1 Disclaimer

This software is delivered as it is. The author assumes no liability for damages, direct or consequential, which may result from it's use.

2 Copyright / Licensing

The software is owned by gig mbh berlin (www.gig-mbh.de).

Two different licenses are available:

1. Free License

Everyone who wants to use the free license has to register with his full name and address via support@gig-mbh.de.

Every software where parts of our free software were used for development has to be free also including source code.

If you derive anything from our software it must be clearly stated that it was derived from it.

Full source code is included.

2. Extended License

Licenses have to be bought by a per developer basis. Site licenses would be available on demand.

Applications built with this software could be deployed without royalty fees. They can be sold and don't need to include source code.

Distribution of a derived version of our software is only allowed with the explicit agreement of the author.

Full source code is included.

3 Support

Support is available via email at support@gig-mbh.de for free but it need not remain so in the future.

4 Introduction

This document describes the function of the component TIBDataProviderEC.

This class implements the interface defined in TDataProviderEC to synchronise our TMemTableEC dataset component with Interbase or Firebird database systems. Therefore it uses core components of InterbaseExpress (IBX) which also has to be included in your project.

For working properly you have to specify a separate query for every task you want the provider to handle (retrieve, modify, delete, insert, refresh) and secondly to specify field assignments where you specify which filelds in the database belong to which fields in the TMemTableEC component.

The component is completely written in C++ and was developed under C++Builder 5 Pro but it should be usable on C++ Builder 6 if compiled in it’s environment.

Questions, bug reports , enhancement requests, suggestions for improving the docs and comments should be send to support@gig-mbh.de.

5 Methods

Open

Description:	Makes the dataprovider active. That means a connection to the database server is established.
Prototype :	void __fastcall Open(void)
Parameters:	none
Return values :	none
Type:	public

Close

Description:	Makes the dataprovider inactive. That means the connection to the database server is closed.
Prototype :	void __fastcall Open(void)
Parameters:	none
Return values :	none
Type:	public

6 Properties

Active

Description: See methods Open and Close.

Definition : __property bool Active = {read=FActive, write=SetActive,
default=false}

Type: published

Database

Description: Specifies a TIBDatabase components (IBX) which will be used to establish a connection to the database.

Definition : __property TIBDatabase *Database = {read=GetDatabase,
write=SetDatabase, default=NULL}

Type: published

Transaction

Description: Specifies a TIBTransaction components (IBX) which will be used to handle transactions for the actions performed.

Definition : __property TIBTransaction *Transaction = {read=GetTransaction,
write=SetTransaction, default=NULL}

Type: published

AutoEndTransaction

Description: Whenever an action is performed by the data provider a transaction is started if not currently done. When the action of the provider has finished it ends the transaction if this property is set to true otherwise the transaction stays active. If the provider is responsible for ending transactions it is guaranteed that no transaction stays active after the provider has performed a set of modify or retrieve actions. If anything went wrong the transaction is allways rolled back otherwise committed.

Definition :

```
__property bool AutoEndTransaction = {read=FAutoEndTransaction,
                                     write=FAutoEndTransaction, default=true}
```

Type: published

FieldAssignments

Description: This property specifies the releation between TMemTableEC and the database fields of the different queries. For every field assignment you specify one line in the following syntax: <memtabfield>;<dbqueryfield>. It is allways assumed that database fieldnames and query variables which belong together also have given the same name.
For update actions you could refer to the unchanged (old) values by inserting the prefix „OLD_” for <memtabfield> and „NEW_” for the changed (new) value. Without prefix the new value is used. Do not use TMemTableEC field names which natively begin with these prefixes as they would confuse the old/new record switching mechanism of the data provider. If you want to have fields in the TMemTabEC data set which have no direct related field in your database but are derived from them in any way you could insert a so called virtual field (simply a field/parameter name which does no exist in any query) and assign/get the transformed values in the data provider's SetMemTabFieldValue and SetDbFieldValue event handlers.

Definition :

```
__property TIBDPFields *FieldAssignment =
                 {read=FFieldAssignment, write=SetFieldAssignment}
```

Type: published

SelectSQL

Description: SQL query for retrieving records. This property is necessary for every data provider.

Definition :
```
__property TStrings *SelectSQL = {read=GetSelectSQL,
                                  write=SetSelectSQL}
```

Type: published

ModifySQL

Description: SQL query for modifying a record. This property is only necessary if you want to apply updates from the TMemTableEC dataset to the database. You should use the unchanged („OLD_" prefixed) field value of a column which uniquely identifies a record row in the WHERE clause of this query. If you want to be sure that no other columns have been changed by concurrent users/transactions meanwhile you could add their unchanged values in the WEHRE clause as well.

Definition :
```
__property TStrings *ModifySQL = {read=GetModifySQL,
                                  write=SetModifySQL}
```

Type: published

InsertSQL

Description: SQL query for inserting new records. This property is only necessary if you want to insert new records from the TMemTableEC dataset to the database.

Definition :
```
__property TStrings *InsertSQL = {read=GetInsertSQL,
                                  write=SetInsertSQL}
```

Type: published

DeleteSQL

Description:	SQL query for deleting records. This property is only necessary if you want to delete records from the database which have been removed from the TMemTableEC.
Definition :	__property TStrings *DeleteSQL = {read=GetDeleteSQL, write=SetDeleteSQL}
Type:	published

RefreshSQL

Description:	SQL query for re-retrieving records after they have been modified or inserted. This property is only necessary if you set the RereadChanges property of the TMemTableEC dataset to true. You should include a column which uniquely identifies a record row in the WHERE clause of this query.
Definition :	__property TStrings *RefreshSQL = {read=GetRefreshSQL, write=SetRefreshSQL}
Type:	published

SelectQry

Description:	Pointer to the TIBSQL component which handles the SelectSQL query statement.
Definition :	__property TIBSQL *SelectQry = {read=FSelectQry}
Type:	public

ModifyQry

Description:	Pointer to the TIBSQL component which handles the ModifySQL query statement.
Definition :	__property TIBSQL *ModifyQry = {read=FModifyQry}
Type:	public

InsertQry

Description: Pointer to the TIBSQL component which handles the InsertSQL query statement.

Definition : __property TIBSQL *InsertQry = {read=FInsertQry}

Type: public

DeleteQry

Description: Pointer to the TIBSQL component which handles the DeleteSQL query statement.

Definition : __property TIBSQL *DeleteQry = {read=FDeleteQry}

Type: public

RefreshQry

Description: Pointer to the TIBSQL component which handles the RefreshSQL query statement.

Definition : __property TIBSQL *RefreshQry = {read=FRefreshQry}

Type: public

7 Events

SetMemTabFieldValue

Description: This event is fired whenever the value from a query field was assigned to a field of the TMemTableEC dataset. Here it is possible to change the assigned value. If you have stated virtual fields in the FieldAssignments property, the assignment of their values can be made inside this event handler. The FieldName parameter contains the TMemTableEC field name of the FieldAssignments property.

Handler :

```
void __fastcall (__closure *TIBDPFieldEvent)(TField *memtabfield,
                        TIBXSQLVAR *dbfield,
                         const AnsiString  &FieldName,
                        TIBXSQLDA *rec)
```

Type: published

SetDbFieldValue

Description: This event is fired whenever the value from a TMemTableEC dataset field was assigned to a field of a query. Here it is possible to change the assigned value. If you have stated virtual fields in the FieldAssignments property, the assignment of their values can be made inside this event handler. The FieldName parameter contains the query field name of the FieldAssignments property.

Handler :

```
void __fastcall (__closure *TIBDPFieldEvent)(TField *memtabfield,
                        TIBXSQLVAR *dbfield,
                         const AnsiString  &FieldName,
                        TIBXSQLDA *rec)
```

Type: published