

Transbase®
Interactive Interface TBI

Transaction Software GmbH
Willy-Brandt-Allee 2
D-81829 München
Germany
Phone: +49-89-62709-0
Fax: +49-89-62709-11
Email: info@transaction.de
<http://www.transaction.de>

Version 6.8.1.40
November 02, 2010

Contents

1	Overview	3
2	Command Line Parameters	5
3	TBI Commands	6
3.1	Database Connections	6
3.2	Transactions	6
3.3	System Commands: e, cd, qu, list, !	7
3.4	TBI Variables	8
3.5	Setting the Consistency Level	8
3.6	Evaluation Plans	9
3.7	TBI Procedures	9
3.8	Database Commands: sc, re, rse	11
3.9	SQL Commands	11
3.10	Command History: history, ee	12
4	TBI Relation Editor RE	13
4.1	Input Mask	13
4.2	RE Commands	13

Chapter 1

Overview

TBI (Transbase Interactive) is a line-oriented user interface for interactive database use. TBI is well suited to develop all kinds of database applications, i.e. to design the database schema and to develop the queries needed for a certain application. TBI exhibits the full functionality of Transbase, including connections to more than one database and distributed transactions.

TBI provides three classes of commands, namely TB/SQL commands, TBI commands, and shell commands. TB/SQL commands may be spread over lines of input and thus have to be delimited by a semicolon (;). TBI commands and shell commands are delimited by the newline character. Shell commands are prefixed with a "!". The table below shows the syntax of all TBI commands. Optional parameters are enclosed in brackets [].

TBI command names are key insensitive, i.e. instead of typing `conn` one may also type `CONN` or any combination of lower case and upper case letters `cOnN`.

Syntax:

```
co[nn] dbname [ login [ password ] ]
dc[onn] [ dbid ]
sw[itc] [ dbid ]
bt
at
ct
acc[ept] var [prompt]
set var string
set consistency <level>
sc[hema] [ tablename ]
re[ledit] tablename
rse tablename [ filename ]
```

```
x [ filename [ param1 [param2] ...] ]  
li[st] [ filename ]  
ee  
hi[story]  
e[dit] [ filename ]  
cd [ directory ]  
qu[it]  
! [shellcommand]
```

Chapter 2

Command Line Parameters

Syntax:

```
tbi [-x | -s] [-f file][ dbname [ login [ password ] ] ]
```

Syntax: (MS/Windows)

```
wintbi [-notify task msg|-notifywindow wnd msg] [-x | -s] [-f file][ dbname [ login [ pa
```

With the `-x` option, tbi echos statements while processing tbi procedures. With the `-s` option (silent), tbi does not output interactive prompts. All other output is still produced.

The `-f` option (evaluate file) evaluates the given file as a command script. At EOF active transactions will be aborted and open connections will be closed.

On MS Windows `-notify/-notify window` makes tbi to send his exitcode to task/wnd by using the message msg. The exitcode will be the wParam component of msg. This option is very usefull in conjunction with the `-f` option when tbi is started from another task.

Beside the option, tbi may be called with 1, 2, or 3 parameters, namely a dbname, the login name and the password.

If no parameters are given, tbi does not connect to a database.

If a dbname is specified, tbi connects to the specified database. A dbname may be either a local name or a remote name.

If login is not specified, the user will be prompted for the name and the password. If no password is specified, the user will be prompted. During typing the password the echo is disabled.

Chapter 3

TBI Commands

3.1 Database Connections

TBI maintains connections to databases. Databases are identified by numbers. One of the connected databases is the active database. The name of the active database is given with the command prompt.

Syntax:

```
co[nn] dbname [ login [ password ] ]\
dc[onn] [ dbid ]\
sw[itch] [ dbid ]
```

The `conn` command connects to the specified database. The meaning of the parameters `login` and `password` is as described for the command line parameters.

The `dconn` command disconnects from a database. If no parameter is specified, the currently active database is disconnected. If other databases are connected to, one of those databases will be chosen as the new current database.

To change the active database, the `switch` command is used. If no parameter is specified, `switch` prints a list of connected databases. Otherwise the specified database is made the active database.

3.2 Transactions

TBI supports a single transaction, which is automatically started with the first TB/SQL statement. A transaction remains active until it is explicitly aborted or committed by the user.

The TBI command prompt shows a plus sign "+" if a transaction is active and a minus sign "-" if no transaction is active.

Syntax:

```
bt
at
ct
```

Effect: bt starts a transaction, at aborts, ct commits the currently active transaction.

If no transaction is active, at and ct have no effect.

If a transaction is already active, bt has no effect.

If the user switches between databases, the current transaction is automatically distributed over all participating databases.

If a transaction is active when TBI is left or when a database is disconnected, TBI prompts the user to abort or commit the transaction.

Warning:

By typing the Ctl-C key, a currently active transaction is always aborted automatically.

3.3 System Commands: e, cd, qu, list, !

A number of system commands are also included in TBI.

Syntax:

```
e[dit] [ filename ]\\
cd [ directory ]\\
li[st] [ filename ]\\
! command\\
qu[it]
```

The edit command edits the file specified by its name. If no name is specified, a temporary file is edited. The temporary file is located either in the directory given by the user's TMPDIR environment variable, or in the current directory. The name of the temporary file is similar to "tbiNNNNN" where NNNNN denotes the process number of the TBI process.

The environment variable EDITOR is used to find the name of the editor. If no variable is defined, a local default is used (e.g. "vi" for UNIX systems).

The cd command changes the current directory. If no directory is specified, cd changes to the home directory of the user. All filenames refer to the current

directory if not stated otherwise. Especially, the commands `edit`, `x`, and `cd` and the `SPOOL` statement refer to files located in this directory.

Note:

On Windows platform the `cd` command not only changes the current working directory for a drive. It also changes the current drive itself.

The `list` command redirects the standard output of TBI into the filename specified. The redirection holds until a subsequent `list` command is issued. The `list` command without a filename restores the `stdout` to where it was at start time of TBI (normally the user's terminal).

The `!` command runs an arbitrary system command within a subshell. Note that command line processing is left to the shell; therefore special characters (e.g. `*` or `[]`) have the usual meaning. If this interpretation shall be omitted, these characters have to be escaped.

Note:

The `!` command is not available on Windows platforms.

If a user wants to start an interactive subshell, it may use one of the commands:

```
! sh
! csh
```

The `quit` command leaves the TBI session. Typing `Ctrl-D` at the beginning of the command line in UNIX systems is equivalent.

3.4 TBI Variables

Syntax:

```
set var string\\
acc[ept] var [prompt]
```

The `set` command sets a variable `var` to the value `string`.

The `accept` command prints a prompt if specified, reads a value from `stdin` and assign it to the variable `var`.

Values of variables can be accessed by the notation `$var` where `var` is a variable which has been assigned a value. If `var` has not been assigned a value, the expression `$var` remains unchanged.

3.5 Setting the Consistency Level

Syntax:

```
set consistency 1
set consistency 2
set consistency 3
```

Setting the consistency level is done via the predefined special variable `consistency`. The allowed values 1, 2, 3 correspond to the values `CONS_1`, `CONS_2`, `CONS_3` used in the explanation of the consistency levels in the TBX and ESQL manuals (chapter *Consistency Levels*).

To summarize:

Level 3 implements serializability and read reproducibility by long write and long read locks.

Level 2 implements short read locks (until end of query).

Level 1 runs read queries without any locks.

Each tbi session starts with level 3 consistency by default.

3.6 Evaluation Plans

Syntax:

```
set plans on
set plans off
```

Enabling **Evaluation Plans** is done via the predefined special variable `"plans"`. The allowed values `ON` and `OFF` enable and disable the mechanism. If `"plans"` is `ON` the evaluation plans are retrieved into file `"plan.txt"` each time a query has been evaluated and closed.

Each tbi session starts with **Evaluation Plans Disabled** by default.

3.7 TBI Procedures

Syntax:

```
x [ filename [ param1 [ param ] ... ] ]
```

Instead of typing TB/SQL commands, TBI commands or shell commands interactively on the TBI command line, all those commands can be typed into a file (using the `"e"` command) and be executed by TBI.

If no filename is given, TBI executes the temporary file. If an explicit filename is given (an absolute or relative pathname referring to the current directory) the given file is executed by TBI.

When calling a named procedure files, up to nine parameters can be passed via the command line. Parameters are separated by space or tab.

Within the procedure file, parameters can be referenced as \$1\$ (the first parameter), \$2\$ (the second parameter) and so forth. For convenience, the parameter \$0\$ is defined, too, which contains the name of the procedure executed. When the procedure is run, TBI substitutes the actual parameter strings (if present) for the formal parameters \$0\$, \$1\$, \$2\$, ..., \$9\$.

TBI executes the procedure silently unless the `-x` option had been specified on the command line when TBI had been started.

The procedure `tel` retrieves the phone number of persons whose name matches the first parameter of `tel`.

```
SELECT name, phone FROM phonenumber
WHERE name LIKE '%$1$'
```

The procedure is called by typing e.g.:

```
x tel TransAction
```

The query produced by the parameter substitution is:

```
SELECT name, phone FROM phonenumber
WHERE name LIKE '%TransAction%'
```

The `x` can be suppressed if the name of the procedure does not conflict with a TBI command prefix or with a TB/SQL prefix.

Bug:

If a parameter is referenced in a procedure and no actual parameter is supplied for it, no text substitution is performed, i.e. \$1\$ is left unchanged instead of substituting the empty string for \$1\$.

If, in the above example, `tel` was called without parameter, TBI would produce the statement:

```
SELECT name, phone FROM phonenumber
WHERE name LIKE '%$1$'
```

and, most probably, no tuples will be found.

3.8 Database Commands: sc, re, rse

Syntax:

```
sc[hema] [ tablename ]\\
re[ledit] tablename\\
rse tablename [ file ]
```

The schema command displays a list of table names that are accessible by the current user, i.e. where the current user has privileges.

If a tablename is specified, detailed field information is displayed for the given tablename.

The reledit command calls a mask-oriented interface to edit the contents of a table. The tablename parameter is mandatory and must specify a valid table or view name. If the table or view cannot be updated by the user (e.g. since the view is not updatable or if the table is locked for update), RE tries to read the table or view. In this case, only retrieval operations are allowed in this RE session.

The reledit subcommands are described in a dedicated section below.

The rse command is a generalized realization of the ALTER TABLE statement defined by ANSI SQL standard. The rse command supplied with a table name generates a script which contains statements to save the contents of the table in a spoolfile, to drop the table, to recreate the table, to spool the saved tuples back into the recreated table and to recreate all secondary indexes as well as all views based on the table.

The generated script is written into the current temporary procedure file or into a file with the specified name. The editor is opened on the file.

The rse command does not change the database. The generated script is a basis to realize schema changes on the specified table.

Note:

The re command is not available on Windows platforms.

3.9 SQL Commands

All TB/SQL commands may be typed to TBI; note that TB/SQL commands have to be suffixed with a semicolon. Thus the user is able to insert arbitrary newline characters into the statement.

SQL commands are recognized by their first word (e.g. SELECT).

3.10 Command History: history, ee

TBI automatically maintains - much like the C-shell - a history of at most 40 interactive commands. The history can be shown using the history command.

There are two possibilities to retrieve a command from the history for reexecution.

```
..          reexecutes the last command
.prefix    reexecutes the most recent command with the specified prefix.
```

```
SELECT tname FROM systable
at
.SE
    ... repeats the select command
..
    ... repeats the select command
```

The `ee` command writes the last executed command into the temporary file and calls the editor. It is a convenient way to correct the last interactively typed command.

Example:

```
SELECT tname FOM systable WHERE ttype = 'S'
    syntax error 'SELECT tname FOM' is reported by Transbase
ee
    correct the error
x
    repeat the command by executing the temporary file
```

Chapter 4

TBI Relation Editor RE

4.1 Input Mask

The input mask contains a line for each field of the table. The user may fill in the fields in order to specify a tuple to be fetched or to insert a new tuple. When browsing through a table the user may delete or update a tuple in place.

Special Characters:

When filling the input mask, printable characters are echoed, non-printable characters are ignored with the following exceptions:

- The tab characters moves the cursor to the next field cyclically.
- The newline character executes the first command listed in the command menu, i.e. either switches to the browsing mode of RE or retrieves the next tuple.
- The escape character displays a menu of applicable RE commands.

4.2 RE Commands

An RE command is selected in one of three ways:

From within the command menu:

By typing the first character of the command followed by a newline character.

From within the command menu:

By hitting the space bar, the prompt is cyclically moved through the menu choices.

When the appropriate command is reached, a newline character activates the command.

From within the input menu, only two commands can be activated simply by hitting the newline character:

If a browse is active, the more command is executed otherwise the browse command is executed.

browse The browse command starts a retrieval query. If the input mask has been filled, only those tuples are retrieved that have the values specified. Note that string fields may specify patterns containing wild cards. See the description of the LIKE predicate in the TB/SQL manual.

more The more command retrieves the next tuple of a previously activated browse. If no further tuples are available, the input mask is cleared. The user may specify the next browse.

insert The insert command may be called to insert a new tuple into the table. If insert is called during a browse, the browse is automatically closed. The insert command is not shown if RE is called for retrieval operations only.

clear The clear command clears the input mask and eventually closes an open browse.

quit The quit command ends the reledit session and switches back to the TBI command interface. Note that you still have to commit or abort the active transaction by a TBI command.

update The update command is only applicable during a browse. If called, the current tuple is updated to the values specified in the input mask. The update command is not shown if RE is called for retrieval operations only.

delete The delete command is only applicable during a browse. If called, the current tuple is deleted from the table. If available the next tuple is shown in the input mask. The delete command is not shown if RE is called for retrieval operations only.