TRANSFER

Data transfer/conversion utility

User guide

In this user guide the functionality of the software is described and explained.

Transfer 3.3 help

Printed: 2021 in (whereever you are located)

Special thanks to:

All the people who contributed to this document, to mum and dad and grandpa, to my sisters and brothers and mothers in law, to our secretary Kathrin, to the graphic artist who created this great product logo on the cover page (sorry, don't remember your name at the moment but you did a great work), to the pizza service down the street (your daily Capricciosas saved our lives), to the copy shop where this document will be duplicated, and and and...

Last not least, we want to thank EC Software who wrote this great help tool called HELP & MANUAL which printed this document.

Contents

Index 81

Foreword

This is just another title page
placed between table of contents
and topics

Top Level Intro

This page is printed before a new
top-level chapter starts

Part

I

1   Contents

Top Level Intro

This page is printed before a new
top-level chapter starts

Part

II

2 What's new in version 3.3?

- DB schema support for destination table/sub-table:
  - transfert syntax - new "owner" keyword added
  - interface for table/sub-table - new "DB schema" field added
  - SQL expressions parsing is performed using DB schema parameter,
    no matter what the user login is in connection profile

Top Level Intro

This page is printed before a new
top-level chapter starts

Part III
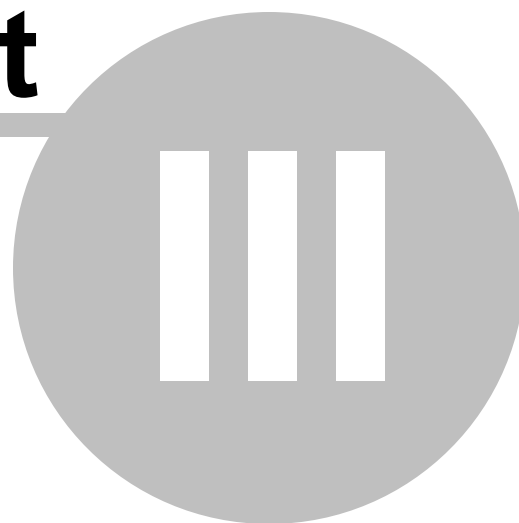
3   License info

NO WARRANTY

TRANSFER IS SOLD "AS IS" AND WITHOUT ANY WARRANTY AS TO MERCHANTABILITY OR
FITNESS FOR A PARTICULAR PURPOSE OR ANY OTHER WARRANTIES EITHER EXPRESSED OR
IMPLIED. THE AUTHOR WILL NOT BE LIABLE FOR DATA LOSS, DAMAGES, LOSS OF PROFITS
OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THIS SOFTWARE.

Evaluation and Registration

Transfer is not free software. You may use this software for evaluation purposes without charge for a
period of 30 days. If you use this software after the 30 day evaluation period, a registration fee is
required. See ORDER.TXT file or 'How to Register' section in on-line help for detailed information about
registration method and price list.

Distribution

You may copy the evaluation version of this software and documentation as you wish, and give exact
copies of the original evaluation version to anyone, and distribute the evaluation version of the software
and documentation in its unmodified form via electronic means. But you should not charge or requesting
donations for any such copies however made and from distributing the software and/or documentation
with other products without the author's written permission.

Registered version

One registered copy of Transfer may either be used by a single person who uses the software personally
on one or more computers, or installed on a single workstation used non-simultaneously by multiple
people, but not both.

Top Level Intro

This page is printed before a new
top-level chapter starts

Part IV

4   **Trial version limitations**

If you use the Trial version of Transfer, then the following limitations apply:

- Program usage is limited by 30 days.
- You cannot process more than 100 records in DBMS table in one transfer session.
- COMPLEX schema cannot be used for connected data processing.
- Automatic data exchange function is not available.

Top Level Intro

This page is printed before a new
top-level chapter starts

Part V

5   How to register

You are limited to 30 days of use for an unregistered version of Transfer.

Registered users are entitled to **FREE** upgrades for the major version, they purchased. That means, that in case you bought registration key for version 3.30, you will receive upgrades for versions 3.xx for **FREE**. If a new authorization code is required it will be issued upon request at no charge to users who have registered for the current major version.

To register:
Write an email to get license: **dagolubev@gmail.com**

Top Level Intro

This page is printed before a new
top-level chapter starts

Part

VI

6   Feadback

Feedback
Please send any bug report, suggestion, question and comment to the author at following address.
dagolubev@gmail.com

Top Level Intro

This page is printed before a new
top-level chapter starts

Part VII

7   Introduction

7.1   What is Transfer?

Transfer is the universal customizable utility for data transfer/conversion from one database to another.

Program is written in Microsoft Visual C++ with ODBC API, wich allows to reach the maximum productivity in massive data blocks processing. Program unification is based on standardized interface provided by ODBC drivers. Wide selection of existing ODBC drivers makes Transfer the universal tool to work with both file databases (dBase, Access) and relational databases (Oracle, Sybase, Microsoft SQL). In its work Transfer uses internal format of data representation, that makes it independent from specific data types peculiar to different databases. Embedded internal data definition language allows to customize the program to perform any tasks for information transferring/conversion.

Transfer main features:

- Work with any databases via ODBC interface.
- High performance in data processing.
- SQL query language support.
- Automatic data type convesion during transfer.
- Embedded flexible data structure (profiles) definition language.
- Support of external arguments for a data profile.
- Convenient data profile editor (GUI).
- Stored database procedures' calls during transfer.
- Trigger-based mechanism to make the data processing scenario.
- Possibility to process master-slave tables in one logical block.
- Internal text data convertor from ANSI to OEM and vice versa.
- Options to run from command line.
- Logging all actions in a protocol file.
- Optional SQL-commands execution tracing.
- Automatic message files creation upon the transfer completion.
- Possibility to run automatic file exchange with a remote host.
- Internal LZW data archiver.
- Multilingual support.

7.2   What is Transfer for?

Initially Transfer was developed as a small utility to move financial data between central office and remote branches. However, in a course of time Transfer has become a powerful and flexible tool for data transferring/conversion using ODBC drivers. The complexity of data processing tasks forced to develop the flexible data structure definition language allowing the program to be highly customizable to the particular user needs. The language has rather simple syntax which is not mandatory to remember. Transfer has a convenient profile editor, that allows to define all neccessary parameters for a data profile using a number of dialogs and then save them in a file.

There could be several fields of using Transfer:

- Program is very useful as a utility for quick one-occasion data transfer/conversion from one database to another. Those databases should not neccessarily be of the same type. For example, quite easy you can transfer your data between Access and Oracle. All you need for that - is to define data

structures (tables) and corellation between source and destination. You can also use the special feature to automatically create tables in destination database. Data can be conveyed unconditionally or using comparision mode, i.e. only missing rows will be added to the destination table.

- It is possible to use Transfer as powerful tool providing data exchange between central office and remote branches for financial applications. Data can be delivered using any media (diskette, USB drive) or by email. Here the powerful mechanizm of integrated data processing can be used for documents exchange (where the document is treated as a monolithic entity which consists of related data). For data processing automation during information import the event-based scenario can be used by means of special triggers. Also, the big advantage of Transfer is the capability of stored database procedures' calls during data transfer.
- Transfer can also serve as a transmission mechanizm to provide an automatic continuous data exchange between distributed databases. The special data exchange setup allows to plan automatic sessions of data transferring in certain data intervals. Program should be run all the time (be resident). For the convenience of users the program can be minimized to system Tray. To optimize the data file size for sending there is an embedded data archiver in LZW format. The remote hosts can be connected, for example, using Microsoft service "Remote-access server".

7.3   How does it work?

The basic requirements for Transfer to work are ODBC drivers for source and destination databases. Further, the data structure (profile) must be defined and stored in a configuration file. To do that one can use any text editor or internal dialog-based editor which eliminates the needs to learn the language syntax. It is possible to input external parameters in the data profile, where the parameters' values can be entered at the execution time (for example data period). Then the created data profile file should be registered in the list of profiles to be available for execution. From this point the data transfer operation can be run just by selection of certain profile. You can set a default profile for import and export operations. In that case the selected profile intialization will be done automatically when using import or export functions.

There are several ways to run Transfer:

- Command line. The program can be run from command line with supplying of a set of options (see Command line options).
- Manual. Program is launched manually, then the neccessary profile is choosen and executed.
- Automatic. Program is always loaded in a computer memory. For convenience the program can be mnimized in a system Tray. Current status is displayed by a special icon. All required data transmission operations are performed automatically in specified data intervals (see Data exchange options).

Top Level Intro

This page is printed before a new
top-level chapter starts

Part

VIII

8 Transfer menu

8.1 File menu

8.1.1 File menu

[Create (Ctrl+N)](#)
[Open (Ctrl+O)](#)
[Close (Ctrl+F4)](#)
[Save (Ctrl+S)](#)
[Save as](#)
[Print setup](#)
[Page setup](#)
[Print (Ctrl+P)](#)
[Import](#)
[Export](#)
[Transfer](#)
[Start exchange (Ctrl+R)](#)
[Stop exchange (Ctrl+T)](#)
[Data profiles](#)
[Options](#)
[Exit (Alt+F4)](#)

8.1.2 Create

Creats a new text file. A new MDI window is opened with simple text editor. It is possible to create and edit several files simultaneously. After the text is imputted, it can be saved using commands: [Save (Ctrl +S)](#) [Save as](#).

8.1.3 Open

Opens an existing file for editting. Only text files can be editted. It is possible to edit several files simultaneously. When editing is completed, the file can be saved using command: [Save (Ctrl+S)](#).

8.1.4 Close

Closes the current file edit window. If several windows are open, then the top window will be closed.

8.1.5 Save

Writing changes made to a file on a disk. If the command is used for a new file, then the program will prompt you to enter a file name. For existing file the changes are written to the same file.

8.1.6   Save as

Allows to write file with a different name. If existing file was changed, the command can save the modified text in a different file. Original file will remain unchanged. For a newly opened file the command functions the same as Save (Ctrl+S).

8.1.7   Print setup

Opens system print setup dialog, which allows you to select your current ptinter and setup its parameters. The dialog depends on the driver of installed printer.

8.1.8   Page setup

Opens page setup dialog, which allows you to setuo the following parameters:

- paper size
- paper source
- page orientation
- margin sizes

8.1.9   Print

Allows to select printer and print text in the open file.

8.1.10   Import

Run import profile.

When using this command, if the default import profile is set, then exactly this profile is executed (see General options). Otherwise a dialog window with the profiles list of "import" type will be opened. Therefore, a user can choose any porfile from the list and run. After profile initialization a dialog window with the list of all defined within a profile data blocks (transfers) is opened where the profile parameters (if any) can be set. Then the profile can be started for execution.

8.1.11   Export

Run export profile.

When using this command, if the default export profile is set, then exactly this profile is executed (see General options). Otherwise a dialog window with the profiles list of "export" type will be opened. Therefore, a user can choose any porfile from the list and run. After profile initialization a dialog window with the list of all defined within a profile data blocks (transfers) is opened where the profile parameters (if any) can be set. Then the profile can be started for execution.

8.1.12 Transfer

Run data transfer procedure.

When using this command, a dialog window with the list of all registered profiles is opened. A user can choose any porfile from the list and run. After profile initialization a dialog window with the list of all defined within a profile data blocks (transfers) is opened where the profile parameters (if any) can be set. Then the profile can be started for execution.

8.1.13 Start exchange

Start data exchange procedure. After procedure is started, all other menu items become disabled. The data processing algorithm depends on the "Department flag" option (see Data exchange options).

Detailed data exchange algorithm description see in Automatic data exchange.

8.1.14 Stop exchange

Stop data exchange procedure. After the procedure is stopped, all other menu items become enabled. It is impossible to stop the procedure during file copy operations.

8.1.15 Data profiles

List of all registered in a program profiles.

File	Name
ImpCnt.ppl	Import into center
ImpFil.ppl	Import into filial branch
ImpOfice.ppl	Import into office
ImpSklad.ppl	Import into warehouse
ImpBnk.ppl	Import bank document
ImpDoc.ppl	Import documents and transactions
ImpBuh.ppl	Import accounting documents
ImpMain.ppl	Import into main database
ImpBaza.ppl	Import into base office
ImpStore.ppl	Import into remote store
ExpCnt.ppl	Export from center
ExpFil.ppl	Export from filial branch
ExpOfice.ppl	Export from office
ExpSklad.ppl	Export from warehouse
ExpBnk.ppl	Export bank documents
ExpDoc.ppl	Export documents and transactions
ExpBuh.ppl	Export accounting documents

Each data profile is a file, that should be placed in a profile directory (see General options). List of registered profiles is stored in Trans30.ini file in [pipeline] section.

Every profile in a list has a special pictogram describing a profile type:

   General purpose profile

   Import profile

   Export profile

To edit a profile you should double-click on it by left mouse button or hit Enter button on a keybord. Then a profile editor window will be opened.

8.1.16   Options

Opens Transfer options dialog.

Options dialog contains the following tabs:
General options
Data exchange options
Processing options
Interface options

8.1.17   Exit

Closes program. All open MDI window are automatically closed without saving.

8.2   Edit menu

8.2.1   Edit menu

Undo (Ctrl+Z
Cut (Ctrl+X)
Copy (Ctrl+C)
Paste (Ctrl+V)
Delete (Del)
Select all (Ctrl+A)
Find (Ctrl+F)
Replace (Ctrl+H)

8.2.2   Undo

Reverse previous editting action in the open file window.

8.2.3 Cut

Deletes currently selected text from open text file window and moves it to Windows clipboard. This command is unavailable if there is no data selected.

8.2.4 Copy

Copies selected text from open text file window to Windows clipboard. This command is unavailable if there is no data selected.

8.2.5 Paste

Inserts text fragment from Windows clipboard into the open text file window. This command is unavailable if the clipboard is empty.

8.2.6 Delete

Deletes currently selected text from open text file window.

8.2.7 Select all

Select the whole text in open file window.

8.2.8 Find

Search open text document for a text string.

8.2.9 Replace

Search open text document for a text string and replace it with another text string.

8.3 Data menu

8.3.1 Data menu

Add (Ctrl+Ins)
Retrieve (F2)
Save (F3)
Edit (F4)
Delete (Ctrl+Del)
First (Ctrl+Home)
Previous (Ctrl+Up)

Next (Ctrl+Down)
Last (Ctrl+End)
Close (Ctr+F4)

8.3.2 Add

Adds an element to the profile list (register profile). Here the new profile card will be opened for data input.

8.3.3 Retrieve

Retrieves (re-reads) data in the profile list or profile card. All data about registered profiles are stored in Trans30.ini file. The command reads this file and displays the list of profiles.

8.3.4 Save

Saves data in the profile list (profile card). When entering a new profile or modifying existing one the command allows to write changes in Trans30.ini file or the profile file.

8.3.5 Edit

Edits data in the profile list (opens profile card). It allows to modify the profile data in a special dialog.

8.3.6 Delete

Deletes profile from the profile list (unregisters profile). However, the profile file is not phisically deleted, only the record about profile is deleted from Trans30.ini file.

8.3.7 First

Moves to the first row in the profile list. If the profile card is open, then the data in a card is not refreshed.

8.3.8 Previous

Moves to the previous row in the profile list. If the profile card is open, then the data in a card is not refreshed.

8.3.9 Next

Moves to the next row in the profile list. If the profile card is open, then the data in a card is not refreshed.

8.3.10 Last

Moves to the last row in the profile list. If the profile card is open, then the data in a card is not refreshed.

8.3.11 Close

Closes current open MDI window. If any data in a window were modified, then program will prompt the user whether the changes should be saved.

8.4 View menu

8.4.1 View menu

Toolbar Main
Toolbar Profiles
Status bar
Customize Main Toolbar
Customize Profiles Toolbar

8.4.2 Toolbar Main

Shows or hides the main toolbar, which includes buttons for some of the most common program commands. A check mark appears next to the menu item when the Toolbar is displayed.

If you move the mouse cursor to the Toolbar and hold it for some time, then the hint to the corresponding menu item will be displayed.

8.4.3 Toolbar Profiles

Shows or hides the profiles toolbar, which includes buttons for some of the most common commands for managing ptofiles. A check mark appears next to the menu item when the Toolbar is displayed.

If you move the mouse cursor to the Toolbar and hold it for some time, then the hint to the corresponding menu item will be displayed.

8.4.4 Status bar

Shows or hides the status bar (residing in the bottom of the program window). A check mark appears

next to the menu item when the Status bar is displayed.

If the status bar is displayed, it shows the selected menu item description. Also, in the right corner it shows the status of Caps Lock, Num Lock and Scroll Lock keys.

8.4.5 Customize Main Toolbar

This menu item shows the main toolbar customization dialog:
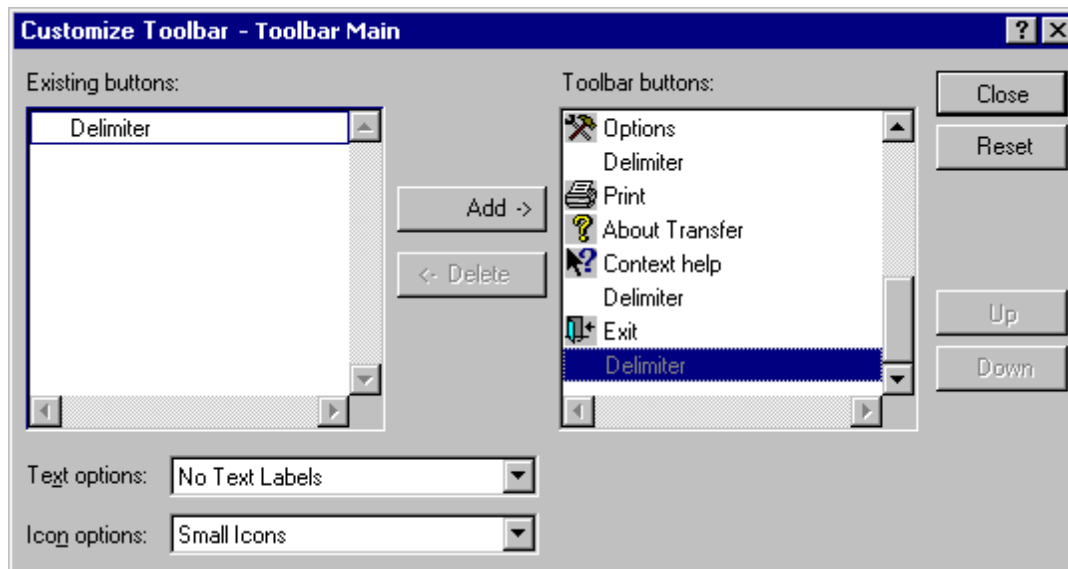
8.4.6 Customize Profiles Toolbar

This menu item shows the profiles toolbar customization dialog:
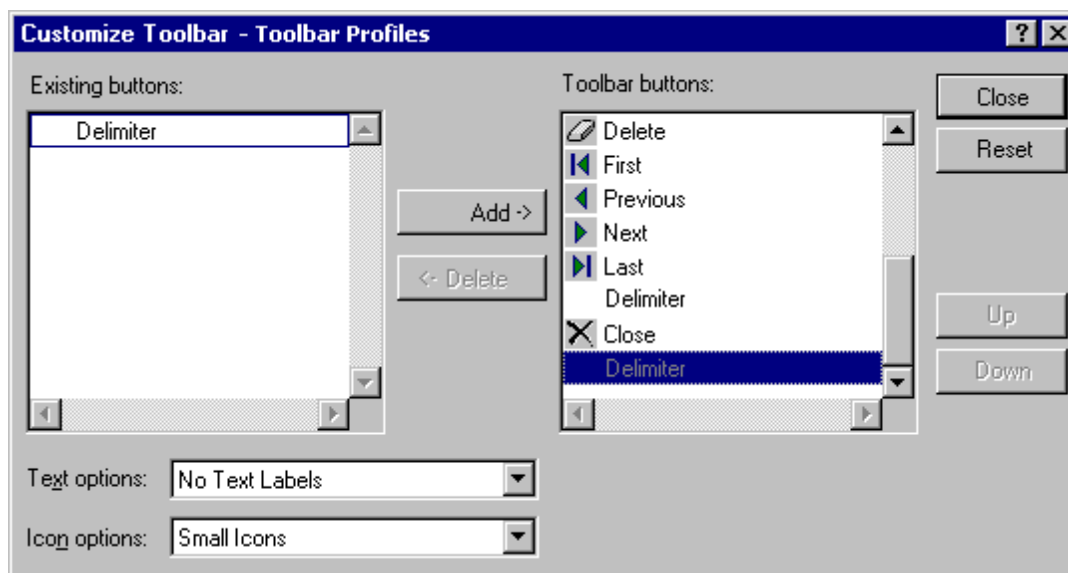
8.5 Window menu

8.5.1 Window menu

Cascade
Tile horizontally
Arrange icons

8.5.2 Cascade

Arranges MDI windows in an overlapped fashion.

8.5.3 Tile horizontally

Arranges MDI windows in non-overlapped horizontal tiles

8.5.4 Arrange icons

Arranges icons of minimized MDI windows.

8.6 Help menu

8.6.1 Help menu

Help topics (F1)
What's this? (Shift+F1)
About Transfer

8.6.2 Help topics

Offers you an index to topics on which you can get help.

8.6.3 What's this?

Context help. To get the context-sensitive help you should press Shift+F1, then move the mouse cursor to the Toolbar button, menu item or interface element, and then press the left mouse button. You will get the corresponding help topic.

8.6.4 About Transfer

This command presents Transfer dialog box which displays the copyright notice and version number of

your copy of Transfer.

Top Level Intro

This page is printed before a new
top-level chapter starts

Part IX

9 Transfer options

9.1 Command line options

-m   (Minimize) Minimize after start
-t   (Tray) Minimize to system Tray
-v   (Verbose) Suppress error messages
-f file (File) Profile file for execution
-p parm (Parameteres) Profile parameters

Profile parameters format:
[parm1=val1, parm2=val2, ..., parmN=valN]

If **-f file** option is given, the program executes profile **file**, then automatically closes.

9.2 General options
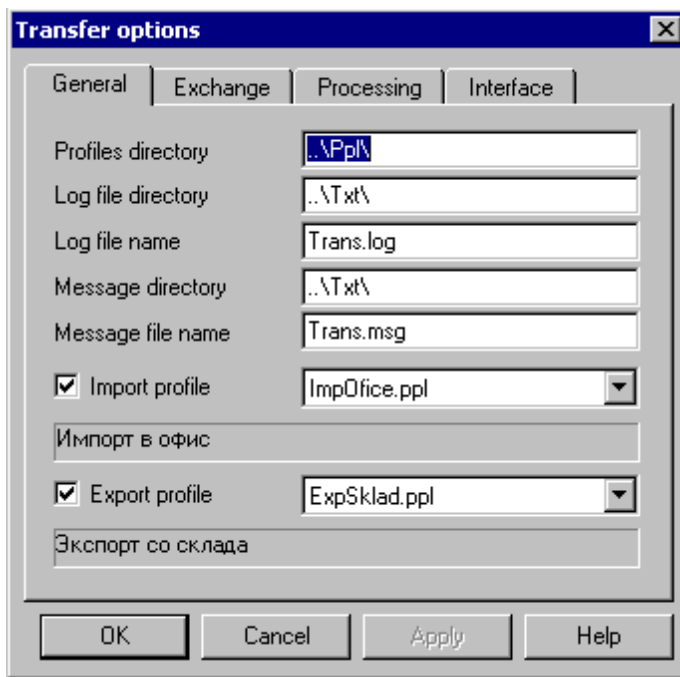
- **Profiles directory**. This is the folder where profile files are located. Relative path to the program work directory can be given here.
- **Log file directory**. This is the folder where the program will create log file during a data transfer session. Relative path to the program work directory can be given here.
- **Log file name**. Name of the file which is used by the program to write all actions and errors during a data transfer sesion.
- **Message directory**. This is the folder where the program creates message files with the transfer results.
- **Message file name**. Name template of the file, where the program writes the brief results of data

transfer session. After each import session the unique message file is created whic name consists of two parts. First part is date and time when the file was created, and second part is the name template (e.g., 05-12-2002_23-28-03_Trans.msg). To avoid message files generation the option "Write import results to a message file" should be unset (see Processing options).

- **Import profile**. Name of the profile file for import by default. Only registered profiles can be selected. Profile description is displayed under the file name. This parameter is mandatory for automated data exchange. If this parameter is checked, then import profile will be atomatically executed when using "Import" command from the program menu.
- **Export profile**. Name of the profile file for export by default. Only registered profiles can be selected. Profile description is displayed under the file name. This parameter is mandatory for automated data exchange. If this parameter is checked, then export profile will be atomatically executed when using "Export" command from the program menu.

9.3   Data exchange options

Those parameters are intended for automatic data exchange between central office and remote branch using Microsoft Access files. Two remote hosts should be connected using Microsoft service "Remote-access server".

- **Department flag**. Determines data processing algorithm. If "Center" option is selected, then program will be running in active mode, meaning that it will itslef initiate transfer sessions. If "Branch" option is selected, then program will be running in passive (waiting) mode.
- **Exchange interval**. Time interval in minutes that determines when the transfer session be started in active mode.
- **Interrogate interval**. Time interval in seconds for polling a remote host for data.
- **Import file path**. File name (with full path) for import. The name should be the same as it is indicated for data source in import data profile.
- **Export file path**. File name (with full path) for export. The name should be the same as it is indicated for data source in export data profile.

- **Empty data file**. File name (with full path) to replace the export file before export operation (to minimize the result file size).
- **Send directory**. Folder (full path) on a remote host where the data are sent.
- **Receive directory**. Folder (full path) on a remote host from where data are received.
- **Archive name**. File name (without path) for archive. Data are compressed in LZW format using embedded archiver.

9.4   Processing options
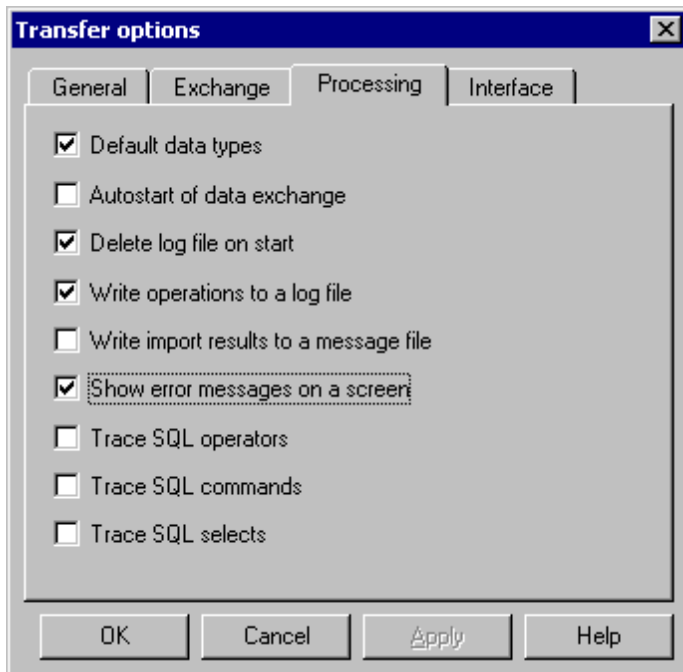
- **Default data types**. Automatic data types casting for parameters in SQL operators, if the data type is not explicitly defined.
- **Autostart of data exchange**. Start data exchange session right after the rogram launch.
- **Delete log file on start**. If set, then log file will be re-created every time the program is started.
- **Write operations to a log file**. Write all executed actions during the data transfer session with date and time indication (e.g., start operation, transfer initialization, transfer execution, file copy or deletion etc).
- **Write import results to a message file**. If set, then after the import session a new message file will be created. Message file will contain brief resuts about the data imported or modified. Messages format is defined separately for every transfer in the data profile.
- **Show error messages on a screen**. If option is set, then each error will be shown in a dialog window. User can cancel current operation, continue with displaying errors or continue without displaying errors. In any case all errors are written in a log file.
- **Trace SQL operators**. If set, then all executed INSERT, DELETE UPDATE operators will be recorded in a protocol file.
- **Trace SQL commands**. If set, then all stored procedures' calls will be written in a log file.
- **Trace SQL selects**. If set, then all executed SELECT commands will be recorded in a log file.

9.5 Interface options

- **Language**. Selecting the current interface language. Changing this parameter will require the program restart.
- **Show about dialog on startup**. If set, then the "About" dialog will be shown after the program start.
- **Minimize on start application**. If set, the program window will be automatically minimized right after application start.
- **Minimize to the system Tray**. If set, then the program will not appear in the task panel when minimized, instead it will add an icon to the system Tray. After that the program can be opened again if double-click on the program icon by a left mouse button. If a user clicks on a program icon by a right mouse button, then context menu appears, allowing to run or stop data exchange sessions as well as setup options.
- **Save main window position**. This option allows to store the position and the size of the main program window. On the next program start the window will be shown at the same position and with the same size.
- **Save MDI windows position**. If set, then the position and the size of the child MDI windows (for example, profiles list) will be displayed at the same position and with the same size as previously.
- **Maximize MDI windows**. If this option is set, the the child windows will be opened maximized.

Top Level Intro

This page is printed before a new
top-level chapter starts

Part

X

10   ODBC source setup

10.1   What is ODBC

Open Database Connectivity - ODBC - is the oldest Microsoft technlogy for database access interface development. The main reason of ODBC introduction was the requirment to provide an easy way to access databases with minimal dependancy on the particular programming language. For example, to access DBF file you would need to know Xbase programming language, and to work with MDB files – Access Basic. With ODBC standard introduction it became possible to work with different databases using structured query language - SQL.

ODBC principals are unified for Windows: for any particular database a special driver (which is supplied as a DLL file) is required. Actually, ODBC always needs two drivers for its work, where the first one is used for data administration, while the second one provides the common interface for programming languges. Drivers interaction via common interface allows to access databases using standard set of function calls.

In Windows 3.1 only 16-bit ODBC drivers were used. The main disadvantage of those drivers was low performance. In Windows 9x, Windows NT, Windows 2000, Windows XP 32-bit ODBC drivers were introduced.

32-bit ODBC drivers advantages:
*   **High performance**. Data processing speed was increased in several times (sometimes in dozens of times).
*   **Capabilities**. New drivers versions have enhanced capabilities for data and tasks control.
*   **Multithreaded data processing**. Decrease the probability of application crash due to one task failure.

10.2   ODBC drivers

In many cases ODBC drivers are supplied by a database provider. Also ODBC drivers are included in some software packets. For example, Microsoft Office contains the following ODBC drivers:
*   Microsoft SQL Server
*   Microsoft ODBC for Oracle
*   Microsoft Access
*   Microsoft Excel
*   Microsoft FoxPro
*   DBase
*   Paradox
*   Text File

To see what ODBC drivers are installed in your Windows system, you need to open Control Panel. Then in Control Panel double-click on "ODBC Data Sources (32 bit)" icon and the "ODBC Data Source Administrator" window will appear. Select Drivers tab and you will see the list of installed ODBC drivers:
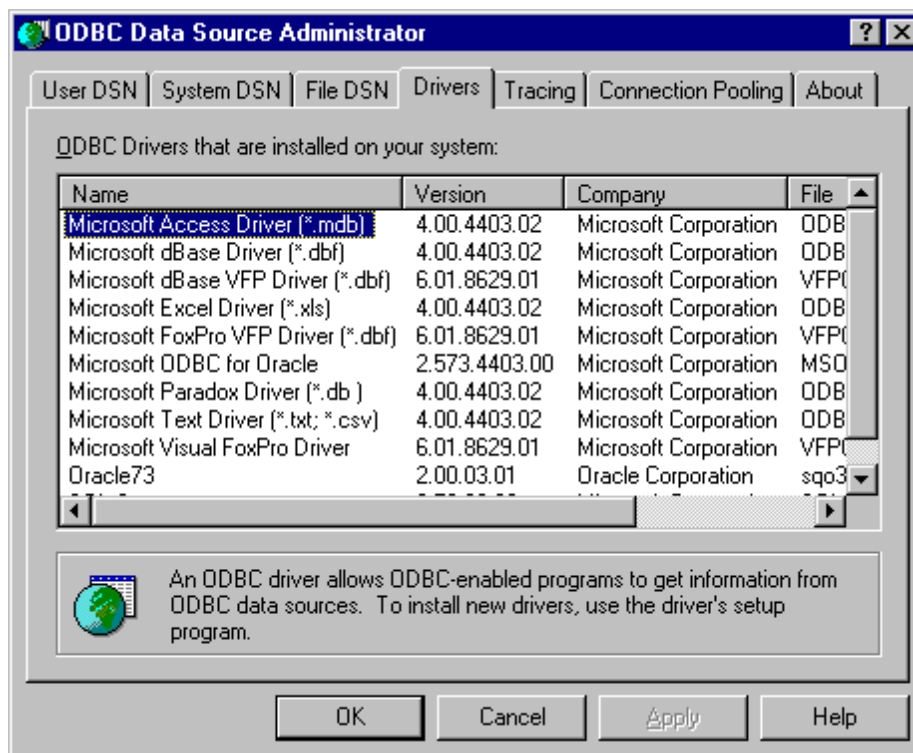
Note that the list contains drivers names and some additional technical info. the most important are Version and Company columns, as they inform who is the driver provider and whether you need to update it. Knowing driver version may also help in problems resolution if they occur during work with specific database. Column File is also useful when you need to remove driver or recover it.

10.3 ODBC source creation

To work with ODBC you should have the database or create ir. Below you may find a technique to make the configured data source for Microsoft Access database.

Step 1
Double-click on "ODBC Data Sources (32 bit)" icon in Control Panel. you will see the "ODBC Data Source Administrator" dialog. Note that this is "User DSN" tab. There are also "System DSN" and "File DSN" tabs, that are used for file data sources.
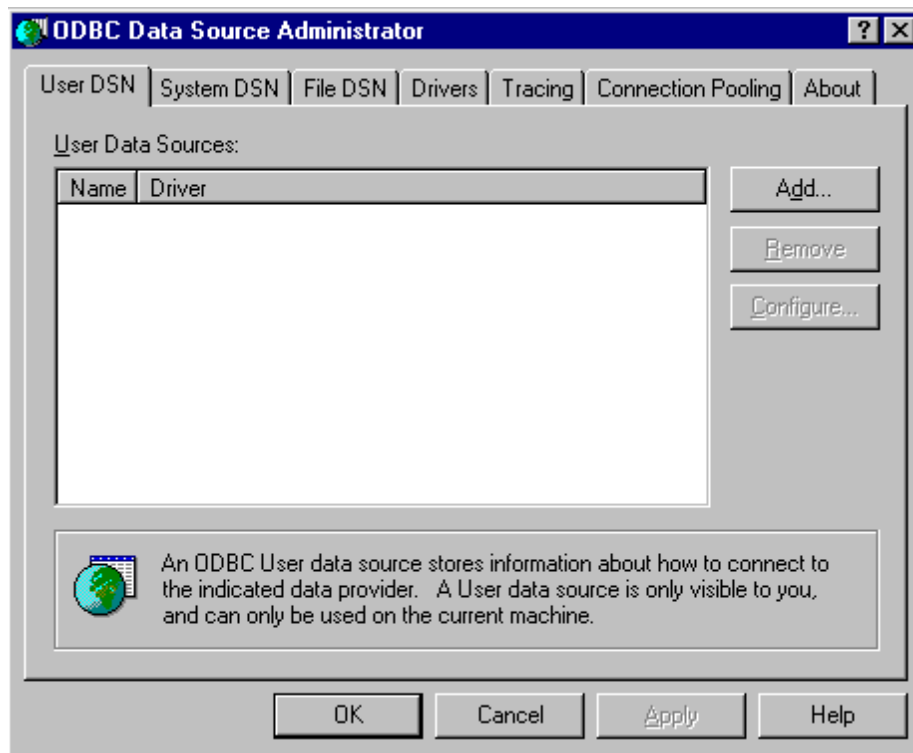
Step 2

Press Add button. You will see the "Create New Data Source" dialog.
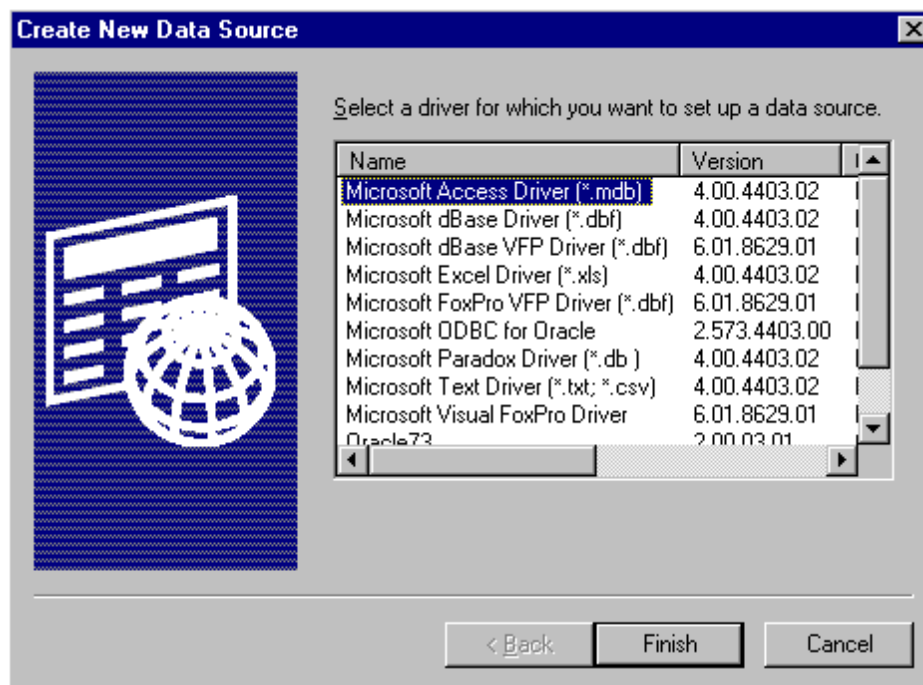
Step 3

- Choose one of the ODBC drivers(for ex., Microsoft Access Driver), and then press Finish button.
  You will see the "ODBC Microsoft Access Setup" dialog window as shown below. Note that this

window may differ for other ODBC drivers, as different drivers may require different configuration data.

- Input the data source name in the "Data Source Name" field. It should be meaningful, but not too long.
- Enter short database description in the "Description" field.
- Select system database option. In most cases you should choose "None", unless you want the system database.
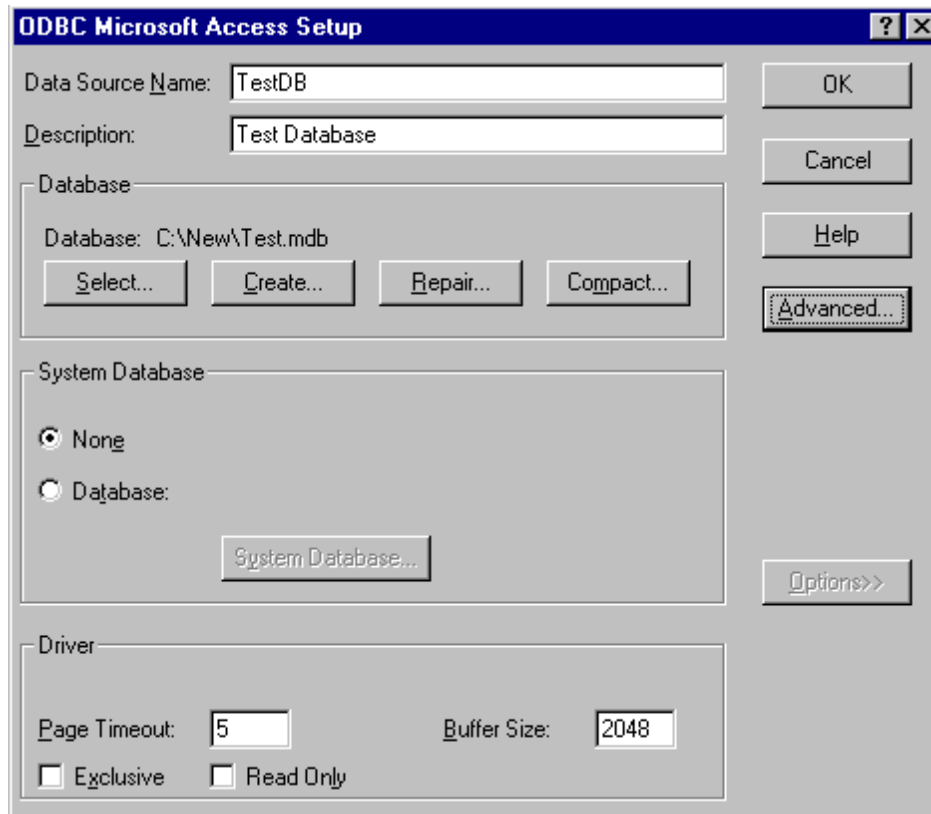
Step 4
To select existing database, press Select button. File Open dialog will appear to pickup the file. ODBC driver will automatically offer correct file extention.
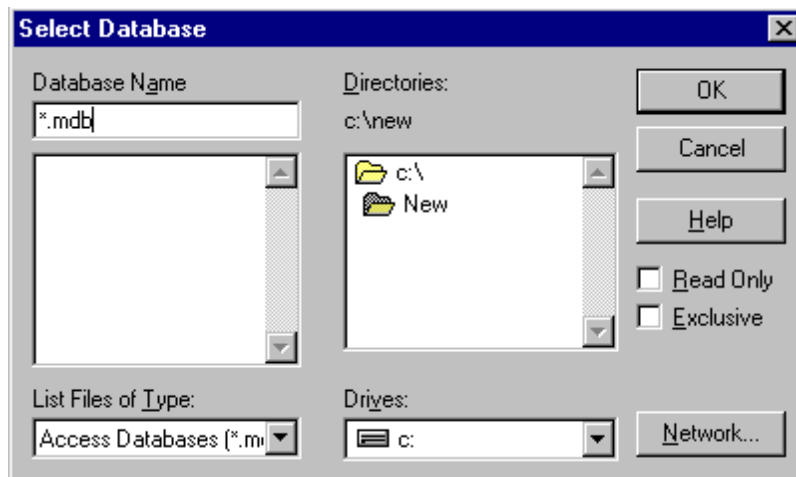
To create a new database press Create button. Besides, ODBS Access drivers allows to shrink and recover database files from this dialog window.

New Database

Database Name: `Test.mdb`

Directories: `c:\new`

- c:\
- New

Locale: `General`

Drives: `c:`

OK

Cancel

Help

Format
- ● Version 4.x
- ○ Version 3.x
- ○ Version 2.x

Options
- ☐ System Databa
- ☐ Encryption

Network...
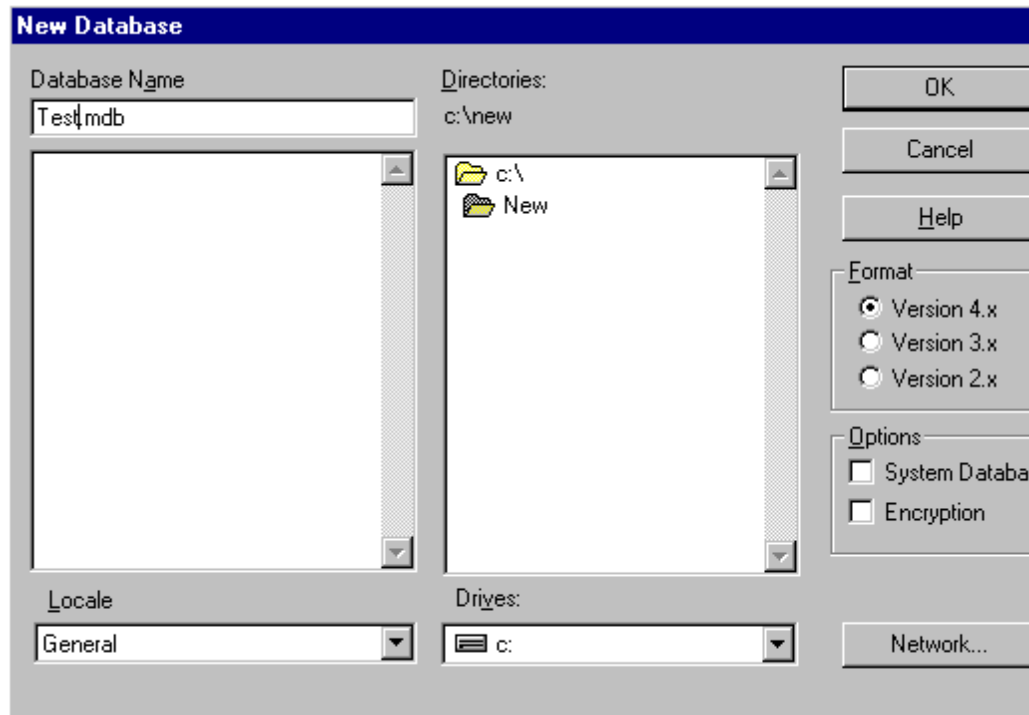
Step 5
Press Advanced button… and you will see the "Set Advanced Options" window. You may setup guest user name and password. You may also setup additional options like number of threads that can used database. Press OK button when you finished with advanced options.
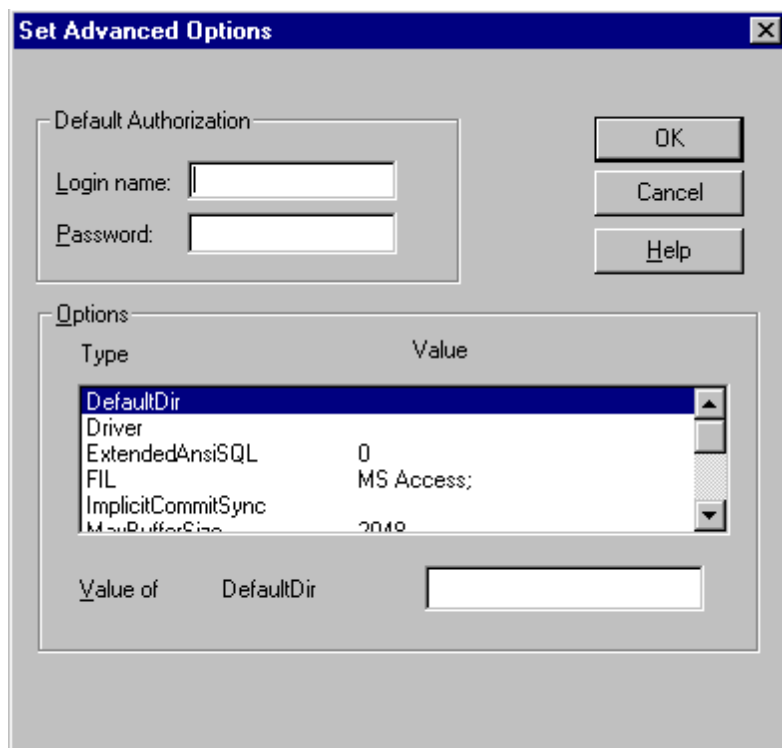
Step 6

Press OK again to close the "ODBC Microsoft Access Setup" dialog. You should see the new element as shown below. If later you need to modify settings for TestDB database, simply select it and press Configure button. To delete database press Remove button.
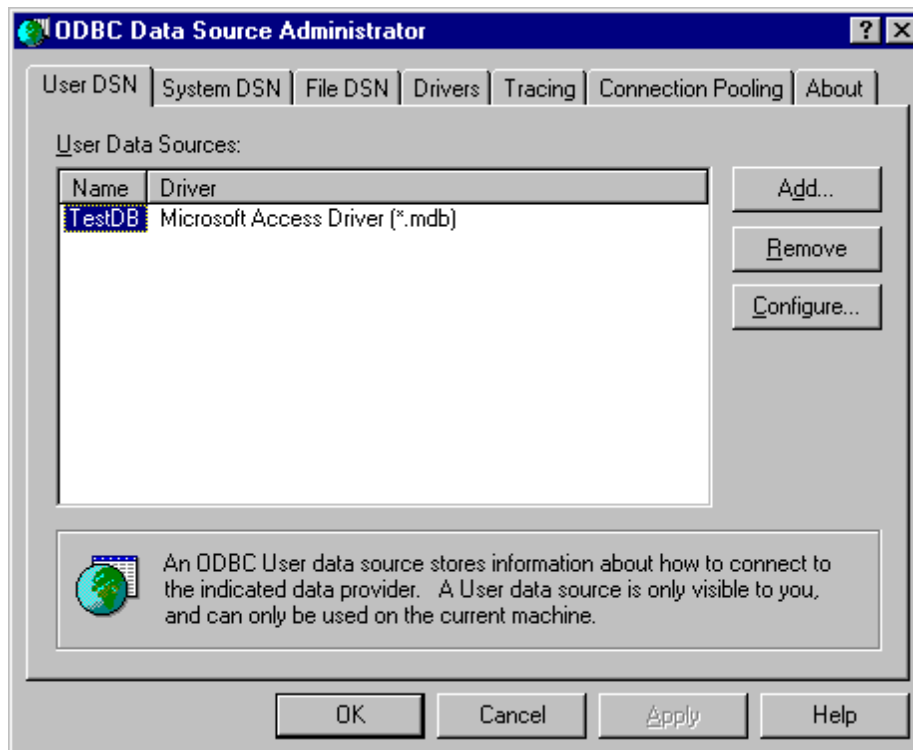
10.4 File ODBC source

If it is neccessary to configure the common data source for several computers in a network, then file ODBC data source is used. The following method gives an example of how to setup File DSN.

Step 1
Double-click on "ODBC Data Sources (32 bit)" icon, and the "ODBC Data Source Administrator" dialog will appear. Choose File DSN tab and you will see the window as shown below. Select the place for storing DSN information. Click on the "Look in" drop-down list. You will see the list of file folders and disk drives on your computer. One possibility is to use the network drive where the database file is placed.
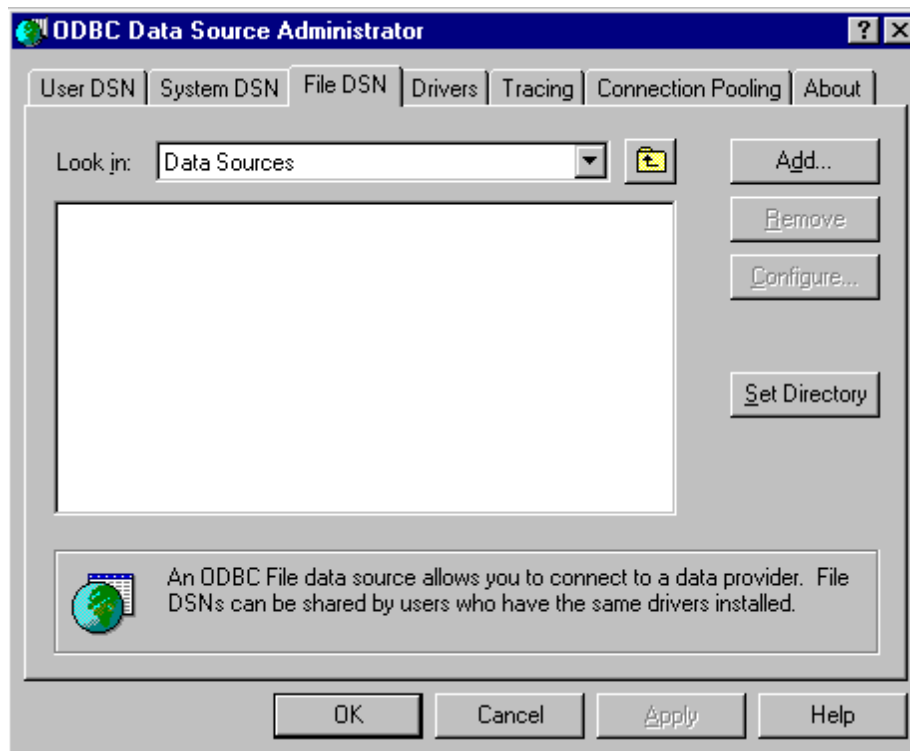
2
Press Add button. The "Create New Data Source" window will appear.
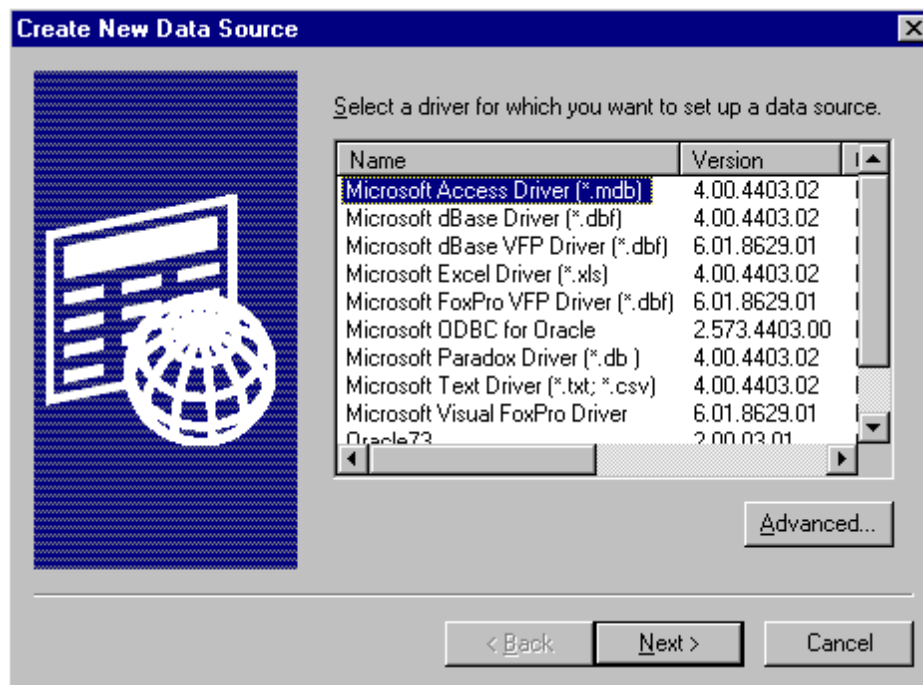
3

Select one of the ODBC drivers and then press NExt button. You will see the following "Create New Data Source" dialog page. Here you should enter the data source name and its location (full path). Press Browse button and you will see the File Open dialog where you can select the storage place. Enter file name and the wizard will automatically add .DSN as an extension.
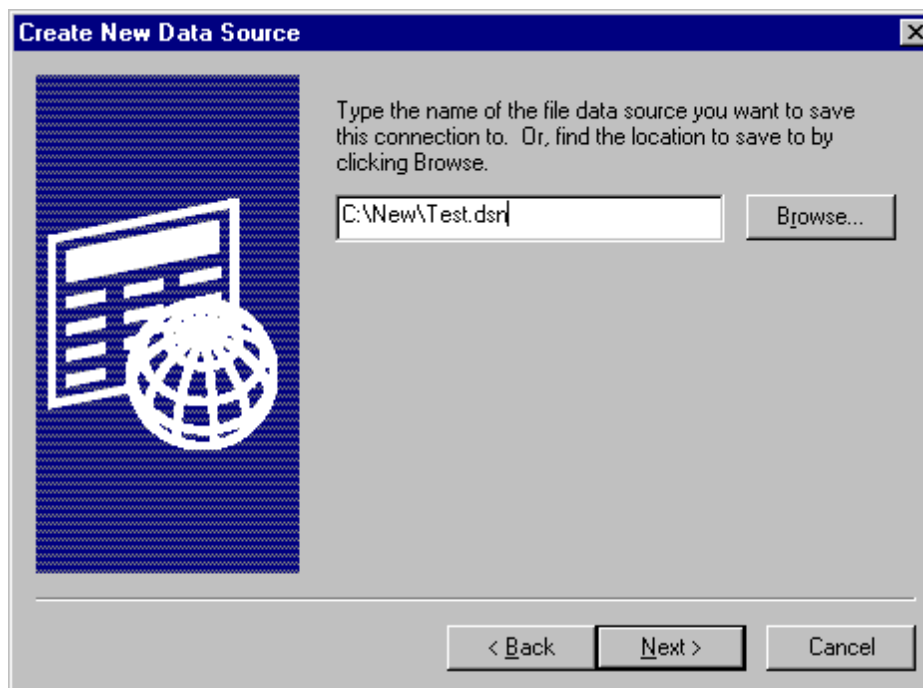
Step 4

Press Next and you will see the final dialog window as shown below. It will report the parameters of DSN

that you are going to create.

5
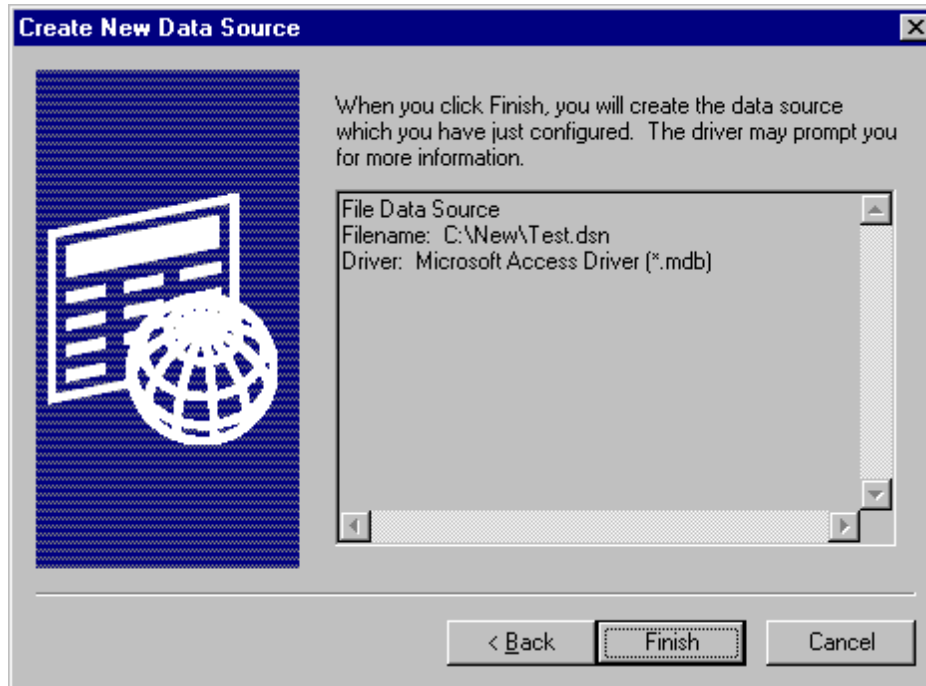
Finish   ODBC Microsoft
Access Setup,   .   Select   ,
.
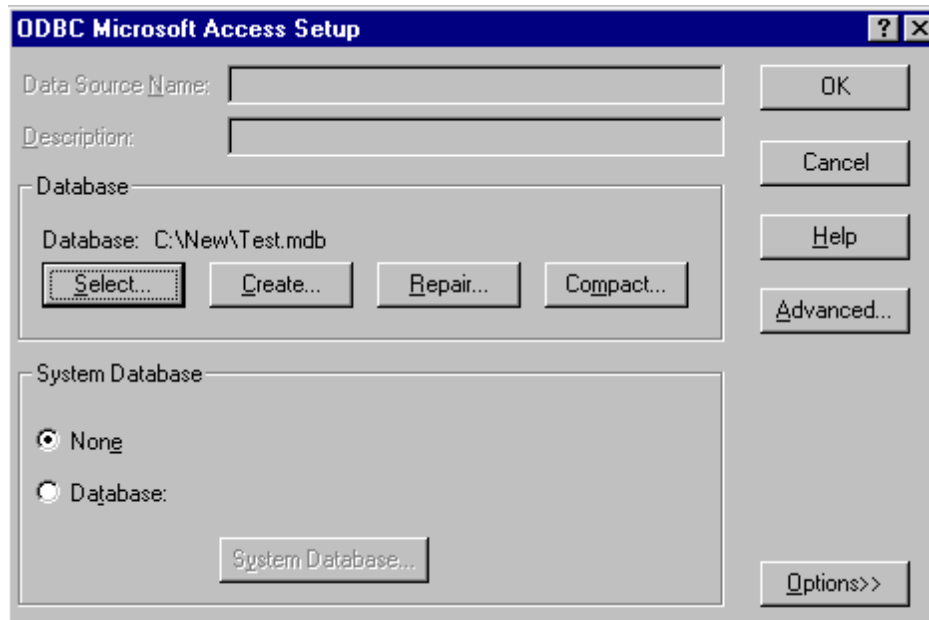
Step 6
Press OK button when you finished the configuration process. You will see a new file DSN element,
similar to shown in "ODBC Data Source Administrator" dialog window.
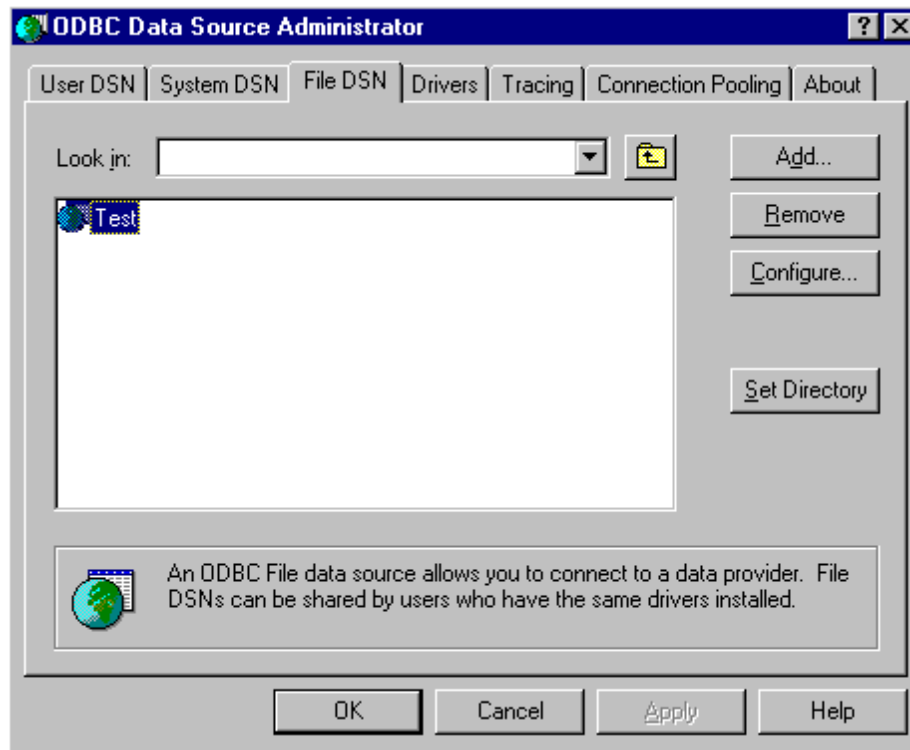
Unlike the usual DSN, this actually creates a file, that you may see and edit in a text editor. Below you may find the file DSN example.
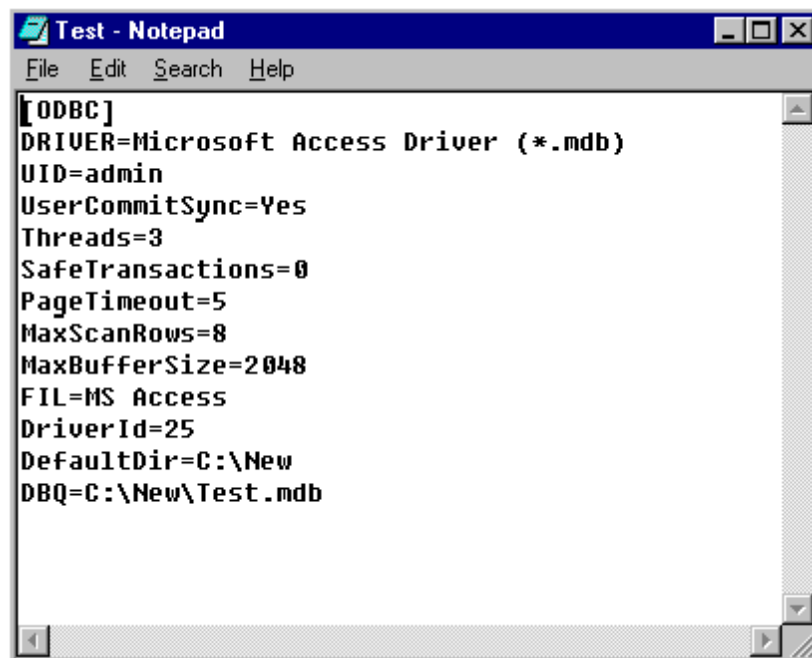
10.5   Tracing ODBC transaction

To trace all operarions being perfomed by ODBC drivers, the transaction tracing can be turned on. To start registering of transactions, you should use "Tracing" tab in the "ODBC Data Source Administrator" window. Here you can setup the name and location of log file indicating it in Log file Path field. Once you press "Start Tracing Now" button, the tracing will begin. the button caption will change to "Stop Tracing Now". Once again press this button to cancel tracing.
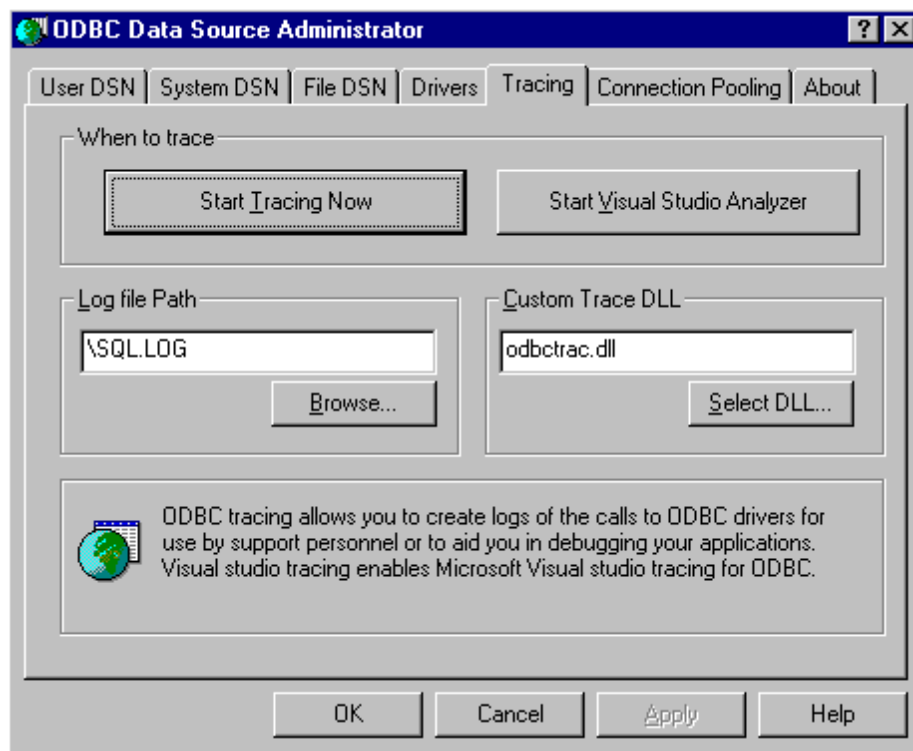
Top Level Intro

This page is printed before a new
top-level chapter starts

Part

XI

11 Data profile setup

11.1 Data profile editting

11.1.1 Data profile editting

New data profile registration
Entering profile parameters
Managing transfers list

11.1.2 New data profile registration

To register a new profile it is neccessary to create profile file and then add it to the Transfer profile list. List of registered profiles is a part of Trans30.ini file. The profile files itself should be located in a special directory, which is configured in the Transfer options (see General options).

All these actions can be done by opening "Data profiles" window from the File menu (see Data profiles). To enter a new profile click the right mouse button on the window and then select "Add" in a context window (you may also use the hot keys combunation - Ctrl+Ins). The new profile window will be opened as shown below.

Enter the following data on this page:
- **File**. File name with PPL extension. this file will be saved in the Transfer profile directory. You can also press [...] button, located next to the field, to select existing profile file. In that case the window will be automatically filled in by data from the selected file.
- **Type**. There are three types – Transfer, Import, Export. Those values identify the profile by its

functional type.

- **Name**. Short profile description reflecting its purpose.
- **Parameters**. Text string containing the list of parameters and their values. To edit parameters press [...] button on the right (see Entering profile parameters).
- **Source**. ODBC data source identifier. It can be selected from the drop-down list showing existing ODBC DSNs (see ODBC source creation).
- **Source UID**. User ID for the source database. It is mandatory for relational databases.
- **Source PWD**. User password for the source database. It is mandatory for relational databases.
- **Destination**. ODBC data destination identifier. It can be selected from the drop-down list showing existing ODBC DSNs (see ODBC source creation).
- **Destination UID**. User ID for the destination database. It is mandatory for relational databases.
- **Destination PWD**. User password for the destination database. It is mandatory for relational databases.

See also Managing transfers list.

Upon completion of entering profile data press Save or Ok button. The profile file will be created and saved in the corresponding directory.

11.1.3 Entering profile parameters

If you press [...] button next to the parameters field in the profile edit window, a dialog window will be opened, as shown below.

The dialog has a table with the following columns:
- **Code**. Parameter identifier. It cannot start with a number and contain blanks or special symbols.
- **Name**. Brief parameter description.
- **Type**. One of available data type – Char, Integer, Numeric, Date, Time, Timestamp (see Data types definition).
- **Value**. Initial value by default. Not mandatory.

To add a parameter in the end of the list use "Add" button. To insert a parameter before the current row use "Insert" button. "Delete" button removes current row from the parameters list. After all parameters are entered press OK button to return to the profile edit window.

11.1.4　Managing transfers list

In the bottom of the profile edit window you may see the list of transfers for this profile. The list shows transfer name and its schema. For more detailed information about possible schemas see Transfer schemas.

There are the following buttons that are used for editting transfer list:
- **Add**. Adds a new transfer to the end of the list.
- **Insert**. Insert a new transfer before the current in the list.
- **Edit**. Opens an edit dialog for the current transfer.
- **Delete**. Deletes the current transfer.
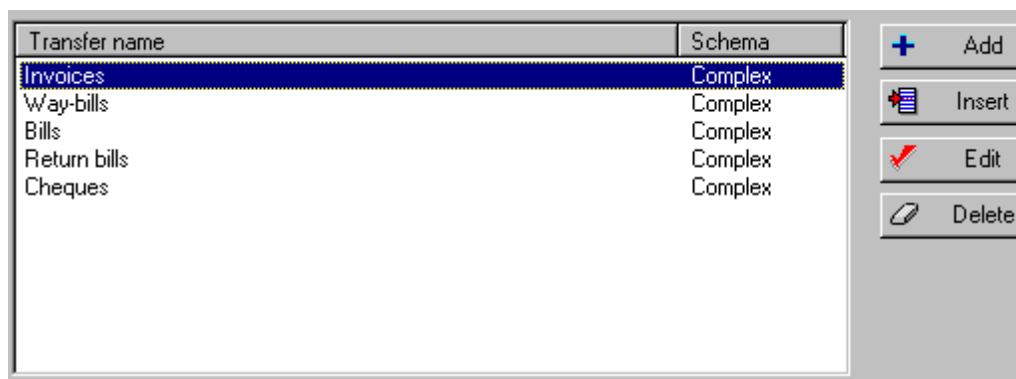
11.2　Transfer edit

11.2.1　Transfer edit

Editting source table data
Editting destination table data
Editting subtable data
Entering fields map
Entering command list

11.2.2　Editting source table data

For transfer editing a dialog window with several pages is opened. Page "Source" contains common transfer parameters, as well as SQL query for the source table:
- **Name**. Brief transfer description.
- **Schema**. Data processing schema: Table, Copy, Complex (for more details see Transfer schemas).
- **Transmission of text data**. Option for possible text data conversion: Without conversion, OEM to ANSI, ANSI to OEM.
- **Message string**. The text string, that will be displayed during transfer execution. It may contain references to the source or destination tables fields (see Prefix symbols). This string is also used for logging into the message file if this option is set.
- **Source SQL Select**. SQL Select to retrieve data from table in the source database. It may contain references to the profile parameters.
- **Subtable SQL Select**. SQL Select to retriev data from the subordinate table in the source database

(see COMPLEX schema processing). It may contain references to the profile parameters and to the master table fields.

11.2.3 Editting destination table data

This page contains the following parameteres:
- **Table**. Mandatory field which determines the table name in the destination database.
- **Process**. Determines the data processing algorithm: Table creation, Add data, Replace data, Full compare (see Transfer execution algorithm).
- **SQL where clause**. Not mandatory, defining additional query condition that will be used for searching data in destination table in case of "Replace data" or "Full compare" algorithms.
- **Table fields map**. For "Table" schema this map is used to define the source and destination tables fileds correlation (see Entering fields map). For "Copy" and "Complex" schemas this can be used for fields redefinition (renaming) or to define special fields flags. See also Field map usage.

11.2.4 Editting subtable data

This page is not mandatory for filling in and is used only for "Complex" schema (see COMPLEX schema processing).

The page contains the following parameteres:
- **Table**. Mandatory field which determines the subtable name in the destination database.
- **Process**. Determines the data processing algorithm: Table creation, Add data, Replace data, Full compare (see Transfer execution algorithm).
- **SQL Were clause**. Not mandatory, defining additional query condition that will be used for searching data in destination subtable in case of "Replace data" or "Full compare" algorithms.
- **Table fields map**. For "Table" schema this map is used to define the source and destination tables fileds correlation (see Entering fields map). For "Copy" and "Complex" schemas this can be used for fields redefinition (renaming) or to define special fields flags. See also Field map usage.

11.2.5 Entering fields map

Fields map is a table with the following columns:
- **Field**. Field identifier in database table.
- **Flag**. Modifier, allowing to define special features for a field. Avauilable values:
  - K - Key field
  - N - Read-only field
  - T - Trigger
  - F - Run/rollback flag
  - For more information about field flags see Field map usage.
- **Type**. One of available data type – Char, Integer, Numeric, Date, Time, Timestamp (see Data types definition).
- **Length**. Maximum length for text data or number of digits for numeric data.
- **Precision**. Maximum precision for numeric data types.
- **Reference (<-)**. Reference prefix. For more information see Prefix symbols.
- **Value**. Not mandatory, depends on the value in "Reference" field. Can be used to assign a constant value to a field or enter a reference.

To add a field in the end of the table use "Add" button. To insert a field before the current row use

"Insert" button. "Delete" button removes current row from the fields map.

11.2.6 Entering commands list

Commands list defines actions to be performed under certain condition during transfer execution. For more information about different command types see Commands list usage. Duplicating commands with the same event type is not allowed - the only exception is OnChange (ReChange) events (in that case values in the "Field" column must differ). Commands list has the following columns:

- **Command**. Any SQL operators (Insert, Update, Delete) or stored procedures calls (using Call command) are allowed here. If it is necessary to run several commands upon one condition, different commands can be written in the same field separated by semicolon. You can use references to the data fields or profile parameters as arguments in SQL operators and stored procedures calls (see Prefix symbols). For more details about commands syntax see Commands list format.
- **Event**. The condition that triggers the command. the following values are used:
  OnStart - When transfer is started
  OnEnd - When transfer is finished
  CallExec - When data block is "registered". Only for COMPLEX schema.
  CallBack - When data block is "unregistered". Only for COMPLEX schema.
  OnInsert - On each row insertion
  OnUpdate - On each row update
  OnDelete - On each row deletion
  ReInsert - On each row insertion rollback
  ReUpdate - On each row update rollback
  ReDelete - On each row deletion rollback
  OnChange - On particular field change
  ReChange - On particular field change rollback
- **Field**. Destination table field identifier. Used only for OnChange event.

To add a command in the end of the list use "Add" button. To insert a command before the current row use "Insert" button. "Delete" button removes current row from the commands list.

11.3 Data profile syntax

11.3.1 Data profile syntax

Profile structure description
Filed map format
Command list format
Data types definition
Prefix symbols
Embedded functions

11.3.2 Profile structure description

PARAMETERS{
[type1 param1(name1)[=val1], …, typeN paramN(nameN)[=valN]]
 type=<char, integer, numeric, date, time, timestamp>
}
CONNECT{
Source(DSN=;UID=;PWD=)

Destination(DSN=;UID=;PWD= **)**
}
TRANSFER{
Name(*Description* **)**
Schema(<COLUMNS, COPY, COMPLEX> **)**
ConvertText(<UNCHANGED, OEMTOANSI, ANSITOOEM> **)**
Source(*source table SQL Select* **)**
SubSQL(*subtable SQL Select* **)**
Message(*message string* **)**
Destination(**process=**<CREATE,ADD,REPLACE,FULL>
owner=<*DB schema*>
table=<*destination table*>
condition(*SQL Where clause for destination table* **)**
columns(*fields map* **)**
see [Fileds map format](#)
)
Detail(**process=**<CREATE,CLEAR,ADD,REPLACE,FULL>
 owner=<*DB schema*>
 table=<*subtable*>
 condition(*SQL Where clause for destination subtable* **)**
 columns(*fields map* **)**
see [Fileds map format](#)
)
TabCommands(*Destination table commands list* **)**
SubCommands(*Subtable commands list* **)**
See [Commands list format](#)
}
…
TRANSFER{
}

Data profile file usually contains one PARAMETERS section, one CONNECT section and at least on or several TRANSFER sections.

PARAMETERS section holds the list of parameters for the whole profile, meaning that the parameters values can be used in the fields map or the commands list. This section is not mandatory.

CONNECT section describes ODBC connections parms for source and destination databases. The section is mandatory for the data profile. For the file ODBC sources UID   PWD parameters are not required.

TRANSFER section defines the structure of the data block to be transmitted.

Name	Data block designation
Schema	Data transfer schema. Available values are:
	COLUMNS - data transmission according to the fields map
	COPY - copying the source table data structure into the destination database
	COMPLEX - transferring connected data using model "master-detail"
	For more information see [Transfer schemas](#).
ConvertText	Option for possible text data conversion:
	UNCHANGED - without conversion

	OEMTOANSI - convert OEM to ANSI format
	ANSITOOEM - covert ANSI to OEM format

Source   SQL Select to the source table (master).

SubSQL   SQL Select to the subordinate table (detail or subtable).

Message   The text string, that will be displayed during transfer execution. It may contain references to the source or destination tables fields (see Prefix symbols). This string is also used for logging into the message file if this option is set.

Destination   This section defines the structure of the master table in the destination database. There are the following subsections:

> **process**   = Data processing algorith, possible values are:
> CREATE   - table creation in the destination database. If the table already exists, it will be deleted and re-created
> ADD   - only adding data to the destination table (unconditional INSERT operators)
> REPLACE   - data replacement in the destination table (INSERT or UPDATE operator depending on whether the row with the same key already exists or not)
> FULL   - full comparison. This is two-pass process. The aim of this process is to have identical data in source and destination (in the scope of given SQL query). Data missing in the source table are deleted from destination. Data missing in the destination table are added to the destination.
> **owner**   = *DB schema or owner of the destination table*
> **table**   = Destination table name
> **condition** = SQL Where clause for destination table
> **columns**   = Fields map (see Fields map format)

Detail   This section defines the structure of the subordinate (detail) table in the destination database. It has several subsections with the same structure as "Destination" section.

TabCommands List of commands for destination table (See Commands list format).

SubCommands List of commands for subordinate table (See Commands list format).

11.3.3 Fileds map format

Fields map is intended to define the correlation between columns of source and destination tables. The syntax includes the destination table column name (optionally starting with the flag separated by a colon) with the data type and length following by an "equal" sign and then reference or a constat (see Prefix symbols):

[flag:]field1[(type[,lengh,precision])]=src ref1;
…
[flag:]fieldN[(type[,lengh,precision])]=src refN;

Flag	**Description**
K	Key field. Defins a unique table index.
N	Read-only field. When table is updated, this field is not modified.
T	Trigger. When the field value is changed during transfer, the

"registration" mechanism is triggered.

F "Registration" flag for data block. Used only in COMPLEX schema.

type=<BT,TI,SI,BI,I,R,F,DB,DC,N,A,VA,LVA,B,VB,LVB,D,T,DT,TS>
See Data types definition

Fields map examples:

Example 1.
columns(T:sklad;T:code_currency;T:cur_code_currency;F:flag;N:typ_rate;N:sum_link_sd))

Example 2.
columns(T:id_ware;proc_tax1=<prc_tax1;proc_tax2=<prc_tax2;proc_tax3=<prc_tax3; N:price_spc))

Example 3.
columns(K:ndprt(I)=$ndprt;K:id_ware(I)=$id_ware;id_parent(I)=$id_parent;
last_ware(I)=$last_ware;flag_ware(I)=$flag_ware;flag_made(I)=$flag_made;
comp_ware(I)=$comp_ware;code_unit(I)=$code_unit;code_country(I)=$code_country;
id_pro(I)=$id_pro;id_ser(I)=$id_ser;name_ware(VA,80)=$name_ware;
name_ware2(VA,80)=$name_ware2;code_ware(VA,30)=$code_ware;
code_gr(VA,255)=$code_gr;barcode(VA,30)=$barcode;marka_ware(VA,40)=$marka_ware;
okdp_ware(VA,20)=$okdp_ware;post_ware(VA,30)=$post_ware;gost_ware(VA,30)=$gost_ware;
acnt_ware(VA,10)=$acnt_ware;weight(DB)=$weight;pack(DB)=$pack;cost_ware(DB)=$cost_ware;
count_ware(DB)=$count_ware;t_tax1(I)=$t_tax1;t_tax2(I)=$t_tax2;t_tax3(I)=$t_tax3;
v_tax1(DB)=$v_tax1;v_tax2(DB)=$v_tax2;v_tax3(DB)=$v_tax3;wdata(TS)=$wdata;t_info(LVA)))

11.3.4 Commands list format

Commands list is used for defining the event-driven data processing during transfer.

Event(*command1*[; … *commandN*])
Event=<OnStart, OnEnd, CallExec, CallBack,
         OnInsert, OnUpdate, OnDelete,
         ReInsert, ReUpdate, ReDelete,
         OnChange.*field*, ReChange.*field*>

Command=<Insert, Update, Delete, Call *procedure*>

It is allowed to define several Insert, Update, Delete operators or procedures calls separated by semicolon within one command.

Command types (events):

Event	Description
OnStart	When transfer is started
OnEnd	When transfer is finished
CallExec	When data block is "registered". Only for COMPLEX schema.
CallBack	When data block is "unregistered". Only for COMPLEX schema.
OnInsert	On each row insertion
OnUpdate	On each row update

OnDelete   On each row deletion
ReInsert   On each row insertion rollback
ReUpdate   On each row update rollback
ReDelete   On each row deletion rollback
OnChange   On particular field change
ReChange   On particular field change rollback

It is possible to use any prefixes in SQL commands or stored procedures calls (see Prefix symbols) for references to profile parameters or data fields of source and destination.

Example of commands list:

TabCommands(
CallExec(update sdoc set flag=1
where type_sd=$type_sd and numref=$numref and year=$year and
    ndprt=$ndprt and autokey_sd=$autokey_sd)
CallBack(update sdoc set flag=0
where type_sd=$type_sd and numref=$numref and year=$year and
    ndprt=$ndprt and autokey_sd=$autokey_sd)
OnInsert(call BAL_MOD($numref,$year,$cli1_ndprt,$cli1_numb_client,
$code_currency,^0,$sumf_sd,$date_sd,$flag))
OnUpdate(call BAL_MOD($numref,$year,$cli1_ndprt,$cli1_numb_client,
$code_currency,#sumf_sd,$sumf_sd,$date_sd,#flag))
OnDelete(call BAL_MOD(#numref,#year,#cli1_ndprt,#cli1_numb_client,
#code_currency,#sumf_sd,^0,#date_sd,#flag))
ReInsert(call BAL_MOD($numref,$year,$cli1_ndprt,$cli1_numb_client,
$code_currency,$sumf_sd,^0,$date_sd,$flag))
ReUpdate(call BAL_MOD($numref,$year,$cli1_ndprt,$cli1_numb_client,
$code_currency,$sumf_sd,#sumf_sd,$date_sd,#flag))
ReDelete(call BAL_MOD(#numref,#year,#cli1_ndprt,#cli1_numb_client,
#code_currency,^0,#sumf_sd,#date_sd,#flag))
OnChange.date_sd(update card_spc set date_card=$date_sd
where numref=$numref and year=$year and sklad=$sklad and
    sds_numref=$numref and sds_year=$year and
    type_sd=$type_sd and sds_ndprt=$ndprt and
    autokey_sd=$autokey_sd)
OnChange.cli1_numb_client(call BAL_MOD(#numref,#year,
#cli1_ndprt,#cli1_numb_client,#code_currency,#sumf_sd,^0,#date_sd,#flag);
call BAL_MOD($numref,$year,
$cli1_ndprt,$cli1_numb_client,$code_currency,^0,#sumf_sd,$date_sd,#flag);
update card_spc set cli_ndprt=$cli1_ndprt,
    numb_client=$cli1_numb_client
where numref=$numref and year=$year and sklad=$sklad and
    sds_numref=$numref and sds_year=$year and
    type_sd=$type_sd and sds_ndprt=$ndprt and
    autokey_sd=$autokey_sd)
ReChange.date_sd(update card_spc set date_card=#date_sd
where numref=$numref and year=$year and sklad=$sklad and
    sds_numref=$numref and sds_year=$year and
    type_sd=$type_sd and sds_ndprt=$ndprt and
    autokey_sd=$autokey_sd)
ReChange.cli1_numb_client(call BAL_MOD($numref,$year,
$cli1_ndprt,$cli1_numb_client,$code_currency,#sumf_sd,^0,$date_sd,#flag);

```
call BAL_MOD(#numref,#year,
#cli1_ndprt,#cli1_numb_client,#code_currency,^0,#sumf_sd,#date_sd,#flag);
update card_spc set cli_ndprt=#cli1_ndprt,
      numb_client=#cli1_numb_client
where  numref=$numref and year=$year and sklad=$sklad and
      sds_numref=$numref and sds_year=$year and
      type_sd=$type_sd and sds_ndprt=$ndprt and
      autokey_sd=$autokey_sd))
```

11.3.5 Data types definition

Definition	ODBC SQL type	Description
BT	Bit	Single bit
TI	TinyInt	Integer with a length of 3 in a range of -128 ~ 127 (unsigned 0 ~ 255)
SI	SmallInt	Integer with a length of 5 in a range of -32768 ~ 32767 (unsigned 0 ~ 65535)
BI	BigInt	Integer with a length of 19 in a range of $-2^{63} \sim 2^{63}-1$ (unsigned 0 ~ $2^{64}-1$)
I	Integer	Integer with a length of 10 in a range of $-2^{31} \sim 2^{31}-1$ (unsigned 0 ~ $2^{32}-1$)
R	Real	Signed numeric value with the binary length of 24 in a range of 0 ~ 10^{38}
F	Float	Signed numeric value with defined binary length p.
DB	Double	Signed numeric value with the binary length of 53 in a range of 0 ~ 10^{308}
DC	Decimal	Signed numeric value with the assigned length of p and precision s. (Maximum length depends on the ODBC driver). 1 <= p < 15, s <= p
N	Numeric	Signed numeric value with the assigned length of p and precision s. 1 <= p < 15, s <= p
A	Char	Text string with the fixed length n.
VA	VarChar	Variable length text string with the maximum length n.
LVA	LongVarChar	Variable length text string. Maximum length depends on the ODBC driver.
B	Binary	Binary data with the fixed length n.
VB	VarBinary	Variable length binary data with the maximum length n.
LVB	LongVarBinary	Variable length binary data. Maximum size depends on the ODBC driver.
T	Time	Time in hours, minutes and seconds.
D	Date	Gregorian date, month and year.
TS	Timestamp	Date and time.

11.3.6 Prefix symbols

Prefix symbols are used to make references to profile parameters or source and destination data fields in the fields map or commands list. After the prefix symbol the data field name or a constant is

indicated.

Symbol	Description	Comments
^	Value (constant)	The constant value will be set in the field for every SQL Insert and Update operation.
+	Autoincrement	The constant in the field indicates the increment value. Autoincrement value will be calculated for every SQL Insert operator, but not for Update. The value is calculated as the maximum field value in the table plus autoincrement constant.
:	Profile parameter	Data from the parameter field will be set in the field for every SQL Insert and Update operations.
$	Field in the master source table	Data from the current row in the main source table will be used in SQL Insert and Update operations.
~	Field in the subordinate source table	Data from the current row in the subordinate source table will be used in SQL Insert and Update operations.
#	Field in the master destination table	Data from the current row in the main destination table will be used in SQL Insert and Update operations.
!	Field in the subordinate destination table	Data from the current row in the subordinate destination table will be used in SQL Insert and Update operations.
<	Field redefinition	Mainly used in COPY schema when some columns names differ in source and destination tables. Data from the current row in the main source table will be set in the field.
%	Embedded functions	Embedded functions allow to call predefined SQL queries for complex data processing. For more information see Embedded functions.

11.3.7 Embedded functions

Embedded functions are the functions that can be used in the "Value" column of the fields map. To indicate the embedded function in the fields map the special prefix % is used. The function value is calculated during processing of each record in the destination table.

Currently Transfer supports the following types of embedded functions:

Function	Parameters	Description
SQLGetVal	field, table, where expr	Returns the *field* value from the *table* based on the *where expr* criterion. If more than one row is returned, then the value in the first row is used.
SQLMaxVal	field, table, where expr	Returns the maximum *field* value from the *table* based on the *where expr* criterion.
SQLMinVal	field, table, where expr	Returns the minimum *field* value from the *table* based on the *where expr* criterion.
SQLAvgVal	field, table, where expr	Returns the average *field* value from the *table* based on the *where expr* criterion.

SQLVarVal	field, table, where expr	Returns variation of the *field* value from the *table* based on the *where expr* criterion.
SQLSumVal	field, table, where expr	Returns sum of the *field* values from the *table* based on the *where expr* criterion.
SQLCount	table, where expr	Returns the number of rows in the *table* satisfying the *where expr* criterion.

Top Level Intro

This page is printed before a new
top-level chapter starts

Part

XII

12 Data processing algorithms

12.1 Transfer schemas

The most important and gloabl attribute of the transfer is the execution schema. Schema defines the common principle of data transmisson within particular transfer. Schema also has an impact to all other transfer parameters.That is why choosing the correct schema for the transfer is very important step.

Tthe following transfer schemas are supported:
- **Table**. This is the most simple type of data transmission. Usually it is used while converting data from one database to another, where the source and destination databases can be absolutely different (for example, ORacle and MS Access). It is mandatory for this schema to define the fields map, as it is assumed that source and destination tables have different structure. Tables in source and destination databases may have different number of columns, column names or types. All those problems can be easily resolved using fields map. In addition all data types conversion is done automatically due to internal universal data types representation.
- **Copy**. This schema is used when source and destination tables have almost (or exact) identical structure. This means that they have equal number of columns with the same name (not mandatory) and similar data type. However this does not mean that source and destination databases should be the same. Using Copy schema allows to avoid detailed fields correlation definition. In that case fields map can be used for fields redefinition (renaming) or for assigiing special field flags (see Fields map usage).
- **Complex**. This is the most complicated data processing algorithm. It is used to define the transmission process of connected data with master-detail model. It is supposed by default that in "Complex" schema source and destination tables should be identical. Therefore this schems is further development of "Copy" schema. For "Complex" schema the subtable term is introduced. Subtable is a subordinate table to the master table in the destination (source) database. Data in those tables are considered as the unified information block: each row in the master table is logically connected with the set of rows in the subordinate. If data in the subordinate table cannot be transmitted for some reason, then the connected row in the master table should not be transmitted either. For "Complex" schema it is mandatory to define parameters for subtable in the transfer profile. More detailed description of this algorithm can be found in COMPLEX schema processing.

12.2 Transfer execution algorithm

Transfer execution algorithm for the given schema is defined by "Process" parameter. This parameter is set for the master table and subtable separately. Let's consider algorithms for different "Process" parameter values.

Create table
This algorithm implies forced table creation in the destination database before running the transfer. If the table with this name already exists, then it is first deleted to guarantee correct data structure. If "Table" schema is defined, then the fields map is used to create a table. It is neccessary that at least one primary key (K flag) was defined in the fields map, otherwise the table will not be created. If "Copy" schema is used, then the table structure is copied from the source table. Since before the transfer the destination table is empty, the transmission process adds up to the unconditional data move from source to destination using SQL Insert operators. Before Insert the destination table is not verified whether the row with the same key already exists. During transfer execution the following events can occur (the corresponding commands can be triggered): OnStart, OnInsert, ReInsert, OnEnd.

Add data

Adding data algorithm is very convenient for transmissin/conversion of large data blocks. It can be used only when it is known that the destination table does not contain data we are going to transmit. The benefit of this method is in data processing speed, because during the transfer execution the destination table is not verified whether the record with a particular key already exists. This saves time that can be critical for the large data blocks transmission. Data is transmitted using SQL Insert statemenets to the destination table with data taken from the source table. If during execution the duplicating record error occurs, the program will show this error and ask a user to either stop or continue operation. User has an option to suppress error messages that will alow him to finish the data transfer session. Anyway, all error messages will be saved in the log file that can be viewed after transfer completion. The following events can be triggered during execution: OnStart, OnInsert, ReInsert, OnEnd.

Replace data

This algorithm is used only when it is required to guarantee that all data (according to the SQL query) is transmitted from source to destination. There can be a situation when records with the same key exist in both source and destination. In that case data in the destination table should be replaced by data from the source table. To provide such functionality it is neccessary to test every data record being transmitted by searching in the destination table by the unigue key. If the record does not exist, then SQL Insert operator is executed. If the row with this key is found the the target table, then this record is modified with the data from the source table using SQL Update operator. The following events can be triggered during transfer execution: OnStart, OnInsert, ReInsert, OnUpdate, ReUpdate, OnChange, ReChange, OnEnd.

Full compare

Previous algoritm ensures that all required data are transmitted from source to destintion, however it does not guarantee that data in the target table is absolutely identical to the source. The point is that there can be records in the destination table that are missing in the source table. But the replacing algorithm does not detect this, since it would require two-pass data comparision process, therefore increasing the overall execution time. Nevertheless, in some cases it is required to provide an absolute identity of data in source and destination, for example in COMPLEX schema. Full comparison algoritm does two passes through the source and destination table during execution. In the first pass through the destination table it detects records missing in destination. Tose rows are deleted using SQL Delete operator. then in the second pass the data replacing algorithm from source to destination is actually performed. Note, that data is compared only within the scope of SQL query to the source table. The following events can be triggered during transfer execution: OnStart, OnDelete, ReDelete, OnInsert, ReInsert, OnUpdate, ReUpdate, OnChange, ReChange, OnEnd.

12.3   COMPLEX schema processing

COMPLEX schema is intended for transmission of connected information using master-detail model. Good example of such information is financial documents.

For instance, the invoice usually has some header information and specification. The header contains document number, date, warehouse, customer name. The specification is a list of goods with quantity and price. When a document is transmitted between two databases, it is usually required that the document is recieved as the whole entity, i.e. both header and specifiation. There should not be a situation when the header exists but the specification is lost or presented just partially. In such cases we cannot guarantee the data integrity. In other words, the main idea is the following: either the document exists or it is not present at all.

There is another aspect of this matter. Usually the financial/bookkeeping documents (like invoice) are used to change some accounting registers, for example payment balance or warehouse rest. Therefore, when registered the document should perform some additional actions in the database. Very often such actions are designed in a form of database stored procedures. When the document is imported in the database of the financial application, it is logical to assume that the same actions should be done as if the document was entered by a user manually. In reality the situation is even much more complex as it may seem on the first glance. The matter is that here we face with the questions on how to transmit documents already present in the system but changed (for example, the invoice was already imported once, but then the specification was changed in the source databse and then the document is being transmitted again). During the repeated transfer it is neccessary to process all changes in the document correctly having in mind the possible modifications in accounting registers (using stored procedures calls). To simplify and automate the document processing technology the "registration" term was introduced in the tool (which is in fact the bookkeeping term). "Registered" document means that all accompanied actions are done (accouting transactions etc). Also, the document is considered "unregistered" if it exists in the database, but the corresponding actions are not executed. Usually, to reflect the "registration" status of the document a special column in the document tables is introduced in the financial/bookkeeping applications.

Taking into account all above, the COMPLEX schema functions in the following way:
1. Depending of the process type defined for the master table (usually it is "Replace data"), the record in the master table is transmitted. First, SQL Insert (Update) operator is performed. If there are no errors, then the associated transfer command (OnInsert, OnUpdate) is executed. If it succeeded, then the program starts processing subtable. If not, then in case of Update data modification is cancelled and Transfer is going to the next record. If SQL Insert operator is failed, then the CallBack transfer command is run (if defined). This can be required to clear the "registration" flag in the destination table. Further, the subtable records are processed, however in that case the associated commands will not be triggered (as the document is treated as "unregistered").
2. If for any field in the master table F flag is defined and the field value in the source data is <u>greater</u> than in the destination, then OnInsert transfer command will be executed. The subordinate table is processed twice. First, data in source and destination tables is made identical (of course by implementing changes in the destination table). Then, for each record in the destination subtable only OnInsert transfer commands are executed. If subtable was not processed successfully, then the transfer command for the master table is rolled back. Also, CallBack command is called to clear the "registration" flag.
3. If for any field in the master table F flag is defined and the field value in the source data is <u>less</u> than in the destination, then OnDelete transfer command will be executed. The subordinate table is processed twice. First, for each row in the subtable only OnDelete transfer commands are executed. If this operation failed, then the transfer command for the master table is rolled back. Also, CallExec command is called to restore the "registration" flag. Then, data in source and destination tables is made identical without running associated commands.
4. Further, the fields that have T (trigger) flag are verified in the master table. If the value of such field is modified, this means the whole record (data block) should be deleted and re-inserted. Therefore, if the record existed in the destination table, then the data is deleted from the subtable with associated commands execution, and then the record itself is deleted. Then the information is transmitted again using associated commands calls.
5. Subtable is processed according to the process type (usually it is defined as "Full compare"). For each row in the subtable the corresponding SQL operators are performed and associated transfer commands are executed. All performed actions are recorded in a special temporary buffer. For each row in the subtable the fields with T (trigger) flag are verified.
6. If during subtable processing any SQL operator returns error for a row, then all previous actions for the whole records set are cancelled using information from the temporary buffer (all inserted rowas are deleted, all deleted or modified - restored). Also all associated commands are rolled back (for that purpose the "reverse" command are defined: ReInsert, ReUpdate, ReDelete). The processing

of subtable is stopped, and for the master table all previous actions are also cancelled. Further, Transfer is moving to the next record in the master table.

7. If for some reason the associated transfer command cannot be executed for the subtable record (i.e., there are no goods in stock), further scenario will depend on the SQL operator type in the master table.

 • If it was SQL Insert, then all previous transfer commands (not SQL operators) in the subtable are rolled back using information from the temporary buffer. However, subtable processing is not stopped, but all subsequent records are processed without associated commands. After the subordinate table is done, the associated commands for the current row in the master table are also rolled back and the CallBack transfer command is called to clear the "registration" flag.

 • If SQL Update was performed for the master table, then all actions in the subtable are cancelled (including SQL operators). Subtable processing is completely stopped, and all actions are cancelled for the master table as well. Then transfer is moving to the next row in the master table.

12.4   Fields map usage

Fields map is used for several purposes: to define the correlation between source and destination table fields in "Table" schema, to redefine fields for "Copy" and "Complex" schemas, and also to define special flags. Fields map usage in the "Table" schema is described in Entering fields map. The syntax to write in the profile file can be found in Fields map format.

Redefinition (renaming) of the fields is used for the "Copy" schema, where the source and destination tables are almost identical, but some columns may have different names (with the similar data type). In that case the record can be added in the fields map where in the "Field" column the destination field name is indicated, and in the "Refernce (<-)" column a special symbol "<" is selected, meaning redefinition. In the "Value" column the source field name should be place. The field type is not required here as it is assumed that the data types are almost identical (i.e., both have numeric type). Also, the special flags can be introduced for the field: N, K, F, T.

Flag N (non-changable field).
If it is neccessary for the field value in the destination table to remain unchanged during the transfer, this flag is set. However, if a new record is inserted in the destination table, the value in this field will be copied from the source.

Flag K (key field).
This flag means that the field is a unique key or a part of the compound key. Key definnition is mandatory for at least one field in the "Table" schema. For "Copy" and "Complex" schemas the information about key columns is taken from the source table as it is assumed that the source and destination tables have identical structure.

Flag F ("registration" flag).
This flag indicates that the field is used as a "registration" indicator for the document. If the value in this field differs during transmisison of the existing row, then the special mechanizm is used for the whole document to "register" or "unregister" it (see COMPLEX schema processing).

Flag T (trigger).
The purpose of this flag is to indicate that this field value is critical for the whole document (if set for the master table). In other words, if the value in a master table record is changed, then the document should be deleted and re-inserted. Example: if the warehouse identifier was changed in the invoice document, then we need to cancel the accounting transaction for the old warehouse and run the corresponding transaction for another warehouse. This is equal to the invoice deletion and registering it again for the

new warehouse. See COMPLEX schema processing).

12.5 Commands list usage

Commands list is used to define the event-driven scenario to perform some associated actions during data transmission. Commands are tied to the particular SQL operators performed in the destination table. Most often the commands list is used in "Complex" schema, however there are no any limitations to use it in any other schemas. The meaning of the columns in the commands list table see in Entering commands list.

To enter a command in the commands list you need first to indicate an event that will trigger this command. Duplicating commands with the same event type is not allowed - the only exception is OnChange (ReChange) events (in that case values in the "Field" column must differ). If it is necessary to run several commands upon one condition, different commands can be written in the same field separated by semicolon. Any SQL operators (Insert, Update, Delete) or stored procedures calls (using Call command) are allowed in the "Command" field. You can use references to the data fields or profile parameters as arguments in SQL operators and stored procedures calls (see Prefix symbols). For more details about commands syntax see Commands list format.

The commands can be defined for the following events:
OnStart
Command is executed before transfer start. For subtable it is executed for each row procession in the master table. If this command fails, then transfer is stopped.

OnEnd
Command is executed after transfer is finished. For subtable it is executed for each row procession in the master table.

OnInsert
Command is executed **after** SQL Insert operator and only if it was successful.

ReInsert
For "Complex" schema it is required to define this command if OnInsert command was defined. It is used to rollback the results of OnInsert command execution. Therefore, here the SQL operators should be entered that perform the reverse actions with respect to OnInsert command.

OnUpdate
Command is executed **after** SQL Update operator and only if it was successful.

ReUpdate
For "Complex" schema it is required to define this command if OnUpdate command was defined. It is used to rollback the results of OnUpdate command execution. Therefore, here the SQL operators should be entered that perform the reverse actions with respect to OnUpdate command.

OnDelete
Command is executed **before** SQL Delete operator. if the command was successful, only then SQL Delete operator is performed.

ReDelete
For "Complex" schema it is required to define this command if OnDelete command was defined. It is used to rollback the results of OnDelete command execution. Therefore, here the SQL operators should

be entered that perform the reverse actions with respect to OnDelete command.

OnChange
It is used for certain actions that needs to be done if the value in this field was modified. For this event "Field" value is mandatory.

ReChange
For "Complex" schema it is required to define this command if OnChange command was defined. It is used to rollback the results of OnChange command execution. Therefore, here the SQL operators should be entered that perform the reverse actions with respect to Onchange command.

CallExec
It is used **only** for "Complex" schema (see COMPLEX schema processing). It is neccessary to define this command if the master table of the document contains "registration" indicator field. CallExec event is happening during document "registration".

CallBack
It is used **only** for "Complex" schema (see COMPLEX schema processing). It is neccessary to define this command if the master table of the document contains "registration" indicator field. CallBack event is happening during document "unregistration".

12.6   Automatic data exchange

This mode is intended for automatic data exchange between central office and remote branch using Microsoft Access files. The
data exchange algorithm is dependant on "Department flag" settings (see Data exchange options). One of the requirement for proper data exchange is the connection between two remote computer via modem or ethernet network. Access to the remote branch computer folders should be provided for the central office computer. For the modem connection this can be achieved using "remote-access server" Mincrosoft service that should be run on the remote host. The Transfer program on the central office computer is playing an active role in the hosts' interaction. The data echange algorithm is the following:

1.   Transfer program on the remote branch computer is running in the waiting mode, inspecting periodically the "Receive directory" on the same computer after the "Interrogate interval" (given in seconds in Data exchange options).
2.   After "Exchange interval" (given in minutes in Data exchange options), the "central" Transfer program runs export profile, wich is set as a default in General options. At the same time, before the actual export the export database file (Microsoft Access file) is replaced by an empty file to minimize the resulting file.
3.   After the export is done, the export file is renamed to the import file name and the is compressed using embedded LZW archiver. As a result the archive export file is created with the "Archive name" as it is configured in Data exchange options.
4.   The obtained file is then copied to the remote "Send directory" (which is configured as "Receive directory" in the remote computer Transfer options). Actually the file is copied with the .tmp extension and then renamed to its natural extention after the copy operation is complete.
5.   Then Transfer on the central office computer goes to the waiting mode, starting to poll the remote "Receive directory" (which is configured as "Send directory" in the remote computer Transfer options) after the "Interrogate interval" (given in seconds in Data exchange options).
6.   As soon as the "branch" Transfer discovers the archive file in its "Receive directory", it immediately copies this file to the local import directory and uncompress it, deleting the archive file.
7.   "Branch" Transfer runs the default import profile.

8. "Branch" Transfer runs the default export profile.
9. "Branch" Transfer compress the resulting export file using embedded LZW archiver. the arcive file is placed in the local "Send directory".
10. As soon as the "central" Transfer discovers the archive file in its "Receive directory" (which is configured as "Send directory" in the remote computer Transfer options), it starts copying it to its local import directory. After the copy is complete the archive file on the remote computer is deleted. The obtained file is uncompressed and the archive file is also deleted.
11. "Central" Transfer runs default import profile. With that the exchange cycle is finished. The next cycle will be initiated by the "central" Transfer after the "Exchange interval".

Top Level Intro

This page is printed before a new
top-level chapter starts

Part

XIII

13   Examples

13.1   Examples list

The following examples are installed with the program:

Example data profile files are located in the folder: .\PPL\. Their names are, respectively: books.ppl, music.ppl, trans.ppl.

In all examples the Microsoft Access database is used. Therefore, to use the examples you need to have the Microsoft Access software installed on your PC. During examples installation the neccessary ODBC data sources are created automatically and the following database files are installed:
.\In\Import.mdb
.\Out\Export.mdb

13.2   Using Table schema - "Books"

File: books.ppl.

In this example the Table schema technic is demonstated. Data profile contains two transfers: Authors and Books. At the same time only one table from Import.mdb database is used as a data source - Books. Intially the table is not normalized, as it contains redundant data about books authors. The data profile is created in such a way so as to provide that each row in the table would contain only unique data. To reach that the transfers are defined to create two tables in the destination database: Authors and Books.

In the first transfer we fill in the Authors table fetching the unique author names from the Books table. To generate the unique key the autoincrement prefix (+) is used in "ID_Author" field definition. Further, in the second transfer all columns are copied except the "author" field. Instead of that, in Books table of the destination database we define the "id_author" field, which should contain the record ID from the Authors table. To obtain the code by the author name, the embedded SQL-function SQLGetVal() is used.

Pay attention to the fact that both transfers contain the fields with the K (key) flag in their table maps. This is the mandatory condition, otherwise the ODBC driver will not create a table. Also, field "size" in the Books table is defined with the different data type: varchar instead of integer. However, the program successfully converts data values due to internal universal data types format used in the program, which is independant of external data sources data types.

13.3   Using Copy schema - "Music"

File: music.ppl.

This example is created to illustrate the Copy schema usage. This kind of data transfer is very convenient for quick transmission of large amount of data from one database to another, when the table in the destination database is not strictly determined. This way is useful because it does not require to define all columns in the fields map. It is enough just to setup the SQL-query for the source table and to define the destination table name. The rest will be done by the program: it will create the table with appropriate structure and will perform data type conversion if neccessary. However, chema Copy allows some flexibility, for eample when you require to customize the destination table structure.

In this example you can see how to redefine some of the table columns. At the same time it is not required to define the columns types, as they are automatically taken from the source table. One of the limitation here is that the key field cannot be redefined. However, source and destination table names must not be identical - in this example they are different.

Pay your attention to the fact that in this data profile the parameters are used, namely the one parameter: max_records. This parameter is used in the SQL-query for the source table. The parameter has the default value that is automatically assigned on transfer start. During transfer execution you can change the default value, thereby modifying the set of data for processing.

13.4   Using functions and commands - "Transactions"

File: trans.ppl.

In this example the Table schema is used also. Almost all fields from the source table are listed in the fields map using redefinition. Also, in the data profile the normalization of the Transactions table is provided - instead of one the two tables are created: Curr   Trans. The currency code from the Curr table is fetched by embedded SQL-function SQLGetVal(). In the Trans table of the destination database additional field "balance" is defined, which should store the accumulative sum resulting from the previous transactions. To do that, embedded SQL-function SQLSumVal() is used, that returns the sum of already recorded values of "amount" field in the Trans table.

However, this example is expecially interesting because of extened Transfer capability to execute SQL-commands upon certain scenario that is used here. In this case in the Transaction transfer the SQL-command is defined that will run upon OnEnd event. This means that the command will be called right after the data transfer completes. In this example the command is used to write the final value in the "balance" field of the Trans table.

Top Level Intro

This page is printed before a new
top-level chapter starts

Part

XIV

14 Reference information

14.1 Glossary

DBF
DBMS
DLL
LZW
MDB
MDI
ODBC
PPL
PWD
SQL
System Tray
UID
xBase
Transfer
Key field
Subtable
Trigger

14.2 Profile structure description

PARAMETERS{
[type1 param1(name1)[=val1], …, typeN paramN(nameN)[=valN]]
 type=<char, integer, numeric, date, time, timestamp>
}
CONNECT{
Source(DSN=;UID=;PWD= **)**
Destination(DSN=;UID=;PWD= **)**
}
TRANSFER{
Name(*Description* **)**
Schema(<COLUMNS, COPY, COMPLEX> **)**
ConvertText(<UNCHANGED, OEMTOANSI, ANSITOOEM> **)**
Source(*source table SQL Select* **)**
SubSQL(*subtable SQL Select* **)**
Message(*message string* **)**
Destination(**process=**<CREATE,ADD,REPLACE,FULL>
owner=<*DB schema*>
table=<*destination table*>
condition(*SQL Where clause for destination table* **)**
columns(*fields map* **)**
see Fileds map format
)
Detail(process=<CREATE,CLEAR,ADD,REPLACE,FULL>
 owner=<*DB schema*>
 table=<*subtable*>
 condition(*SQL Where clause for destination subtable* **)**

 columns(*fields map* **)**
see [Fileds map format](#)
)
TabCommands(*Destination table commands list* **)**
SubCommands(*Subtable commands list* **)**
See [Commands list format](#)
}
…
TRANSFER{
}

Data profile file usually contains one PARAMETERS section, one CONNECT section and at least on or several TRANSFER sections.

PARAMETERS section holds the list of parameters for the whole profile, meaning that the parameters values can be used in the fields map or the commands list. This section is not mandatory.

CONNECT section describes ODBC connections parms for source and destination databases. The section is mandatory for the data profile. For the file ODBC sources UID   PWD parameters are not required.

TRANSFER section defines the structure of the data block to be transmitted.

Name	Data block designation
Schema	Data transfer schema. Available values are: COLUMNS - data transmission according to the fields map COPY - copying the source table data structure into the destination database COMPLEX - transferring connected data using model "master-detail" For more information see [Transfer schemas](#).
ConvertText	Option for possible text data conversion: UNCHANGED - without conversion OEMTOANSI - convert OEM to ANSI format ANSITOOEM - covert ANSI to OEM format
Source	SQL Select to the source table (master).
SubSQL	SQL Select to the subordinate table (detail or subtable).
Message	The text string, that will be displayed during transfer execution. It may contain references to the source or destination tables fields (see [Prefix symbols](#)). This string is also used for logging into the message file if this option is set.
Destination	This section defines the structure of the master table in the destination database. There are the following subsections: **process** = Data processing algorith, possible values are: CREATE - table creation in the destination database. If the table already exists, it will be deleted and re-created ADD - only adding data to the destination table (unconditional INSERT operators) REPLACE - data replacement in the destination table (INSERT or UPDATE operator

depending on whether the row with the same key already exists or not)

FULL — full comparison. This is two-pass process. The aim of this process is to have identical data in source and destination (in the scope of given SQL query). Data missing in the source table are deleted from destination. Data missing in the destination table are added to the destination.

owner = *DB schema or owner of the destination table*
table = Destination table name
condition = SQL Where clause for destination table
columns = Fields map (see Fields map format)

Detail This section defines the structure of the subordinate (detail) table in the destination database. It has several subsections with the same structure as "Destination" section.

TabCommands List of commands for destination table (See Commands list format).

SubCommands List of commands for subordinate table (See Commands list format).

14.3 Fields map format

Fields map is intended to define the correlation between columns of source and destination tables. The syntax includes the destination table column name (optionally starting with the flag separated by a colon) with the data type and length following by an "equal" sign and then reference or a constat (see Prefix symbols):

[flag:]field1[(type[,lengh,precision])]=src ref1;
…
[flag:]fieldN[(type[,lengh,precision])]=src refN;

Flag **Description**
K   Key field. Defins a unique table index.
N   Read-only field. When table is updated, this field is not modified.
T   Trigger. When the field value is changed during transfer, the "registration" mechanism is triggered.
F   "Registration" flag for data block. Used only in COMPLEX schema.

type=<BT,TI,SI,BI,I,R,F,DB,DC,N,A,VA,LVA,B,VB,LVB,D,T,DT,TS>
See Data types definition

Fields map examples:

Example 1.
columns(T:sklad;T:code_currency;T:cur_code_currency;F:flag;N:typ_rate;N:sum_link_sd))

Example 2.
columns(T:id_ware;proc_tax1=<prc_tax1;proc_tax2=<prc_tax2;proc_tax3=<prc_tax3; N:price_spc))

Example 3.
columns(K:ndprt(I)=$ndprt;K:id_ware(I)=$id_ware;id_parent(I)=$id_parent;
last_ware(I)=$last_ware;flag_ware(I)=$flag_ware;flag_made(I)=$flag_made;
comp_ware(I)=$comp_ware;code_unit(I)=$code_unit;code_country(I)=$code_country;

id_pro(I)=$id_pro;id_ser(I)=$id_ser;name_ware(VA,80)=$name_ware;
name_ware2(VA,80)=$name_ware2;code_ware(VA,30)=$code_ware;
code_gr(VA,255)=$code_gr;barcode(VA,30)=$barcode;marka_ware(VA,40)=$marka_ware;
okdp_ware(VA,20)=$okdp_ware;post_ware(VA,30)=$post_ware;gost_ware(VA,30)=$gost_ware;
acnt_ware(VA,10)=$acnt_ware;weight(DB)=$weight;pack(DB)=$pack;cost_ware(DB)=$cost_ware;
count_ware(DB)=$count_ware;t_tax1(I)=$t_tax1;t_tax2(I)=$t_tax2;t_tax3(I)=$t_tax3;
v_tax1(DB)=$v_tax1;v_tax2(DB)=$v_tax2;v_tax3(DB)=$v_tax3;wdata(TS)=$wdata;t_info(LVA)))

14.4   Commands list format

Commands list is used for defining the event-driven data processing during transfer.

Event(*command1*[; … *commandN*])
Event=<OnStart, OnEnd, CallExec, CallBack,
        OnInsert, OnUpdate, OnDelete,
        ReInsert, ReUpdate, ReDelete,
        OnChange.*field*, ReChange.*field*>

Command=<Insert, Update, Delete, Call *procedure*>

It is allowed to define several Insert, Update, Delete operators or procedures calls separated by semicolon within one command.

Command types (events):

Event	Description
OnStart	When transfer is started
OnEnd	When transfer is finished
CallExec	When data block is "registered". Only for COMPLEX schema.
CallBack	When data block is "unregistered". Only for COMPLEX schema.
OnInsert	On each row insertion
OnUpdate	On each row update
OnDelete	On each row deletion
ReInsert	On each row insertion rollback
ReUpdate	On each row update rollback
ReDelete	On each row deletion rollback
OnChange	On particular field change
ReChange	On particular field change rollback

It is possible to use any prefixes in SQL commands or stored procedures calls (see Prefix symbols) for references to profile parameters or data fields of source and destination.

Example of commands list:

TabCommands(
CallExec(update sdoc set flag=1
where type_sd=$type_sd and numref=$numref and year=$year and
    ndprt=$ndprt and autokey_sd=$autokey_sd)

CallBack(update sdoc set flag=0
where type_sd=$type_sd and numref=$numref and year=$year and
ndprt=$ndprt and autokey_sd=$autokey_sd)
OnInsert(call BAL_MOD($numref,$year,$cli1_ndprt,$cli1_numb_client,
$code_currency,^0,$sumf_sd,$date_sd,$flag))
OnUpdate(call BAL_MOD($numref,$year,$cli1_ndprt,$cli1_numb_client,
$code_currency,#sumf_sd,$sumf_sd,$date_sd,#flag))
OnDelete(call BAL_MOD(#numref,#year,#cli1_ndprt,#cli1_numb_client,
#code_currency,#sumf_sd,^0,#date_sd,#flag))
ReInsert(call BAL_MOD($numref,$year,$cli1_ndprt,$cli1_numb_client,
$code_currency,$sumf_sd,^0,$date_sd,$flag))
ReUpdate(call BAL_MOD($numref,$year,$cli1_ndprt,$cli1_numb_client,
$code_currency,$sumf_sd,#sumf_sd,$date_sd,#flag))
ReDelete(call BAL_MOD(#numref,#year,#cli1_ndprt,#cli1_numb_client,
#code_currency,^0,#sumf_sd,#date_sd,#flag))
OnChange.date_sd(update card_spc set date_card=$date_sd
where numref=$numref and year=$year and sklad=$sklad and
sds_numref=$numref and sds_year=$year and
type_sd=$type_sd and sds_ndprt=$ndprt and
autokey_sd=$autokey_sd)
OnChange.cli1_numb_client(call BAL_MOD(#numref,#year,
#cli1_ndprt,#cli1_numb_client,#code_currency,#sumf_sd,^0,#date_sd,#flag);
call BAL_MOD($numref,$year,
$cli1_ndprt,$cli1_numb_client,$code_currency,^0,#sumf_sd,$date_sd,#flag);
update card_spc set cli_ndprt=$cli1_ndprt,
numb_client=$cli1_numb_client
where numref=$numref and year=$year and sklad=$sklad and
sds_numref=$numref and sds_year=$year and
type_sd=$type_sd and sds_ndprt=$ndprt and
autokey_sd=$autokey_sd)
ReChange.date_sd(update card_spc set date_card=#date_sd
where numref=$numref and year=$year and sklad=$sklad and
sds_numref=$numref and sds_year=$year and
type_sd=$type_sd and sds_ndprt=$ndprt and
autokey_sd=$autokey_sd)
ReChange.cli1_numb_client(call BAL_MOD($numref,$year,
$cli1_ndprt,$cli1_numb_client,$code_currency,#sumf_sd,^0,$date_sd,#flag);
call BAL_MOD(#numref,#year,
#cli1_ndprt,#cli1_numb_client,#code_currency,^0,#sumf_sd,#date_sd,#flag);
update card_spc set cli_ndprt=#cli1_ndprt,
numb_client=#cli1_numb_client
where numref=$numref and year=$year and sklad=$sklad and
sds_numref=$numref and sds_year=$year and
type_sd=$type_sd and sds_ndprt=$ndprt and
autokey_sd=$autokey_sd))

14.5 Data types definition

Definition	ODBC SQL type	Description
BT	Bit	Single bit

TI	TinyInt	Integer with a length of 3 in a range of -128 ~ 127 (unsigned 0 ~ 255)
SI	SmallInt	Integer with a length of 5 in a range of -32768 ~ 32767 (unsigned 0 ~ 65535)
BI	BigInt	Integer with a length of 19 in a range of -2^{63} ~ 2^{63}-1 (unsigned 0 ~ 2^{64}-1)
I	Integer	Integer with a length of 10 in a range of -2^{31} ~ 2^{31}-1 (unsigned 0 ~ 2^{32}-1)
R	Real	Signed numeric value with the binary length of 24 in a range of 0 ~ 10^{38}
F	Float	Signed numeric value with defined binary length p.
DB	Double	Signed numeric value with the binary length of 53 in a range of 0 ~ 10^{308}
DC	Decimal	Signed numeric value with the assigned length of p and precision s. (Maximum length depends on the ODBC driver). 1 <= p < 15, s <= p
N	Numeric	Signed numeric value with the assigned length of p and precision s. 1 <= p < 15, s <= p
A	Char	Text string with the fixed length n.
VA	VarChar	Variable length text string with the maximum length n.
LVA	LongVarChar	Variable length text string. Maximum length depends on the ODBC driver.
B	Binary	Binary data with the fixed length n.
VB	VarBinary	Variable length binary data with the maximum length n.
LVB	LongVarBinary	Variable length binary data. Maximum size depends on the ODBC driver.
T	Time	Time in hours, minutes and seconds.
D	Date	Gregorian date, month and year.
TS	Timestamp	Date and time.

14.6 Prefix symbols

Prefix symbols are used to make references to profile parameters or source and destination data fields in the fields map or commands list. After the prefix symbol the data field name or a constant is indicated.

Symbol	Description	Comments
^	Value (constant)	The constant value will be set in the field for every SQL Insert and Update operation.
+	Autoincrement	The constant in the field indicates the increment value. Autoincrement value will be calculated for every SQL Insert operator, but not for Update. The value is calculated as the maximum field value in the table plus autoincrement constant.
:	Profile parameter	Data from the parameter field will be set in the field for every SQL Insert and Update operations.
$	Field in the master source table	Data from the current row in the main source table will be used in SQL Insert and Update operations.
~	Field in the subordinate source table	Data from the current row in the subordinate source table will be used in SQL Insert and Update

		operations.
#	Field in the master destination table	Data from the current row in the main destination table will be used in SQL Insert and Update operations.
!	Field in the subordinate destination table	Data from the current row in the subordinate destination table will be used in SQL Insert and Update operations.
<	Field redefinition	Mainly used in COPY schema when some columns names differ in source and destination tables. Data from the current row in the main source table will be set in the field.
%	Embedded functions	Embedded functions allow to call predefined SQL queries for complex data processing. For more information see Embedded functions.

14.7 Embedded functions

Embedded functions are the functions that can be used in the "Value" column of the fields map. To indicate the embedded function in the fields map the special prefix % is used. The function value is calculated during processing of each record in the destination table.

Currently Transfer supports the following types of embedded functions:

Function	**Parameters**	**Description**
SQLGetVal	field, table, where expr	Returns the *field* value from the *table* based on the *where expr* criterion. If more than one row is returned, then the value in the first row is used.
SQLMaxVal	field, table, where expr	Returns the maximum *field* value from the *table* based on the *where expr* criterion.
SQLMinVal	field, table, where expr	Returns the minimum *field* value from the *table* based on the *where expr* criterion.
SQLAvgVal	field, table, where expr	Returns the average *field* value from the *table* based on the *where expr* criterion.
SQLVarVal	field, table, where expr	Returns variation of the *field* value from the *table* based on the *where expr* criterion.
SQLSumVal	field, table, where expr	Returns sum of the *field* values from the *table* based on the *where expr* criterion.
SQLCount	table, where expr	Returns the number of rows in the *table* satisfying the *where expr* criterion.

Index

- A -

About Transfer   27
Add data   24
Arrange icons   27
Automatic data exchange   68

- C -

Cascade   27
Close file   19
Close row   25
Command line options   30
Commands list format   57, 77
Commands list usage   67
COMPLEX schema processing   64
Copy   23
Create new file   19
Customize Main Toolbar   26
Customize Profiles Toolbar   26
Cut   23

- D -

Data exchange options   31
Data profile editting   47
Data profile syntax   54
Data profiles   21
Data types definition   59, 78
Delete   23
Delete data   24

- E -

Edit data   24
Edit menu   22
Editting destination table data   50
Editting source table data   49
Editting subtable data   51
Embedded functions   60, 80
Entering commands list   53
Entering fields map   52

Entering profile parameters   48
Examples list   71
Exit   22
Export data   20

- F -

Feadback   14
Fields map usage   66
File menu   19
File ODBC source   41
Fileds map format   56, 76
Find   23
First row   24

- G -

General options   30
Glossary   74

- H -

Help menu   27
Help topics   27
How does it work?   17
How to register   12

- I -

Import data   20
Interface options   33

- L -

Last row   25
License info   8

- M -

Managing transfers list   49
Menu data   23

- N -

New data profile registration   47

Endnotes 2... (after index)