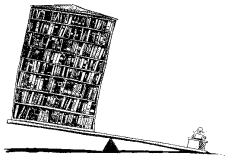
**KE Texpress**

****

**Client/Server API Guide** KE Software Pty Ltd

Copyright © 1993-2004 KE Software Pty Ltd This work is copyright and may not be reproduced except in accordance with the provisions of the Copyright Act.

Contents i

**Contents**

**Chapter 1 Introduction................................ ......................... 1-1**

Terminology............................................................................ 1-3 Compiling an Application Program........................................... 1-4

**Chapter 2 Error Handling................................ ..................... 2-1**

Error Number.......................................................................... 2-3 Error Message......................................................................... 2-4 Error Offset in Command......................................................... 2-5

**Chapter 3 Initialisation and Termination............................ 3-1**

Connection Parameters............................................................ 3-3 API Initialisation...................................................................... 3-6 Server Connection................................................................... 3-8 Session Parameters.................................................................. 3-9 API Version........................................................................... 3-10 Table Access.......................................................................... 3-11 Server Configuration.............................................................. 3-12 Server Interruption................................................................. 3-13 Server Disconnection............................................................. 3-15 API Termination.................................................................... 3-16

**Chapter 4 Cursors................................ ................................ 4-1**

Texql Command...................................................................... 4-3 Cursor Type............................................................................. 4-6 Close Cursor............................................................................ 4-7 Merge Cursors......................................................................... 4-8 Sort Cursor.............................................................................. 4-9

**Chapter 5 Row Access................................ ......................... 5-1**

Next Row................................................................................ 5-3 Get Row.................................................................................. 5-5 Move Row............................................................................... 5-6 Row Position........................................................................... 5-7 Row Reset............................................................................... 5-8 Count Rows............................................................................. 5-9

*KE Texpress API Guide*

ii Contents

Number of Row Hits.............................................................. 5-10 Lock Row.............................................................................. 5-11 Unlock Row.......................................................................... 5-12 Row Status............................................................................ 5-13 New Row.............................................................................. 5-14 Save Row.............................................................................. 5-16 Discard Row.......................................................................... 5-17 Delete Row............................................................................ 5-18

**Chapter 6 Column Access................................ ................... 6-1**

Column Names........................................................................ 6-3 Column Kind........................................................................... 6-4 Column Type........................................................................... 6-5 Column Nested Cursor............................................................. 6-6 Column Data Get..................................................................... 6-7 Column Data Set...................................................................... 6-9

**Chapter 7 Convenience Functions................................ ..... 7-1**

Item Names............................................................................. 7-3 Item Number of Fields............................................................. 7-4 Item Data Get.......................................................................... 7-5 Item Data Set........................................................................... 7-7 Field Type................................................................................ 7-8 Field Data Get......................................................................... 7-9 Field Data Set........................................................................ 7-10

**Appendix A Sample Program................................ .............. A-1 Appendix B Error Codes................................ ..................... B-1**

*KE Texpress API Guide*

Introduction 1-1

**Chapter 1**

**Introduction**

Terminology............................................................................ 1-3 Compiling an Application Program........................................... 1-4

*KE Texpress API Guide*

1-2 Introduction

**Overview**

The KE Texpress Information Management System is an object-oriented database package which provides numerous extensions to the traditional relational database model. The most significant extensions are in the area of complex object support. KE Texpress supports the inclusion of the following object components into an object definition:

• Text.

• Multi-valued fields.

• References to foreign objects (objects in different formats controlled by software other than KE Texpress).

This manual describes the KE Texpress C Language Applications Programming Interface (C-API). The C-API provides a back-end library of C functions which enable developers to harness the speed and flexibility of the KE Texpress Information Management System.

The library of functions provides a wrapper around the Texql language so that the full functionality of Texql is available through the API. The KE Texpress C API may also be accessed using the C++ programming language. For a complete description of Texql refer to the Texql Guide.

The C-API can also be used in conjunction with Titan 3.4 databases.

Function descriptions in this manual provide formal C code definitions for the function call, parameters passed and values returned. Typically an example section of code which utilises the function is also shown.

The remainder of this manual is divided into the following chapters.

Chapter 2 describes the method by which function error codes and messages can be accessed.

In chapter 3 the API initialisation and termination functions are discussed.

Chapter 4 provides an overview of the primary API functions used for performing commands. Chapters 5 and 6 describe the row access and column access functions respectively.

Convenience functions are described in Chapter 7. These functions provide short hand methods of accessing data in a style particular suited to the data layout of Titan 3.4 databases.

A complete example program is provided in Chapter 8.

*KE Texpress API Guide*

Introduction 1-3

The sample code sections and the complete example program utilise the *contacts*, *loantypes* and *loans* tables described in the Texql Guide.

*KE Texpress API Guide*

1-4 Introduction

**Terminology**

KE Texpress uses terminology which reflects the object-oriented nature of the product, and thus highlights the distinction between it and relational database systems. However, Texql provides an interface to KE Texpress databases which attempts is similar to a standard SQL interface to a relational database.

This section describes the terminology used by Texql and the C-API in terms of the appropriate terminology of KE Texpress. Refer to the KE Texpress Guides for a description of KE Texpress terminology.

The following terms are used throughout this manual:

**Texql KE Texpress**

**table** This refers to a single KE Texpress database. All KE Texpress databases, although controlled separately in

terms of access privileges, etc., are accessible as Texql

tables.

**column** This refers to an item in a KE Texpress database.

**nested table** This refers to a KE Texpress multi-field item which is not of type text or a multi-field text item without an

associated Look-up table. Multi-field text items without

Look-up tables are considered to be Texql text boxes, i.e.

single atomic value of (continuous) text.

**tuple** or **row** This refers to a record in a KE Texpress table or a record derived by Texql as the result of a query.

**nested tuple** The multi-field Key and library items of KE Texpress are represented as nested tuples in Texql. This means that

these items can be treated as atomic values or,

alternatively, their components can be individually

manipulated.

**atomic value** This refers to a value in a column of a tuple, i.e. the value of a field within a KE Texpress record

To assist in the portability of API client programs between various platforms C language typedefs are used for function arguments (e.g. TEXCURSOR, TEXS32, TEXSTRING). Refer to the API C language header files (the "include" directory) for further information.

*KE Texpress API Guide*

Introduction 1-5

**Compiling an Application Program**

Application programs which use the KE Texpress C-API need access to the API header file and the API library. These files are kept under the KE Texpress directory. On a UNIX system, for example, if the KE Texpress home directory is:

/home/kestrel/ texpress

the API related information resides in the following directories:

/home/kestrel/ texpress/include

/home/kestrel/ texpress/lib

All C source files which use the KE Texpress C-API function calls must include the KE Texpress API header file. This file is included by using the compiler directive:

#include "texapi.h"

or

#include <texapi.h>

To ensure maximum portability of applications the first form is preferred. The include file is located in the include directory under the API directory. On a UNIX system, this directory is typically specified on the C compiler command line as one to search for header files.

The API libraries reside in the lib directory under the API directory. On a UNIX system, this directory is typically specifed on the C compiler or loader line as one to search for library files.

Thus to compile an application program in the C source code file *prog.c* into an executable program *prog* on a UNIX system the following command can be used:

cc -I/home/kestrel/ texpress/include prog.c \

-L/home/kestrel/ texpress/lib \

-ltex - los \

-o prog

Alternatively, an environment variable (or a macro in a Makefile) can be set to point to the texpress directory:

setenv TEXAPI /home/kestrel/ texpress

cc -I${TEXAPI}/include prog.c \

-L${TEXAPI}/lib - ltex - los -o prog

*KE Texpress API Guide*

Error Handling 2-1

**Chapter 2**

**Error Handling**

Error Number.......................................................................... 2-3 Error Message......................................................................... 2-4 Error Offset in Command......................................................... 2-5

*KE Texpress API Guide*

2-2 Error Handling

**Overview**

All KE Texpress API functions report errors in a consistent manner. Each function returns a status value of 0, indicating success, or -1, indicating an error. If an error status is returned then the error function described in this chapter may be used to obtain information about the type of error that has occurred.

It is considered good programming practice to always check the return value of an API function call.

*KE Texpress API Guide*

Error Handling 2-3

**Error Number**

**NAME**

TexError - error number

**SYNOPSIS**

int

TexError()

**DESCRIPTION**

Gets the error number of an error generated by the last API call. A full list of error numbers is contained in the "texapi.h" header file.

**RETURN VALUES**

The error number.

**ERRORS**

None

**EXAMPLE**

...

printf("API call failed: no. = % d\n", TexError());

...

**SEE ALSO**

TexErrMsg, TexErrOff

*KE Texpress API Guide*

2-4 Error Handling

**Error Message**

**NAME**

TexErrMsg - error message

**SYNOPSIS**

TEXSTRING

TexErrMsg()

**DESCRIPTION**

Gets the error message of the error generated by the last API call. The text of these error messages is kept in the standard KE Texpress text file.

**RETURN VALUES**

A pointer to the text of the error message.

**ERRORS**

None

**EXAMPLE**

...

printf("API call failed: \"% s\"\n", TexErrMsg());

...

**SEE ALSO**

TexError

*KE Texpress API Guide*

Error Handling 2-5

**Error Offset in Command**

**NAME**

TexErrOff - offset of error in texql statement text

**SYNOPSIS**

int

TexErrOff()

**DESCRIPTION**

Gets the character offset in the texql statement text of the last error generated by a call to the KE Texpress API. If the error was not directly associated with a texql statement, this value is -1.

**RETURN VALUES**

The character offset.

-1 if error not associated with texql statement.

**ERRORS**

None

**EXAMPLE**

TEXCURSOR cursor

char cmd[128];

int off;

...

sprintf( cmd, "select all frim contacts");

if (TexCommand( cmd, &cursor) < 0)

{

switch ( TexError())

{

case TESYNTAX:

off = TexErrOff();

printf("Syntax error in\n%s\n", cmd);

printf("Near offset % d\n", off);

break;

...

}

...

**SEE ALSO**

TexError

*KE Texpress API Guide*

Initialisation and Termination 3-1

**Chapter 3**

**Initialisation and Termination**

Connection Parameters............................................................ 3-3 API Initialisation...................................................................... 3-6 Server Connection................................................................... 3-8 Session Parameters.................................................................. 3-9 API Version........................................................................... 3-10 Table Access.......................................................................... 3-11 Server Configuration.............................................................. 3-12 Server Interruption................................................................. 3-13 Server Disconnection............................................................. 3-15 API Termination.................................................................... 3-16

*KE Texpress API Guide*

3-2 Initialisation and Termination

**Overview**

Before using other functions, each program must first initialise the KE Texpress API and then connect to at least one KE Texpress server. A termination call is also provided for when access to the API is no longer required.

If a connection is established correctly a session identifier is returned. This session identifier is used in subsequent calls to indicate which server is being accessed.

The version number of the current API installation may be obtained to allow the calling program to check for compatibility.

The KE Texpress API make extensive use of KE Texpress tables. The TexTable() function allows the calling program to determine the availibility of a KE Texpress table at program startup.

*KE Texpress API Guide*

Initialisation and Termination 3-3

**Connection Parameters**

**NAME**

TEXPARAMS - structure holding parameters used when establishing API connections.

**SYNOPSIS**

#include "texapi.h"

**DESCRIPTION**

The connection function TexConnect is passed parameters which define the connection to be established in a TEXPARAMS structure. The stucture holds the following entries:

TEXSTRING p\_name;

int p\_type;

TEXSTRING p\_host;

TEXSTRING p\_port;

TEXSTRING p\_service;

TEXSTRING p\_prog;

TEXS32 p\_read;

TEXS32 p\_write;

TEXSTRING p\_user;

TEXSTRING p\_passwd;

char p\_escape;

TEXS32 p\_baud;

int p\_parity;

int p\_stop;

The *p\_name* field may hold a name which can be used to refer to the connection parameters. It is currently unused by the API.

The *p\_type* field holds a flag indicating the type of connection to be established. The value in this field should be one of the pre-defined constants:

IO\_SOCKET

IO\_PIPE

IO\_SERIAL

The *p\_host* field holds the name of the host machine to connect to when establishing a socket connection. The *p\_port* field holds the name of the serial port to use when establishing a serial connection. Note that the use of these two fields is mutually exclusive.

The *p\_service* field holds the name of the service to connect to when establishing a socket connection. This name should appear in the file /etc/services when establishing socket connections from UNIX machines.

*KE Texpress API Guide*

3-4 Initialisation and Termination

The *p\_prog* holds the name of the program to run when establishing using a pipe to communicate with the server. Note that the use of these two fields is mutually exclusive.

The *p\_read* field holds the size of buffer to use for receiving data from the server. The *p\_write* field holds the size of buffer to use when transmitting data to the server.

The *p\_user* field holds the login name of the user to run as on the server machine. This user's privileges will control the degree of access provided to the tables used by the API. The *p\_passwd* field holds the unencrypted password of this user. These values are transmitted to the server during the TexConnect call to authenticate access to the server machine. The values are not used for pipe-based connections. A NULL password may be sent for connections on a socket. This will force the server to use the remote command authentication scheme based on the UNIX hosts.equiv and .rhosts files This is the mechanism used by the standard UNIX remote command utilities rlogin, rsh, and rcp. (For more information, see the hosts.equiv entry in section 4 of the UNIX manuals).

The remaining fields are only used when establishing serial line connections to the server.

Serial-based connections use XON/XOFF flow control to prevent the loss of information during transmission of large amounts of data. The *p\_escape* field holds the character to be used to escape the special meaning of certain characters (primarily the XON (control-Q) and XOFF (control-S) characters themselves) during serial transmission.

The *p\_baud* field holds the data transmission rate to be used. A set of pre defined constants is supplied to specify the baud rate:

IO\_BAUD\_300

IO\_BAUD\_1200

IO\_BAUD\_9600

...

etc. Numeric values may also be specified. Not all baud rates may be supported by all connections. When a baud rate is requested, the nearest slower rate supported by both client and server machines is selected.

The *p\_parity* field holds the parity to use when transmitting each character. The value in this field should be one of the pre-defined constants:

IO\_PARITY\_NONE

IO\_PARITY\_EVEN

IO\_PARITY\_ODD

*KE Texpress API Guide*

Initialisation and Termination 3-5

When selecting which parity to use, it is important to realise that all data transmission must be with 8-bit characters. Unless extended parity generation is supported (i.e. a ninth parity bit is transmitted for each character) IO\_PARITY\_NONE should be used.

The *p\_stop* field holds the number of stop bits to transmit between each character. The value in this field should be one of the pre-defined constants:

IO\_STOP\_10

IO\_STOP\_20

IO\_STOP\_15

which represent using 1, 2 and 1.5 stop bits respectively. Not all of these values will be supported by all clients and servers .

**DEFAULT VALUES**

*p\_type* IO\_PIPE for UNIX clients

IO\_SERIAL for DOS, Windows and Macintosh clients.

*p\_host* "localhost"

*p\_port* "COMM1" for DOS and Windows clients.

"modem" for Macintosh clients.

*p\_service* "texserver"

*p\_prog* "texserver"

*p\_read* 1024

*p\_write* 1024

*p\_user* The login name of the effective user on the client machine for UNIX clients.

The host name (if any) for DOS, Windows and Macintosh

clients.

*p\_passwd* NULL

*p\_escape* ESC (Octal 033)

*p\_baud* IO\_BAUD\_9600

*p\_parity* IO\_PARITY\_NONE

*p\_stop* IO\_NONE

**SEE ALSO**

TexInitialise, TexConnect, TexParams

*KE Texpress API Guide*

3-6 Initialisation and Termination

**API Initialisation**

**NAME**

TexInitialise - initialise the API.

**SYNOPSIS**

int

TexInitialise( argc, argv, params)

int \*argc;

char \*\*argv;

TEXPARAMS \*params;

**DESCRIPTION**

Performs the necessary initialisation for the front-end of the KE Texpress API. No other API functions should be called prior to this function. Command line arguments should be passed through to TexInitialise. A pointer to a TEXPARAMS structure is also passed. This structure is loaded with default connection parameters, based on the client machine type, the values of certain environment variables and the values specified by any API-specific command line arguments. Any arguments specific to the API are removed from the argument list.

The environment variables which are interpreted by TexInitialise are:

Environment Variable

TEXPARAMS field affected

Value

TEXTYPE p\_type "socket", "pipe", or "serial" TEXHOST p\_host Host for a socket connection. TEXPORT p\_port Port for a serial connection TEXSERVICE p\_service Service for a socket connection TEXPROG p\_prog Program for a pipe connection TEXREAD p\_read Receive buffer size in bytes TEXWRITE p\_write Transmit buffer size in bytes TEXUSER p\_user User login name on server machine TEXESCAPE p\_escape Escape character

TEXBAUD p\_baud Data transmission rate TEXPARITY p\_parity "none", "even", or "odd" TEXSTOP p\_stop "1", "2", or "1.5"

*KE Texpress API Guide*

Initialisation and Termination 3-7

The arguments which the TexInitialise consumes are all introduced on the command line by '-T'. The options which may be specified are:

Command-line Argument

TEXPARAMS field affected

Meaning

-Tb*size* p\_read p\_write

Receive and transmit with *size* buffer size

-Tc p\_type Use a serial connection.

-Th*host* p\_host Connect to *host* for socket

connection

-Tn p\_type Use a socket connection

-Tp p\_type Use a pipe connection

-Tr*size* p\_read Receive with *size* buffer size

-Ts*service* p\_service Use *service* for socket connection -Tw*size* p\_write Transmit with *size* buffer size

**RETURN VALUES**

0 API front-end was successfully initialised.

-1 Initialisation procedure failed.

**ERRORS**

None.

**EXAMPLE**

main(argc, argv)

int argc;

char \*\*argv;

{

TEXPARAMS params;

...

if (TexInitialise(& argc, argv, & params) < 0)

/\* initialisation failure \*/

...

**SEE ALSO**

TexConnect, TexParams, TexDisconnect, TexTerminate

*KE Texpress API Guide*

3-8 Initialisation and Termination

**Server Connection**

**NAME**

TexConnect - connect to a KE Texpress server.

**SYNOPSIS**

int

TexConnect( params, session)

TEXPARAMS \*params;

TEXSESSION \*session;

**DESCRIPTION**

Connects to a KE Texpress back-end server. The connection is established according to the configuration held in the structure pointed to by the *params* argument. The connection invokes the server on the host machine. If the connection is successfully established then a session identifier is returned in the *session* parameter. The session identifier is used in subsequent API calls.

More than one call to TexConnect can be made. This means that more than one server, running on more than one host machine, can be invoked by the one API application program.

**RETURN VALUES**

0 Server connection was successfully established.

-1 Connection procedure failed.

**ERRORS**

TELICENCEERR Licencing error.

TEWHOAREYOU User information could not be determined. TEPERMISSION No permission to connect to this server.

**EXAMPLE**

TEXPARAMS params;

TEXSESSION session;

...

if (TexConnect(& params, &session) < 0)

/\* connection failure \*/

...

**SEE ALSO**

TexInitialise, TexParams, TexDisconnect, TexTerminate

*KE Texpress API Guide*

Initialisation and Termination 3-9

**Session Parameters**

**NAME**

TexParams - get current session parameters

**SYNOPSIS**

int

TexParams(session, params)

TEXSESSION session;

TEXPARAMS \*params;

**DESCRIPTION**

Retrieves the current connection parameters for the session whose identifier is passed in the argument *session*. The *params* argument points to a TEXPARAMS structure which is loaded with the session's connection parameters. These parameters may be used to check that the appropriate connection has been made.

**RETURN VALUES**

0 Parameters were retrieved successfully.

-1 Parameters could not be retrieved.

**ERRORS**

TESESSIONBAD An incorrect session identifier was supplied TESESSIONCLOSED The connection is no longer open.

**EXAMPLE**

TEXSESSION session;

TEXPARAMS params;

...

if (TexParams(session, & params) < 0)

/\* connection failure \*/

...

printf("connection host is % s\n", params.p\_host);

...

**SEE ALSO**

TexInitialise, TexConnect, TexDisconnect, TexTerminate

*KE Texpress API Guide*

3-10 Initialisation and Termination

**API Version**

**NAME**

TexVersion - determine API version

**SYNOPSIS**

int

TexVersion(session, version)

TEXSESSION session;

TEXSTRING \*version;

**DESCRIPTION**

Retrieves the version number of the server running on the machine with the session identifier *session*. This function can useful for providing version release verfication for front end applications which use the API.

**RETURN VALUES**

0 Version number succesfully determined.

-1 Error in determining the version number.

**ERRORS**

TESESSIONBAD An incorrect session identifier was supplied TESESSIONCLOSED The connection is no longer open.

**EXAMPLE**

TEXSTRING version;

...

TexVersion(session, &version);

if (strcmp(version, "5.0.12") != 0)

{

printf("Program requires version 5.0.12\n");

exit(1);

}

**SEE ALSO**

TexInitialise, TexConnect, TexDisconnect

*KE Texpress API Guide*

Initialisation and Termination 3-11

**Table Access**

**NAME**

TexTable - open a table for access

**SYNOPSIS**

int

TexTable(session, table)

TEXSESSION session;

TEXSTRING table;

**DESCRIPTION**

Opens a KE Texpress table on the machine whose session identifier is *session* for access by subsequent API calls. By default tables are opened when first referenced in any session. Tables remain open until the session is terminated.

This function may be used to explicitly open a table so as to provide more specialised error diagnostics.

**RETURN VALUES**

0 Table was opened successfully.

-1 Table could not be opened.

**ERRORS**

TESESSIONBAD An incorrect session identifier was supplied TESESSIONCLOSED The connection is no longer open.

TETABLEFAIL No such KE Texpress table.

TENOREG User is not a registered user of the table.

TENOINIT Table is not initialised.

TELOGON Unable to use the table at this time.

TETABLESTART General table start up error.

TETABLEREAD Unable to read table description.

TETABLENOISE Table noise word file start up error.

**EXAMPLE**

if (TexTable(session, "loans") < 0)

{

if (TexError() == TETABLEFAIL)

printf("loans table not found\n");

...

}

*KE Texpress API Guide*

3-12 Initialisation and Termination

**SEE ALSO**

TexCommand, TexInitialise, TexTerminate

*KE Texpress API Guide*

Initialisation and Termination 3-13

**Server Configuration**

**NAME**

TexConfQuote - set the text quoting character

**SYNOPSIS**

int

TexConfQuote(session, quote)

TEXSESSION session;

char \*quote;

**DESCRIPTION**

Sets the text quoting character to the character pointed to by the *quote* argument for the server running on the machine identified by *session*.

This function is used to modify the text quoting character. If the character pointed to by the *quote* argument is null (the '\0' character) the default quote character is used (usually the single quote, '\''). The value pointed to by the *quote* argument is filled with the character which is used.

**RETURN VALUES**

0 Server was configured.

-1 Server could not be configured.

**ERRORS**

TESESSIONBAD An incorrect session identifier was supplied TESESSIONCLOSED The connection is no longer open.

**EXAMPLE**

TEXSESSION session;

TEXCURSOR cursor;

char chr, \* cmd;

...

chr = ':';

if (TexConfQuote(Session, & chr) < 0)

/\* handle error \*/

cmd = "contacts where surname = : o'connor:";

if (TexCommand(session, cmd, &cursor) < 0)

/\* handle error \*/

**SEE ALSO**

TexError, TexCommand, TexInitialise, TexTerminate

*KE Texpress API Guide*

3-14 Initialisation and Termination

**Server Interruption**

**NAME**

TexInterrupt - interrupt a server function

**SYNOPSIS**

int

TexInterrupt(session)

TEXSESSION session;

**DESCRIPTION**

Interrupts the function which is executing on the machine whose session identifier is *session*.

This function is typically called by a signal handler which has responded to the user interrupting the front-end process during the execution of another API operation. The function which is interrupted will return with an error status and a subsequent call to TexError() will return the result code TEINTERRUPT.

**RETURN VALUES**

0 Server was interrupted.

-1 Server could not be interrupted.

**ERRORS**

TESESSIONBAD An incorrect session identifier was supplied TESESSIONCLOSED The connection is no longer open.

**EXAMPLE**

TEXSESSION Session;

...

extern void handler( int);

...

signal( SIGINT, handler);

...

if (TexCommand(Session, "contacts where remarks \

contains ' f\*'") < 0)

{

switch ( TexError())

{

case TEINTERRUPT:

printf(" TexCommand was interrupted\n");

break;

}

...

*KE Texpress API Guide*

Initialisation and Termination 3-15

}

...

void

handler( sig)

int sig;

{

/\* User interrupted front-end so request any

\*\* back-end o peration is terminated.

\*/

TexInterrupt(Session);

}

**SEE ALSO**

TexError, TexCommand, TexInitialise, TexTerminate

*KE Texpress API Guide*

3-16 Initialisation and Termination

**Server Disconnection**

**NAME**

TexDisconnect - close an established connection

**SYNOPSIS**

int

TexDisconnect(session)

TEXSESSION session;

**DESCRIPTION**

Closes a previously established connection to a KE Texpress server. The session identifier is given in the *session* argument. All open cursors associated with the session are closed.

**RETURN VALUES**

0 API was successfully terminated.

-1 The terminate request failed.

**ERRORS**

TESESSIONBAD An incorrect session identifier was supplied TESESSIONCLOSED The connection is no longer open.

**EXAMPLE**

main(argc, argv)

int argc;

char \*\*argv

{

...

if (TexConnect(& params, &session) < 0)

/\* connection failure \*/

...

TexDisconnect(session);

}

**SEE ALSO**

TexInitialise, TexConnect, TexParams

*KE Texpress API Guide*

Initialisation and Termination 3-17

**API Termination**

**NAME**

TexTerminate - terminate the API

**SYNOPSIS**

int

TexTerminate()

**DESCRIPTION**

Terminates the KE Texpress API. No other calls should be made to API functions after this function is called. This call shutdowns all open connections thereby closing all previously accessed tables. All memory used by the API is freed.

**RETURN VALUES**

0 API was successfully terminated.

-1 The terminate request failed.

**ERRORS**

None

**EXAMPLE**

main(argc, argv)

int argc;

char \*\*argv

{

TEXPARAMS params;

...

if (TexInitialise(& argc, argv, & params) < 0)

/\* initialisation failure \*/

...

TexTerminate();

exit(0);

}

**SEE ALSO**

TexInitialise, TexConnect

*KE Texpress API Guide*

Cursors 4-1

**Chapter 4**

**Cursors**

Texql Command...................................................................... 4-3 Cursor Type............................................................................. 4-6 Close Cursor............................................................................ 4-7 Merge Cursors......................................................................... 4-8 Sort Cursor.............................................................................. 4-9

*KE Texpress API Guide*

4-2 Cursors

**Overview**

All access to tables is initially via a Texql statement sent to the API. Any valid Texql command may be sent to the API for processing. When the command succesfully completes a cursor is assigned and returned to the calling function. This cursor is used for subsequent column and data access.

Two groups of functions are provided for use with a cursor. The TexCol() group of functions can be used to access the column structure for the operation as well as obtain an actual column data value. The TexRow() group of commands can be used to manipulate the row markers which indicate the next area of data to be retrieved.

It is perfectly acceptable to have multiple cursors in operation at the same time. When a cursor is no longer required it should be closed.

A cursor may refer to the result of any Texql statement. This means that a cursor may refer to a table, a tuple or an atom. Generally a cursor is used to manipulate the rows and columns of a table. However, nested cursors (see Column Nested Cursor) may be used to access the nested columns and rows of nested tables or the columns of a nested tuple.

A cursor is associated with a specific connection. The connection's session identifier is stored with the cursor and so need not be passed to subsequent API calls which use the cursor.

*KE Texpress API Guide*

Cursors 4-3

**Texql Command**

**NAME**

TexCommand - perform a Texql command

**SYNOPSIS**

int

TexCommand(session, command, cursor)

TEXSESSION session;

TEXSTRING command;

TEXCURSOR \*cursor;

**DESCRIPTION**

Performs the requested command on the server identifer by the *session* argument. The *command* argument is interpreted as a texql command. If the command completes successfully then a Texql cursor is opened and assigned to the *cursor* parameter. This cursor is then used to access data and other information associated with the command.

For Texql query commands the cursor can be used in subsequent API calls to retrieve the row and column data that matched the query. Initially the row marker is placed at the first row that matched the query. The TexRowNext() and TexRowGet() functions may be used as required to manipulate the row marker. The TexCol() group of functions are used to actually access the data.

For Texql describe commands the cursor can be used to retrieve the resulting column structure for the command. The TexCol() group of functions are used to access the column structure. As no actual data is associated with a describe cursor the TexColDataGet() and TexColDataSet() functions and the TexRow() group of functions are invalid in this instance.

For Texql data manipulation (DML) commands the cursor is made available so the number of rows manipulated can be determined using TexRowCnt(). No other operations (other than TexClose()) are valid on DML cursors.

**RETURN VALUES**

0 The command succeeded.

-1 The command failed.

**ERRORS**

TECURNOMORENo more cursors are available.

*KE Texpress API Guide*

4-4 Cursors

TESYNTAX Syntax error in command.

Many other error codes are possible. Refer to "texapi.h" for a complete list of error codes. Also refer to the Texql Guide for a complete description of the Texql language and of the error codes generated.

**EXAMPLES**

TEXSESSION session;

TEXCURSOR cursor;

...

if (TexCommand(session, "select all from contacts",

&cursor) < 0)

/\* Texql command failed - check error \*/

...

TEXSESSION session;

char \*cntry, cmd[128];

int maxexp;

TEXCURSOR cursor;

..

cntry = "Japan";

maxexp = 300000;

...

sprintf( cmd, "contacts where country = '%s'

and exposure <= %d", cntry, maxexp);

if (TexCommand(session, cmd, &cursor) < 0)

/\* check error \*/

...

else

/\* can now process matching values \*/

...

TEXSESSION session;

char table[30], cmd[128];

TEXCURSOR cursor;

...

printf("Table to describe? ");

if (gets(table))

{

...

sprintf( cmd, "describe %s", table);

if (TexCommand(session, cmd, &cursor) < 0)

/\* check error \*/

...

else

/\* can now determine column structure \*/

...

...

}

*KE Texpress API Guide*

Cursors 4-5

TEXSESSION session;

int contact;

char cmd[128];

TEXCURSOR cursor;

...

contact = 9;

...

sprintf( cmd, "delete from contacts

where contactno = %d", contact);

if (TexCommand(session, cmd, &cursor) < 0)

/\* check error \*/

...

else

/\* deletion was successful \*/

**SEE ALSO**

TexError, TexErrMsg, TexErrOff, TexColCursor, TexClose

*KE Texpress API Guide*

4-6 Cursors

**Cursor Type**

**NAME**

TexType - type of cursor

**SYNOPSIS**

int

TexType(cursor, type)

TEXCURSOR cursor;

int \*type;

**DESCRIPTION**

Determines the type of a cursor. The type is assigned to the type variable passed as a parameter.

Cursor types are: TEXCURQUERY

TEXCURDESCRIBE

TEXCURINSERT

TEXCURUPDATE

TEXCURDELETE

**RETURN VALUES**

0 Cursor type determined successfully.

-1 Unable to determine cursor type.

**ERRORS**

TECURBAD Bad cursor.

**EXAMPLE**

TEXCURSOR curs;

int curtype;

...

if (TexType(curs, & curtype) < 0)

/\* check error \*/

...

...

switch ( curtype)

{

case TEXCURQUERY:

printf("cursor from Texql query\n");

break;

case TEXCURDELETE:

printf("cursor from Texql delete\n");

break;

*KE Texpress API Guide*

Cursors 4-7

**SEE ALSO**

TexCommand

*KE Texpress API Guide*

4-8 Cursors

**Close Cursor**

**NAME**

TexClose - close a cursor

**SYNOPSIS**

int

TexClose(cursor)

TEXCURSOR cursor;

**DESCRIPTION**

Closes and frees a cursor. The cursor may be an outer cursor or a nested cursor. Closing a cursor will result in closure of all associated nested cursors.

**RETURN VALUES**

0 The cursor was closed successfully.

-1 The cursor could not be closed.

**ERRORS**

TECURBAD Bad cursor.

**EXAMPLE**

TEXCURSOR cursor;

...

if (TexCommand("select...", &cursor) < 0)

...

TexClose(cursor);

**SEE ALSO**

TexCommand, TexCursor

*KE Texpress API Guide*

Cursors 4-9

**Merge Cursors**

**NAME**

TexMerge - merge two cursors

**SYNOPSIS**

int

TexMerge(cursor1, cursor2, dups)

TEXCURSOR cursor1, cursor2;

TEXS32 \*dups;

**DESCRIPTION**

Merge the two cursors, cursor1 and cursor2. More specifically the rows in cursor2 are appended to cursor1, with duplicates being removed. The cursor2 is left unchanged. The number of duplicates removed is set in dups.

Both cursors must be referencing the same base table.

**RETURN VALUES**

0 Cursors were merged successfully.

-1 Cursors could not be merged.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

TEBASETABLE Not a base table cursor.

TEMRGNAMESBAD Cursors reference different base tables.

**EXAMPLE**

TEXCURSOR cursor1, cursor2;

TEXS32 dups;

...

if (TexCommand("contacts where surname = 'Johnson',

&cursor1, & dups) < 0)

...

if (TexCommand("contacts where postcode = '3220',

&cursor2, & dups) < 0)

...

if (TexMerge(cursor1, cursor2, & dups) < 0)

...

printf("%d duplicates", dups);

*KE Texpress API Guide*

4-10 Cursors

**SEE ALSO**

TexCommand, TexCursor

*KE Texpress API Guide*

Cursors 4-11

**Sort Cursor**

**NAME**

TexSort - sort a cursor

**SYNOPSIS**

int

TexSort(cursor, sortinfo)

TEXCURSOR cursor;

TEXSORT sortinfo;

**DESCRIPTION**

Sort the rows of the cursor. The argument sortinfo contains an array of column names and sorting direction flags. The array must be terminated by an element with a NULL column name.

This function resets the row marker to the first row.

**RETURN VALUES**

0 Sorting was performed successfully.

-1 Sorting failed.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

TECOLNAMEBAD Bad column name.

TESORTCURSOR Sorting error.

**EXAMPLE**

TEXCURSOR cursor;

TEXSORT sortinfo[3];

...

sortinfo[0]. s\_colname = "surname";

sortinfo[0]. s\_flags = SORT\_ASCEND;

sortinfo[1]. s\_colname = " firstnam";

sortinfo[1]. s\_flags = SORT\_DESCEND;

sortinfo[2]. s\_colname = ( TEXSTRING) NULL;

sortinfo[2]. s\_flags = 0; /\* doesn't really matter \*/

if (TexSort(cursor, sortinfo) < 0)

/\* error \*/

**SEE ALSO**

TexCommand, TexCursor

*KE Texpress API Guide*

Row Access 5-1

**Chapter 5**

**Row Access**

Next Row................................................................................ 5-3 Get Row.................................................................................. 5-5 Move Row............................................................................... 5-6 Row Position........................................................................... 5-7 Row Reset............................................................................... 5-8 Count Rows............................................................................. 5-9 Number of Row Hits.............................................................. 5-10 Lock Row.............................................................................. 5-11 Unlock Row.......................................................................... 5-12 Row Status............................................................................ 5-13 New Row.............................................................................. 5-14 Save Row.............................................................................. 5-16 Discard Row.......................................................................... 5-17 Delete Row............................................................................ 5-18

*KE Texpress API Guide*

5-2 Row Access

**Overview**

The TexRow() group of commands can be used to manipulate the row marker which indicates the next row to be accessed.

*KE Texpress API Guide*

Row Access 5-3

**Next Row**

**NAME**

TexRowNext - retrieve next row

**SYNOPSIS**

int

TexRowNext(cursor)

TEXCURSOR cursor;

**DESCRIPTION**

Retrieves the next row of the table associated with the cursor. The data is loaded into internal buffers in readiness for access using the TexColDataGet() function.

Repeated calls to TexRowNext() will eventually result in the TEEOF error code being set when all rows are exhausted. A TEEOF error code on a TexRowNext() call should not be viewed as an error but rather an indication that there is no more row data for the specified cursor.

A call to the TexRowGet() function alters the current row marker.

**RETURN VALUES**

0 The next row was retrieved successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

TEEOF No more rows.

**EXAMPLES**

TEXCURSOR cursor;

TEXSTRING name;

...

if (TexCommand("contacts where contno = 13", &cursor) < 0) ...

if (TexRowNext(cursor) < 0)

/\* check error \*/

...

if (TexColDataGet(cursor, "surname", &name) < 0)

/\* check error \*/

...

*KE Texpress API Guide*

5-4 Row Access

printf("Surname of contact 13 is % s\n", name);

TEXCURSOR cursor;

TEXSTRING name;

...

printf("Loan types\n");

if (TexCommand(" loantypes", &cursor) < 0)

/\* check error \*/

while ( TexRowNext(cursor) == 0)

{

if (TexColDataGet(cursor, " loanname", &name) < 0)

/\* check error \*/

printf("% s\n", name);

}

if (TexError() != TEEOF)

/\* real error \*/

...

TEXCURSOR cursor, catcur;

TEXSTRING category;

...

if (TexCommand("loans where contno = 13",&cursor) < 0) ...

if (TexRowNext(cursor) < 0)

...

if (TexColCursor(cursor, " category\_tab", & catcur) < 0) ...

printf("Loan categories of contact 13:\n");

while( TexRowNext( catcur) == 0)

{

...

if (TexColDataGet( catcur, " category",&category) <

0)

/\* check error \*/

...

printf("% s\n", category);

...

**SEE ALSO**

TexRowGet, TexColDataGet

*KE Texpress API Guide*

Row Access 5-5

**Get Row**

**NAME**

TexRowGet - retrieve a specific row number

**SYNOPSIS**

int

TexRowGet(cursor, rownum)

TEXCURSOR cursor;

TEXS32 rownum;

**DESCRIPTION**

Retrieves the specified row of the table associated with the cursor. The data is loaded into internal buffers in readiness for access using the TexColDataGet() function. Rows are numbered from 1 to TexRowCnt(cursor). Subsequent calls to TexRowNext() will retrieve rows commencing from one greater than the row specified in the TexRowGet() call.

**RETURN VALUES**

0 The next row was retrieved successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

TEEOF The rownum is out of range.

**EXAMPLE**

TEXCURSOR cursor;

TEXS32 numrows, row;

if (TexCommand("order contacts on exposure", &cursor) < 0) /\* check error \*/

if (TexRowCnt(cursor, & numrows) < 0)

/\* check error \*/

/\* print rows in reverse order \*/

for (row = numrows; row; row--)

{

if (TexRowGet(cursor, row) < 0)

/\* check error \*/

/\* print row \*/

...

**SEE ALSO**

TexRowNext, TexColDataGet

*KE Texpress API Guide*

5-6 Row Access

**Move Row**

**NAME**

TexRowMove - move the row number relative to the current position

**SYNOPSIS**

int

TexRowMove(cursor, rownum)

TEXCURSOR cursor;

TEXS32 rownum;

**DESCRIPTION**

Retrieves a row of the table associated with the cursor relative to the current row position. The data is loaded into internal buffers in readiness for access using the TexColDataGet() function. The supplied row number may be positive (specifying a move forward) or negative (specifying a move backward). Subsequent calls to TexRowNext() will retrieve rows commencing from one greater than the row specified by the TexRowMove() call.

**RETURN VALUES**

0 The next row was retrieved successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

TEEOF The derived rownum is out of range.

**EXAMPLE**

TEXCURSOR cursor;

TEXS32 numrows, row;

if (TexCommand("contacts", &cursor) < 0)

/\* check error \*/

if (TexRowCnt(cursor, & numrows) < 0)

/\* check error \*/

/\* print rows in reverse order \*/

for (row = numrows; row; row--)

{

...

if (TexRowMove(cursor, -1) < 0)

/\* check error \*/

**SEE ALSO**

TexRowGet, TexColDataGet

*KE Texpress API Guide*

Row Access 5-7

**Row Position**

**NAME**

TexRowPos - determine row marker position

**SYNOPSIS**

int

TexRowPos(cursor, rownum)

TEXCURSOR cursor;

TEXS32 \*rownum;

**DESCRIPTION**

Determine the current row number position of the row marker.

**RETURN VALUES**

0 Row marker determined.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

**EXAMPLE**

TEXCURSOR cursor;

TEXS32 rownum;

...

if (TexRowPos(cursor, & rownum) < 0)

/\* check error \*/

...

printf("Row marker at row % ld\n", (long) rownum);

**SEE ALSO**

TexRowNext, TexRowGet, TexRowReset

*KE Texpress API Guide*

5-8 Row Access

**Row Reset**

**NAME**

TexRowReset - reset the cursor row marker

**SYNOPSIS**

int

TexRowReset(cursor)

TEXCURSOR cursor;

**DESCRIPTION**

Resets the row marker associated with the cursor such that the next call to TexRowNext() will retrieve the first row of the table.

**RETURN VALUES**

0 Reset successfully completed.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

**EXAMPLE**

TEXCURSOR cursor;

...

if (TexRowGet(cursor, (TEXS32) 7) < 0)

/\* check error \*/

...

/\* process row 7 \*/

...

if (TexRowReset(cursor) < 0)

/\* check error \*/

...

/\* start processing again from row 1 \*/

while ( TexRowNext(cursor) == 0)

{

...

**SEE ALSO**

TexRowNext

*KE Texpress API Guide*

Row Access 5-9

**Count Rows**

**NAME**

TexRowCnt - count the number of rows

**SYNOPSIS**

int

TexRowCnt(cursor, nrows)

TEXCURSOR cursor;

TEXS32 \*nrows;

**DESCRIPTION**

Counts the number of rows associated with the cursor. For a Texql query command, TEXCURQUERY, this function will assign to the parameter the number of rows that matched the query.

For a Texql data manipulation command, TEXCURINSERT, TEXCURUPDATE or TEXCURDELETE, this function will assign to the parameter the number of rows that were inserted, updated or deleted respectively.

This function resets the row marker to the first row of the table.

**RETURN VALUES**

0 Row count successfully accessed.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECOLTYPE Bad column type.

**EXAMPLE**

TEXCURSOR cursor;

TEXS32 numloans;

...

if (TexCommand("loans", &cursor) < 0)

...

if (TexRowCnt(cursor, & numloans) < 0)

/\* check error \*/

...

printf("Table contains % ld loans\n", (long) numloans);

...

**SEE ALSO**

TexRowNext, TexRowGet, TexRowReset

*KE Texpress API Guide*

5-10 Row Access

**Number of Row Hits**

**NAME**

TexRowHits - determine the number of row hits

**SYNOPSIS**

int

TexRowHits(cursor, nhits)

TEXCURSOR cursor;

TEXS32 \*nhits;

**DESCRIPTION**

Determines the number of hits associated with a query cursor (TEXCURQUERY. For query commands in which column attributes were provided that enabled the index to be utilised this function sets the number of row hits. In this case TexRowHits is significantly faster than TexRowCnt.

If indexing information was not able to be utilised by the query then -1 is returned in the hits variable and TexRowCnt must be used to determine the number of matching rows.

This function resets the row marker to the first row of the table.

**RETURN VALUES**

0 Function completed normally.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

**EXAMPLE**

TEXCURSOR cursor;

TEXS32 hits;

...

if (TexCommand("loans where term = 12", &cursor) < 0)

...

if (TexRowHits(cursor, &hits) < 0)

/\* check error \*/

if (hits == (TEXS32) -1)

/\* unable to determine hits \*/

else

printf("Hit % ld rows\n", (long) hits);

**SEE ALSO**

TexRowNext, TexRowGet, TexRowCnt, TexRowReset

*KE Texpress API Guide*

Row Access 5-11

**Lock Row**

**NAME**

TexRowLock - lock a row

**SYNOPSIS**

int

TexRowLock(cursor, rownum)

TEXCURSOR cursor;

TEXS32 rownum;

**DESCRIPTION**

Place a lock on the rownum'th row of the cursor. This provides the cursor with exclusive update access to that row. Locking will fail if some other cursor or Texpress program has already obtained an exclusive row lock.

If the rownum is TEXROWCURRENT then the current row is locked. If the rownum is TEXROWALL then all rows of the cursor are locked.

The cursor must be referencing a base table.

**RETURN VALUES**

0 The row(s) was locked successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

TEROWLOCK Lock failed.

**EXAMPLE**

TEXCURSOR cursor;

...

/\* lock row 3 \*/

if (TexRowLock(cursor, (TEXS32) 3) < 0)

/\* check error \*/

...

**SEE ALSO**

TexRowUnlock, TexRowStatus

*KE Texpress API Guide*

5-12 Row Access

**Unlock Row**

**NAME**

TexRowUnlock - unlock a row

**SYNOPSIS**

int

TexRowUnlock(cursor, rownum)

TEXCURSOR cursor;

TEXS32 rownum;

**DESCRIPTION**

Unlock on the rownum'th row of the cursor. It is only possible to unlock a row that was previously locked by a call to TexRowLock using the same cursor.

If the rownum is TEXROWCURRENT then the current row is unlocked. If the rownum is TEXROWALL then all rows of the cursor are unlocked.

The cursor must be referencing a base table.

**RETURN VALUES**

0 The row was unlocked successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

TEROWUNLOCK Unlock failed.

**EXAMPLE**

TEXCURSOR cursor;

...

/\* unlock row 3 \*/

if (TexRowUnlock(cursor, (TEXS32) 3) < 0)

/\* check error \*/

...

**SEE ALSO**

TexRowLock, TexRowStatus

*KE Texpress API Guide*

Row Access 5-13

**Row Status**

**NAME**

TexRowStatus - determine status of a row

**SYNOPSIS**

int

TexRowStatus(cursor, rownum, status)

TEXCURSOR cursor;

TEXS32 rownum;

TEXU32 \*status;

**DESCRIPTION**

Determine the status of the rownum'th row of the cursor. The status indicates if the row has been locked. If the row is the current row the status also indicates if the row has been modified. The cursor must be referencing a base table.

The status flag is a bit map where any of the following may be set:

TEXROWMODIFIED /\* row has been modified \*/

TEXROWCURSORLOCK /\* row locked by this cursor \*/

TEXROWOTHERLOCK /\* row locked by other cursor \*/

TEXROWUPDATED /\* row updated by other cursor \*/

TEXROWDELETED /\* row deleted by other cursor \*/

**RETURN VALUES**

0 Row status determined successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TEROWSTATUS Row status failed.

**EXAMPLE**

TEXCURSOR cursor;

TEXU32 status;

if (TexRowStatus(cursor, (TEXS32) 3, &status) < 0)

/\* check error \*/

if (status & TEXROWOTHERLOCK)

...

**SEE ALSO**

TexRowLock, TexRowUnlock

*KE Texpress API Guide*

5-14 Row Access

**New Row**

**NAME**

TexRowNew - create a new row

**SYNOPSIS**

int

TexRowNew(cursor, rownum, keydata)

TEXCURSOR cursor;

TEXS32 rownum;

TEXARRAY \*keydata;

**DESCRIPTION**

Create a new row in the cursor. The row is created imediately before the row at position, rownum, or if rownum is TEXROWAPPEND or is greater than TexRowCnt() the row is added to the end of the table. The row marker is set to the new row. The cursor must be referencing a base table.

The keydata parameter is used to specify the primary key value to be assigned to the new row. It comprises an NULL terminated array of TEXSTRING values which are assigned in turn to each of the columns of the primary key tuple. If the table does not have a primary key then the primary key parameter must be NULL.

If keydata is NULL or values are provided for only some of the primary key columns then, if possible, an automatic primary key will be assigned.

A new row does not become permanent until TexRowSave() is called. A new row mau be discarded prior to saving by calling TexRowDiscard() with the appropriate row number.

Following a TexRowNew() call and prior to any TexRowSave() or TexRowDiscard() call, calls made to any other TexRow function which may move the row marker will result in the new row being silently discarded.

**RETURN VALUES**

0 Row created successfully.

-1 An error occurred.

*KE Texpress API Guide*

Row Access 5-15

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

TEKEYNONE Table does not have a primary key. TEKEYFAIL Failed to assign primary key.

TEKEYBAD Badly formed primary key.

TEKEYDUP Duplicate primary key.

**EXAMPLE**

TEXCURSOR cursor;

TEXSTRING keydata[2];

...

/\* Create a new row at row position 3 of the cursor,

\*\* and assign a primary key value of 10.

\*/

keydata[0] = "10";

keydata[1] = ( TEXSTRING) NULL;

if (TexRowNew(cursor, (TEXS32) 3, keydata) < 0)

/\* check error \*/

...

*/\* perform further editing using TexColDataSet()*

*\*/*

...

if (*want to permanently save row*)

{

if (TexRowSave(cursor) < 0)

/\* check error \*/

}

else /\* discard row \*/

{

if (TexRowDiscard(cursor, TEXROWCURRENT) < 0)

/\* check error \*/

}

**SEE ALSO**

TexRowSave, TexRowDiscard

*KE Texpress API Guide*

5-16 Row Access

**Save Row**

**NAME**

TexRowSave - save the current row

**SYNOPSIS**

int

TexRowSave(cursor)

TEXCURSOR cursor;

**DESCRIPTION**

Save the current row of the cursor. This creates or updates a permanent row in the table. The cursor must be referencing a base table.

The TEXROWMODIFIED flag is cleared by this function. A row lock is not removed by this function.

**RETURN VALUES**

0 Row status determined successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

TEROWSTATUS Row status failed.

**EXAMPLE**

TEXCURSOR cursor;

...

if (TexRowSave(cursor) < 0)

/\* check error \*/

**SEE ALSO**

TexRowNew, TexColDataSet, TexRowDiscard

*KE Texpress API Guide*

Row Access 5-17

**Discard Row**

**NAME**

TexRowDiscard - discard a row

**SYNOPSIS**

int

TexRowDiscard(cursor, rownum)

TEXCURSOR cursor;

TEXS32 rownum;

**DESCRIPTION**

Discard the rownum'th row of the cursor. If rownum is TEXROWCURRENT then the current row is discarded. This discards the row from the current cursor only, it does not delete the row from the table.

The cursor must be referencing a base table.

After a call to TexRowNew(), but prior to a call to TexRowSave(), TexRowDiscard() may be called to discard the new row.

**RETURN VALUES**

0 Row discarded successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

**EXAMPLE**

TEXCURSOR cursor;

...

/\* discard row 3 from cursor \*/

if (TexRowDiscard(cursor, (TEXS32) 3) < 0)

/\* check error \*/

**SEE ALSO**

TexRowDelete, TexRowNew

*KE Texpress API Guide*

5-18 Row Access

**Delete Row**

**NAME**

TexRowDelete - delete a row

**SYNOPSIS**

int

TexRowDelete(cursor, rownum)

TEXCURSOR cursor;

TEXS32 rownum;

**DESCRIPTION**

Delete the rownum'th row of the cursor. If rownum is TEXROWCURRENT then the current row is deleted. This permanently deletes the row from the table.

The cursor must be referencing a base table.

**RETURN VALUES**

0 Row deleted successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECURSNOTQRY Not a query cursor.

TECURSDESC This is a describe cursor.

TELOCKREC Unable to lock row.

**EXAMPLE**

TEXCURSOR cursor;

...

/\* delete row 3 \*/

if (TexRowDelete(cursor, (TEXS32) 3) < 0)

/\* check error \*/

**SEE ALSO**

TexRowDiscard

*KE Texpress API Guide*

Column Access 6-1

**Chapter 6**

**Column Access**

Column Names........................................................................ 6-3 Column Kind........................................................................... 6-4 Column Type........................................................................... 6-5 Column Nested Cursor............................................................. 6-6 Column Data Get..................................................................... 6-7 Column Data Set...................................................................... 6-9

*KE Texpress API Guide*

6-2 Column Access

**Overview**

The TexCol() group of functions can be used to access the column structure for the operation as well as get and set column data values.

*KE Texpress API Guide*

Column Access 6-3

**Column Names**

**NAME**

TexColNames - access column names

**SYNOPSIS**

int

TexColNames(cursor, colnames)

TEXCURSOR cursor;

TEXARRAY \*colnames;

**DESCRIPTION**

Retrieves an ordered list of the column names associated with the cursor and assigns it to the colnames parameter. Upon successful completion colnames will point to a NULL terminated array of column names.

**RETURN VALUES**

0 Column names accessed successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECOLTYPEBAD Bad column type.

**EXAMPLE**

TEXCURSOR cursor;

TEXARRAY loannames;

int i;

...

if (TexCommand("describe loans", &cursor) < 0)

...

if (TexColNames(cursor, & loannames) < 0)

/\* check error \*/

...

for (i = 0; loannames[ i]; i++)

printf("% s\n", loannames[ i]);

**SEE ALSO**

TexColKind, TexColType

*KE Texpress API Guide*

6-4 Column Access

**Column Kind**

**NAME**

TexColKind - determine column kind

**SYNOPSIS**

int

TexColKind(cursor, colname, kind)

TEXCURSOR cursor;

TEXSTRING colname;

int \*kind;

**DESCRIPTION**

Retrieves the kind of column for colname. The colname must be the name of a valid column of the table or tuple associated with cursor.

Columns kinds are:TEXKINDTABLE

TEXKINDTUPLE

TEXKINDATOM

**RETURN VALUES**

0 Column kind accessed successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECOLNAMEBAD Bad column name.

TECOLTYPEBAD Bad column type.

**EXAMPLE**

TEXCURSOR cursor;

int kind;

...

if (TexCommand("describe loantypes", &cursor) < 0)

...

if (TexColKind(cursor, " modon", &kind) < 0)

/\* check error \*/

switch (kind)

{

...

case TEXKINDTUPLE:

printf(" modon is a tuple\n");

break;

...

*KE Texpress API Guide*

Column Access 6-5

**SEE ALSO**

TexColType

*KE Texpress API Guide*

6-6 Column Access

**Column Type**

**NAME**

TexColType - determine column type

**SYNOPSIS**

int

TexColType(cursor, colname, type)

TEXCURSOR cursor;

TEXSTRING colname;

int \*type;

**DESCRIPTION**

Retrieves the type of column for the atomic column colname. The colname must be a valid atomic column of the table or tuple associated with cursor.

Columns types are:TEXTYPETEXT

TEXTYPEINTEGER

TEXTYPEFLOAT

**RETURN VALUES**

0 Column kind accessed successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECOLNAMEBAD Bad column name.

TECOLTYPEBAD Not an atomic column type.

TEATOMTYPEBAD Bad atom type.

**EXAMPLE**

TEXCURSOR cursor;

int type;

...

if (TexCommand("loans where loanno = 10", &cursor) < 0)

...

if (TexColType(cursor, "amount", &type) < 0)

/\* check error \*/

...

if (type != TEXTYPEFLOAT)

printf("amount must be a real value\n");

...

*KE Texpress API Guide*

Column Access 6-7

**SEE ALSO**

TexColKind

*KE Texpress API Guide*

6-8 Column Access

**Column Nested Cursor**

**NAME**

TexColCursor - obtain a nested cursor

**SYNOPSIS**

int

TexColCursor(cursor, colname, nested)

TEXCURSOR cursor;

TEXSTRING colname;

TEXCURSOR \*nested;

**DESCRIPTION**

Obtain a nested cursor for the column name colname associated with cursor. This nested cursor may then used in subsequent TexRow() and TexCol() calls. Typically a nested cursor is used to access values in a nested table or tuple. A nested cursor is closed using TexClose(). Closing a cursor will result in closure of all associated nested cursors.

**RETURN VALUES**

0 Nested cursor obtained successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECOLNAMEBAD Bad column name.

TECOLTYPEBAD Not an atomic column type.

**EXAMPLE**

TEXCURSOR cursor, datecur;

int type;

if (TexCommand("describe loantypes", &cursor) < 0)

...

if (TexColCursor(cursor, " modon", & datecur) < 0)

/\* check error \*/

...

if (TexColType( datecur, "modon\_2", &type) < 0)

/\* check error \*/

switch (type)

{

case TEXTYPEINTEGER:

printf(" modon: second field is integer\n");

break;

*KE Texpress API Guide*

Column Access 6-9

**SEE ALSO**

TexColNames, TexColKind

*KE Texpress API Guide*

6-10 Column Access

**Column Data Get**

**NAME**

TexColDataGet - access data for an atomic column

**SYNOPSIS**

int

TexColDataGet(cursor, colname, data)

TEXCURSOR cursor;

TEXSTRING colname;

TEXSTRING \*data;

**DESCRIPTION**

Access the data of column colname associated with the cursor. The column must be an atomic column.

If the data value is NULL then the data variable is assigned a NULL pointer.

**RETURN VALUES**

0 Data obtained successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECOLNAMEBAD Bad column name.

TECOLTYPEBAD Not an atomic column type.

**EXAMPLES**

TEXCURSOR cursor;

TEXSTRING name;

...

if (TexCommand("contacts where contno = 13", &cursor) < 0) ...

if (TexRowNext(cursor) < 0)

/\* check error \*/

...

if (TexColDataGet(cursor, "surname", &name) < 0)

/\* check error \*/

...

printf("Surname of contact 13 is % s\n", name);

*KE Texpress API Guide*

Column Access 6-11

TEXCURSOR cursor;

TEXSTRING name;

...

printf("Loan types\n");

if (TexCommand(" loantypes", &cursor) < 0)

/\* check error \*/

while ( TexRowNext(cursor) == 0)

{

if (TexColDataGet(cursor, " loanname",&name) < 0)

/\* check error \*/

printf("% s\n", name);

}

if (TexError() != TEEOF)

/\* real error \*/

...

TEXCURSOR cursor, catcur;

TEXSTRING category;

...

if (TexCommand("loans where contno = 13",&cursor) < 0) ...

if (TexRowNext(cursor) < 0)

...

if (TexColCursor(cursor, " category\_tab", & catcur) < 0) ...

printf("Loan categories of contact 13:\n");

while( TexRowNext( catcur) == 0)

{

...

if (TexColDataGet( catcur, " category",&category) <

0)

/\* check error \*/

...

printf("% s\n", category);

...

**SEE ALSO**

TexColKind, TexColType, TexColCursor, TexColDataSet

*KE Texpress API Guide*

6-12 Column Access

**Column Data Set**

**NAME**

TexColDataSet - assign data for an atomic column

**SYNOPSIS**

int

TexColDataSet(cursor, colname, data)

TEXCURSOR cursor;

TEXSTRING colname;

TEXSTRING data;

**DESCRIPTION**

Set the column, colname, to the value, data. Updates using the function may only be made to columns of base tables.

This command sets the in memory value only. Data is not permanently stored until TexRowSave() is called.

The TEXROWMODIFIED row flag is set by this function.

**RETURN VALUES**

0 Data assigned successfully.

-1 An error occurred.

**ERRORS**

TECURBAD Bad cursor.

TECOLNAMEBAD Bad column name.

TECOLTYPEBAD Not an atomic column type.

**EXAMPLES**

TEXCURSOR cursor;

TEXSTRING name;

...

if (TexCommand("contacts where contno = 13", &cursor) < 0) ...

if (TexRowNext(cursor) < 0)

/\* check error \*/

...

if (TexColDataSet(cursor, "surname", " Roberts") < 0)

/\* check error \*/

*KE Texpress API Guide*

Column Access 6-13

**SEE ALSO**

TexColCursor, TexColDataGet, TexRowStatus, TexRowSave

*KE Texpress API Guide*

Convenience Functions 7-1

**Chapter 7**

**Convenience Functions**

Item Names............................................................................. 7-3 Item Number of Fields............................................................. 7-4 Item Data Get.......................................................................... 7-5 Item Data Set........................................................................... 7-7 Field Type................................................................................ 7-8 Field Data Get......................................................................... 7-9 Field Data Set........................................................................ 7-10

*KE Texpress API Guide*

7-2 Convenience Functions

**Overview**

KE Texpress Texql and the C-API have inherent support for data

structures more flexible than those provided by version 3.4 databases. To assist programmers in utilising the C-API several version 3.4 convenience functions are provided. These convenience functions provide simplified access to existing version 3.4 databases.

*KE Texpress API Guide*