1

DBD::XBase

Version

Version 0.130.

Author and Contact Details

The driver author is Jan Pazdziora. He can be contacted at *adelton@fi.muni.cz* or *via* the *dbi-users* mailing list.

Supported Database Versions and Options

The `DBD::XBase` module supports dBaseIII and IV and Fox* flavors of *dbf* files, including their *dbt* and *fpt* memo files.

Connect Syntax

The `DBI->connect()` Data Source Name, or *DSN*, should include the directory where the *dbf* files are located as the third part.

```
dbi:XBase:/path/to/directory
```

It defaults to current directory.

There are no driver specific attributes for the `DBI->connect()` method.

Numeric Data Handling

`DBD::XBase` supports generic `NUMBER(p,s)`, `FLOAT(p,s)` and `INTEGER(l)` types. The maximum scale and precision is limited by Perl's handling of numbers. In the *dbf* files, the numbers are stored as ASCII strings, binary integers, or floats.

Existing *dbf* files come with the field types defined in the *dbf* file header. Numeric types can be either stored as ASCII string or in some binary format. `DBD::XBase` (via `XBase.pm`)

parses this information and reads and writes the fields in that format.

When you create a new *dbf* file (*via* `CREATE TABLE`), the numeric fields are always created in the traditional XBase way, as an ASCII string. (The `XBase.pm` module offer more control over this.)

Numeric fields are always returned as Perl numeric values, not strings. Consequently, numbers outside of Perl's valid range are not possible. This restriction might be withdrawn in the future.

String Data Handling

`DBD::XBase` has `CHAR(length)` and `VARCHAR(length)` data types.

The maximum length is 65535 characters for both types*.

```
* This limit if effective even though the older dBase's only allowed
  254 characters. Therefore, created dbf files might not be portable
  to other XBase compatible software.
```

Both CHAR *and* VARCHAR are blank padded so *ChopBlanks* applies to both.

Data with the 8th bit set are handled transparently. No national language character set conversions are done. Since the string types can store binary data, Unicode strings can be stored.

Date Data Handling

`DBD::XBase` supports these date and time types:

```
DATE
DATETIME
TIME
```

The DATE type holds an eight character string in the format `YYYYMMDD`. Only that format can be used for input and output. `DBD::XBase` doesn't check for validity of the values.

The DATETIME and TIME types internally store a 4 byte integer day value and a 4 byte integer seconds value (counting 1/1000's of a second). `DBD::XBase` inputs and outputs these types using a floating point unix-style ''seconds-since-epoch'' value (possibly with decimal part). This might change in the future.

There is no way to get the current date/time, and no SQL date time functions are supported. There is also no concept of time zones.

LONG/BLOB Data Handling

DBD::XBase supports a MEMO data type. BLOB can be used as an alias for MEMO. Strings up to 2 GB can be stored in MEMO fields (for all types of XBase memo files).

With dBaseIII *dbt* files, the memo field cannot contain a 0x1A byte. With dBaseIV and Fox* dbt/fpts, any character values can be stored.

No special handling is required for fetching or inserting MEMO fields. The *LongReadLen* and *LongTruncOk* attributes are currently ignored.

Other Data Handling issues

The DBD::XBase driver supports the type_info() method.

DBD::XBase supports automatic conversions between data types wherever it's reasonable.

Transactions, Isolation and Locking

DBD::XBase currently does not support transactions and does not lock the tables it is working on.

No-Table Expression Select Syntax

You can't select a constant expression using DBD::XBase. Only table field names or *, for all columns, can be selected.

Table Join Syntax

DBD::XBase does not support table joins.

Table and Column Names

The XBase format stores each table as a distinct file. Memo fields are stored in an additional file. The table names are limited by the filesystem's maximum filename length. They are stored and treated as entered. The case sensitivity depends on the filesystem that the file is stored on.

Column names are limited to 11 characters. They are stored as uppercase, but are not case sensitive.

Table and field names have to start with letter. Any combination of letters, digits, and underscores may follow. National character sets can be used.

DBD::XBase does not support putting quotes around table or column names.

Case Sensitivity of LIKE Operator

The LIKE operator is not case sensitive. There is currently no case sensitivity operator.

Row ID

`DBD::XBase` does not support a ''row ID'' pseudocolumn.

Automatic Key or Sequence Generation

`DBD::XBase` does not support automatic key generation or sequence generators owing to the limitations of the XBase format.

Automatic Row Numbering and Row Count Limiting

`DBD::XBase` does not support a row-numbering pseudocolumn.

Parameter Binding

Parameter binding is implemented in the driver and supports the `?` style of placeholder. The `:1` placeholder style is not (yet) supported.

The TYPE attribute to `bind_param()` is ignored. Consequently, unsupported values of the TYPE attribute do not currently generate a warning.

Stored Procedures

Stored procedures are not applicable in the XBase format.

Table Metadata

`DBD::XBase` supports the `table_info` method.

There is no way to get detailed information about the columns of a table (at the moment) other than doing a `SELECT * FROM table` and using the *NAME* and *TYPE* attributes of the statement handle.

Keys and Indexes are not supported.

Driver-specific Attributes and Methods

`DBD::XBase` has just one driver-specific attribute and that is valid for both database and statement handles.

xbase_ignorememo

> Ignore memo files and thus don't fail to read a table where the memo file is missing or corrupt.

DBD::XBase has no generally useful private methods.

Positioned updates and deletes

DBD::XBase does not support positioned updates or deletes.

Differences from the DBI Specification

DBD::XBase has no known significant differences in behavior from the current DBI specification.

Note that DBD::XBase does not fully parse the statement until it is executed. Thus attributes like *$sth->{NUM_OF_FIELDS}* are not available until after `$sth->execute()` has been called. This is valid behavior but is important to note when porting applications written originally for other drivers.

URLs to More Database/Driver Specific Information

Very comprehensive information about the XBase format, along with many references, can be found at:

```
http://www.e-bachmann.dk/docs/xbase.htm
```

Concurrent use of Multiple Handles

DBD::XBase supports an unlimited number of concurrent database connections to one or more databases.

It also supports the preparation and execution of a new statement handle while still fetching data from another statment handle associated with the same database handle.