# ODBC Text Driver

## For All Users

The following topics discuss the ODBC text driver and how to install it.

Overview
Hardware and Software Requirements
Setting Up the ODBC Text Driver
Adding, Modifying, and Deleting a Text Data Source
Connecting to a Text Data Source
Using the ODBC Text Driver

## For Advanced Users

The following topics discuss how to use the ODBC text driver directly.

Connection Strings (Advanced)
Text File Format (Advanced)
SQL Statements (Advanced)
Data Types (Advanced)
Error Messages (Advanced)

## For Programmers

The following topics provide programming information on the ODBC text driver.  They are intended for application programmers and require knowledge of the Open Database Connectivity (ODBC) application programming interface (API).

SQLGetInfo Return Values (Programming)
ODBC API Functions (Programming)
Implementation Issues (Programming)

## Overview

See Also

A text database is a directory that contains text files (tables). You specify information called a schema that describes the content and format of text files you want to access. The schema is what the ODBC text driver uses to open and create new text tables.

The ODBC text driver allows you open and query a text database through the Open Database Connectivity (ODBC) interface.

The application/driver architecture is:

Application

|

ODBC Driver Manager (ODBC.DLL)

|

ODBC Text Driver (SIMBA.DLL, TXTISAM.DLL)

|

Text File

**See Also**

For All Users

- Adding, Modifying, and Deleting a Text Data Source
- Connecting to a Text Data Source
- Hardware and Software Requirements
- Setting Up the ODBC Text Driver
- Using the ODBC Text Driver

## Hardware and Software Requirements

See Also

To access text files, you must have:

- The ODBC text driver.
- The ODBC Driver Manager 1.0 (ODBC.DLL).
- A computer running MS-DOS 3.3 or later.
- Microsoft Windows 3.0a or later.

To add, modify, or delete drivers or data sources, you should have the ODBC Control Panel option (or the ODBC Administrator program if you're running Windows 3.0a) installed on your computer.

**See Also**

For All Users

Setting Up the ODBC Text Driver

## Setting Up the ODBC Text Driver

See Also

**To set up the ODBC text driver**

1 In the Main group in the Program Manager window, double-click the Control Panel icon. In the Control Panel window, double-click the ODBC icon.

---

**Note** For Windows 3.0a, start the ODBC Administrator by double-clicking the ODBC Administrator icon in the Microsoft ODBC group.

---

2 In the Data Sources dialog box, choose the Drivers button.

3 In the Drivers dialog box, choose the Add button.

4 In the Add Driver dialog box, enter the name of the drive and directory containing the ODBC text driver in the text box. Or choose the Browse button to select a drive and directory name.

5 Choose the OK button.

6 In the Install Drivers dialog box, choose Text from the Available ODBC Drivers list.

7 Choose the OK button to install the driver.

---

**Note** The ODBC text driver may share some of the same dynamic link libraries (DLLs) with other drivers installed on your computer. If so, you will be asked to overwrite the ODBC text driver, regardless of whether it has been installed. Choose the Yes button to install the driver.

---

After installing the driver, you must then add a data source for it.

**To delete the ODBC text driver**

1 In the Main group in the Program Manager window, double-click the Control Panel icon. In the Control Panel window, double-click the ODBC icon.

---

**Note** For Windows 3.0a, start the ODBC Administrator by double-clicking the Microsoft ODBC Administrator icon in the Microsoft ODBC group.

---

2 In the Data Sources dialog box choose the Drivers button.

3 In the Drivers dialog box, select the ODBC text driver from the list.

4 Choose the Delete button.

The ODBC text setup program asks if you want to remove the driver and all the data sources that use the driver.

5 Choose the Yes button.

**See Also**

For All Users

- Hardware and Software Requirements
- Adding, Modifying, and Deleting a Text Data Source

## Adding, Modifying, and Deleting a Text Data Source

See Also

Before you can access data with the ODBC text driver, you must add a data source for it. The ODBC text driver uses the information you enter to access the data. You can change or delete a data source at any time.

**To add a text data source**

1 In the Main group in the Program Manager window, double-click the Control Panel icon. In the Control Panel window, double-click the ODBC icon.

**Note** For Windows 3.0a, start the ODBC Administrator by double-clicking the Microsoft ODBC Administrator icon in the Microsoft ODBC group.

2 In the Data Sources dialog box, choose the Add button.

3 In the Add Data Source dialog box, select Text from the Installed ODBC Drivers list and choose OK.

4 In the ODBC Text Setup dialog box, enter information to set up the data source. To define the format of tables in the data source, choose the Define Format button. This displays the Define Text Format dialog box, with which you can specify the schema for the data source.

**To modify a text data source**

1 In the Main group in the Program Manager window, double-click the Control Panel icon. In the Control Panel window, double-click the ODBC icon.

**Note** For Windows 3.0a, start the ODBC Administrator by double-clicking the Microsoft ODBC Administrator icon in the Microsoft ODBC group.

2 In the Data Sources dialog box, select the data source from the Data Sources list.

3 Choose the Setup button.

4 In the ODBC Text Setup dialog box, enter information to set up the data source. To define the format of tables in the data source, choose the Define Format button. This displays the Define Text Format dialog box, with which you can specify the schema for the data source.

**To delete a text data source**

1 In the Main group in the Program Manager window, double-click the Control Panel icon. In the Control Panel window, double-click the ODBC icon.

**Note** For Windows 3.0a, start the ODBC Administrator by double-clicking the Microsoft ODBC Administrator icon in the Microsoft ODBC group.

2 In the Data Sources dialog box, select the data source from the Data Sources list.

3 Choose the Delete button, and then choose the Yes button to confirm the deletion.

**schema**

The schema includes information about each table (text file) in a data source, including the table's format, the number of rows to scan to determine column types, whether the first row of the table contains column names, and each column's name, data type, and width.

**See Also**

For All Users

- Connecting to a Text Data Source
- Setting Up the ODBC Text Driver

## ODBC Text Setup Dialog Box

See Also

The ODBC Text Setup dialog box contains the following fields:

**Data Source Name**

A name that identifies the data source, such as Payroll or Personnel.

**Description**

An optional description of the data in the data source; for example, "Hire date, salary history, and current review of all employees."

**Directory**

Displays the currently selected directory. Before you add the data source, you must either use the Select Directory button to select a directory, or select the Use Current Directory check box to use the application's current working directory.

When defining a text data source directory, specify the directory where your most commonly used text files are located. The ODBC Text driver uses this directory as the default directory. Copy other text files into this directory if they are used frequently. Alternatively, you can qualify file names in a SELECT statement with the directory name:

```
SELECT * FROM C:\MYDIR\EMP.TXT
```

Or, you can use the USE statement to specify a new default directory:

```
USE C:\MYDIR
```

**Options**

Displays the following options:

**Extensions List**

Lists the filename extensions of the text files on the data source. To use all files in the directory, select the Default (*.*) check box. To use only those files with certain extensions, clear the Default (*.*) check box and add each extension you want to use.

To add an extension, type the extension in the Extension box and click the Add button. The extension must use the format **.xxx*. For example, to use .DAT files, type the extension *.DAT. To remove an extension, highlight the extension in the Extensions list and click the Remove button.

**Define Format**

Displays the Define Text Format dialog box and enables you to specify the schema for individual tables in the data source directory.

**data source (text)**

A text data source specifies the default data directory in which the ODBC text driver searches for text files you want to access, as well as other driver information.

**See Also**

For All Users

- Adding, Modifying, and Deleting a Text Data Source
- Define Text Format Dialog Box

## Define Text Format Dialog Box

See Also

The Define Text Format dialog box enables you to define the format for columns in a selected file.

---

**Note** The ODBC text driver does not change the format of a text file to match the format defined with the Define Text Format dialog box. If the format of the text file does not match the format defined with this dialog box, the ODBC text driver returns an error when it uses the format, such as when it attempts to retrieve data from the text file.

---

The Define Text Format dialog box contains the following fields:

**Tables**

Lists text files in the data source directory having filename extensions that were listed in the Extensions list of the ODBC Text Setup dialog box. Select the file you want to define the format for, or select <default> to specify the format for files not explicitly defined. When <default> is selected, the Columns list and associated fields are disabled.

**Column Name Header**

Check this box if the first row of the file lists column names.

**Format**

Allows you to select the file format:

- CSV Delimited (Comma Separated Value)
  Fields are separated by a comma. Character columns can be enclosed in double quotation marks ("").
- Tab Delimited
  Fields are separated by a tab. Character columns can be enclosed in double quotation marks ("").
- Custom Delimited
  Fields are separated by a delimiter other than a comma or tab. Use the Delimiter field to specify the delimiter. Character columns can be enclosed in double quotation marks ("").
- Fixed Length
  Fields are of a fixed length. On output, column values less than the fixed length are padded with spaces.

**Delimiter**

Specify the delimiter character used in Custom Delimited text files (it is disabled otherwise). For example, * specifies that asterisks separate columns. You can also specify the delimiter in hexadecimal (\xHH) or decimal (\dDDD) format. For example, an asterisk (ASCII 42) in hexadecimal format is \x2A and in decimal format is \d042.

---

**Note** Double quotes (") may not be used as the delimiter character.

---

**Rows to Scan**

Enter the number of rows scanned to determine information about the text file. To scan the entire file, enter 0. This field is enabled only with Delimited text files.

**Characters**

If the text file uses a non-ANSI character set, select the OEM option button.

If the text file uses the ANSI character set, select the ANSI option button.

**Columns**

Lists the columns in the selected table.

**Data Type, Name, Width, and Date Separator**

Enables you to specify the schema for each data source. This information is written to a SCHEMA.INI file in the data source directory. There is a separate SCHEMA.INI for each text data source directory.

For fixed-length tables, Width is displayed for all data types but cannot be changed except in the Parse dialog. For other formats, Width is enabled only for Char or LONGCHAR data types.

The Date Separator field is enabled only for columns that have a Date data type.

**Guess (CSV Delimited, Custom Delimited, and Tab Delimited Formats)**

Automatically generates the column's data type, name and width values for the columns in the selected table by scanning the table's contents according to the Format list box selection. Any previously defined columns in the Columns list are cleared and replaced with new entries.

**Note** For fixed-length tables, the Guess button is replaced by the Parse button.

**Parse (Fixed-Length Format)**

Displays the Parse dialog box. Vertical lines separate the columns and a vertical line marks the end of the last column. To add a line separating two columns, place the cursor after the last character in the first column and double-click the mouse button or press the SPACEBAR. It is not necessary to specify the start of the first column or the end of the last column. To remove a line separating two columns, place the cursor on the line and double-click the mouse button or press the SPACEBAR.

Choose OK to close the dialog box and return to the Define Text Format dialog box. The column names are displayed in the Columns list and the data type of each column is set to Char. Use the Data Type, Name, Width, and Date Separator fields and the Modify button to modify the attributes of each column.

**Add**

Adds an entry to the schema for the data source. When you choose the Add button, the entry identified by the Data Type, Name, Width, and Date Separator fields is added to the end of the Column list.

**Modify**

Modifies a column entry. To update an entry, select a column from the Column list, enter new values in the Data Type, Name, Width, and Date Separator fields, and then choose the Modify button.

**Remove**

Deletes a column entry. To delete an entry, select a column from the column list, and then choose the Remove button.

**See Also**

For All Users

ODBC Text Setup Dialog Box

## Connecting to a Text Data Source

See Also

When you connect to a text data source, an application may prompt you to enter the name of a directory. If you are prompted, enter or select the directory containing the text files you want to access.

**See Also**

For All Users

Adding, Modifying, and Deleting a Text Data Source
Using the ODBC Text Driver

For Advanced Users

Connection Strings (Advanced)

## Using the ODBC Text Driver

The following information may be useful when using the ODBC text driver to access text files:

### Column and Table Names

If column or table names contain any characters except letters, numbers, and underscores, they must be delimited. To delimit a column or table name, enclose the name in double quotes(").

### Columns

- Column names over 30 characters are truncated.
- The driver allows column names to contain any valid text characters (for example, spaces). If column names contain any characters except letters, numbers, and underscores, they must be delimited. To delimit a column name, enclose the name in double quotes(").
- If you do not specify a column name, the driver provides a default name. For example, the driver calls the first column Col1, the second column Col2, and so on.
- Although not required, character columns in delimited text files can be enclosed in double quotation marks (""). Fixed-length tables should not use double quotation-mark delimiters.
- The maximum size of a LONGCHAR column is 65,500 characters.
- For all columns, null values are represented by a blank padded string in fixed-length files, but are represented by no spaces in delimited files. For example, in the following row containing three fields, the second field is a null value:

  "Smith", , 123
- All column values may be padded with leading spaces.

### Literals

- The maximum length of any literal (for example, a string) is 1000 characters.
- A character string literal can be any ANSI character (1 - 255 decimal). Use two consecutive single quotation marks ('') to represent one single quotation mark (').

### Rows

The length of any row must be less than or equal to 65, 532 bytes.

### Tables

- The maximum width of a table is 256 columns for both fixed and delimited tables.
- All tables are opened shared, except for tables created for inserting. Those tables are opened exclusive and can only be modified by one user at a time. Data in tables cannot be updated or deleted.
- The driver allows table names to contain any valid MS-DOS characters. If table names contain any characters except letters, numbers, and underscores, they must be delimited. To delimit a table name, enclose the name in double quotes(").

## Connection Strings (Advanced)

See Also

The connection string for the ODBC text driver uses the following keywords:

| Keyword | Description |
|---|---|
| **DSN** | Name of the text data source. |
| **DBQ** | The text directory. |
| **FIL** | File type (TEXT). |

For example, to connect to the Accounting data source in the directory C:\ACCT, use the following connection string:

```
DSN=Accounting;DBQ=C:\ACCT;FIL=TEXT
```

**See Also**

For All Users

Connecting to a Text Data Source

## Text File Format (Advanced)

See Also

The ODBC text driver supports both delimited and fixed-width text files. A text file consists of an optional header line and zero or more text lines.

Although the header line uses the same format as the other lines in the text file, the ODBC text driver interprets the header line entries as column names, not data.

A delimited text line contains one or more data values separated by delimiters: commas, tabs, or a custom delimiter. The same delimiter must be used throughout the file. Null data values are denoted by two delimiters in a row with no data between them. Character strings in a delimited text line may be enclosed in double quotation marks (" ").

The width of each data entry in a fixed-width text line is specified in a schema. Null data values are denoted by blanks.

The following grammar, written for programmers, defines the format of a text file that can be read by the ODBC text driver. Non-italics represent characters that must be entered as shown, italics represent arguments that are defined elsewhere in the grammar, brackets ([]) represent optional items, braces ({}) delimit a list of mutually exclusive choices, vertical bars (|) separate these choices, and ellipses (...) represent items that can be repeated one or more times.

The format of a text file is:

*text-file* ::=
  [*delimited-header-line*] [*delimited-text-line*]... *end-of-file* |
  [*fixed-width-header-line*] [*fixed-width-text-line*]... *end-of-file*

*delimited-header-line* ::= *delimited-text-line*

*delimited-text-line* ::=
  *blank-line* |
  *delimited-data* [*delimiter delimited-data*]... *end-of-line*

*fixed-width-header-line* ::= *fixed-width-text-line*

*fixed-width-text-line* ::=
  *blank-line* |
  *fixed-width-data* [*fixed-width-data*]... *end-of-line*

*end-of-file* ::= <EOF>

*blank-line* ::= *end-of-line*

*delimited-data* ::= *delimited-string* | *number* | *date* | *delimited-null*

*fixed-width-data* ::= *fixed-width-string* | *number* | *date* | *fixed-width-null*
The width of each column in a fixed width text file is specified in the SCHEMA.INI file.

*end-of-line* ::= <CR> | <LF> | <CR><LF>

*delimited-string* ::= *unquoted-string* | *quoted-string*

*unquoted-string* ::= [*character* | *digit*] [*character* | *digit* | *quote-character*]...

*quoted-string* ::=
  *quote-character*
  [*character* | *digit* | *delimiter* | *end-of-line* | *embedded-quoted-string*]...
  *quote-character*

*embedded-quoted-string ::=*
  *quote-character quote-character*
  [*character* | *digit* | *delimiter* | *end-of-line*]
  *quote-character quote-character*

*fixed-width-string* ::= [*character* | *digit* | *delimiter* | *quote-character*] ...

*character* ::= any character except:
  *delimiter*
  *digit*
  *end-of-file*
  *end-of-line*
  *quote-character*

*digit* ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 *delimiter* ::= , | <TAB> | *custom-delimiter*

*custom-delimiter* ::= any character except:
    *end-of-file*
    *end-of-line*
    *quote-character*
The delimiter in a custom-delimited text file is specified in the SCHEMA.INI file.

*quote-character* ::= "

*number* ::= *exact-number* | *approximate-number*

*exact-number* ::= [+ | -] {*unsigned-integer*[.*unsigned-integer*] |
    *unsigned-integer*. |
    .*unsigned-integer*

*approximate-number* ::= *exact-number*{e | E}[+ | -]*unsigned-integer*

*unsigned-integer* ::= {*digit*}...

*date* ::=
    *mm date-separator dd date-separator yy* |
    *mmm date-separator dd date-separator yy* |
    *dd date-separator mmm date-separator yy* |
    *yyyy date-separator mm date-separator dd* |
    *yyyy date-separator mmm date-separator dd*

*mm* ::= *digit* [*digit*]

*dd* ::= *digit* [*digit*]

*yy* ::= *digit digit*

*yyyy* ::= *digit digit digit digit*

*mmm* ::= Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec

*date-separator* ::= - | / | .

*delimited-null* ::=
For delimited files, a NULL is represented by no data between two delimiters.

*fixed-width-null* ::= <SPACE>...
For fixed width files, a NULL is represented by spaces.

**See Also**

For All Users

Adding, Modifying, and Deleting a Text Data Source

## SQL Statements (Advanced)

See Also

The ODBC text driver supports most SQL statements and clauses in the ODBC minimum grammar. While the driver supports the grammar for reading data, it has limitations for writing data.

For information about ODBC SQL grammar limitations, and additional and driver-specific grammar supported, see the following topics:

For Advanced Users

Additional Supported ODBC SQL Grammar (Advanced)

Driver-specific ODBC SQL Grammar (Advanced)

Limitations to ODBC SQL Grammar (Advanced)

**See Also**

For Advanced Users

Data Types (Advanced)

## Additional Supported ODBC SQL Grammar (Advanced)

See Also

The ODBC text driver completely supports the following SQL statements and clauses in the Core and Extended ODBC grammar:

| **Core and Extended Grammar Supported** | **Comments** |
|---|---|
| Approximate numeric literal | Supported |
| AVG(*expression*), COUNT(*), MAX(*expression*), MIN(*expression*), and, SUM(*expression*) | See also the description of COUNT(*expression*) in Driver-specific ODBC SQL Grammar. |
| BETWEEN predicate | Supported |
| Correlation names are fully supported, including within the table list. | For example, in the following string, E1 is the correlation name for the table named Emp: `SELECT * FROM Emp E1` `WHERE E1.LastName = 'Smith'` |
| Exact numeric literal | Supported |
| [GROUP BY *column-name* [,*column-name*]...] | Supported |
| [HAVING *search-condition*] | Supported. |
| IN (*value-list*) | Implemented as specified in the ODBC core grammar.   For example: `SELECT * FROM Emp` `WHERE Dept` `IN ('Sales','Marketing')` |
| INSERT supports full paths for table names | For example: `INSERT INTO R:\MYDIR\EMP\MYTABLE` |

### See Also

For Advanced Users

Driver-specific ODBC SQL Grammar Supported (Advanced)

Limitations to ODBC SQL Grammar (Advanced)

## Driver-specific ODBC SQL Grammar (Advanced)

See Also

The ODBC text driver supports the following driver-specific ODBC SQL grammar:

| Driver-specific ODBC SQL Grammar | Comments |
|---|---|
| BETWEEN predicate | The syntax:<br>*expression1* BETWEEN *expression2* AND *expression3*<br>returns True only if *expression1* is greater than or equal to *expression2* and *expression1* is greater than or equal to *expression3*. |
| COUNT(*expression*) | Counts all non-null values for an expression across a predicate. This function behaves like other set functions, such as SUM, AVG, MIN, and MAX. For example, the following statement counts all the rows in Q where A+B does not equal NULL:<br>`SELECT COUNT(A+B) FROM Q` |
| Date arithmetic | The driver supports adding and subtracting an integer from a date column. The integer specifies the number of days to add or subtract. The driver also supports subtracting of one date column from another to return a number of days. |
| Date literals | The YYYY-MM-DD format is supported. |
| GROUP BY *expression-list* | GROUP BY supports an expression list as well as a column name. |
| ORDER BY *expression-list* | If the expression is a single integer literal, it is interpreted as the number of the column in the result set. Ordering is done on one of the result table columns. No ordering is allowed on set functions or an expression that contains a set function.<br>For example, in the following clauses the table is ordered by three key expressions: a+b, c+d, and e.<br>`SELECT * FROM emp`<br>`ORDER BY a+b,c+d,e` |
| ORDER BY with GROUP BY | ORDER BY can be performed on any expression in the GROUP BY *expression-list* or any column in the result set. |
| Outer Joins | A SELECT statement can contain a list of OUTER JOIN clauses. |
| Scalar Functions | Supported. |
| Table names that occur in the FROM clause of SELECT, after the INTO clause in INSERT, and after CREATE and DROP TABLE can contain a valid path, primary name, and filename extension. | Filename extensions are required for text databases. Use of a table name elsewhere in a SQL statement does not support the use of paths or extensions but will accept only the primary name (for example, C:\ABC\EMP). Correlation names can be used. You can access a table from the current directory:<br>`SELECT * FROM EMP.CSV`<br>`WHERE EMP.COL1 = 'aaa'`<br>You can also specify the full path of a file: |

| | |
|---|---|
| | `SELECT * FROM C:\TXTDIR\EMP.TXT` |
| USE [*drive*:]\*dir* | Sets the current database directory. *drive* is a valid drive name and *dir* is any valid MS-DOS directory name.<br>For example, the following changes the current directory to C:\DBDIR:<br>`USE C:\DBDIR`<br>USE is same as setting DataDirectory to an MS-DOS directory in your ODBC.INI file. |
| Using paths with CREATE INDEX and DROP INDEX | You can specify a path with a table name. For example:<br>`CREATE INDEX` *`index-name`*<br>`ON R:\MYDIR\EMP (A,B)`<br>`DROP INDEX R:\MYDIR\EMP.`*`ext`* |

**See Also**

For Advanced Users

Additional Supported ODBC SQL Grammar (Advanced)

Limitations to ODBC SQL Grammar Supported (Advanced)

## Outer Joins (Advanced)

See Also

The ODBC text driver extends the OUTER JOIN syntax to support nested outer joins. The OUTER JOIN syntax is:

*left-outer-join* ::=
    *table-reference* LEFT OUTER JOIN *table-reference*
   ON *search-condition*

*table-reference* ::=
   *table-name* | [(] *left-outer-join* [)]

where *table-name* can be a table name or a table name followed by a correlation name. For example, the following statement uses a three-way outer join to create a list of sales orders. For each sales order, all line numbers (if any) are listed, and for each line number, the part and description (if any) are listed.

```
SELECT Order.SONum,
  Line.LineNum,
  Part.PartNum,
  Part.Description
FROM Order LEFT OUTER JOIN
  (Line LEFT OUTER JOIN Part
  ON Line.PartNum=Part.PartNum)
  ON Order.SONum=Line.SONum
```

**Note** The rightmost ON corresponds to the leftmost LEFT OUTER JOIN.

**See Also**

For Advanced Users

Additional Supported ODBC SQL Grammar (Advanced)

Limitations to ODBC SQL Grammar (Advanced)

## Limitations to ODBC SQL Grammar (Advanced)

See Also

The ODBC text driver imposes the following limitations on the ODBC SQL grammar:

| Grammar | Limitation |
|---|---|
| AND predicates | A maximum of 300 supported. |
| CREATE INDEX, DELETE, DROP INDEX, and UPDATE | Not supported. |
| LIKE predicate | If data in a column is longer than 255 characters, the LIKE comparison will be based only on the first 255 characters. |
| Sort Keys | The maximum length of a sort key in a GROUP BY clause, ORDER BY clause, SELECT DISTINCT statement, or outer join is 255 bytes; the maximum length of all sort keys in a sort row is 65,500 bytes.<br>If the length of the data in a column is greater than 255 characters, sorting will be based on the first 255 characters. |

**See Also**

For Advanced Users

Additional Supported ODBC SQL Grammar (Advanced)

Driver-specific ODBC SQL Grammar (Advanced)

## Data Types (Advanced)

See Also

The following table shows how text data types are mapped to ODBC SQL data types. Note that not all ODBC SQL data types are supported by the ODBC text driver.

| **Text Data Type** | **ODBC Data Type** |
|---|---|
| CHAR WIDTH *colwidth* | SQL_CHAR |
| DATE *dateformat* WIDTH *colwidth* | SQL_DATE |
| FLOAT WIDTH | SQL_DOUBLE |
| INTEGER WIDTH *colwidth* | SQL_INTEGER |
| LONGCHAR WIDTH *colwidth* | SQL_LONGVARCHAR |

where *colwidth* is the maximum width of the data (number of characters) in the text file.

**Note** **SQLGetTypeInfo** returns ODBC data types. All conversions in Appendix D of the *Microsoft ODBC Programmer's Reference* are supported for the SQL data types listed above.

**See Also**

For Advanced Users

Data Type Limitations (Advanced)

For Programmers

Implementation Issues (Programming)

## Data Type Limitations (Advanced)

See Also

The ODBC text driver imposes the following limitations on the data types:

| Data Type Limitation | Description |
|---|---|
| CHAR columns | Maximum length (fixed length or delimited) is 255 bytes. |
| DATE format | MM-DD-YY (for example, 01-17-92)<br>MMM-DD-YY (for example, Jan-17-92)<br>DD-MMM-YY (for example, 17-Jan-92)<br>YYYY-MM-DD (for example, 1992-01-17)<br>YYYY-MMM-DD (for example, 1992-Jan-17) |
| Float columns | The maximum width includes the sign and decimal point.   In SCHEMA.INI, the width is denoted as follows:<br>14.083 is Float Width 6<br>-14.083 is Float Width 7<br>+14.083 is Float Width 7<br>14083. is Float Width 6<br>ODBC always returns 8 for float columns.<br>Float columns can also be in scientific notation, for example:<br>-3.04E+2 is Float Width 8<br>25E4 is Float Width 4<br>**Note**   Decimal and scientific notation cannot be mixed in a column. |
| Integer columns | The maximum width is 11 for delimited columns, and include the sign but no decimal point.   The length can be larger for fixed-length columns because of the blanks added.<br>In SCHEMA.INI, the width is denoted as follows:<br>14083 is Integer Width 5<br>0 is Integer Width 1<br>ODBC always returns 4 for integer columns. |
| LONGCHAR columns | Maximum length (fixed length or delimited) is 65,500 bytes. |
| SQL_C_TINYINT | When converting text data to the C data type SQL_C_TINYINT, numbers from 0 to 127 are converted correctly.   Numbers from 128 to 255 are converted to numbers from -128 to -1. Numbers less than 0 or greater than 255 cannot be converted.<br>When converting data from the C data type SQL_C_TINYINT to text data, numbers from 0 to 127 are converted correctly.   Numbers from -128 to -1 are converted to numbers from 128 to 255.<br>This occurs because SQL_C_TINYINT is signed, but the ODBC text driver uses unsigned single-byte integers. |

**See Also**

For Advanced Users

SQL Statements (Advanced)

## Error Messages (Advanced)

When an error occurs, the ODBC text driver returns the native error number, the SQLSTATE (an ODBC error code), and an error message.

### Native Error

For errors that occur in the data source, the ODBC text driver returns the native error returned to it by the ODBC File Library. For errors that are detected by the driver or the Driver Manager, the ODBC text driver returns a native error of zero.

### SQLSTATE

For errors that occur in the data source, the ODBC text driver maps the returned native error to the appropriate SQLSTATE. For errors that are detected by the driver or the Driver Manager, the ODBC text driver or Driver Manager generates the appropriate SQLSTATE.

### Error Message

For errors that occur in the data source, the ODBC text driver returns an error message based on the message returned by text. For errors that occur in the ODBC text driver or the Driver Manager, the ODBC text driver returns an error message based on the text associated with the SQLSTATE.

Error messages have the following format:

[*vendor*][*ODBC-component*][*data-source*]*message-text*

where the prefixes in brackets ([ ]) identify the location of the error. When the error occurs in the Driver Manager or Simba driver, *data-source* is not given. When the error occurs in the data source, the [*vendor*] and [*ODBC-component*] prefixes identify the vendor and name of the ODBC component that received the error from the data source.

The following table shows the error messages returned by the Driver Manager, Simba driver and Text ISAM:

| **Error Message** | **Error location** |
|---|---|
| [Microsoft][ODBC DLL]*message-text* | Driver Manager (ODBC.DLL) |
| [Microsoft][ODBC Single-Tier Driver]*message-text* | Simba Driver (SIMBA.DLL) |
| [Microsoft][ODBC Single-Tier Driver][ODBC File Library]*message-text* | Text ISAM (TXTISAM.DLL) |

## SQLGetInfo Return Values (Programming)

See Also

The following table lists the C language #defines for the fInfoType argument and the corresponding values returned by SQLGetInfo. This information can be retrieved by passing the listed C language #defines to SQLGetInfo in the fInfoType argument. Where **SQLGetInfo** returns a 32-bit bitmask, a vertical bar (|) represents a bitwise OR. For more information about the values return by **SQLGetInfo**, see the *Microsoft ODBC SDK Programmer's Reference, Version 1.0*.

| ***fInfoType* Value (#define)** | **Returned Value** |
|---|---|
| SQL_ACCESSIBLE_PROCEDURES | "N" |
| SQL_ACCESSIBLE_TABLES | "N" |
| SQL_ACTIVE_CONNECTIONS | 0 |
| SQL_ACTIVE_STATEMENTS | 0 |
| SQL_CONCAT_NULL_BEHAVIOR | 1 |
| SQL_CONVERT_BIGINT | 0 |
| SQL_CONVERT_BINARY | 0 |
| SQL_CONVERT_BIT | SQL_CVT_CHAR \| SQL_CVT_DOUBLE \| SQL_CVT_INTEGER \| SQL_CVT_LONGVARCHAR |
| SQL_CONVERT_CHAR | SQL_CVT_CHAR \| |

| | |
|---|---|
| | SQL_CVT_DATE \| SQL_CVT_DOUBLE \| SQL_CVT_INTEGER \| SQL_CVT_LONGVARCHAR |
| SQL_CONVERT_DATE | SQL_CVT_CHAR \| SQL_CVT_DATE |
| SQL_CONVERT_DECIMAL | 0 |
| SQL_CONVERT_DOUBLE | SQL_CVT_CHAR \| SQL_CVT_DOUBLE \| SQL_CVT_INTEGER \| SQL_CVT_LONGVARCHAR |
| SQL_CONVERT_FLOAT | SQL_CVT_CHAR \| SQL_CVT_DOUBLE \| SQL_CVT_INTEGER \| SQL_CVT_LONGVARCHAR |
| SQL_CONVERT_FUNCTIONS | SQL_FN_CVT_CONVERT |
| SQL_CONVERT_INTEGER | SQL_CVT_CHAR \| SQL_CVT_DOUBLE \| SQL_CVT_INTEGER \| SQL_CVT_LONGVARCHAR |
| SQL_CONVERT_LONGVARBINARY | 0 |
| SQL_CONVERT_LONGVARCHAR | SQL_CVT_CHAR \| SQL_CVT_LONGVARCHAR |
| SQL_CONVERT_NUMERIC | SQL_CVT_CHAR \| SQL_CVT_DOUBLE \| SQL_CVT_INTEGER \| SQL_CVT_LONGVARCHAR |
| SQL_CONVERT_REAL | SQL_CVT_CHAR \| SQL_CVT_DOUBLE \| SQL_CVT_INTEGER \| SQL_CVT_LONGVARCHAR |
| SQL_CONVERT_SMALLINT | SQL_CVT_CHAR \| SQL_CVT_DOUBLE \| SQL_CVT_INTEGER \| SQL_CVT_LONGVARCHAR |
| SQL_CONVERT_TIME | SQL_CVT_CHAR |
| SQL_CONVERT_TIMESTAMP | SQL_CVT_CHAR \| SQL_CVT_DATE |
| SQL_CONVERT_TINYINT | SQL_CVT_CHAR \| SQL_CVT_DOUBLE \| SQL_CVT_INTEGER \| SQL_CVT_LONGVARCHAR |
| SQL_CONVERT_VARBINARY | 0 |
| SQL_CONVERT_VARCHAR | 0 |
| SQL_CORRELATION_NAME | 2 |
| SQL_CURSOR_COMMIT_BEHAVIOR | 2 |
| SQL_CURSOR_ROLLBACK_BEHAVIOR | 0 |
| SQL_DATA_SOURCE_READ_ONLY | "Y" |
| SQL_DBMS_NAME | "TEXT" |
| SQL_DBMS_VER | "1.0" |
| SQL_DEFAULT_TXN_ISOLATION | 0 |
| SQL_DRIVER_NAME | "SIMBA.DLL" |
| SQL_DRIVER_VER | " 1.01.*nnnn*"   (*nnnn* specifies the |

| | |
|---|---|
| | build date.) |
| SQL_EXPRESSIONS_IN_ORDERBY | "Y" |
| SQL_FETCH_DIRECTION | SQL_FD_FETCH_NEXT |
| SQL_IDENTIFIER_CASE | 4 |
| SQL_IDENTIFIER_QUOTE_CHAR | """ (double quotation mark). |
| SQL_MAX_COLUMN_NAME_LEN | 30 |
| SQL_MAX_CURSOR_NAME_LEN | 18 |
| SQL_MAX_OWNER_NAME_LEN | 0 |
| SQL_MAX_PROCEDURE_NAME_LEN | 0 |
| SQL_MAX_QUALIFIER_NAME_LEN | 66 |
| SQL_MAX_TABLE_NAME_LEN | 12 |
| SQL_MULT_RESULT_SETS | "N" |
| SQL_MULTIPLE_ACTIVE_TXN | "N" |
| SQL_NON_NULLABLE_COLUMNS | 0 |
| SQL_NUMERIC_FUNCTIONS | SQL_FN_NUM_MOD |
| SQL_ODBC_API_CONFORMANCE | 1 |
| SQL_ODBC_SAG_CLI_CONFORMANCE | 1 |
| SQL_ODBC_SQL_CONFORMANCE | 0 |
| SQL_ODBC_SQL_OPT_IEF | "N" |
| SQL_OUTER_JOINS | "Y" |
| SQL_OWNER_TERM | "" |
| SQL_PROCEDURE_TERM | "" |
| SQL_PROCEDURES | "N" |
| SQL_QUALIFIER_NAME_SEPARATOR | "\\" (backslash) |
| SQL_QUALIFIER_TERM | "DIRECTORY" |
| SQL_ROW_UPDATES | "N" |
| SQL_SCROLL_CONCURRENCY | SQL_SCCO_READ_ONLY |
| SQL_SCROLL_OPTIONS | SQL_SO_FORWARD_ONLY |
| SQL_SEARCH_PATTERN_ESCAPE | "\\" (backslash) |
| SQL_SERVER_NAME | "TEXT" |
| SQL_STRING_FUNCTIONS | SQL_FN_STR_CONCAT \| SQL_FN_STR_LCASE \| SQL_FN_STR_LEFT \| SQL_FN_STR_LENGTH \| SQL_FN_STR_LOCATE \| SQL_FN_STR_LTRIM \| SQL_FN_STR_RIGHT \| SQL_FN_STR_RTRIM \| SQL_FN_STR_SUBSTRING \| SQL_FN_STR_UCASE |
| SQL_SYSTEM_FUNCTIONS | SQL_FN_SYS_DBNAME \| SQL_FN_SYS_USERNAME |
| SQL_TABLE_TERM | "TABLE" |
| SQL_TIMEDATE_FUNCTIONS | SQL_FN_TD_CURDATE \| SQL_FN_TD_CURTIME \| SQL_FN_TD_DAYOFMONTH \| SQL_FN_TD_DAYOFWEEK \| SQL_FN_TD_MONTH \| SQL_FN_TD_YEAR |
| SQL_TXN_CAPABLE | 0 |
| SQL_TXN_ISOLATION_OPTIONS | 0 |

**See Also**

For Advanced Users

- Data Types (Advanced)
- SQL Statements (Advanced)

For Programmers

- Scalar Functions (Programming)

## Scalar Functions (Programming)

The following table lists the scalar functions supported by the ODBC text driver.

| | | |
|---|---|---|
| CONCAT | LCASE | RIGHT |
| CONVERT | LEFT | RTRIM |
| CURDATE | LENGTH | SUBSTRING |
| CURTIME | LOCATE | UCASE |
| DATABASE | LTRIM | USER |
| DAYOFMONTH | MOD | YEAR |
| DAYOFWEEK | MONTH | |

For information about the arguments and return values of scalar functions, see Appendix G of the *Microsoft ODBC SDK Programmer's Reference.*

## ODBC API Functions (Programming)

See Also

The ODBC text driver supports all Core and Level 1 functions and the following Level 2 functions:

- SQLDataSources
- SQLMoreResults

These ODBC API functions have the following implementations with the ODBC text driver:

| Function | Description |
|---|---|
| **SQLDriverConnect** | The following keywords are supported in the connection string: **DSN**, **DBQ**, and **FIL**. |
| **SQLGetConnectOption** and **SQLSetConnectOption** | These functions support the SQL_ACCESS_MODE, SQL_CURRENT_QUALIFIER, SQL_OPT_TRACE, and SQL_OPT_TRACEFILE connection options. **SQLGetConnectOption** also supports the SQL_AUTOCOMMIT option. |
| **SQLGetCursorName** and **SQLSetCursorName** | These functions are supported, but cannot be used for positioned updates or deletes (for example, WHERE CURRENT OF *cursor-name*). |
| **SQLGetData** | This function can retrieve data from any column, whether or not there are bound columns after it and regardless of the order in which the columns are retrieved. |
| **SQLGetInfo** | **SQLGetInfo** supports a driver-specific information type, SQL_FILE_USAGE (65002). The returned value is a 16-bit integer that indicates how the driver directly treats files in a data source:<br>0 (SQL_FILE_NOT_SUPPORTED) = The driver is not a single-tier driver.<br>1 (SQL_FILE_TABLE) = A single-tier driver treats files in a data source as tables.<br>3 (SQL_FILE_QUALIFIER) = A single-tier driver treats files in a data source as a qualifier.<br>The ODBC text driver returns 1, since each text file is a table. |
| **SQLGetStmtOption** and **SQLSetStmtOption** | These functions support the SQL_MAX_LENGTH, SQL_MAX_ROWS, and SQL_NOSCAN statement options. |
| **SQLGetTypeInfo** | Only data type names returned by **SQLGetTypeInfo** can be used with CREATE statements. |
| **SQLMoreResults** | **SQLMoreResults** always returns SQL_NO_DATA_FOUND. It cannot return additional results. |

| | |
|---|---|
| **SQLSpecialColumns** | This function always returns SQL_SUCCESS and an empty result set. |
| **SQLTables** | The table names returned by **SQLTables** have no filename extensions. |
| **SQLTransact** | This function supports COMMIT, but not ROLLBACK. |

**See Also**

For Advanced Users

- Error Messages (Advanced)

For Programmers

- Implementation Issues (Programming)

## Implementation Issues (Programming)

See Also

The following implementation-specific issues might affect the use of the ODBC text driver.

### Arithmetic Errors

The Text driver evaluates the WHERE clause in a SELECT statement as it fetches each row. If a row contains a value that causes an arithmetic error, such as divide-by-zero or numeric overflow, the driver returns all rows, but returns errors for columns with arithmetic errors. When inserting or updating, however, the Text driver stops inserting or updating data when the first arithmetic error is encountered.

### Closing Open Tables (Files)

Calling **SQLFreeStmt** with the SQL_CLOSE option changes the statement state but does not close the tables used by the *hstmt*. To close the tables currently used by *hstmt*, you must call **SQLFreeStmt** with the SQL_DROP option.

In the following example, when **SQLFreeStmt** is called, the emp and dept tables remain open:

```
SQLPrepare(hStmt,"SELECT * FROM emp,dept
WHERE emp.dept = dept.dept_id",SQL_NTS);

SQLExecute(hStmt);

/*.SQLFetch until NO_DATA_FOUND

SQLFreeStmt(hStmt,SQL_CLOSE);

SQLPrepare(hStmt,"SELECT * FROM emp",SQL_NTS);
```

**Note** Each file used by the ODBC text driver requires a file handle. Because tables (files) remain open until **SQLFreeStmt** is called with the SQL_DROP option, reusing an *hstmt* for different tables without dropping it can result in an error caused by attempting to open too many files.

### Creating and Opening Tables

A new table is created using the format specified in ODBC.INI. If not specified, tables are created in CSVDELIMITED format. By default, INTEGER columns default to 11 characters and FLOAT columns default to 22 characters. DATE columns use the YYYY-MM-DD format. CHAR and LONGCHAR columns are the width specified in the CREATE statement.

### Sorting with DISTINCT, GROUP BY, ORDER BY

DISTINCT, GROUP BY, and ORDER BY always result in a sort. If indexes are found, data is dynamically fetched and the sort is based using those indexes. If indexes are not found, a temporary table is created from the data and the sort occurs on the temporary table. This type of sort is not based on dynamic data since the temporary table is built from data found in the original data file at statement execution time.

**See Also**

For All Users

Using the ODBC Text Driver

**API**

Application programming interface. A set of routines that an application, such as Microsoft Access, uses to request and carry out lower-level services.

**character set**

A character set is a set of 256 letters, numbers, and symbols specific to a country or language. Each character set is defined by a table called a code page. An OEM (Original Equipment Manufacturer) character set is any character set except the ANSI character set. The ANSI character set (code page 1007) is the character set used by Microsoft Windows.

**conformance level**

Some applications can use only drivers that support certain levels of functionality, or conformance levels. For example, an application might require that drivers be able to prompt the user for the password for a data source. This ability is part of the Level 1 conformance level for the application programming interface (API).

Every ODBC driver conforms to one of three API levels (Core, Level 1, or Level 2) and one of three SQL grammar levels (Minimum, Core, or Extended). Drivers may support some of the functionality in levels above their stated level.

For detailed information about conformance levels, programmers should see the *Microsoft ODBC SDK Programmer's Reference*.

**data source**

A data source includes the data a user wants to access and the information needed to get to that data. Examples of data sources are:

- A SQL Server database, the server on which it resides, and the network used to access that server.
- A directory containing a set of dBASE files you want to access.

**DBMS**

Database management system. The software used to organize, analyze, search for, update, and retrieve data.

**DDL**

Data definition language. Any SQL statement that can be used to define data objects and their attributes. Examples include CREATE TABLE, DROP VIEW, and GRANT statements.

**DLL**

Dynamic-link library. A set of routines that one or more applications can use to perform common tasks. The ODBC drivers are DLLs.

**DML**

Data manipulation language. Any SQL statement that can be used to manipulate data. Examples include UPDATE, INSERT, and DELETE statements.

**ODBC**

Open Database Connectivity.   A Driver Manager and a set of ODBC drivers that enable applications to access data using SQL as a standard language.

**ODBC Driver Manager**

A dynamic-link library (DLL) that provides access to ODBC drivers.

**ODBC driver**

A dynamic-link library (DLL) that an ODBC-enabled application, such as Microsoft Excel, can use to gain access to a particular data source. Each database management system (DBMS), such as Microsoft SQL Server, requires a different driver.

**SQL**

Structured Query Language. A language used for retrieving, updating, and managing data.

**SQL statement**

A command written in Structured Query Language (SQL); also known as a query. An SQL statement specifies an operation to perform, such as SELECT, DELETE, or CREATE TABLE; the tables and columns on which to perform that operation; and any constraints to that operation.

**translation option**

An option that specifies how a translator translates data. For example, a translation option might specify the character sets between which a translator translates character data. It might also provide a key for encryption and decryption.

**translator**

A dynamic-link library (DLL) that translates all data passing between an application, such as Microsoft Access, and a data source. The most common use of a translator is to translate character data between different character sets. A translator can also perform tasks such as encryption and decryption or compression and expansion.