```

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

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

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

```
DROP INDEX emp.emphire
```

Pack Statement

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

```
PACK filename
```

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

```
PACK emp
```

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

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

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

Locking

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

Levels of Database Locking

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

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

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

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

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

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

Using Locks on Local Files

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

Limit on Number of Locks

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

How Transactions Affect Record Locks

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

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

Isolation and Lock Levels Supported

dBASE supports isolation level 1. It supports both file- and record-level locking.

ODBC Conformance Level

The dBASE driver supports the Core, Level 1, and Level 2 API functions listed in Supported ODBC Functions. The dBASE driver also supports backward and random fetching in SQLExtendedFetch.

The driver supports the minimum SQL grammar.

Number of Connections and Statements Supported

dBASE supports multiple connections and multiple statements per connection.

dBASE ODBC Setup Dialog

Use the dBASE ODBC Setup dialog to create new dBASE data sources or configure existing data sources.

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

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

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

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, FoxBASE, FoxPro1, FoxPro25, or FoxPro30. The default is dBASE V.

Advanced: Displays the ODBC Advanced Driver Setup dialog box to configure optional data source settings, such as locking and file caching.

OK

Creates or modifies the current data source using the options you specify.

Cancel

Exits the ODBC Setup dialog without creating or modifying a data source.

FoxPro 3.0 ODBC Setup Dialog

Use the FoxPro 3.0 ODBC Setup dialog to configure a data source for FoxPro 3.0 database containers.

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

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

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

Click **Select** to browse the available FoxPro 3.0 database containers.

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

Advanced: Displays the ODBC Advanced Driver Setup dialog box to configure optional data source settings, such as locking and file caching.

OK

Creates or modifies the current data source using the options you specify.

Cancel

Exits the ODBC Setup dialog without creating or modifying a data source.

Advanced Dialog

To configure optional settings for a dBASE data source, specify values as follows:

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.

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

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

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

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

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

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.

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

Data File Extension: Specifies the file extension to use for data files. The default Data File Extension setting is DBF. The Data File Extension 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 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.

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

Use Long Names: Set this check box to use long filenames as table names. The maximum table name

length is specific to the enviroment in which you are running (for example, in Windows 95, the maximum table name length is 128 characters).

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.

Define

Displays the Define Index dialog box to define the index attributes for your data files. See Defining Index Attributes for step-by-step instructions.

Note The Define button is unavailable if you are configuring advanced options for a FoxPro 3.0 DBC data source.

Translate

Displays the Select Translator dialog box to perform a translation of your data from one character set to another. INTERSOLV provides a translator named INTERSOLV OEM ANSI that translates your data from the IBM PC character set to the ANSI character set.

The translators that are listed in the Select Translator dialog box are determined by the values listed in the ODBC Translators section of your ODBCINST.INI file.

Click **OK** to leave this dialog box and perform the translation.

Close

Returns to the dBASE or FoxPro ODBC Setup dialog box, where you can click the **OK** button to write these settings to the ODBC.INI file.

Defining dBASE Index Attributes

To define the index files that are associated with a dBASE file:

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

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

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

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

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

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

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

6 Click **OK** to save this information, or press Cancel.