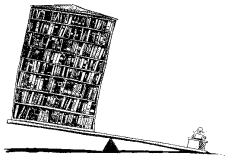
**KE Texpress Texql Guide**

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 Notational Conventions............................................................ 1-4 Texql Features......................................................................... 1-4 Special Symbols....................................................................... 1-6 Reserved Words....................................................................... 1-7 Example Tables........................................................................ 1-8 Contacts Table........................................................... 1-9

Loantypes Table....................................................... 1-11

Loans Table............................................................. 1-12

**Chapter 2 User Interface................................ ...................... 2-1**

Invocation................................................................................ 2-3 Default Invocation................................................................... 2-6 Comments................................................................................ 2-6 Statement Terminator.............................................................. 2-7 Help......................................................................................... 2-7 Command History.................................................................... 2-8 Editing Previous Commands....................................... 2-8

Saving Command Histories......................................... 2-9

Restoring Command Histories.................................... 2-9

Describe Command................................................................ 2-10 Listing Tables........................................................................ 2-11 Reading Input from a File....................................................... 2-12 Writing Output to a File......................................................... 2-13 Termination........................................................................... 2-13

**Chapter 3 Data Types and Attributes................................ ..3-1**

Atomic Data Types.................................................................. 3-3 Null Values.............................................................................. 3-5 Tuples...................................................................................... 3-5 Tables and Nested Tables......................................................... 3-7 Reference Attributes................................................................ 3-7 Type Compatibility................................................................... 3-8 Identifiers................................................................................. 3-8 Renaming Identifiers using As................................................ 3-10

*KE Texpress Texql Guide*

ii Contents

**Chapter 4 Query Language................................ ................. 4-1**

Select-From-Where Expressions.............................................. 4-4 Select......................................................................... 4-5 All Columns.................................................. 4-5

Excluding Columns....................................... 4-6

Structured Attributes.................................... 4-6

Reference Attributes..................................... 4-8

Nested Selection........................................... 4-8

From.......................................................................... 4-9 Where...................................................................... 4-10 Aliasing a Sub-expression using With..................................... 4-11 Scoping Rules........................................................................ 4-12 Extracting Attributes from Tuples ............................ 4-12 Scoping Rules for SFW Expressions......................... 4-13 Functional Notation............................................................... 4-14 Boolean Expressions.............................................................. 4-14 And.......................................................................... 4-14 Or............................................................................ 4-15 Not .......................................................................... 4-15 Relational Operators................................................. 4-16 Comparisons with Null........................................................... 4-17 Null Values............................................................................ 4-17 Relational Operators for Numeric Operands........................... 4-18 Between................................................................................. 4-18 Text Operators....................................................................... 4-19 Equality.................................................................... 4-19 Relational Operators................................................. 4-20 Like ......................................................................... 4-20 Contains................................................................... 4-21 Table and Tuple Operators..................................................... 4-24 Equality.................................................................... 4-24 Tuple Existence........................................................ 4-24 Subset and Superset................................................. 4-25 Comparing Atomic Expressions with Tables........................... 4-26 In............................................................................. 4-26 Has .......................................................................... 4-27 Arithmetic Expressions.......................................................... 4-28 Unary Operators....................................................... 4-28 Binary Operators...................................................... 4-29 Aggregate Functions................................................ 4-30 Min............................................................. 4-30

Max............................................................ 4-31

Sum............................................................ 4-31

Avg............................................................ 4-32

Counting Tuples....................................................... 4-32

*KE Texpress Texql Guide*

Contents iii

Forming Tuples from Tables.................................................. 4-33 Tuple Numbers...................................................................... 4-33 Tuple Access......................................................................... 4-34 Table Expressions.................................................................. 4-35

Times....................................................................... 4-35 Join.......................................................................... 4-35 Union....................................................................... 4-36 Intersect................................................................... 4-36 Except...................................................................... 4-37 Nesting Data into Tables.......................................... 4-37 Unnesting Tables...................................................... 4-38 Removing Duplicate Tuples...................................... 4-39 Sorting Tuples.......................................................... 4-40

Miscellaneous Text Functions................................................ 4-41 Stem ........................................................................ 4-41 Phonetic................................................................... 4-42 Numwords............................................................... 4-42 Word........................................................................ 4-43 Words...................................................................... 4-43

**Chapter 5 Data Manipulation Language............................. 5-1**

Insert....................................................................................... 5-3 Update..................................................................................... 5-5 Delete...................................................................................... 5-7

**Appendix A Example Tables and Data.............................. A-1**

Contacts ................................................................................. A-2 Loan Types............................................................................. A-4 Loans...................................................................................... A-5

**Appendix B Error and Status Codes................................ .. B-1**

Error Codes.............................................................................B-2 Status Codes............................................................................B-5

*KE Texpress Texql Guide*

Introduction 1-1

**Chapter 1**

**Introduction**

Terminology............................................................................ 1-3 Notational Conventions............................................................ 1-4 Texql Features......................................................................... 1-4 Special Symbols....................................................................... 1-6 Reserved Words....................................................................... 1-7 Example Tables........................................................................ 1-8 Contacts Table........................................................... 1-9

Loantypes Table....................................................... 1-11

Loans Table............................................................. 1-12

*KE Texpress Texql 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 Texpress).

This manual describes the query language interface to Texpress. This query language, called Texql, includes both query and data manipulation facilities. While modelled on the industry standard SQL definition, it contains language extensions compatible with the object-oriented features of Texpress. This results in a distinctive SQL "feel" to the interface, significantly reducing the learning time for experienced SQL programmers. But Texql is still able to harness the power of the object-oriented model.

The most significant features of Texql are as follows:

• Support of text attributes.

• Tuples.

• Nested tables.

• Implicit joins using references.

Texql can also be used in conjunction with Titan Version 3.4 databases.

In this manual, each of the Texql language features is presented with a definition and an example of its use and result. The remainder of this chapter discusses the conventions and terminology used throughout this manual as well as the Texpress tables used in the examples.

Chapter 2 describes how Texql is invoked and the many command line options available. A general description of the Texql user interface is also provided.

Chapter 3 discusses the types supported by Texql and how fields can be referenced.

*KE Texpress Texql Guide*

Introduction 1-3

The query interface of Texql is described in Chapter 4. This includes a description of all operators and functions.

The data manipulation interface, covering insertions, updates and deletions, is described in Chapter 5.

Appendix A includes a definition of the three example databases including all of the sample data.

Appendix B provides a list of the error and status messages which can be generated by Texql.

**Terminology**

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 Texpress databases which attempts to simulate a standard SQL interface to a relational database.

This section describes the terminology used by Texql in terms of the appropriate terminology of Texpress. Refer to the Texpress Reference Manuals for a description of Texpress terminology.

The following terms are used throughout this manual:

**Texql** KE Texpress

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

access privileges, etc., are accessible as Texql tables.

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

**nested table** This refers to a 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. one

atomic value of (continuous) text.

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

*KE Texpress Texql Guide*

1-4 Introduction

**nested tuple** The multi-field Key and library items of 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 Texpress record.

**Notational Conventions**

Texql is case insensitive except within text constants. Examples within this manual use the following conventions:

• Texql keywords and symbols are shown in **bold** font and in lower case. • Table and nested table identifiers are shown in *italics*.

• Tuple identifiers and atomic attribute identifiers are shown in normal font. For example within this manual, commands are displayed in the format:.

**select** surname **,** *maillist\_tab*

**from** contacts

**where** surname **= '**Johnson **';**

This command is a common Select - From - Where (SFW) clause. Texql key words may be entered in upper case if desired.

**Texql Features**

Texql provides an SQL-like interface to Texpress databases. In keeping with the object-oriented nature of Texpress, Texql incorporates extensions to the relational model to support the text data type and multi-field items.

To provide text retrieval capabilities on text items in Texpress, Texql supports the **contains** operator which permits text matching facilities identical to Texpress. These facilities include word based queries, stemming and phonetic retrieval operators and phrase retrieval.

Multi-field items are supported in Texql by the definition of nested tables (with the exception of Texpress multi-field text items without a Look-up table which are considered to be atomic values of continuous text). These nested tables

*KE Texpress Texql Guide*

Introduction 1-5

consist of a single column only with each atomic value in each tuple in the nested table equating to a value in one of the fields of the appropriate multi-field item.

Nested tables maintain the object-oriented nature of Texpress applications by accurately representing hierarchically defined complex object structures. The equivalent relational model requires significant decomposition of these objects into a series of separate tables. This results in greatly increased complexity of design and subsequent query access. Although queries can be performed on these decomposed objects and the results of these queries combined, the appropriate hierarchical structure cannot be accurately recreated (as it can never be accurately represented in such systems). Also in relational systems the continual requirement for table joins is potentially very process expensive.

To access information in nested tables, Texql supports a recursive language definition which allows a query expression to contain nested query expressions. In fact, a select-from-where (SFW) expression in a Texql query can be used in any place that an explicit table reference can be used.

Texpress linked Keys are defined in Texql as references. A reference is an implicit join between two tables using a Key value identifier. Key operations are extremely efficient in Texpress and reference attributes mean that an entire table can appear to be nested as an attribute in another table.

In relational databases all join conditions must be provided explicitly. Texql also supports such explicit joins but, in addition, through the use of reference attributes, Texql supports implicit joins. A reference attribute has a domain consisting of the key values of the referenced table. In formulating a query on a table containing a reference attribute, the user is able to view the referenced table entries as belonging to the outer table.

Texql commands can also be expressed in an alternative functional notation. For example, a query to select the surname and mailing list columns for all tuples in the contacts table can be expressed using the standard notation as:

**select** surname, *maillist\_tab*

**from** *contacts*;

where the nested table, *maillist\_tab*, in the same way atomic values are selected. The same query can also be expressed using the functional notation as:

contacts[surname, maillist\_tab];

*KE Texpress Texql Guide*

1-6 Introduction

**Special Symbols**

The following symbols are used in Texql: **( )** Tuple definition and grouping **{}** Row number selection.

**[ ]** Column selection.

**,** Separator.

**= <> < <= > >=**

Logical and relational operators.

**+ - \* / %** Arithmetic operators.

**|** Multiple tuple separator.

**'** Text constant begin and end delimiter. **"** Identifier delimiter.

**:** Inner unnest operator.

**:=** Temporary table assignment, used in with clause. **;** Statement delimiter.

**# --**

Comments indicator

The following operators are supported within a Texql contains clause: **!** Not.

**=** Exact word equivalence (case sensitive). **&** Fold case word equivalence (case insensitive). **~** Word stemming.

**@** Phonetic.

**"** Phrase begin and end delimiter.

*KE Texpress Texql Guide*

Introduction 1-7

**^ $ \* ? [ ] { }**

Pattern matching.

**Reserved Words**

The following are reserved words in Texql and in general should not be used as identifiers.

after all and as

asc avg

before between boolean but

column contains count

default delete desc describe

distinct

e edit except exists

exit

false float forming from

h has help history

ifnull in inner insert

integer intersect into is

join

key

like list

max min

nest not null numwords

of on or order

outer

phonetic preserve

quit

read ref restore rownum

save select set stem

*KE Texpress Texql Guide*

1-8 Introduction

subset sum superset

table text times to

totuple true

union unnest update

values

where with word words

write

If an identifier the same as a reserved word is required then a delimited identifier (identifier in double quotes) may be used. For example a column name of **words** may be referenced using **"words"**.

**Example Tables**

The examples in this manual are based on a simple bank loan registration system. This example illustrates many of the features of Texql. However, it is not intended to be a complete design for such a loan registration system.

The loan registration system comprises the following three tables:

**contacts** This table contains information about the people or organisations who currently have a loan with the bank or are targeted by the bank

for various loan promotions.

**loantypes** This table describes each of the available loan types available and the current applicable interest rate.

**loans** This table contains information about all current loans. It makes references to contacts, for the person or organisation taking out the

loan, and loantypes, for the loan category information.

These tables can be described by the Texql describe command (refer to Chapter 2 for more information). The table descriptions are provided in the following sections.

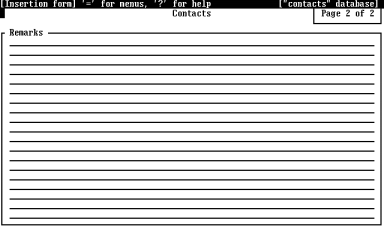
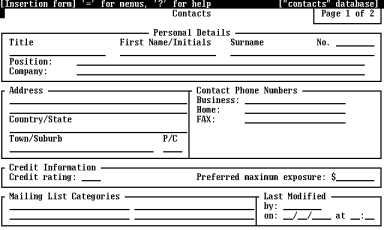
Note that these table descriptions are repeated in Appendix A of this manual together along with a complete list of the data in each table.

*KE Texpress Texql Guide*

**Contacts Table**

Introduction 1-9

The Insertion Form for the *contacts* table is displayed in the folowing two figures.

*KE Texpress Texql Guide*

1-10 Introduction

The *contacts* Texql table description can be obtained using the command: **describe** *contacts***;**

and is as follows:.

contacts[

contno **integer**,

title **text**,

firstnam **text**,

surname **text,**

position **text,**

company **text,**

address **text**,

country **text**,

town **text**,

postcode **text**,

phone **text**,

rating **text,**

exposure **integer**,

*maillist\_tab*[

maillist **text**

],

modby **text**,

*modon*(

modon\_1 **integer**,

modon\_2 **integer**,

modon\_3 **integer**

),

*modat*(

modat\_1 **integer**,

modat\_2 **integer**

),

remarks **text**

];

This table contains a column for the contact number (an integer Key value) plus a variety of simple columns, containing atomic values only, for contact identification and address details.

Each record in this table can include from zero to four mailing list categories. In Texpress terminology, this is a multi-field item. In Texql, it is defined as a nested table. The nested table name is defined as the Texpress item id concatenated with the string \_tab. The nested table contains a single column whose name is the Texpress item id.

The *contacts* table also contains last modified information including Texpress date and time library items. These are defined in Texql nested tuples.

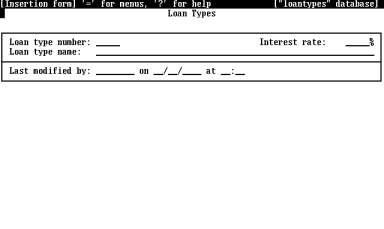
Finally, the remarks item of Texpress is a multi-field text item and is considered by Texql to be a simple atomic value of continuous text.

*KE Texpress Texql Guide*

Introduction 1-11

**Loantypes Table**

The Insertion Form for the *loantypes* table is displayed below.

The *loantypes* Texql table description can be obtained using the command: **describe** *loantypes***;**

and is as follows:

loantypes[

loanno **integer**,

interest **float**,

loanname **text**,

modby **text**,

*modon(*

modon\_1 **integer**,

modon\_2 **integer**,

modon\_3 **integer**

),

*modat*(

modat\_1 **integer**,

modat\_2 **integer**

)

];

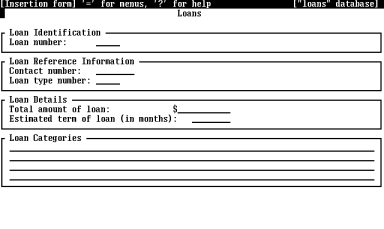
This table is contains columns for loan number (an integer Key value), interest rate, loan type name and last modified information.

*KE Texpress Texql Guide*

1-12 Introduction

**Loans Table**

The Insertion Form for the *loans* table is displayed below.

The *loans* Texql table description can be obtained using the command: **describe** *loans*;

and is as follows:

loans[

loanno **integer**,

*contno*(

contno **integer**

) ref *contacts*,

*typeno(*

loanno **integer**

) ref *loantypes*,

amount **float**,

term **integer**,

*category\_tab*[

category **text**

]

];

The *loans* table contains several simple columns and a nested table for the multi field category item. However, this table also contains references to the *contacts* and *loantypes* tables.

*KE Texpress Texql Guide*

User Interface 2-1

**Chapter 2**

**User Interface**

Invocation................................................................................ 2-3 Default Invocation................................................................... 2-6 Comments................................................................................ 2-6 Statement Terminator.............................................................. 2-7 Help......................................................................................... 2-7 Command History.................................................................... 2-8 Editing Previous Commands....................................... 2-8

Saving Command Histories......................................... 2-9

Restoring Command Histories.................................... 2-9

Describe Command................................................................ 2-10 Listing Tables........................................................................ 2-11 Reading Input from a File....................................................... 2-12 Writing Output to a File......................................................... 2-13 Termination........................................................................... 2-13

*KE Texpress Texql Guide*

2-2 User Interface

**Overview**

Texql is a command line based interpreter. Texql commands may be entered interactively or saved in script files to be run in batch mode. Numerous command line options may be set in order to alter the default behaviour of Texql.

This chapter also describes the user interface features of Texql which are not part of the query or data manipulation languages. These facilities are provided for the entry and editing of Texql commands.

*KE Texpress Texql Guide*

User Interface 2-3

**Invocation**

Texql is invoked from the Unix command interpreter or shell by typing the command:

texql

This commences execution of the interpreter and results in the display of the Texql prompt.

There are many command line options available for Texql which alter the default operation of the command. The complete Texql usage message is as follows:

Usage: texql [-R] [- Ttermtype] [- ceis] [- hn]

[-lfile] [- pstr] [- tstr] [table ...]

Options are as follows:

-R Read only mode. Tables may be accessed on a read only basis. Data manipulation commands are not permitted.

-T*termtype*

Terminal type. This command line option can be used to invoke texql using the Texpress terminal description, *termtype*. This can be useful for the correct interpretation and display of extended ASCII characters. Generally the terminal type setting is determined via the TERM environment variable but may set explicitly using this option.

-c Echo commands As each command line is read it is immediately echoed.

-e Print error/status codes with all system messages. The complexity of a language such as Texql results in a large number of possible input or execution errors. For programs developed to interface to Texql, it is essential to be able to identify and interpret returned error and status messages so that appropriate action can be taken.

This option prefixes a unique numeric identifier to each error or status message. Refer to Appendix C for a complete list of error and status messages.

-h*n* Maintain a command history of *n* commands.

By default, Texql retains a history of the last 100 commands performed during interactive sessions. By default, for non-interactive sessions, no history is retained.

A history of commands enables a user to select a previously performed command and re-execute it or modify it before re-executing. It can be very

*KE Texpress Texql Guide*

2-4 User Interface

useful for correction of simple syntax errors in complex statements or for step-wise refinement of complex operations.

The editing of history commands and the saving and restoring of histories is discussed in more detail later in this chapter.

-i Disable keyboard interrupts for the duration of the session

Texql responds to the keyboard interrupt, (typically generated by pressing the Ctrl+C or DEL key). During interactive sessions, an interrupt results in the current Texql operation being aborted and the next prompt displayed. For non-interactive sessions, an interrupt terminates the session.

-l*file*

Log error messages in *file*. Texql displays error messages directly on the screen before displaying the next prompt. This command line option can be used to also place all error messages in a specified file. This can be useful for debugging Texql scripts or files of commands.

-p*str*

Use a prompt of **str**. The default interactive prompt is:

texql *n*>

where *n* is the command number. There is no prompt by default for non interactive sessions.

This command line option can be used to explicitly set the prompt. If *str* contains the character **!**, then this character is replaced by the current command number (unless it is preceded in the prompt by a backslash character, **\**). For example, the command:

texql - p"next !> "

produces the Texql prompt:

next n>

(where n is the command number), whereas the command: texql - p"next\! "

produces the prompt:

next!>

-s Do not display status messages The data manipulation commands of Texql (insert, update and delete) provide a status message on completion which

*KE Texpress Texql Guide*

User Interface 2-5

indicates the number of rows (records) affected. This command line option can be used to suppress the display of these messages.

-t*str*

Text output delimiter. By default text on output is delimited by single quotes. This command line option can be used to change the begin and end output text delimiters to a different character(s) sequence. For example the command:

texql -t++

would produce text strings on output in the format:

++Johnson++

rather than the standard:

'Johnson'

Note that input text strings are not affected by this option and must still be delimited by single quotes.

*table* ...

Texql opens each Texpress table as it is required, i.e. the first time it is referenced. However, it can be forced to access certain tables upon invocation by appending the table names to the command line. This results in all checks for appropriate access being performed before the first Texql statement involving that table is entered. If the user does not have appropriate access rights then Texql terminates.

The use of table names on the command line does not restrict the tables which can be accessed during that session.

*KE Texpress Texql Guide*

2-6 User Interface

**Default Invocation**

By default, Texql determines whether it is running interactively or its input has been redirected from a file, pipe or socket. For interactive sessions, Texql automatically displays a prompt between commands and sets the prompt to:

texql n>

where n is the command number, commencing from 1.

Commands are not echoed by Texql as echoing is generally performed by the terminal driver. The number of commands retained in the history is set to 100.

For non-interactive sessions, Texql automatically suppresses the display of the prompt. It also does not echo commands or status messages.

**Comments**

Comments are defined in Texql by using a **#** character or **--** character sequence. The remainder of the line after the comment marker is ignored. Comment markers are ignored within text literals (i.e. text enclosed in single or double quotes).

An example Texql command script including comments is as follows:

# select all columns from loans table

#

**select all from** *loans***;**

-- select all columns from loantypes table

-- (SQL style comments)

**select all** -- another comment

**from** *loantypes***;**

*KE Texpress Texql Guide*

User Interface 2-7

**Statement Terminator**

All statements in Texql are terminated by a semi-colon, '**;**'. Statements are free format and may extend over numerous lines. The statement terminator is not recognised in text literals (text enclosed in single or double quotes). No other text (other than comments) should follow on the same line as the statement terminator.

The following two statements are completely equivalent:

# select all columns from loans table.

#

**select all from** *loans***;**

# select all columns from loans table.

#

**select**

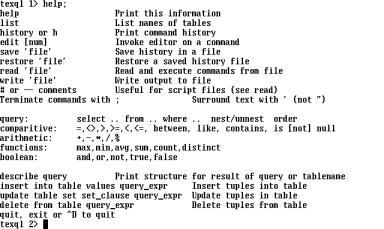
**all**

**from**

*loans***;**

**Help**

The help command prints out some simple help information. An example of the help display is shown below.

*KE Texpress Texql Guide*

2-8 User Interface

**Command History**

By default, for all interactive sessions, Texql maintains a history of the last 100 commands performed in the current session. Non-interactive sessions, by default do not maintain a history. As described earlier in this chapter, Texql supports a command line argument to explicitly set the number of commands retained in the history. This is independent of whether the session is interactive or non interactive..

The **history** command, (also selected by **h**), displays all of the commands retained in the history together with their command numbers.

In conjunction with this command history, Texql implements a simple history substitution mechanism based on that provided by the Unix csh. Previous commands in the history can be included in the current command using **!** followed by a number. If the number is greater than zero, then it is treated as an absolute command number. Otherwise, it is considered to be relative to the current command number. The derived number identifies a command in the command history and the exclamation mark and the command number are replaced by the text of that command.

For example, !5, refers to command number five. The sequence !-1, or alternatively !!, refers to the previous command.

texql 5> # Display the history of commands

texql 5> # in this session.

.......> #

.......> **history;**

1 **count**(*loantypes*)

2 **select all**

**from** *loantypes*

**where** interest **<** 10

3 **max**(*loantypes*[interest])

4 **min**(*loantypes*[interest])

texql 5> # Select the second command,

.......> # add another condition and repeat it.

.......> #

.......> !2 **and** interest **>** 8**;**

The command statement terminator is not saved in the history. Also, the history command itself is never placed in the command history.

**Editing Previous Commands**

A command from the command history can be recalled and edited using the user's preferred editor. The preferred editor is designated during the Texpress installation process. If desired the shell environment variable, EDITOR, may be set to indicate a particular editor.

*KE Texpress Texql Guide*

User Interface 2-9

To edit a command, the user simply types **edit** (or **e**) and the command number, followed by a semi-colon. This invokes the preferred editor on a temporary file containing the command. If no command number is specified, the previous command is used. When the user saves the file and exits the editor, the new command is then added to the history. It is not executed automatically. The !! command must then be used to actually execute the command.

texql 5> # Display the history of commands

.......> # in this session.

.......> #

.......> **history**;

1 **count**(*loantypes*)

2 **select all**

**from** *loantypes*

**where** interest **<** 10

3 **max**(*loantypes*[interest])

4 **min**(*loantypes*[interest])

texql 5> # Edit the second command.

.......> #

.......> **edit** 2;

texql 5> # Invoke the newly edited command.

.......> #

.......> !!;

The edit command itself is never placed in the command history.

**Saving Command Histories**

A copy of the command history can be saved in a file using the command: **save** '*filename*';

This saves the command history in a format suitable for use with the restore command described below.

**Restoring Command Histories**

A command history saved in a file can be restored using the command: **restore** '*filename*';

This restores the saved command history, replacing any existing history.

*KE Texpress Texql Guide*

2-10 User Interface

**Describe Command**

Any valid Texql query statement can be preceded by the keyword, **describe**. This can be used to describe the table returned as a result of that query.

If the query accesses a single table only, then the describe command returns the name of that table. Otherwise the table name is left blank. Then it describes each of the columns of the table, giving the name of the column and its type, both derived from the original tables.

**Statement:**

# Describe the loantypes table

#

**describe** *loantypes*;

**Output:**

*loantypes*[

loanno **integer**,

interest **float**,

loanname **text**,

modby **text**,

modon(

modon\_1 **integer**,

modon\_2 **integer**,

modon\_3 **integer**

),

modat(

modat\_1 **integer**,

modat\_2 **integer**

)

];

**Statement:**

# Describe the result of a query joining two tables.

#

# The query incorporates an explicit join

# rather than the use of the reference.

#

**describe**

(

**select** loanname, term

**from** *loantypes*, *loans*

**where** *loantypes*.loanno = *loans*.loanno

)**;**

**Output:**

[

loanname **text**,

term **integer**

]**;**

*KE Texpress Texql Guide*

User Interface 2-11

**Listing Tables**

The command, **list**, can be used to list all of the tables available on the system. The format of the list is similar to the format produced by the Texpress command, **tlsdb**, when used with the **-l** option.

**Statement:**

# List all of the available tables.

#

**list**;

**Output:**

contacts (texpress) 3 records 4.2%

loans (texpress) 4 records 4.5%

loantypes (texpress) 7 records 4.6%

A table name may also be specified as an optional argument.

**Statement:**

# List just the contacts table.

#

**list** *contacts*;

**Output:**

contacts (texpress) 3 records 4.2%

*KE Texpress Texql Guide*

2-12 User Interface

**Reading Input from a File**

Input to texql can be redirected from the shell command line, as well as by using the Texql **read** command. This command temporarily redirects input to come from the specified file. Commands are read until the end of the file is reached. These commands are not added to the history and history substitution is not performed. Any input errors result in the termination of the read and input again coming from standard input.

Assume that a file called ***saved.commands*** exists and contains the following Texql commands:

# Commands saved in a file for later re-use.

#

**select all**

**from** *loantypes*

**where** interest < 10;

**count**(*loantypes*);

**max**(*loantypes*[interest]);

**min**(*loantypes*[interest]);

**Statement:**

# Read commands from the file, saved.commands.

#

**read** 'saved.commands';

**Output:**

Reading from " saved.commands"

>> # Commands saved in a file for later re-use.

>> #

>> **select all**

>> **from** *loantypes*

>> **where** interest < 10;

(1,9.50,'First home buyer','john',(15,06,1993),(11,50))

>> **count**(*loantypes*);

8

>> **max**(*loantypes*[interest]);

18.00

>> **min**(*loantypes*[interest]);

9.50

>>

Finished reading from " saved.commands"

It is possible to nest read commands so that a file containing commands to be executed can also contain a further read command.

*KE Texpress Texql Guide*

User Interface 2-13

**Writing Output to a File**

The command, **write**, causes all output from Texql to be duplicated and sent to both the standard output and to a file. The write command requires a single argument of a file name surrounded by single quotes.

Entering the write command without specifying a file name indicates that the duplication of output should be ceased.

A sequence such as:

# Write output to the file, saved.output and then

# perform a variety of commands.

#

**write** 'saved.output';

**select all**

**from** *loantypes*

**where** interest **<** 10;

**count**(*loantypes*);

**max**(*loantypes*[interest]);

**min**(*loantypes*[interest]);

**write**;

would result in the following output being saved in the file ***saved.output*** as well as being displayed on the standard output of Texql.:

**select all**

**from** *loantypes*

**where** interest **<** 10;

(1,9.50,'First home buyer','john',(15,06,1993),(11,50))

**count**(*loantypes*);

8

**max**(*loantypes*[interest]);

18.00

**min**(*loantypes*[interest]);

9.50

**write**;

**Termination**

An interactive Texql session can be terminated be entering **exit** or **quit** (followed by a semi-colon). Alternatively, Control-D (EOF) may be typed.

Non interactive sessions terminate when input end of file is detected.

# Exit from Texql.

#

**quit**;

*KE Texpress Texql Guide*

Data Types and Attributes 3-1

**Chapter 3**

**Data Types and Attributes**

Atomic Data Types.................................................................. 3-3 Null Values.............................................................................. 3-5 Tuples...................................................................................... 3-5 Tables and Nested Tables......................................................... 3-7 Reference Attributes................................................................ 3-7 Type Compatibility................................................................... 3-8 Identifiers................................................................................. 3-8 Renaming Identifiers using As................................................ 3-10

*KE Texpress Texql Guide*

3-2 Data Types and Attributes

**Overview**

This chapter describes the format and use of each of the Texql data types and more complex structure attributes of Texql. The mapping of Texpress data types to Texql data types is also described.

*KE Texpress Texql Guide*

Data Types and Attributes 3-3

**Atomic Data Types**

Texql supports the following atomic data types:

**text**

This consists of a sequence of printable characters. Depending on the retrieval operators used text is either treated as a sequence of words, or a single string. Text is interpreted as words using the same rules as Texpress (i.e. the space character and most punctuation characters are word-break characters for most languages, while for multi-byte languages each ideographic character is considered as a separate word).

Text constants are surrounded by single quote characters. The following sequences can be embedded text constants:

**\\** Replaced by **\**.

**\'** Replaced by **'**.

**\"** Replaced by **".**

**\b** Replaced by backspace.

**\f** Replaced by form feed.

**\n** Replaced by newline.

**\r** Replaced by carriage return

**\t** Replaced by tab

**\nnn** Replaced by the character with octal ASCII code nnn (\0 not allowed).

**\xnn**

**\Xnn** Replaced by the character with hexadecimal ASCII code nn (\x0 not allowed).

**\<return>**

Removed from the text constant but results in joining two lines, e.g.

'The quick brown fox jum\

ped over the lazy dog.'

is equivalent to the continuous string:

'The quick brown fox jumped over the lazy dog.'

*KE Texpress Texql Guide*

3-4 Data Types and Attributes

Texpress items of type Text or String map directly to the Texql text atomic type. Expressions which return text values, together with the boolean operators for text expressions, are described in Chapter 4.

**integer**

This consists of an optional leading plus or minus sign followed by one or more digits, e.g.:

3053

The range of valid integer values is machine dependent but generally at least from -231 to 231 - 1.

Texpress items of type Integer map directly to the Texql integer atomic type. Arithmetic expressions which return integers are described in Chapter 4.

**float**

This consists of an optional leading plus or minus sign followed by zero or more digits, then a decimal point and zero or more digits, e.g.:

-128.25

The range of valid float values is machine dependent.

Texpress items of type Real map directly to the Texql float atomic type. Arithmetic expressions which return floating point numbers are described in Chapter 4.

**boolean**

Boolean constants are the keywords **true** or **false**.

Expressions which return boolean values are described in Chapter 4.

Other Texpress data types, such as dates time, latitude and longitude, are defined as tuples of the appropriate basic data types.

*KE Texpress Texql Guide*

Data Types and Attributes 3-5

**Null Values**

The Texql keyword, **null**, returns the null value. It is given a type that will match any other atomic type, with the exception of tuple and nested table attributes which cannot have null values. Therefore null can be used with text, integer, float and boolean columns.

Chapter 4 discusses how null values are handled by various operators.

**Tuples**

A tuple is a row or part of a row within a table or nested table. A tuple can be a subset of the attributes in another tuple. For example, Texpress multi-field Key and library items are represented as tuples. These tuples are composed of tuples and atomic attributes.

A constant of type tuple is a list of constants separated by commas and surrounded by ( and ). For example, the following tuple would be suitable for the loantypes database:

(9, 9.90, 'Bank Transfer', 'john', (18, 6, 1993), (12, 55))

Individual attributes can be extracted from a tuple by use of the **.** (dot) extract operator, and so are referenced as:

*tuple\_identifier*.attribute\_name.

If the above tuple identifier is loantype, then a reference to *loantype*.loanname would be replaced by the text 'Bank Transfer'. Note that where there can be no ambiguity, the tuple\_identifier and extract operator can be omitted and so the above attribute could be referenced simply as:

loanname

For any reference to a tuple attribute, it is only necessary to provide sufficient identifiers to avoid ambiguity. So, for example, the query:

**select** *loans*.*typeno*.*modon*.modon\_2

**from** *loans*;

could be written more simply as:

**select** *typeno*.modon\_2

**from** *loans*;

or even:

*loans*[*typeno*.modon\_2];

*KE Texpress Texql Guide*

3-6 Data Types and Attributes

It is necessary to provide typeno in the above example as the *loans* table contains another modon\_2 attribute through its reference to the *contacts* table.

The rules for resolving implicit extraction are described in Chapter 4.

Parentheses, ( and ), are used for the projection of attributes from a tuple while still maintaining the tuple structure. For example, the following statement creates a tuple structure for the year and month attributes of the modon nested tuple:

**select** loanname, *modon*(modon\_3, modon\_2)

**from** *loantypes*;

Parentheses are also used to construct tuples from other attributes. This is described in more detail in Chapter 4.

Tuple projection always returns a tuple structure whereas tuple extraction returns atomic values.

*KE Texpress Texql Guide*

Data Types and Attributes 3-7

**Tables and Nested Tables**

A **table** is a top-level object which equates to a Texpress database. All tables are global and can be accessed by all users who have the appropriate privileges. A **nested table** is an attribute of a table or a structured attribute which could be another nested table.

The primary distinction between tables and nested tables is that a nested table is an ordered list of tuples whereas a table is an unordered collection of tuples.

The *contacts* table contains a nested table for mailing lists (called *maillist\_tab*). An attribute of a nested table, such as maillist can be referenced as the nested select:

(

**select** maillist

**from** *maillist\_tab*

)

or:

maillist\_tab[ maillist]

which projects the maillist attribute from *maillist\_tab* and returns a table of values.

A constant of type table or nested table is a list of rows separated by **|** and surrounded by **[** and **]**. For example, a table of data suitable for loading into the *loantypes* table could be expressed as:

[

9, 9.0, 'Bank Transfer','john', (17, 6, 1993), (12, 55) | 10, 15, 'Investment', ' ian', (18, 6, 1993), (12, 55)

]

It is not possible for Texql to derive the structure of a table when it includes an empty nested table, unless this structure can be derived from another tuple in the table.

**Reference Attributes**

As described previously, Texql supports reference attributes. A reference attribute is a tuple comprising the key of the table referenced. It is defined in Texpress by use of a linked Key item.

Reference attributes provide support for implicit joins which are described in Chapter 4.

*KE Texpress Texql Guide*

3-8 Data Types and Attributes

**Type Compatibility**

Two expressions are type compatible if the types of the expressions are one of the following:

• The same type (for structured types this implies that all component attributes are also type compatible); or

• One is an integer and the other a floating point number, in which case the integer is converted to a floating point number; or

• One is a tuple with one attribute and the other is type compatible with that attribute.

**Identifiers**

All outer level tables and atomic, structured and reference attributes are accessed by use of the **identifier name** associated with them. An identifier is any sequence of letters, digits and underscores (\_). The first character of the identifier name must be a letter.

Identifiers are case sensitive. Thus Loantypes, LOANTYPES and loantypes are the different identifiers.

A reserved word can be used as an identifier by enclosing it in double quotes, e.g. "from".

Atomic constants have an attribute identifier equal to the type of the constant. All other constants have empty attribute identifiers and so cannot be specified on a select line. Many queries also return tables with empty identifiers.

A query consisting of a single identifier which is a table name retrieves all of the records in that table.

**Statement:**

# Display all data from all records in

# the loantypes table.

#

*loantypes*;

**Output:**

(1,9.50,'First home buyer','john',(15,06,1993),(11,50))

(2,12.90,'Investment property','john',(15,06,1993),(11,50)) (3,15.50,'Personal loan','john',(15,06,1993),(11,50))

(4,14.25,'Car','john',(15,06,1993),(11,50))

(5,10.75,'Home improvement','john',(15,06,1993),(11,51))

(6,16.50,'General loan','john',(15,06,1993),(11,51))

(7,18.00,'Overdraft','john',(15,06,1993),(11,51))

*KE Texpress Texql Guide*

Data Types and Attributes 3-9

(8,17.00,'Travel','john',(15,06,1993),(14,19))

Within a table the attributes can also be referenced using the keyword, **column**, followed by the number of a column within that table. Column number referencing also applies to nested tables.

**Statement:**

# Display columns 2 and 3 from all rows

# in the loantypes table.

#

**select column** 2, **column** 3

**from** *loantypes***;**

**Output:**

(9.50,'First home buyer')

(12.90,'Investment property')

(15.50,'Personal loan')

(14.25,'Car')

(10.75,'Home improvement')

(16.50,'General loan')

(18.00,'Overdraft')

(17.00,'Travel')

**Statement:**

# Display information via a combination

# of column numbers and names, from all

# rows in the loans table.

#

**select** amount, **column** 2.surname

**from** loans **;**

**Output:**

(65000.00,'Citizen')

(40000.00,'Citizen')

(5000.00,'Johnson')

(10000.00,'Rustings')

In general it is advisable to avoid the use of column number referencing as it is too heavily dependent on the table definition. However, occasionally it can be more convenient than using **as** to rename otherwise empty attribute identifiers (see Section 3.8).

*KE Texpress Texql Guide*

3-10 Data Types and Attributes

**Renaming Identifiers using As**

Sometimes it is desirable to define an **alias** for a table, tuple or attribute identifier. Such an action is essential to clarify otherwise ambiguous references to variables, e.g. joining a table to itself, or to assign an identifier to an attribute or table for which no identifier exists, such as a derived table.

Aliases can be assigned to attribute and table identifiers by using the **as** keyword.

**Statement:**

# Select title, firstnam and surname columns

# from contacts renaming each using as.

#

**select** title **as** t, firstnam **as** f, surname **as** s

**from** *contacts* **as** *c*;

**Output:**

('Mr','John','Citizen')

('Ms','Jennifer','Johnson')

('Mr','Peter','Rustings')

Functional notation can also be applied when using aliases (refer to Chapter 4 for more information on functional notation).

**Statement:**

# Select title, firstnam and surname columns

# from contacts renaming each using as

# and using functional notation.

#

*contacts*[title, firstnam, surname] **as** *c*[t, f, s];

**Output:**

('Mr','John','Citizen')

('Ms','Jennifer','Johnson')

('Mr','Peter','Rustings')

**Statement:**

# Create a temporary table of names and

# select from it using as.

#

**select** name

**from** [ 'Albert Jones' |

'Bob Brown' |

'Craig Thomas' |

'David Jeans' |

'Eric Davis'

] as *names*[name]

**where** name **=** 'David Jeans' **;**

**Output:**

('David Jeans')

*KE Texpress Texql Guide*

Data Types and Attributes 3-11

It is also possible to use as to convert an attribute into a tuple with one attribute. **Statement:**

# Convert an atomic value into a tuple using as.

#

'john' **as** *names*(name);

**Output:**

('john')

*KE Texpress Texql Guide*

Query Language 4-1

**Chapter 4**

**Query Language**

Select-From-Where Expressions.............................................. 4-4 Select......................................................................... 4-5

All Columns.................................................. 4-5

Excluding Columns....................................... 4-6

Structured Attributes.................................... 4-6

Reference Attributes..................................... 4-8

Nested Selection........................................... 4-8

From.......................................................................... 4-9

Where...................................................................... 4-10

Aliasing a Sub-expression using With..................................... 4-11 Scoping Rules........................................................................ 4-12 Extracting Attributes from Tuples ............................ 4-12

Scoping Rules for SFW Expressions......................... 4-13

Functional Notation............................................................... 4-14 Boolean Expressions.............................................................. 4-14 And.......................................................................... 4-14

Or............................................................................ 4-15

Not .......................................................................... 4-15

Relational Operators................................................. 4-16

Comparisons with Null........................................................... 4-17 Null Values............................................................................ 4-17 Relational Operators for Numeric Operands........................... 4-18 Between................................................................................. 4-18 Text Operators....................................................................... 4-19 Equality.................................................................... 4-19

Relational Operators................................................. 4-20

Like ......................................................................... 4-20

Contains................................................................... 4-21

Table and Tuple Operators..................................................... 4-24 Equality.................................................................... 4-24

Tuple Existence........................................................ 4-24

Subset and Superset................................................. 4-25

Comparing Atomic Expressions with Tables........................... 4-26 In............................................................................. 4-26

Has .......................................................................... 4-27

*KE Texpress Texql Guide*

4-2 Query Language

Arithmetic Expressions.......................................................... 4-28 Unary Operators....................................................... 4-28 Binary Operators...................................................... 4-29 Aggregate Functions................................................ 4-30 Min............................................................. 4-30

Max............................................................ 4-31

Sum............................................................ 4-31

Avg............................................................ 4-32

Counting Tuples....................................................... 4-32 Forming Tuples from Tables.................................................. 4-33 Tuple Numbers...................................................................... 4-33 Tuple Access......................................................................... 4-34 Table Expressions.................................................................. 4-35

Times....................................................................... 4-35 Join.......................................................................... 4-35 Union....................................................................... 4-36 Intersect................................................................... 4-36 Except...................................................................... 4-37 Nesting Data into Tables.......................................... 4-37 Unnesting Tables...................................................... 4-38 Removing Duplicate Tuples...................................... 4-39 Sorting Tuples.......................................................... 4-40

Miscellaneous Text Functions................................................ 4-41 Stem ........................................................................ 4-41 Phonetic................................................................... 4-42 Numwords............................................................... 4-42 Word........................................................................ 4-43 Words...................................................................... 4-43

*KE Texpress Texql Guide*

Query Language 4-3

**Overview**

This chapter describes the query language component of Texql. Queries take the form of select-from-where. The following sections describe each of these components. Then follows a description of the scoping rules used in Texql.

Later in the chapter descriptions of Texql operators and built-in functions are provided.

*KE Texpress Texql Guide*

4-4 Query Language

**Select-From-Where Expressions**

A Select-From-Where (SFW) expression specifies a selection constraint on a sub-expression restricting which tuples may be retrieved (the where component) and a list of attributes, or expressions based on attributes, to for each tuple retrieved (the select component or projection).

A SFW expression is used to retrieve data stored in Texpress databases as well as to derive new data from this stored data.

The most common form of SFW expressions uses the words, **select**, **from** and **where**. An alternative form uses functional notation where the table referenced in the **from** component is specified first. This is followed by a list of projected attributes or expressions, in square brackets, equating to the **select** component.

A simple example is to project the loanname attribute from the loantypes table, selecting only the tuples (Texpress records) which contain an interest rate less than 10%.

**Statement:**

**select** loanname

**from** *loantypes*

**where** interest **<** 10;

**Output:**

('First home buyer')

The functional notation for this query could be expressed as:

*loantypes*[loanname]

**where** interest **<** 10**;**

Each of the **select**, **from** and **where** components are described in more detail in the following sections.

*KE Texpress Texql Guide*

Query Language 4-5

**Select**

The select component of an SFW expression provides a projection from the referenced tables. It creates a table structure from these projections. However, functions (such as the **min** function, described in Section 4.10), can be applied to SFW expressions to produce atomic values.

The select component is a list of attributes or sub-expressions forming new attributes from the table sub-expression on the from line. The order of attributes in the select line defines the attribute order in the resulting table.

The projected list of expressions can include atomic values, tuples and nested tables.

An example SFW expression to project the loan name and date/time of last modification to the record could be entered as follows:

**Statement:**

**select** loanname **,** *modon***,** *modat*

**from** *loantypes*

**where** interest **<** 10**;**

**Output:**

('First home buyer',(15,06,1993),(11,50))

**All Columns**

A projection of the keyword, **all** (or **\***), selects all attributes of the tables listed in the from component. For example, to select all columns from the loantypes table, the following SFW expression can be used:

**Statement:**

**select all**

**from** *loantypes***;**

**Output:**

(1,9.50,'First home buyer','john',(15,06,1993),(11,50))

(2,12.90,'Investment property','john',(15,06,1993),(11,50)) (3,15.50,'Personal loan','john',(15,06,1993),(11,50))

(4,14.25,'Car','john',(15,06,1993),(11,50))

(5,10.75,'Home improvement','john',(15,06,1993),(11,51))

(6,16.50,'General loan','john',(15,06,1993),(11,51))

(7,18.00,'Overdraft','john',(15,06,1993),(11,51))

(8,17.00,'Travel','john',(15,06,1993),(14,19))

The above SFW expression could be replaced by the following functional notation command:

loantypes **;**

*KE Texpress Texql Guide*

4-6 Query Language

**Excluding Columns**

Texql allows designated attributes to be excluded from a selection by the addition of a **but** component, as in the following example:

**Statement:**

**select all but** loanno **,** *modby*

**from** *loantypes***;**

**Output:**

(9.50,'First home buyer',(15,06,1993),(11,50))

(12.90,'Investment property',(15,06,1993),(11,50))

(15.50,'Personal loan',(15,06,1993),(11,50))

(14.25,'Car',(15,06,1993),(11,50))

(10.75,'Home improvement',(15,06,1993),(11,51))

(16.50,'General loan',(15,06,1993),(11,51))

(18.00,'Overdraft',(15,06,1993),(11,51))

(17.00,'Travel',(15,06,1993),(14,19))

which is equivalent to the SFW expression:

**select** interest **,** loanname **,** *modon***,** *modat*

**from** *loantypes***;**

The excluded attributes in a but component can be a tuple or nested table or attributes from a tuple or nested table.

**Structured Attributes**

A projection of a tuple (or nested table) can be achieved by simply referencing that tuple's (or table's) identifier. However, a subset of the attributes of a tuple or nested table can also be selected.

A nested select component can be used to select a subset of the attributes of a nested table, as in the following example:

**Statement:**

**select** surname **, (**

**select** maillist

**from** *maillist\_tab*

**)**

**from** *contacts***;**

**Output:**

('Citizen',['Home improvement'|'Boating'])

('Johnson',['Home buyer'|'Travel'])

('Rustings',['Better finance'])

Alternatively, a subset of the attributes of a nested tuple or a table can be projected using the following functional notation:

*KE Texpress Texql Guide*

Query Language 4-7

**Statement:**

**select** surname **,** *maillist\_tab***[**maillist **]**

**from** *contacts***;**

**Output:**

('Citizen',['Home improvement'|'Boating'])

('Johnson',['Home buyer'|'Travel'])

('Rustings',['Better finance'])

New tuple attributes can be constructed in the select component by use of round brackets, **(** and **)**. For example, to create a tuple of the columns of a name from the contacts table, the following SFW expression could be used:

**Statement:**

**select (**title**,** firstnam **,** surname **) as** *name***,** company **from** *contacts***;**

**Output:**

(('Mr','John','Citizen'),'Acme Electronics Pty. Ltd.') (('Ms','Jennifer','Johnson'),'Channel Ten')

(('Mr','Peter','Rustings'),' Rustings Pty. Ltd.')

Nested tuple attributes can be collapsed into a collection of columns of atomic values. For example, the following query collapses the tuple structure for the column *modon*.

**Statement:**

**select** surname **,** *modon***.all**

**from** *contacts***;**

**Output:**

('Citizen',15,06,1993)

('Johnson',15,06,1993)

('Rustings',15,06,1993)

The sequence **.\*** could have been used in place of **.all** in this example. This query is equivalent to the following query:

**select** surname **,** modon\_1 **,** modon\_2 **,** modon\_3

**from** *contacts***;**

If the identifier used was a reference attribute pointing to another table, then the **.all** would have caused the reference to be followed and so would have been replaced by all of the columns of the referenced table. Unnesting nested table attributes requires the special functional operators, **inner unnest** and **outer unnest**, which are described later in this Chapter.

Whenever a table name appears in both the select component and the from component then it refers to each tuple of the table, thus the above query could also be written as:

*KE Texpress Texql Guide*

4-8 Query Language

**select** surname **,** *contacts***.***modon***.all**

**from** *contacts***;**

However the following query returns a tuple containing a nested tuple equivalent in structure to the contacts table query:

**select** *contacts*

**from** *contacts***;**

**Reference Attributes**

Reference attributes represent implicit joins between tuples in the table containing the reference and tuples in the referenced table. Each of the attributes in the referenced tuple is directly available, through this implicit join, when querying on the tuple containing the reference.

In the example tables, the *loans* table contains a reference to the *contacts* table and to the *loantypes* table. The following query results in two implicit joins being performed between the *loans* table and the *contacts* and *loantypes* tables following the defined references:

**Statement:**

**select** surname **,** loanname

**from** *loans***;**

**Output:**

('Citizen','First home buyer')

('Citizen','General loan')

('Johnson','Travel')

('Rustings','Overdraft')

As there is no name clash for surname or loanname in the above example, the reference attribute identifier does not have to precede the referenced table attribute identifier.

**Nested Selection**

The recursively defined syntax of Texql allows an expression that returns a table to be used within a query at any place that a table can be used. This property of Texql is referred to as orthogonality and, in many cases, simplifies query formulation.

The recursive syntax allows a table to be joined at any level within a hierarchical structure. For example, the following query joins the nested table of mailing list entries with the outer table of loans.

**Statement:**

**select** surname, *maillist\_tab*[maillist]

**from** *loans*;

*KE Texpress Texql Guide*

Query Language 4-9

**Output:**

('Citizen',['Home improvement'|'Boating'])

('Citizen',['Home improvement'|'Boating'])

('Johnson',['Home buyer'|'Travel'])

('Rustings',['Better finance'])

When implicit joins refer to the same attributes from the same table it may necessary to rename attributes using **as**.

**From**

The **from** component within a SFW expression specifies the table (or tables) from which tuples are to be selected. Theoretically, when more than one table is given in the from component, tuples are selected from the cartesian product of tuples from each table. However, the Texql optimiser may use information from the where component to provide enhanced performance.

Specifying multiple table names in the from component is the means by which a join query can be performed with the where component providing the actual join condition.

An explicit join could be used in the absence of a reference attribute. For example, the following query could be used to extract the surname and loanname of tuples in the loans table:

**Statement:**

**select** *contacts*.surname, *loantypes*.loanname

**from** *contacts*, *loantypes*, *loans*

**where** *loans*.*contno*.contno = *contacts*.contno

**and** *loans*.*typeno*.loanno = *loantypes*.loanno **;**

**Output:**

('Citizen','First home buyer')

('Citizen','General loan')

('Johnson','Travel')

('Rustings','Overdraft')

It is also possible to use a query expression in the from component of a Texql query. This is most useful when a query expression is to be applied to a derived table structure and not to a base or outer table. For example:

**Statement:**

**select** surname **,** amount **,** annual

from

(

**select** surname, amount,

amount \* interest / 100 **as** annual

**from** *loans*

)

**where** annual > 5000 **;**

*KE Texpress Texql Guide*

4-10 Query Language

**Output:**

('Citizen',65000.00,6175.000000)

('Citizen',40000.00,6600.000000)

**Where**

The **where** component of a query determines the tuples from the tables in the corresponding from component which are used in computing the result of the SFW expression. The where component is optional.

The condition of the where component within a SFW expression must be a boolean expression, returning a result of either true or false. If the boolean expression evaluates to true for a particular tuple from the table(s) in the from component then it is returned in the result of the SFW-expression, projected through the select component.

**Statement:**

**select** surname, firstnam, amount

**from** *loans*

**where** amount > 10000;

**Output:**

('Citizen','John',65000.00)

('Citizen','John',40000.00)

A where component can accompany any nested select-from expression. **Statement:**

# Select loans with amounts which are above average.

# Consider only loans of amounts >= 10000.

#

**select** *out*.surname, *out*.firstnam, *out*.amount

**from** *loans* **as** *out*

**where** *out*.amount > **avg**

(

**select** *tmp*.amount

**from** *loans* **as** *tmp*

**where** *tmp*.amount >= 10000

)**;**

**Output:**

('Citizen','John',65000.00)

('Citizen','John',40000.00)

*KE Texpress Texql Guide*

Query Language 4-11

**Aliasing a Sub-expression using With**

SFW queries can use aliases to simplify their formulation and avoid duplication of some of their clauses. This can be achieved using the keyword, **with**.

A with clause follows a where clause, if any, and assigns a clause to an identifier. All occurrences of the identifier throughout the SFW query are replaced by the clause assigned to that identifier.

A query to select tuples from the contacts database which include mailing list entries containing the word 'home' could be expressed as follows:

**Statement:**

# Select tuples with a mailing list entry

# containing the word 'home'.

#

**select** surname

**from** *contacts*

**where exists**

(

*maillist\_tab*

**where** maillist **contains** 'home'

)**;**

**Output:**

('Citizen')

('Johnson')

However, if it were necessary to project the appropriate mailing list entries as well, then the following statement would be required:

**Statement:**

# Select tuples with a mailing list entry containing

# the word 'home'. Then project the surname and the

# relevant mailing list entries only.

#

**select** surname,

(

*maillist\_tab*

**where** maillist **contains** 'home'

)

**from** *contacts*

**where exists**

(

*maillist\_tab*

**where** maillist **contains** 'home'

)**;**

**Output:**

('Citizen',['Home improvement'])

('Johnson',['Home buyer'])

*KE Texpress Texql Guide*

4-12 Query Language

This could be expressed more simply using a **with** clause, as follows: **Statement:**

# Select tuples with a mailing list entry containing

# the word 'home'. Then project the surname and the

# relevant mailing list entries only. Use a with clause

# to simplify the query.

#

**select** surname, homelist

**from** *contacts*

**where exists**(homelist)

**with** homelist :=

(

*maillist\_tab*

**where** maillist **contains** 'home'

)**;**

**Scoping Rules**

Resolution of identifiers is determined by the scoping rules of Texql. These rules specify how an identifier in an expression is bound to an attribute or table. There are two primary scoping rules in Texql:

1. Given a tuple and an attribute to extract from the tuple, Texql attempts to locate the attribute in the tuple.

2. Given an attribute identifier in an SFW expression, Texql determines which tables are searched (and in which order) to locate the attribute.

When locating an attribute, the former rule (1) is repeatedly applied for each table searched by the latter rule (2) until the attribute is located.

**Extracting Attributes from Tuples**

If the extract operator is used, (i.e. an identifier reference of the form, *tuple*.attribute), then the following steps are pursued until the attribute identifier is found:

1. Check the attributes of the tuple structure.

2. Recursively check the attributes in any nested tuple structures.

3. Recursively follow any reference attributes of the tuple.

Texql reports an error if any of the above steps results in an ambiguity, i.e. the identifier appears twice in a tuple structure or appears in two different nested tuple structures or appears following two different reference attributes.

*KE Texpress Texql Guide*

Query Language 4-13

Tuple extraction does not consider columns in nested tables. If this is necessary, then the unnest operators must be used (see Section 4.16.6).

**Scoping Rules for SFW Expressions**

A reference to a nested table (or tuple or attribute) refers to the table (or tuple or attribute) within the current tuple. Consider the following example query which contains selections from nested tables:

**select** amount,

(

**select** category

**from** *category\_tab*

**where** category **contains** 'home'

**or** category **contains** 'car'

),

(

**select** maillist

**from** *contno*.*maillist\_tab*

**where** maillist **contains** 'home'

)

**from** loans**;**

The outer select is applied to every tuple in the loans table. The inner selects are applied to each tuple of the appropriate nested table within the current loans tuple.

The following rules are used to resolve references to table and attribute identifiers. The rules are followed, in order, until the identifier is resolved or until the completion of the rules at which point Texql gives an appropriate error message.

1. If an identifier appears on the right of an extract operator then it must be extracted from the tuple on the left.

2. If an identifier in a select or where component appears as an extractable attribute of a tuple in one of the tables in the from component then Texql attempts to locate the identifier in that table.

3. If an identifier in a select or where component appears in a from component then the attribute identifier refers to that table.

4. Steps 2 and 3 above are repeated, checking recursively the from components from innermost to outermost SFW expression

5. If the identifier is still unresolved but it is an outer level table identifier then it refers to that table.

*KE Texpress Texql Guide*

4-14 Query Language

**Functional Notation**

As described previously, Texql also supports a functional notation for the representation of SFW expressions. This notation replaces the select and from components with the table name followed by a projection list in square brackets. For example, the following SFW expression:

**select** attrs1 **but** attrs2

**from** *table\_name***;**

could be expressed as:

*table\_name*[attrs1 **but** attrs2] **;**

This notation also applies to projections from nested tables and derived tables.

**Boolean Expressions**

The boolean operators, **and**, **or**, and **not**, perform the appropriate logical boolean operations on the given boolean sub-expressions. Boolean sub expressions may return null as a valid value. So the boolean operators must use a three-valued logic where null represents an unknown value.

The result of each boolean operator on all operand combinations is defined in the following sections. An example operator use is also provided.

**And**

The **and** evaluation table is:

**and** true null false

true true null false

null null null false

false false false false

**Statement:**

# Find the surnames and loan amounts of loans which

# exceed $20,000 and have a term of < 100 months.

#

*loans*[surname, amount]

**where** amount > 20000

**and** term < 100;

*KE Texpress Texql Guide*

Query Language 4-15

**Output:**

('Citizen',40000.00)

**Or**

The **or** evaluation table is:

**or** true null false

true true true true

null true null null

false true null false

**Statement:**

# Find the title, first name and surname of each # contact in Melbourne, Nunawading or Blackburn. #

*contacts*[title, firstnam, surname]

**where** town = ' melbourne'

**or** town = ' nunawading'

**or** town = ' blackburn';

**Output:**

('Ms','Jennifer','Johnson')

('Mr','Peter','Rustings')

**Not**

The **not** evaluation table is:

**not**

true false

null null

false true

**Statement:**

# Find details of loans to customers other

# than Mr Citizen.

#

*loans*

**where not** surname = 'citizen' **;**

*KE Texpress Texql Guide*

4-16 Query Language

**Output:**

(3,(2),(8),5000.00,12,['Overseas Travel'])

(4,(3),(7),10000.00,36,['Overdraft'])

**Relational Operators**

The relational operators are as follows:

**=** Equals.

**<>** Does not equal.

**<** Less than.

**<=** Less than or equal to.

**>** Greater than.

**>=** Greater than or equal to.

These relational operators can be applied to boolean operands resulting in a boolean comparison being performed. If either operand is null, the expression returns null. Otherwise, for the purposes of boolean comparisons, true is considered greater than false.

*KE Texpress Texql Guide*

Query Language 4-17

**Comparisons with Null**

The operator, **is null**, returns true if the atomic sub-expression returns a null value and false otherwise. The operator, **is not null**, returns true if the atomic sub-expression does not return a null value and false otherwise.

These operators are not equivalent to = null and <> null, respectively. **Statement:**

# Identify contact tuples with unknown surnames.

#

**select** contno, company

**from** *contacts*

**where** surname **is null**;

**Output:**

*(no output)*

**Null Values**

A **null value** in a tuple represents an unknown value. Generally, processing any list of values including a null results in a null value being returned (e.g. the **sum** function). In order to prevent such a result, it is generally necessary to explicitly prevent the selection of null values using a where clause including **is not null**.

To simplify processing of null values, Texql provides the function, **ifnull**, which returns its first argument if this is not null. If the first argument is null, ifnull returns its second argument.

**Statement:**

# Example use of the ifnull function.

#

**sum**

(

**select ifnull**(amount, 0)

**from** *loans*

)**;**

**Output:**

120000.000000

*KE Texpress Texql Guide*

4-18 Query Language

**Relational Operators for Numeric Operands**

The relational operators available for numeric operands are the same as those for boolean operands, described in Section 4.5.1.

If both operands of a relational operator are of the same type (i.e. either integer or floating point) then no conversion is necessary and the appropriate comparison is performed.

If one operand is integer and the other float then the integer operand is converted to float and the comparison is performed.

The null value is treated as an unknown value and so it is not considered equal to any other value including null. Hence, the comparison:

**null = null**

returns the value:

**null**

**Between**

A closed range of values can be requested using two relational operators combined using the boolean and operator. For example,

(val **>=** a) **and** (val **<=** b)

This can also be expressed using the **between** keyword as follows:

val **between** a **and** b

The between operator can only be applied to atomic values and all three expressions (val, a and b) must be type compatible.

**Statement:**

# Find loans with amounts between 10000 and 50000

#

*loans*

**where** amount **between** 10000 **and** 50000 **;**

**Output:**

(2,(1),(6),40000.00,60,['Extension to family home'|'Car

purchase'|'Overseas Travel'])

(4,(3),(7),10000.00,36,['Overdraft'])

*KE Texpress Texql Guide*

Query Language 4-19

**Text Operators**

Most of the operators previously described can be applied to text operands. However, there are several other operators exclusively provided for text operations.

**Equality**

The standard equality operators, = (equals) and <> (does not equal) can be applied to text operands. The two operands are otherwise compared literally. Implicitly character case is insignificant.

**Statement:**

# Find all contacts who are design engineers.

#

*contacts*[firstnam, surname]

**where** position = 'design engineer' **;**

**Output:**

('John','Citizen')

**Statement:**

# Find all loan types except personal

# and general loans.

#

*loantypes*

**where** loanname <> 'general loan'

**and** loanname <> 'personal loan' **;**

**Output:**

(1,9.50,'First home buyer','john',(15,06,1993),(11,50))

(2,12.90,'Investment property','john',(15,06,1993),(11,50)) (4,14.25,'Car','john',(15,06,1993),(11,50))

(5,10.75,'Home improvement','john',(15,06,1993),(11,51))

(7,18.00,'Overdraft','john',(15,06,1993),(11,51))

(8,17.00,'Travel','john',(15,06,1993),(14,19))

An explicit case sensitive or case insensitive text comparison can be performed by preceding the text operand with the transformation operator, **=**, or, **&**, respectively.

**Statement:**

# Find all contacts who are design engineers.

# Character case sensitive.

#

*contacts*[firstnam, surname]

**where** position = ='design engineer' **;**

**Output:**

*(no output)*

*KE Texpress Texql Guide*

4-20 Query Language

**Statement:**

# Find all contacts who are design engineers.

# Character case insensitive.

#

*contacts*[firstnam, surname]

**where** position = &'design engineer' **;**

**Output:**

('John','Citizen')

**Relational Operators**

The relational operators, **>**, **>=**, **<** and **<=** and **between**, can be applied to text operands. A comparison is performed based on the ASCII value of the characters in the operands. If either operand is null , the expression returns null.

**Statement:**

# Find contacts with surnames beginning

# with the letter A through M.

#

*contacts*[title, firstnam, surname]

**where** surname **between** 'A' **and** 'M'**;**

**Output:**

('Mr','John','Citizen')

('Ms','Jennifer','Johnson')

**Like**

As with SQL, Texql supports the specification of substrings in queries on text operands using the **like** operator. The like operator compares a text string operand using the pattern matching rules of Texpress.

If either operand of the like operator is a null , the expression returns null. The like operator accepts the following pattern matching characters:

**\*** Matches zero or more characters at that position in the operand.

**?** Matches any single character at that position in the operand.

**[*str*]** Matches a single character if that character is one of those in *str*. The *str* string may contain a sequence such as a-z which

indicates the ASCII range of characters from a through to z.

**[^*str*]** Matches a single character if that character is not one of those in *str*. Character ranges can be included as before.

*KE Texpress Texql Guide*

Query Language 4-21

**{*str*}** Matches zero or more characters which are contained in *str*. The *str* string is as defined above, supporting character ranges.

**{^*str*}** Matches zero or more characters which are not contained in *str*. The *str* string is as defined above, supporting character

ranges.

Although the anchor characters,, **^** for start of string and **$** for end of string, are supported, they are redundant as the whole text must be matched by the pattern, not just a single word in the text.

**Statement:**

# Find contacts with first names beginning with

# a 'J', followed by any number of 'o', 'h' or 'e',

# then followed by 'n' and then any other characters.

#

**select** title, firstnam, surname

**from** *contacts*

**where** firstnam **like** 'j{ohe}n\*';

**Output:**

('Mr','John','Citizen')

('Ms','Jennifer','Johnson')

**Contains**

Texql supports the searching of text columns using the **contains** operator. This operator enables the selection of tuples based on words, stems of words, sounds of words, phrases or parts of words within the text attribute.

Within Texpress, a word is defined to be any sequence of letters, digits and underscore characters. The space character and most punctuation characters are word-break characters for most European languages, while for multi-byte languages, each ideographic character is considered a separate word.

The contains operator returns true if a text attribute (left hand side) contains the series of terms on the right hand side. The contents of the right hand side can be anything acceptable in a single text field of a Texpress Query form. The rules for matching are as defined by Texpress (refer to the Texpress Reference Manuals).

If multiple terms are specified in the text literal on the right hand side of the expression then all terms must match for the column to match (boolean 'and' relationship). Each term in the text literal can be one of the following:

***term*** The word, *term*, must appear in the text. Implicitly case is insensitive.

**=*term*** The word, *term*, must appear in the text. Case is sensitive.

*KE Texpress Texql Guide*

4-22 Query Language

**&*term*** The single word, *term*, must appear in the text. Case is insensitive.

**~*term*** A word which is based on the same word stem as *term* must appear in the text.

**@*term*** A word which sounds like *term* must appear in the text.

***pattern*** A word which matches the *pattern* (using the pattern matching rules described for the like operator) must appear

in the text.

**!*term*** The single word, *term*, must not appear in the text. Case is insensitive.

**"*term1 term2* ..."** The text must contain a phrase consisting of *term1* followed directly by *term2*, etc. Note that noise words are ignored

within phrases. Each of the terms may take the form of any

of the previous forms (exact terms, stems, sounds or

patterns).

One of the transformation operators, =, &, @ or ~, can

immediately precede a phrase, in which case it is applied to

all terms in the phrase.

Also, the not operator, !, can immediately precede a phrase,

in which case the search is for 'not' that phrase.

Any one of the transformation operators, =, &, ~ or @ may precede the literal text in which case the operator is applied to all terms in the text. A transformation explicitly specified within the text overrides any transformation operator specified outside the text.

Following are several examples of the use of the contains operator. **Statement:**

# Find contacts where the remarks column contains

# the word, 'tattslotto'.

#

*contacts*[firstnam, surname, position]

**where** remarks **contains** 'tattslotto' **;**

**Output:**

('Peter','Rustings','Director')

**Statement:**

# Find contacts where the position column contains # a word with the stem of 'market'.

*KE Texpress Texql Guide*

Query Language 4-23

#

*contacts*[firstnam, surname, position]

**where** position **contains** '~market' **;**

**Output:**

('Jennifer','Johnson','Marketing Officer')

**Statement:**

# Find contacts where the surname column contains # a name sounding like 'jansen'.

#

*contacts*[firstnam, surname, position]

**where** surname **contains** '@jansen';

**Output:**

('Jennifer','Johnson','Marketing Officer')

**Statement:**

# Find contacts where the remarks column contains # a series of terms.

#

*contacts*

**where** remarks

**contains** 'excellent "good loan ~prospects" @jenafer'; **Output:**

(2,'Ms','Jennifer','Johnson','Marketing Officer','Channel Ten',

'34 James Court','VIC','NUNAWADING','3131','859 2717 844 7891', 'UNK',80000,['Home buyer'|'Travel'], 'john', (15,06,1993), (14,11),

'Jennifer is a good loan prospect but with no previous credit history (apart from a good bankcard record). She has an excellent position, albeit with a difficult job ahead of her, and so we can expect her to be able to service loans. However, current thinking prefers to minimise our exposure until there is more track record.')

*KE Texpress Texql Guide*

4-24 Query Language

**Table and Tuple Operators**

Many of the basic Texql operators can be applied to variables of type table, nested table or tuple. As described previously, a table is an unordered collection of tuples, while a nested table is an ordered list of tuples. Each can, of course, contain nested tuples.

**Equality**

The equality and inequality operators (= and <>) may be used for tables and tuples.

Two tuples are considered equal if they are type compatible and the pairs of values for each attribute in the tuples are equal.

Two tables are equal if they are type compatible, each of the tuples in the tables are equal and the tuples are in the same order.

**Statement:**

# Check that all home loans and only home

# loans are under 12% interest rate.

#

(

*loantypes*

**where** loanname **contains** 'home'

) =

(

*loantypes*

**where** interest < 12

)**;**

**Output:**

T

**Tuple Existence**

The **exists** operator returns true if its single operand, a table sub-expression, contains at least one tuple. It returns false otherwise.

**Statement:**

# Display the surnames of loans for which

# there exists a type containing 'car'.

#

#

*loans*[surname]

**where exists**

(

*category\_tab*

**where** category **contains** 'car'

)**;**

*KE Texpress Texql Guide*

Query Language 4-25

**Output:**

('Citizen')

**Subset and Superset**

Two operators exist to determine if a table is a subset or a superset of another table. These operators, **subset of** and **superset of**, take two operands which must be tables identical in structure.

The subset of operator returns true if all of the tuples in the first table exist in the second, while the superset operator returns true if all of the tuples of the second table exist in the first.

The order of tuples in the tables is not important. Duplicate tuples are ignored during the evaluation of these operators.

**Statement:**

# Ensure that all loans are registered to

# people listed in the contacts table.

#

(

*loans*[*contno*.contno]

) **subset of**

(

*contacts*[contno]

);

**Output:**

T

*KE Texpress Texql Guide*

4-26 Query Language

**Comparing Atomic Expressions with Tables**

Two operators exist to compare an atomic or tuple expression with a table. These are the operators, **in** and **has**.

**In**

The **in** operator returns true if the left atomic or tuple sub-expression is contained within the set of values returned by the right table sub-expression. Otherwise false is returned.

**Statement:**

# Find the surname of all contacts for

# which there is a current loan tuple.

#

*contacts*[surname]

**where** contno **in**

(

*loans*[*contno*]

);

**Output:**

('Citizen')

('Johnson')

('Rustings')

*KE Texpress Texql Guide*

Query Language 4-27

**Has**

The **has** operator is functionally similar to the **in** operator. It returns true if the set of values returned by the left table sub-expression contains the right atomic or tuple sub-expression. Otherwise false is returned.

The right hand side of a has expression can also begin with one of the following operators (if none is specified then = is assumed):

• **=**

• <>

• >

• >=

• <

• <=

• between

• like

• contains

• subset of

• superset of

**Statement:**

# Find the surname of all contacts for

# which there is a current loan tuple.

#

*contacts*[surname]

**where**

(

*loans*[*contno*]

) **has** contno **;**

**Output:**

('Citizen')

('Johnson')

('Rustings')

*KE Texpress Texql Guide*

4-28 Query Language

**Arithmetic Expressions**

Texql supports a rich suite of arithmetic operators, each of which is described in this section.

**Unary Operators**

The unary plus, **+**, and unary minus, **-**, operators can be applied to any sub expression that returns a numeric value. The unary minus operator changes the sign of its operand. Both operators return null if the operand evaluates to null.

The type of the unary plus and unary minus expression is the same as the type of their operand, with the exception of tuples with a single numeric attribute which are first converted to an atomic numeric value.

**Statement:**

# Prepare outstanding loan amounts

# for inclusion on assets register.

#

**select** -amount

**from** *loans***;**

**Output:**

(-65000.000000)

(-40000.000000)

(-5000.000000)

(-10000.000000)

*KE Texpress Texql Guide*

Query Language 4-29

**Binary Operators**

The binary arithmetic operators take two operands which are numeric, atomic sub-expressions. The operators are as follows:

**+** Addition.

**-** Subtraction.

**\*** Multiplication.

**/** Division.

**%** Modulus.

If either operand returns a value of type float, then a floating point value is returned. Otherwise an integer value is returned. If either sub-expression returns null then null is returned.

**Statement:**

# Recalculate loan amounts

# given a 10% reduction.

#

**select** amount \* 0.9

**from** *loans***;**

**Output:**

(58500.000000)

(36000.000000)

(4500.000000)

(9000.000000)

The **+** operator can also be applied to text operands, in which case it performs a **concatenation** of its operands.

**Statement:**

# Produce a table with a single text

# attribute which is the concatenation

# of the title, first name and surname

# of all contacts.

#

**select** title + ' ' + firstnam + ' ' + surname

**from** *contacts***;**

**Output:**

('Mr John Citizen')

('Ms Jennifer Johnson')

('Mr Peter Rustings')

*KE Texpress Texql Guide*

4-30 Query Language

**Aggregate Functions**

Texql supports the aggregate functions, **min**, **max**, **sum** and **avg**, which are designed to process a vector of values (such as a table with one column only). These functions generally take a SFW expression as their operand and produce a numeric result.

For each of these functions, the existence of a null value in an argument to the function causes the function to return the value null. Also if the table passed to these functions is empty, then the min, max and avg functions return null, while sum returns zero. This action can be overridden if a **default** component is added after the where component, if any. Thus a query such as:

**avg**(*loans*[amount] **where** amount > 100000) **;**

would return null if no loan met the where condition. However, the statement: **avg**(*loans*[amount] where **amount** > 100000 **default** 0)**;**

would return 0 rather than null.

These aggregate functions also create an implicit attribute identifier equal to the name of the function.

**Min**

The **min** function can be applied to any single attribute table. The attribute must be atomic and not a tuple or nested table.

If any of the values in the table are null, null is returned. If the table is empty, then the min function returns the default component, if any, otherwise null. The min function returns an atomic value whose type is the same as the attribute in its table operand.

When min is applied to a table whose single column is of text type, then it returns the value which is the first in lexicographic order (actually ASCII order).

**Statement:**

# Determine the minimum loan amount.

#

**min**(*loans*[amount]) **;**

**Output:**

5000.00

*KE Texpress Texql Guide*

Query Language 4-31

**Max**

The **max** function can be applied to any single attribute table. The attribute must be atomic and not a tuple or nested table.

If any of the values in the table are null, null is returned. If the table is empty, then the max function returns the default component, if any, otherwise null. The max function returns an atomic value whose type is the same as the attribute in its table operand.

When max is applied to a table whose single column is of type text or string, then it returns the value which is the last in lexicographic order (actually ASCII order).

**Statement:**

# Determine the surname which is

# last in lexicographic order.

#

**max**(**contacts**[surname]) **;**

**Output:**

'Rustings'

**Sum**

The **sum** function can be applied to any single attribute table. The attribute must be atomic and not a tuple or nested table. The column of this table must be of a numeric type (integer or float).

If any of the values in the table are null, null is returned. If the table is empty, then the sum function returns the default component, if any, otherwise 0. The sum function returns an atomic value whose type is the same as the attribute in its table operand.

**Statement:**

# Determine the sum of home loans.

#

**sum**

(

*loans*[amount]

**where exists**

(

*category\_tab*

**where** category **contains** 'home'

)

)**;**

**Output:**

105000.000000

*KE Texpress Texql Guide*

4-32 Query Language

**Avg**

The **avg** function can be applied to any single attribute table. The attribute must be atomic and not a tuple or nested table.

The column of this table must be of a numeric type (integer or float). If any of the values in the table are null, null is returned. If the table is empty, then the avg function returns the default component, if any, otherwise null. The atomic value returned by avg is always a floating point number.

**Statement:**

# Create a temporary table of the loans columns plus

# an additional column set to the average loan amount.

# Then project from this table, the amount and the

# average amount.

#

**select** amount, "AVG"

**from**

(

**select all**,

**avg**

(

*loans*[amount]

)

**from** *loans* **as** *tmp*

)**;**

**Output:**

(65000.00,30000.000000)

(40000.00,30000.000000)

(5000.00,30000.000000)

(10000.00,30000.000000)

**Counting Tuples**

The **count** function can be applied to any table of any structure. The table may contain nested tables or null values. The count function always returns an integer value which is the number of tuples in the table. If an empty table is passed to this function, then zero is returned.

**Statement:**

# Determine the number of loans

# exceeding the average loan amount.

#

**count**

(

*loans*[amount] **as** tmp

**where** *tmp*.amount > **avg**(*loans*[amount])

)**;**

**Output:**

2

*KE Texpress Texql Guide*

Query Language 4-33

**Forming Tuples from Tables**

The function, **totuple**, makes the syntactic conversion of a table containing one tuple into the type, tuple. If the table does not contain exactly one tuple, this function returns an error. This function is necessary when passing the result of a SFW expression to a function, such as numwords (see Section 4.17), which operates on atomic values only.

**Statement:**

# Output the amount of loan number 3

# plus an extra $5,000.

#

**totuple**

(

*loans*[amount]

**where** loanno = 3

) + 5000 **;**

**Output:**

10000.000000

**Tuple Numbers**

The variable, **rownum**, returns the tuple number of the current tuple. This is the integer value representing the number of the tuple within the table or nested table. Tuples are numbered commencing from one.

**Statement:**

# Select the rownum and amount of

# tuples in the loans table from

# row number 2 onwards.

#

**select rownum**, amount

**from** *loans*

**where rownum** > 1**;**

**Output:**

(2,40000.00)

(3,5000.00)

(4,10000.00)

The rownum function also returns an attribute identifier of rownum.

*KE Texpress Texql Guide*

4-34 Query Language

**Tuple Access**

A single tuple can be selected from a table by adding {number} after the table reference. This is called **tuple access**. If the specified tuple does not exist in the table, a tuple of null values is returned.

**Statement:**

# Access the fourth tuple of the loantypes table.

#

*loantypes*{4};

**Output:**

(4,14.25,'Car','john',(15,06,1993),(11,50))

**Statement:**

# Access the name and second mailing list

# entry of all contacts.

#

*contacts*[surname, *maillist\_tab*{2}]**;**

**Output:**

('Citizen',['Boating'])

('Johnson',['Travel'])

('Rustings',[])

A sub-range of tuples can be accessed from a table by adding the sequence: {num1 **to** num2}

after the table reference. This forms a table from rows *num1* through to *num2* or the number of rows in the table, whichever is smaller.

**Statement:**

# Access the surname and first two

# category entries for each loan.

#

*loans*[surname, *category\_tab*{1 **to** 2}]**;**

**Output:**

('Citizen',['First home purchase'])

('Citizen',,['Extension to family home'|'Car purchase'])

('Johnson',['Overseas Travel'])

('Rustings',['Overdraft'])

*KE Texpress Texql Guide*

Query Language 4-35

**Table Expressions**

There are several operators available for manipulating tables. Each of these return results which are also tables.

**Times**

The operator, **times**, returns the **cartesian product** of its operands, each of which must be tables. The resultant structure is a table containing two tuple attributes.

**Join**

The operator, **join**, implements a **natural join**. A natural join performs the cartesian product of its table operands, selecting only those tuples where all attributes with the same identifiers in the different tables have the same values, and then removes duplicate attributes.

Unlike the times operator, join does not form tuples in the result.

**Statement:**

# Without using the references, perform a

# natural join of the loans, contacts and

# loantypes tables and then select the

# amount, surname and interest rate from

# the resulting table.

#

**select** amount, surname, interest

**from**

(

(

*loans*[loanno, *contno*.contno,

*typeno*.loanno **as** lno, amount]

)

**join**

(

*contacts*[contno, surname]

)

**join**

(

*loantypes*[loanno **as** lno, interest]

)

)**;**

**Output:**

(65000.00,'Citizen',9.50)

(40000.00,'Citizen',16.50)

(5000.00,'Johnson',17.00)

(10000.00,'Rustings',18.00)

*KE Texpress Texql Guide*

4-36 Query Language

**Union**

The **union** operator returns a table of the union of all tuples in each of its operands i.e. it returns any tuples which appear in either of its operands. Each operand must be of type table. Duplicate tuples are removed. If **union all** is used then duplicate tuples are not removed.

The table operands must be **union compatible**, which means that they must have the same attribute types and the attributes must appear in the same order. This must also be true recursively for any nested attributes.

**Statement:**

# Find all loans of more than $5,000

# together with all travel loans.

#

(

*loans*

**where** amount > 5000

)

**union all**

(

*loans*

**where exists**

(

*category\_tab*

**where** category **contains** 'travel

)

)**;**

**Output:**

(1,(1),(1),65000.00,120,['First home purchase'])

(2,(1),(6),40000.00,60,['Extension to family home'|'Car

purchase'| 'Overseas Travel'])

(4,(3),(7),10000.00,36,['Overdraft'])

(2,(1),(6),40000.00,60,['Extension to family home'|'Car

purchase'| 'Overseas Travel'])

(3,(2),(8),5000.00,12,['Overseas Travel'])

**Intersect**

The **intersect** operator returns the intersection of all of the tuples in its operands, i.e. it only returns tuples that appear in both of the operands. The two operands must be union compatible tables.

If **union all** is specified, then the number of copies of any specific row value in the result is equal to the lesser number of such copies in the table indicated by the left operand and the number in the table indicated by the right operand.

*KE Texpress Texql Guide*