RtDbCopy Ver. 1.00

Copyright 1995 Tomasz Stanczak
Compuserve: 100735,3273

TBatchMove is a very powerful component, but it has several shortcomings. What it cannot handle:
- if you don't define a problem table and an error occurs, the whole operation will be canceled
- it cannot change field data types in most cases
- it cannot influence field data, it's copied as it is

RtDbCopy is a replacement component, which offers:
- before and after copy event, to do housekeeping actions
- on every record event, to gauge the whole process or implement function filter
- on every field event, where you get the data being read and set data being written, so that you can feed three destination fields from one source field or do whatever is appropriate
- on every error event, where you can ask the user to decide if he wants to continue or break
- through Mappings definition it lets you change field data types and/or sizes

RtDbCopy comes very close to the TBatchMove speed-wise. It is not necessarily faster, though it uses low-level BDE calls accessing data. With Paradox/dBase TBatchMove were faster but with Local IB it was the other way round. I found out that the differences were most offen minor.

TRtDbCopy

Properties Events Methods

The TRtDbCopy component is in fact very similiar to TBatchMove. The information about tables involved is saved in Source and Destination properties, the type of operation in Mode. The logical fields mapping between both tables is saved in Mappings property in almost the same manner as in TBatchMove, the different part is that field data type, size and field event triggerring information for every field is also stored in there. To put the whole process into a transaction the property of the same name must be set True. It works only for SQL based destination tables, and if it shouldnt work for any particular database it is possible to make manually with SQL passthough in OnBeforeCopy and OnAfterCopy events. Then comes OnError event with the possibility to decide if the copying should be canceled or continued. In such a case CopyNo readonly property contains the sequential number of the record being copied. OnField event lets put the hands on data for every field separately before it will be saved in destination table. OnRecord event is called for every record being processed. As with TBatchMove the method which starts the operation is called Execute, it needs some memory to be allocated. If memory problems be experienced ReadBlockSize and WriteBlockSize runtime properties can be used to decrease the amount of memory needed.

CopyNo

Declaration

property CopyNo: longint;

Description

Readonly runtime only property. It contains the sequential number of record being processed. You can check it for example in the error event.

Mappings

Declaration

property Mappings: TString;

Description

This is similiar to Mappings of TBatchMove in that it defines a relation between source field names and destination field names. It has been changed though to let you define date type and size of the destination fields and if a field event should be fired on the particular field. The structure of a Mapping should be as follows:

```
[*]<dest.fieldname>[:<fieldtype>,<field size>]=<source fieldname>
```

The asterix at the beginnig cause the field event to be called on that field. Fields data type and size are optional. Most probably it will be set using a property editor, that handles proper structure.

ReadBlockSize

Declaration

property ReadBlockSize: byte;

Description

Runtime property. It lets you define how big a read buffer should be as multiplicity of 4096 bytes. Default value is 16, which gives a buffer of 65536.

WriteBlockSize

Declaration

property WriteBlockSize: byte;

Description

Runtime property. It lets you define how big a write buffer should be as multiplicity of 4096 bytes. Default value is 16, which gives a buffer of 65536.

Transaction

Declaration

property Transaction: Boolean;

Description

If this property is set true and Destination Table is SQL based, a transaction will be started befory copy, and comitted after succesful end. It uses Destination Table Databases methods, so if it doesnt work for you use OnBeforeCopy and OnAfterCopy events to manage transactions manually.

OnAfterCopy

Declaration

property OnAfterCopy: TNotifyEvent;

Description

This event is fired only once per copy - after all work is done. It could be used to commit or a transaction using SQL-passthrough.

OnBeforeCopy

Declaration

property OnBeforeCopy: TNotifyEvent;

Description

This event is fired only once per copy - before it all starts. It could be used to start a transaction using SQL-passthrough.

OnError

Declaration

property OnError: TRtErrorEvent;

Description

This event will be fired on every exception. Use it to get the error information and decied if the process should be aborted.

OnField

Declaration

property OnField: TRtFieldEvent;

Description

TRtFieldEvent will be called on every field defined in Mappings property for every record being copied. Use it to examine and change Destination fields data before they will be saved.

OnRecord

Declaration

property OnRecord: TRtRecordEvent;

Description

This event will be called after every record has been read from Source Table, but before it would be written into the Destination Table. It first lets you implement for example a progress gauge or a function filter.

TRtErrorEvent

Unit
RtDbCopy

Declaration

TRtErrorEvent = **procedure** (Sender: TObject; E: Exception; var Cancel: Boolean) **of object;**

Description

This procedure is called by OnError event. Parameter E is the actual Exception being raised to be examined. Setting Cancel to False lets RtDbCopy proceed, otherwise the operation will be canceled. Default value for Cancel is True;

TRtFieldEvent

Unit
RtDbCopy

Declaration

```
TRtFieldEvent = procedure (Sender: TObject; FieldNo: word; DataType: TFieldType; Data: Pointer;
                         var IsBlank: Boolean) of object;
```

Description

TRtFieldEvent procedure will be in OnField event. FieldNo is position of the field in Destination Table, DataType defines ist type, and Data is a pointer to the aktual fields data. In order to use it you should cast it to the appropriate type:

```
Integer(Data^) := 20;          {DataType=ftSmallint}
Longint(Data^) := 100000;      {DataType=ftInteger}
StrUpper(PChar(Data));         {DataType=ftString}
```

IsBlank informs if the source field was blank, setting it True stops RtDbCopy from saving it in the Destination Table.

TRtRecordEvent

Unit
RtDbCopy

Declaration

TRtErrorEvent = **procedure** (Sender: TObject; Value: integer; var Cancel,Handled: Boolean) **of object;**

Description

This procedure will be called in OnRecord event. Value variable contains a sequential number of the record being copied. Setting Cancel to True causes the whole operation to be aborted. Setting Handled to True moves RtDbCopy to the next record. The first lets you implement a progress gauge, the second a function filter - for all record that shouldnt be copied set Handled:=False.

TRtCopyMode

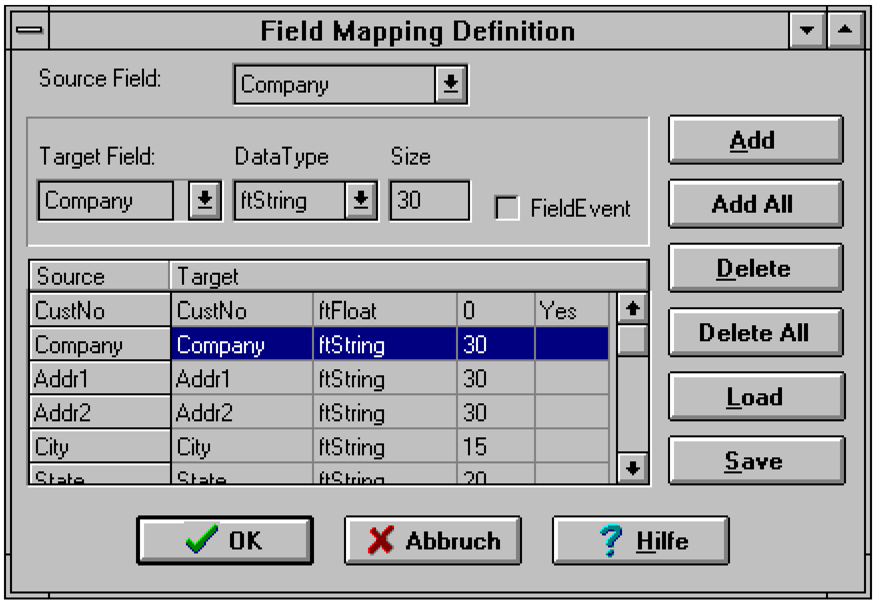
Declaration

type TRtCopyMode = (batAppend, batUpdate, batAppendUpdate, batDelete, batCopy);

Description

It is the same as TBatchMode, I just didnt want to be dependent on TBatchMove changes.

Field Mapping Property Editor



Source Field: Company

Target Field: Company — DataType: ftString — Size: 30 — FieldEvent: ☐

Source	Target			
CustNo	CustNo	ftFloat	0	Yes
Company	Company	ftString	30	
Addr1	Addr1	ftString	30	
Addr2	Addr2	ftString	30	
City	City	ftString	15	
State	State	ftString	20	

Field Mapping editor lets you visually define all of the destination fields attributes. It is also possible to load a text file in appropriate format.

Registration

Product	*Price*	*s&h*
RtDbCopy	$20 (30 DM)	2$ (3 DM)
RtControls	$39 (60 DM)	2$ (3 DM)

Only following payment methods will be accepted:

Compuserve: GO SWREG

ID# 7405 RtDbCopy alone

ID# 7197 RtControls

Checks: Only checks drawn on a US bank valued in US$ or drawn on a German bank and valued in DM will be accepted. The checks should be made payable to **Tomasz Stanczak**. Please let us time after sending check, we cannot send you our product before the money is transferred.

Please fill in the Order Form and send with a payment to:

Tomasz Stanczak
Hardenbergstr. 8
31275 Lehrte
Germany

Credit Card Orders: You can order with MC, Visa, Amex, or Discover from Public (software) Library by calling 800-2424-PsL or 713-524-6394 or by FAX to 713-524-6398 or by CIS Email to 71355,470. You can also mail credit card orders to PsL at P.O.Box 35705, Houston, TX 77235-5705.

THE ABOVE NUMBERS ARE FOR CREDIT CARD ORDERS ONLY.
THE AUTHOR OF THIS PROGRAM CANNOT BE REACHED AT THESE NUMBERS.

Any questions about the status of the shipment of the order, refunds, registration options, product details, technical support, volume discounts, dealer pricing, site licenses, non-credit card orders, etc, must be directed to Tomasz Stanczak:

CIS:	100735,3273
Internet:	100735.3273@compuserve.com
WWW:	http://ourworld.compuserve.com/homepages/tomasz

To order please specify Item# 14429, RtControls Delphi VCL Pack, price $39 plus s&h $2 (e-mail) or $4 (Europe) or $6 overseas.

To insure that you get the latest version, PsL will notify us the day of your order and we will ship the product directly to you.

Shipping:

Currently only e-mail shipping is supported, so please always state your e-mail address!. In the future post shipping including printed documentation will be possible (no promise!).

Registered users will receive components full source code. Contact the author for multi-user discounts and site licenses.

Order Form

Name (block letters please):	______________________________
Company Name:	______________________________
Street Address:	______________________________
	______________________________
City, State, Zip Code:	______________________________
Country:	______________________________
Phone Number:	______________________________
E-Mail Address:	______________________________

Product Name	Quantity	Price

Shipping______________________________

Total Fee______________________________

Technical Notes

RtDbCopy have been designed with speed and flexibility in mind. It wasnt possible to achieve the same speed as TBatchMove, but through the use of low-level BDE calls wherever I could, they are very close.

There is also one known problem. RtDbCopy cannot create tables through Foxpro/dBase ODBC drivers from both Microsoft and Intersolv. It is not the problem with the component itself but more with TTables CreateTable method - TBatchMove has the same problem.

The shareware version of this component runs only if Delphi will be found in memory.

Licence

Warranty

Support

Support is available from the author via Compuserve Mail to 100735,3273 , Internet Mail to 100735.3273@compuserve.com or http://ourworld.compuserve.com/homepages/tomasz. Currently there is no postal, telephone, or fax support. All support is on a "best effort" basis, and no promise is made that all questions can be answered, problems solved, features added nor bugs exterminated.

Revision History

1.00	this is the intial version.

Properties

CopyNo	Mappings
ReadBlockSize	Transaction
WriteBlockSize	

Events

OnAfterCopy	OnError	OnRecord
OnBeforeCopy	OnField	

Methods

Create	Destroy	Execute

Destination Field Name

ComboBox ontaining all field names of the source table.

Destination Field Name

ComboBox filled with the field names of the destination table, or if Mode=batCopy, source table.

Destination Field Data Type

ComboBox with all possible data types. After a source field or a field in the grid have been chosen it is set to ist data type.

Destination Field Size

Here goes a destination fields data size. After a source field or a field in the grid have been chosen it is set to ist data size. This field is enabled only if data type is one of ftString,ftMemo,ftBytes,ftVarBytes,ftBlob or ftGraphic.

Destination Field Event

Check it if a field event should be fired on the selected field.

StringGrid

This StringGrid shows all Field Mappings.

Ok Button

Saves changes into Mappings property.

Cancel Button

Mappings property stays unchanged.

Help Button

Calls this help file.

Add Button

Adds field mapping into Mappings.

AddAll Button

Adds field mappings for all source fields. If Field Event CheckBox is set, the OnField event would be fired for all fields.

Delete Button

Deletes selected row from Mappings property.

DeleteAll Button

Clears Mapping property.

Load Button

Loads Mappings property from a text file.

Save Button

Saves Mappings into a text file.