1

DBD::Empress and DBD::EmpressNet

Version

Version 0.51.

These drivers use the same Perl interface but use a different underlying database interface. `DBD::Empress` is for direct access of databases, whilst `DBD::EmpressNet` is for distibuted database connected *via* the Empress Connectivity Server (referred to in Empress v8.10 and earlier as the Empress ODBC server).

Author and Contact Details

The driver was written by Steve Williams. He can be contacted at *swilliam@empress.com*.

Supported Database Versions and Options

`DBD::Empress` supports Empress V6.10 and later.

Connect Syntax

The `DBI->connect()` Data Source Name, or *DSN* can be one of the following:

```
dbi:Empress:physical_database
dbi:EmpressNet:logical_database
dbi:EmpressNet:SERVER=server_name;DATABASE=physical_database;PORT=port_number
```

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

Numeric Data Handling

Empress RDBMS supports the following numeric data types:

```
DECIMAL(p,s)      1 to 15 digits
DOLLAR(p,type)    1 to 13 digits
REAL              Typically 4-byte single precision float
FLOAT(p)          Typically 4 or 8-byte float as required
LONGFLOAT         Typically 8-byte double precision float
SHORTINTEGER             -127 to 127
INTEGER                -32767 to 32767
LONGINTEGER       -2147483647 to 2147483647
```

The the DBD driver supports Empress Generic datatypes only. This means that all data for a specific group will be retrieved as the same data type. For example, SHORTINTEGER, INTEGER, and LONGINTEGER will all be retrieved as LONGINTEGER.

`DBD::Empress` always returns all numbers as strings.

String Data Handling

Empress RDBMS supports the following string data types:

```
CHAR (length, type)
NLSCHAR (length, type)
TEXT (display_length, primary, overflow, extent)
NLSTEXT (display_length, primary, overflow, extent)
```

All arguments have default values. See Empress SQL Reference (A4) for details. The maximum size for all string types is typically 2**31-1 bytes (2GB). None of the string types are blank padded.

NLSCHAR and NLSTEXT are can be used for storage of 8 bit and multibyte characters but UTF-8 is not currently supported.

Strings can be concatenated using the `s1 CONCAT(s2)` SQL function.

Date Data Handling

Empress RDBMS supports the following date time data types:

```
DATE(t)            = 0000-01-01 to 9999-12-31 at 1 day resolution
TIME(t)            = 1970-01-01 to 2035-12-31 at 1 second resolution
MICROTIMESTAMP(t)  = 0000-01-01 to 9999-12-31 at 1 microsecond resolution
```

The (t) is the format type for default output. This is one of the nine types defined in the section on date/time formats.

Empress supports 9 formats for date/time types:

Type	Date	Time	MicroTimestamp
0	yyyymmdd	yyyymmddhhmmss	yyyymmddhhmmssffffff
1	dd aaaaaaaaa yyyy	dd aaaaaaaaa yyyy hh:mm:ss	dd aaaaaaaaa yyyy hh:mm:ss.fffff
2	aaaaaaaaa dd, yyyy	aaaaaaaaa dd, yyyy hh:mm:ss	aaaaaaaaa dd, yyyy hh:mm:ss.fffff
3	mm/dd/yy	mm/dd/yy hh:mm:ss	mm/dd/yy hh:mm:ss.ffffff
4	dd/mm/yy	dd/mm/yy hh:mm:ss	dd/mm/yy hh:mm:ss.ffffff
5	dd aaa yy	dd aaa yy hh:mm:ss	dd aaa yy hh:mm:ss.ffffff
6	aaa dd, yy	aaa dd, yy hh:mm:ss	aaa dd, yy hh:mm:ss.fffff
7	mm/dd/yyyy	mm/dd/yyyy hh:mm:ss	mm/dd/yyyy hh:mm:ss.ffffff
8	dd/mm/yyyy	dd/mm/yyyy hh:mm:ss	dd/mm/yyyy hh:mm:ss.ffffff

For input, the DBD drivers recognize all formats.

The date part for all types is not optional. If you specify a value without a time component, the default time is 00:00:00 (midnight). If only two digits of the year are input then the century pivots on the Empress variable MSDATELIMIT. For Empress v8.*xx* and above the default for this is 1950. Earlier versions of Empress defaulted to 1900.

Empress accepts any of the 9 specified types as input. The only limitation is that you cannot insert a four digit year into a date type that uses a two digit format. It always uses MSDATELIMIT for input dates.

For output, `DBD::Empress` uses just `yyyymmddhhmmssffffff` and `DBD::EmpressNet` uses just `yyyy-mm-dd hh:mm:ss.ffffff`. Empress does not support changing of the default display formats. It is not possible to format a date time value in other styles for output. The best approach is to select the components of the date time, using SQL functions like `DAYOF(d)` and `MONTHOF(d)`, and format them using Perl.

The current date/time at the server, can be obtained using the NOW or TODAY pseudo constants. NOW returns the current date and time. TODAY returns the date portion only.

Date and time arithmetic can be done using the Empress date/time operators. For example:

```
NOW + 2 MINUTES + 5 SECONDS
TODAY - 3 DAYS
```

Empress provides a wide range of date functions including DAYOF, MONTHOF, YEAROF, HOUROF, MINUTEOF, SECONDOF, WEEKOFYEAR, DAYNAME, DAYOFWEEK, DAYOFYEAR, and DATENEXT.

The following SQL expression:

```
'1 jan 1970' + unix_time_field SECONDS
```

would convert to a local time from 1 Jan 1970, but the GMT base cannot be generated directly.

The number of seconds since 1 Jan 1970 for date granularity can be obtained for the local time zone (not GMT) using:

```
    (date_field - '1 jan 1970') * 86400
```

Empress does no automatic time zone adjustments.

LONG/BLOB Data Handling

Empress RDBMS supports the following LONG data types:

TEXT	Variable length 7-bit character data
NLSTEXT	As TEXT but allows 8-bit characters
BULK	User Interpreted (Byte Stream)

The maximum size for all these types is typically 2**31-1 bytes (2GB). None of the types are passed to and from the database as pairs of hex digits.

LongReadLen works as defined for `DBD::EmpressNet` but is ignored for `DBD::Empress`. The maximum *LongReadLen* is limited to 2GB typically. *LongTruncOk* is not implemented.

No special handling is required for binding LONG/BLOB data types. The TYPE attribute is currently not used when binding parameters. The maximum length of `bind_param()` parameters is limited by the capabilities of the OS or the size of the C `int`, whichever comes first.

Other Data Handling issues

The `type_info()` method is not supported.

Empress automatically converts strings to numbers and dates, and numbers and dates to strings, as needed.

Transactions, Isolation and Locking

`DBD::Empress` support transactions. The default isolation level is Serializable.

Other transaction isolation levels are not explicitly supported. However Read Uncommited is supported on a single query basis. This is activated by adding the BYPASS option into each SQL statement; for example:

```
    SELECT BYPASS * FROM table_name
```

Record level locking is the default. Read locks do not block other read locks, but read locks block write locks, and write locks block all other locks. Write locks can be bypassed for read using the BYPASS option.

When in transaction mode (AutoCommit off), selected rows are automatically locked against update unless the BYPASS option is used in the SELECT statement.

The `LOCK TABLE table_name IN lock_mode` statement can be used to apply an explicit lock on a table. Lock mode can be EXCLUSIVE or SHARE. SHARE requires the user to have SELECT or UPDATE privileges on the table. EXCLUSIVE requires the user to have UPDATE, INSERT, or DELETE privileges. Locks are only valid for the duration of a transaction.

No-Table Expression Select Syntax

To select a constant expression—that is, one that doesn't involve data from a database table or view—you need to select from a real table. Use the DISTINCT keyword in the query to prevent multiple values being returned; or, better yet, write the query to only match one row in a table you know exists, such as a system catalog.

Table Join Syntax

For outer joins, the Empress keyword OUTER should be placed before the `table(s)` that should drive the outer join. For example:

```
SELECT customer_name, order_date
FROM OUTER customers, orders
WHERE customers.cust_id = orders.cust_id;
```

This returns all the rows in the customers table that have no matching rows in the orders table. Empress returns c<NULL> for any select list expressions containing columns from the orders table.

Table and Column Names

The names of Empress identifiers, such as tables and columns, cannot exceed 32 characters in length. The first character must be a letter, but the rest can be any combination of letters, numerals, and underscores (_). Empress table/column names are stored as defined. They are case sensitive.

Empress tables and fields can contain most ASCII characters (except `$` and `?`) if they are quoted. For example:

```
SELECT field_name "a simple *&" FROM table_name "what a dumb ^"
```

However, this practice is not recommended.

Any ISO-Latin characters can be used in the base product. Specific products for other languages, such as Japanese, can handle those character sets.

Case Sensitivity of LIKE Operator

The LIKE operator is case sensitive. The MATCH operator is case insensitive.

Row ID

A table row identifier can be referenced as MS_RECORD_NUMBER. It can be treated as a string during fetch. But it must be treated as an integer when used in a WHERE clause. It is only useful for explicit fetch; inequalities are not allowed.

```
SELECT * FROM table_name WHERE MS_RECORD_NUMBER = ?
```

Automatic Key or Sequence Generation

Empress has no "auto increment" or "system generated" key mechanism, and does not support sequence generators.

Automatic Row Numbering and Row Count Limiting

There is no pseudocolumn that sequentially numbers the rows fetched by a select statement.

Parameter Binding

Parameter binding is directly suported by Empress. Only the default `?` style of placeholders is supported.

`DBD::Empress` recognizes the `bind_param()` TYPE attribute SQL_BINARY. All other types are automatically bound correctly without TYPE being used. Unsupported types are ignored without warning.

Stored Procedures

`DBD::Empress` does not explicitly support stored procedures. Implicit support is available for stored procedures in SQL statements, e.g.:

```
$sth->prepare("SELECT func(attribute) FROM table_name");
```

Table Metadata

`DBD::Empress` does not support the `table_info` method.

The SYS_ATTRS and SYS_TABLES system tables can be used to obtain detailed information about the columns of a table. For example:

```
SELECT * FROM sys_attrs
WHERE attr_tabnum = (SELECT tab_number FROM sys_tables WHERE tab_name='x')
```

However, this requires SELECT privileges on these system tables.

Detailed information about indices or keys cannot currently be easily retrieved though `DBD::Empress`. It is possible, though difficult, to interpret the contents of the system tables to obtain this information.

Driver-specific Attributes and Methods

`DBD::Empress` has no significant driver-specific handle attributes or private methods.

Positioned updates and deletes

`DBD::Empress` does not currently support positioned updates and deletes.

Differences from the DBI Specification

`DBD::Empress` was written against DBI 0.89 and has not been updated yet. The DBD was conformant to the behavior at that time. Newer features of DBI that require driver changes may not behave as specified.

Note the transaction limitation of `DBD::Empress` (not `DBD::EmpressNet`) outlined in the section on below.

URLs to More Database/Driver Specific Information

```
http://www.empress.com
```

Concurrent use of Multiple Handles

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

`DBD::Empress` also supports multiple concurrent database connections to one or mode databases. However, these connections are simulated, and there are therefore a number of limitations. Most of these limitations are associated with transaction processing: 1) Auto-commit must be on or off for all connections; and 2) Switching processing from one database to another automatically commits any transactions on the first database.

`DBD::Empress` supports the preparation and execution of a new statement handle while still fetching data from another statment handle associated with the same database handle.