Database field type

From Free Pascal wiki

| **English (en)** | **español (es)** | **français (fr)** | **中文（中国大陆）(zh_CN)** |

Contents

Overview

FCL-DB database fields can have different data types. In datasets, a subset of these types is available. What types are supported differ per dataset type.

Types

Currently, the following field types are defined: See FCL field type documentation (http://lazarus-ccr.sourceforge.net/docs/fcl/db/tfieldtype.html) To do: add assignment information (e.g. can you use .AsString) and additional information below.

Field type	Description
ftADT	
ftArray	represents an Interbase 6/Firebird array datatype (array of simple datatypes like varchar or integer). However, SQLDB does not support the array datatype currently.
ftAutoInc	an autoincrementing integer field.
ftBCD	a binary coded decimal floating point value
ftBlob	a binary large object (BLOB), meant to store arbitrary binary data.
ftBoolean	a boolean value (yes/no).
ftBytes	presumably a fixed number of bytes stored as-is. Needs to have its Size property set to work.
ftCurrency	a format to precisely store currency values.
ftCursor	
ftDataSet	presumably meant to store an entire dataset (possibly to implement master/detail table).
ftDate	a date without time information.
ftDateTime	date and time information.
ftDBaseOle	presumably meant to store OLE objects in a DBase database. Needs to have its Size property set to work.
ftFMTBcd	a form of Binary Coded Decimal (BCD) number field. Needs to have its Size property set to work.
ftFixedChar	a fixed width character field, similar to a Pascal shortstring. Needs to have its Size property set to work.
ftFixedWideChar	a fixed width multibyte character field. Needs to have its Size property set to work.
ftFloat	a floating point numeric type.
ftFmtMemo	Needs to have its Size property set to work.
ftGraphic	Needs to have its Size property set to work.
ftGuid	a fie0ld used to store a GUID (Globally Unique Identifier). With the current code, this field needs to have its Size property set to 38.
ftIDispatch	
ftInteger	an integer field
ftInterface	
ftLargeint	a field that contains an integer, stores more bytes than an integer and therefore has a larger range.
ftMemo	stores a variable amount of string/text data; needs no size set.
ftOraBlob	presumably stores Oracle BLOB.
ftOraClob	presumably stores Oracle CLOB: an Oracle data type that can hold up to 4 GB of data [1] (http://www.orafaq.com/wiki/CLOB)
ftParadoxOle	presumably meant to store OLE objects in a Paradox database.
ftReference	
ftSmallint	an integer type field with less bytes than ftInteger.
ftString	stores string data; needs to have its Size property set to the maximum number of characters possible in that field.

Field type	Description
ftTime	stores time-only data.
ftTimeStamp	stores date/time data. Probably equivalent to ftDateTime
ftTypedBinary	some kind of blob-like field?
ftUnknown	
ftVarBytes	presumably a variant with byte/binary data?
ftVariant	presumably meant to store variant data.
ftWideMemo	an ftMemo with widestring (UTF16) characters.
ftWideString	an ftString with widestring (UTF16) characters.
ftWord	presumably stores an integer value

Size, DataSize and Unicode

Note that for string type fields, Size indicates the number of characters that can be stored. As indicated in FPC Unicode support#Introduction, FPC up to and including 2.6 only deals with **ANSI/ASCII** single byte characters; it does not support Unicode/UTF8/UTF16/Unicodestring characters.

The read-only property DataSize indicates the field size in bytes.

If you use multibyte characters (e.g. UTF8 or UTF16/Unicodestring encoded), DataSize and Size do not mean the same thing. If you use only ANSI/ASCII characters, DataSize and Size are effectively the same thing.

Defining types in your dataset

Todo: write me. Explain various ways of doing things (TFieldDef, TFields) and which dataset supports which methods.

Assigning and retrieving values

Once you have the fields in your dataset defined, you can assign and retrieve data like this - suppose you have a dataset called FTestDataset:

```
 FTestDataset.Open; //Open for viewing/editing/inserting
 FTestDataset.Append;
 FTestDataset.Fieldbyname('YourFieldName').Asstring := 'This is my field data'; //Suppose field YourFieldName
is a memo field
 FTestDataset.Post; //"Commit"/save changes in the record to the dataset.
 writeln('YourFieldName has data:' + FTestDataset.Fieldbyname('YourFieldName').Asstring
 { Retrieve the field value of the current record. Because we haven't moved the record, we should get what we
just entered }
```

For text/memo fields, use the AsString method. For date/time fields, use the AsDateTime method. For binary fields, I suppose you can use the AsString method - *but there must be another way, too*

See also

- Databases

Retrieved from "https://wiki.freepascal.org/index.php?title=Database_field_type&oldid=103357"
