jBASE SQL Engine

Version 4.1.5

Copyright

Acknowledgements

Information regarding Unicode has been provided in part courtesy of the Unicode Consortium. The Unicode Consortium is a non-profit organization founded to develop, extend and promote use of the Unicode Standard, which specifies the representation of text in modern software products and standards. The membership of the consortium represents a broad spectrum of corporations and organizations in the computer and information processing industry. The consortium is supported financially solely through membership dues. Membership in the Unicode Consortium is open to organizations and individuals anywhere in the world who support the Unicode Standard and wish to assist in its extension and implementation.

Portions of the information included herein regarding IBM's ICU has been reprinted by permission from International Business Machines Corporation copyright 2001

jBASE, jBASE BASIC, jED, jSHELL, jLP, jEDI, jCL, jQL, j3 j4 and jPLUS files are trademarks of TEMENOS Holdings NV.

REALITY is a trademark of Northgate Solutions Limited.

PICK is a trademark of Raining Data Inc.

All other trademarks are acknowledged.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Windows, Windows NT, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names used in this publication may be trademarks or service marks of others.

Errata and Comments

If you have any comments regarding this manual or wish to report any errors in the documentation, please document them and send them to the address below:

Technical Publications Department

TEMENOS UK Limited

2 People building

Maylands Ave.

Hemel Hempstead

Herts.

HP2 4NWEngland

Tel SB: +44 (0) 1442 431000

Fax: +44 (0) 1442 431001

Please include your name, company, address, and telephone and fax numbers, and email address if applicable. documentation@temenos.com

Contents

Documentation Conventions

This manual uses the following conventions:

Convention	Usage
BOLD	In syntax, bold indicates commands, function names, and options. In text, bold indicates keys to press, function names, menu selections, and MS-DOS commands.
UPPERCASE	In syntax, uppercase indicates JBASE commands, keywords, and options; BASIC statements and functions; and SQL statements and keywords. In text, uppercase also indicates JBASE identifiers such as filenames, account names, schema names, and Windows NT filenames and pathnames.
UPPERCASE *Italic*	In syntax, italic indicates information that you supply. In text, italic also indicates UNIX commands and options, filenames, and pathnames.
Courier	Courier indicates examples of source code and system output.
Courier Bold	**Courier Bold** In examples, courier bold indicates characters that the user types or keys (for example, <Return>).
[]	Brackets enclose optional items. Do not type the brackets unless indicated.
{}	Braces enclose nonoptional items from which you must select at least one. Do not type the braces.
ItemA | itemB	A vertical bar separating items indicates that you can choose only one item. Do not type the vertical bar.
. . .	Three periods indicate that more of the same type of item can optionally follow.

$\Rightarrow$ A right arrow between menu options indicates you should choose each option in sequence. For example, "Choose **File** $\Rightarrow$**Exit**" means you should choose **File** from the menu bar, and then choose **Exit** from the File pull-down menu.

Syntax definitions and examples are indented for ease in reading.

All punctuation marks included in the syntax—for example, commas, parentheses, or quotation marks—are required unless otherwise indicated.

Syntax lines that do not fit on one line in this manual are continued on subsequent lines. The continuation lines are indented. When entering syntax, type the entire syntax entry, including the continuation lines, on the same input line.

Overview

One of the main benefits of providing a SQL engine for jBASE is such that the database can be used with external tools and APIs. This document is meant to be used with the jDBC Driver manual which gives a description of how the JAVA API for jDBC can be used with jBASE. In addition, there is an API for jBASE BASIC that is covered in this manual.

SQL has many benefits that can be applied to the multi-valued, hierarchical, database jBASE. In particular with jBASE, SQL allows users to query data where there might be tables within tables and no primary-key/foreign key relationship (these relationships are defined in the dictionary). This is an extreme advantage not available in most other RDBMS systems. Some of the advantages of using SQL over the traditional query language of jQL are discussed below:

1) SQL allows sub-queries, UNION/INTERSECT/MINUS statements, and allows joins. jQL does not. jQL might take 2 or 3 queries to do the work of one SQL statement. jQL may programmatically require more lines of code to accomplish the same task.

2) To call user defined functions in jQL, there needs to be a dictionary item representing this, usually expressed as an Itype. This clutters the dictionary. SQL allows use of functions directly in the language (ex. SELECT MYFUNC(a.FIRSTNAME,a.AGE) FROM MYCUSTS a). One can build complex virtual columns without having to modify the dictionary to do it.

3) SQL has support for grouping records with GROUP BY and further selecting on those grouped records with the HAVING keyword. While jQL has group functionality with some verbs (grouping is not supported with the most commonly used jQL verbs that return select lists), it doesn't have the HAVING functionality.

4) SQL is a more structured language that has no implications in it. Therefore, it is more readable and more easily understood. For example,

SELECT MYCUSTS WITH FIRSTNAME = "JIM" OR WITH FIRSTNAME = "JOHN"

SELECT MYCUSTS WITH FIRSTNAME = "JIM" OR FIRSTNAME = "JOHN"

SELECT MYCUSTS WITH FIRSTNAME = "JIM" OR "JOHN"

are all valid jQL statements that return the exact same results. In addition, jQL allows one to put the ordering clause before the selection criteria clause and vice versa.

Assumptions

While it is not entirely necessary that the reader understand jQL syntax, it is assumed that the reader is familiar with SQL syntax. It is also assumed that the user understand that jBASE is a hierarchical database and not a relational database (meaning that data is not necessarily normalized to 3rd normal form as in relational databases). Other assumptions follow:

Filename and tablename are used synonymously in this document.

$JBCRELEASEDIR refers to the directory where jBASE was installed.

$JEDIFILEPATH refers to the search path jBASE uses to find files or tables.

SQL Functions

Unless otherwise noted, all jBASE SQL functions match the syntax of those found in Oracle. Functions not supported are documented later in this document.

SQLSELECT PROGRAM

The SQLSELECT program is the program that runs SQL statements on the jsh. It displays headers that are supplied by the dictionary. Some important notes:

1. By default, data is truncated according to the size of the display length. For example, if the dictionary item looks like this,

```
jsh -->jed MYCUSTS]D ADDR1

File MYCUSTS]D , Record 'ADDR1'

Command->

0001 A

0002 3

0003 Address Line 1

0004

0005

0006

0007

0008

0009 L

0010 6
```

then any data beyond 6 characters will be truncated as shown below.

```
jsh -->SQLSELECT ADDR1 FROM MYCUSTS WHERE FIRSTNAME = 'JIM'

ADDR1

------

1 SUN

1 SUN

64 HAD

121 EL

1 SUN

10260

10260


Selected 7 rows.
```

jBASE is different than Oracle and other relational databases in that the size of the variable is not declared (it can be any size up to the max size of the file). If the user wishes to display all data, one can do so by setting the environment variable JSQLSHOWRAWDATA as shown below.

```
jsh -->set JSQLSHOWRAWDATA=1

jsh -->SQLSELECT LASTNAME, ADDR1 FROM MYCUSTS WHERE FIRSTNAME = 'JIM'

HARRISON^1 SUN AVENUE

SUE^1 SUN AVENUE

LAMBERT^64 HADDOCKEN PLACE

FLETCHER^121 ELEVEN SQUARE

COOPER^1 SUN AVENUE

FENCES^10260 SW GREENBURG RD

FREEMAN^10260 SW GREENBURG RD


Selected 7 rows.
```

Running in the mode JSQLSHOWRAWDATA will ignore header processing. You will also find that the addr1 field above is no longer truncated. Note as well, that attribute marks are displayed as the '^' character in this reporting mode.

2) Headers can be turned off as well by setting the environment variable JSQLHEADER=OFF. Formatting will be preserved in this mode, but JSQLSHOWRAWDATA overrides any setting of JSQLHEADER.


```
jsh -->set JSQLHEADER=off

jsh -->SQLSELECT LASTNAME, ADDR1 FROM MYCUSTS WHERE FIRSTNAME = 'JIM'

HARRISON           1 SUN

SUE                1 SUN

LAMBERT            64 HAD

FLETCHER           121 EL

COOPER             1 SUN

FENCES             10260

FREEMAN            10260


Selected 7 rows.
```

Examples of SQL

The example files

These examples are located in the $JBCRELEASEDIR/samples/SQL directory and can be run in or outside a jsh by simply adding this directory to the $JEDIFILEPATH. The MYCUSTS and MYCUSTS2 files are indexed and these indexes are utilized.

On the jsh

Attribute Marks (a character to delimit database fields) are represented with the character '^'.

1) Example of a join between two files.

```
jsh-->SQLSELECT a.FIRSTNAME, b.AGE, a.LASTNAME,b.LASTNAME FROM MYCUSTS a, CUSTOMERS b
WHERE a.FIRSTNAME = b.FIRSTNAME AND a.FIRSTNAME = 'JIM' AND a.LASTNAME = 'FLETCHER'
```

FIRSTNAME	AGE	LASTNAME	LASTNAME
JIM	41	FLETCHER	STALLED
JIM	50	FLETCHER	JAMES
JIM	58	FLETCHER	SUE

```
Selected 3 rows.
```

2) Multivalued example of a join between two files

(SYSTEMTYPE is a multivalued field and hence one can see the implied JOIN (as SYSTEMTYPE is a table within a table) as well as the JOIN beween the two tables. Note that we can thwart the implied JOIN by creating an ASSOCIATION. See ASSOCIATIONS).

```
jsh -->SQLSELECT  a.FIRSTNAME, a.SYSTEMTYPE, b.AGE, b.LASTNAME  FROM  MYCUSTS  a,
CUSTOMERS b WHERE a.FIRSTNAME = b.FIRSTNAME AND a.FIRSTNAME = 'JIM' AND a.LASTNAME =
'FLETCHER'
```

FIRSTNAME	SYSTEMTYPE	AGE	LASTNAME
JIM	ROS	41	STALLED
JIM	Boo! Not jBASE	41	STALLED
JIM	ROS	50	JAMES
JIM	Boo! Not jBASE	50	JAMES
JIM	ROS	58	SUE
JIM	Boo! Not jBASE	58	SUE

```
Selected 6 rows.
```

3) Example of a subquery

```
jsh-->SQLSELECT DISTINCT a.FIRSTNAME, a.LASTNAME FROM MYCUSTS a WHERE a.FIRSTNAME IN
( SELECT b.FIRSTNAME FROM CUSTOMERS b WHERE b.AGE = 50)

FIRSTNAME               LASTNAME
----------------------- --------------------
JIM                     FLETCHER
CLIVE                   PIPENSLIPPERS
JIM                     FREEMAN
CLIVE                   DELL
CLIVE                   COOPER
CLIVE                   JACKSON
JIM                     HARRISON
JIM                     SUE
JIM                     LAMBERT
JIM                     COOPER
JIM                     FENCES
CLIVE                   GATES


FIRSTNAME               LASTNAME
----------------------- --------------------
CLIVE                   FLETCHER
CLIVE                   WALKER
CLIVE                   BOYCOTT


Selected 15 rows.
```

4) Example of the LIKE keyword

(results not shown)

```
jsh-->SQLSELECT a.FIRSTNAME, a.LASTNAME FROM MYCUSTS a WHERE a.FIRSTNAME LIKE 'JIM%'
```

5) Example of the BETWEEN keyword

(results not shown)

```
jsh-->SQLSELECT a.FIRSTNAME, a.LASTNAME FROM CUSTOMERS a WHERE a.FIRSTNAME BETWEEN
'JIM' AND 'JOHNO'
```

6) Example of the EXISTS keyword

(results not shown)

```
jsh-->SQLSELECT a.FIRSTNAME, a.LASTNAME FROM CUSTOMERS a WHERE EXISTS ( SELECT
FIRSTNAME FROM MYCUSTS WHERE FIRSTNAME = 'DONNA' )
```

In a BASIC Program

Retrieving raw data is not difficult to do programmatically using the jQL API.

The following code below is taken directly from;

$JBCRELEASEDIR/samples/SQL/MYSQLLIST.b.

It is meant to show how rows are returned with user given selection criteria. Example of output follows the code. (To compile: the commands are (1) BASIC . MYSQLLIST.b (2) CATALOG . MYSQLLIST.b)

```
*

* This program is an example program that shows how to fetch data

*  with SQL Syntax.

*

*

PROGRAM MYSQLLIST


INCLUDE JQLINTERFACE.h


ResultCode = 0


* Get Selection Criteria

CRT "Enter Selection Criteria :":

INPUT SelCriteria

IF SelCriteria = "" THEN STOP


* Compile the statement

Options = JQLOPT_USE_SQLSELECT

SelCriteria =  "SELECT ":SelCriteria

ResultCode = JQLCOMPILE(Statement, SelCriteria,Options,ErrorString)

IF ResultCode < 0 THEN STOP ErrorString


* Start execution

sel = ""

Status = JQLEXECUTE(Statement,sel)


* Main Output Loop

ProcessedItems = 0

LOOP

      Status = JQLFETCH(Statement,Control,Data)

WHILE Status >= 0 DO

      IF Control<1> >= 0 THEN

            * Detail Line

            ProcessedItems++
```

```
                  CRT "Data :": CHANGE(CHANGE(Data, @VM, "]"),@AM,"^")

          END
REPEAT
CRT "Processed ":ProcessedItems
```

Example run

```
jsh -->MYSQLLIST

Enter Selection Criteria :?a.FIRSTNAME FROM CUSTOMERS a

Data :JIM

Data :JIM

Data :DONNAYA

Data :JOHNO

Data :^

Data :CLIVE

Data :JIM

Processed 7
```

limiting multi-values in display

When a multi-valued attribute is presented in the SQLSELECT clause, and that same attribute is also present in the WHERE clause, a question arises as to how the data is to be displayed. Let's take an easy example. Below is the data as it is stored on disk. Attribute1 is the FIRSTNAME column, Attribute2 is the LASTNAME column and Attribute13 is the SYSTEMTYPE multi-valued column (different values are separated with a] character...)

```
MYCUSTS2.. 0000162

FIRSTNAME. JIM

Last Name. FREEMAN

SYSTEMTYPE... Another Pick ] Boo! Not jBASE  ]  jBASE  ] ROS ]  UNI*
]  Another Pick
```

First, let's look at a jQL listing of the file. (Note that the bolded words at the end of the query are the output specification).

```
jsh-->LIST MYCUSTS2 WITH FIRSTNAME = "JIM" AND LASTNAME = "FREEMAN"
AND SYSTEMTYPE >= 'ROS' AND SYSTEMTYPE != 'Boo! Not jBASE' FIRSTNAME
LASTNAME SYSTEMTYPE

PAGE    1                                          09:33:43  17
OCT 2003


MYCUSTS2.. 0000162
```

```
FIRSTNAME. JIM

Last Name. FREEMAN

SYSTEMTYPE... Another Pick   Boo! Not jBASE   jBASE   ROS   UNI*
Another Pick


 1 Records Listed
```

You can see that the whole item is returned and every attribute in SYSTEMTYPE is returned even though we've attempted to narrow SYSTEMTYPE in the query with two conditions. Why does this happen? **Because we are selecting on the item and not the multi-values in the jQL language**. In other words, each ITEM meets the criteria of SYSTEMTYPE >= 'ROS' AND SYSTEMTYPE != 'Boo! Not jBASE', not each multi-value. (There is an ITEM that has at least one multi-value that meets the condition, hence the AND clauses can be thought of as OR clauses).

In jQL there is a way to "limit" the display of multi-values. This is shown underlined below where the output specification of SYSTEMTYPE is met with added conditions.

```
jsh-->LIST MYCUSTS2 WITH FIRSTNAME = "JIM" AND LASTNAME = "FREEMAN"
AND SYSTEMTYPE >= 'ROS' AND SYSTEMTYPE != 'Boo! Not jBASE' FIRSTNAME
LASTNAME SYSTEMTYPE GE "ROS" AND NE "Boo! Not jBASE"
PAGE    1                                           09:44:29  17
OCT 2003


MYCUSTS2.. 0000162

FIRSTNAME. JIM

Last Name. FREEMAN

SYSTEMTYPE... jBASE   ROS   UNI*


 1 Records Listed
```

The effect is that there are only 3 values now displayed for SYSTEMTYPE. Now let's look at a query that is returning results for SQL. This is what jBASE will return by default:

```
jsh -->SQLSELECT FIRSTNAME, LASTNAME, SYSTEMTYPE FROM MYCUSTS2 WHERE
FIRSTNAME = 'JIM' AND LASTNAME = 'FREEMAN' AND SYSTEMTYPE >= 'ROS'
AND SYSTEMTYPE != 'Boo! Not jBASE'
FIRSTNAME               LASTNAME            SYSTEMTYPE

----------------------- -------------------- ----------------------
--
JIM                     FREEMAN             jBASE

JIM                     FREEMAN             ROS

JIM                     FREEMAN             UNI*
```

```
Selected 3 rows.
```

Here the WHERE clause itself limits what is returned.

So here is the dichotomy of limiting. Are we selecting on the ITEM or are we selecting on the MULTI-VALUES being displayed on the item? Which one does the WHERE clause refer to? By default, the SQL engine selects on the multi-values and the AND clauses are treated as AND clauses when limiting the display. Such that the query in SQL will produce no results as shown below where it will display the item with the LIST command.

```
jsh-->SQLSELECT FIRSTNAME, LASTNAME, SYSTEMTYPE FROM MYCUSTS2 WHERE
FIRSTNAME = 'JIM' AND LASTNAME = 'FREEMAN' AND SYSTEMTYPE = 'UNI*'
AND SYSTEMTYPE = 'jBASE'

Selected 0 rows.
```

But what if you really want to select on the ITEM in SQL and in effect have the AND clauses be treated as OR clauses? This behaviour can be changed by setting the environment variable JQL_LIMIT_WHERE to any value or setting the option programmatically as shown below.

```
Options = JQLOPT_LIMIT_WHERE

ResultCode = JQLCOMPILE(Statement, SelCriteria,Options,ErrorString)
```

With this variable set, the following query would produce the results shown below.

```
jsh-->SQLSELECT FIRSTNAME, LASTNAME,SYSTEMTYPE FROM MYCUSTS2 WHERE
FIRSTNAME = 'JIM' AND LASTNAME = 'FREEMAN' AND SYSTEMTYPE = 'UNI*'
AND SYSTEMTYPE = 'jBASE'

FIRSTNAME                   LASTNAME              SYSTEMTYPE
----------------------- -------------------- ----------------------
--

JIM                         FREEMAN               jBASE
JIM                         FREEMAN               UNI*


Selected 2 rows.
```

In addition, the user can choose to ignore limiting the display all together by setting the environment variable JQL_DONT_LIMIT or setting the Option JQLOPT_DONT_LIMIT.

Associations

A common question is how data is associated if one column or more columns are multi-valued and the rest are not. Take this example where both NUMBEERSPERBRAND and NUMCALSPERBRAND are multi-valued:

```
jsh -->SQLSELECT a.LASTNAME, a.NUMBEERSPERBRAND, a.NUMCALSPERBRAND
FROM CUSTOMERS a WHERE a.FIRSTNAME = 'JIM'


LASTNAME             NUMBEERSPERBRAND              NUMCALSPERBRAND
-------------------- ----------------------------- ----------------
--------------
STALLED              10                            105
STALLED              12                            100
JAMES                6                             150
JAMES                12                            100
SUE                  4                             200
SUE                  12                            100


Selected 6 rows.
```

The data on disk for JIM STALLED is shown below. Attribute 5 (NUMBEERSPERBRAND) and Attribute 6 (NUMCALSPERBRAND) are both multi-valued. Yet, only two rows are returned from the SQL query above. Why? Attribute 5 and Attribute 6 are associated in the dictionary.

```
0001 JIM

0002 STALLED

0003 41

0004 2

0005 10]12

0006 105]100

0007 OLY]BUD

0008 35 JIM IDLE RD.

0009 PORTLAND

0010 97210

0011 US

0012 FIDO\JACK
```

Attribute 7 (BRANDS) in the dictionary (CUSTOMERS]D) is the controlling attribute for NUMBEERSPERBRAND (Attribute 5) and NUMCALSPERBRAND (Attribute 6). This is defined in bold in attribute 4 below.


```
    BRANDS
```

```
001 A

002 7

003 BRANDS

004 C;5;6

005

006

007

008

009 L

010 30


     NUMBEERSPERBRAND

001 A

002 5

003 NUMBEERSPERBRAND

004 D;7

005

006

007

008

009 R

010 30


     NUMCALSPERBRAND

001 A

002 6

003 NUMCALSPERBRAND

004 D;7

005

006

007

008

009 L

010 30
```

The dependant attributes (NUMBEERSPERBRAND and NUMCALSPERBRAND) define their controlling attribute in attribute 4 as well.

Without this relationship defined, the same query would yield vastly different results.

```
jsh -->SQLSELECT a.LASTNAME, a.NUMBEERSPERBRAND, a.NUMCALSPERBRAND
FROM CUSTOMERS a WHERE a.FIRSTNAME = 'JIM'

STALLED^10^105

STALLED^10^100

STALLED^12^105

STALLED^12^100

JAMES^6^150

JAMES^6^100

JAMES^12^150

JAMES^12^100

SUE^4^200

SUE^4^100

SUE^12^200

SUE^12^100


Selected 12 rows.
```

Now we see that there is a JOIN taking place, so please take note that multi-valued attributes (tables within a table) need to be related to one another in the dictionary, otherwise a JOIN will occur.

Dictionaries

jBASE has different mechanisms to represent dictionary items or meta data. These are documented in the jQL documentation and the following section assumes a cursory knowledge of dictionary definitions. The main thing to keep in mind is that there are Dictionary files which hold the meta-data, and data files which hold the application data.

To view the MYCUCSTS dictionary, issue the following command.

```
jsh -->LIST-ITEM MYCUSTS]D
```

```
You will see records like this:
    FIRSTNAME
001 A
002 1
003 FIRSTNAME
004
005
006
007
008
009 L
010 24


    LASTNAME
001 A
002 2
003 Last Name
004
005
006
007
008
009 L
010 20
etc.
```

FIRSTNAME maps to Attribute 1 in the datafile and LASTNAME maps to Attribute 2 in the data file. Now let's look at the raw data for an item (jed is a jBASE editor similar to ED. 0000011 below is the record key of the shown record.)

```
jsh -->jed MYCUSTS 0000011

File MYCUSTS , Record '0000011'                              Inser

Command->

001 JIM

002 HARRISON

003 1 SUN AVENUE

004

005 SAN JOSE

006 IN

007 09324

008 (125) 555-1337

009 (124) 555-1337

010 JIMH@compe.com

011 SPARC]INTEL PII]ALPHA AXP]DIGITAL]DIGITAL]DELL]ALPHA AXP

012 HPUX]SOLARIS]DGUX]TRU64]DGUX]TRU64]SOLARIS

013 UNI*]ROS]Another Pick]Another Pick]ROS]UNI*]jBASE

014 1980]1315]1475]1016]843]1436]879
```

One can see the FIRSTNAME "JIM" in Attribute 1 and the LASTNAME "HARRISON" in Attribute2. Attribute13 is SYSTEMTYPE and is multi-valued.

Dictionary Considerations

1) For numeric comparisons, the item must be defined as right justified as shown below in attribute 9.

```
    NUMBEERSPERBRAND
001 A
002 5
003 NUMBEERSPERBRAND004 D;7
005
006
007
008
009 R
010 30
```

This allows one to do queries such as the following;

```
jsh-->SQLSELECT LASTNAME,  NUMBEERSPERBRAND FROM CUSTOMERS2 WHERE
FIRSTNAME = 'JIM' AND LASTNAME = 'JAMES' AND NUMBEERSPERBRAND > 5
AND NUMBEERSPERBRAND < 10
```

2) Columns that belong to the same relation have to be associated, otherwise a JOIN will occur if it appears in the SELECT clause (See the Chapter on Associations)

Dictionary Descriptors

(not inclusive)

A-descriptor

The 10-line **attribute definition** which is used on generic MultiValue systems including jBASE and on **UniVerse**

	BIRTH.DATE	AGE	dataname
001	A	A	D/CODE
002	1	0	A/AMC
003	Date of birth	Age	TAG
004			V/STRUCT
005			
006			
007	D	MD02	V/CONV
008		F;D;1;-;'100';'365.25';/	V/CORR
009	R	R	V/TYPE
010	11	3	V/MAX

17

D-descriptor

```
001 D                               TYPE
002 7                               LOC
003 D2/                             CONV
004 Date of Birth                   NAME
005 8R                              FORMAT
006 M                               SM
007 DETS4                           ASSOC
```

I-descriptor

```
001 I                               TYPE
002 @RECORD<1,2>*@RECORD<1,3>        LOC = the I-expression to be
evaluated
003 MD2                             CONV
004 Value                           NAME
005 10R                             FORMAT
006 S                               SM
007 DETS5                           ASSOC
```

Current limitations

Unsupported SQL Features

Correlated subqueries are not yet supported, as well as outer joins. Joins on multi-valued attributes are not supported when these attributes are coming from different tables.

Unsupported SQL Functions

TRUNC

TRANSLATE

STDDEV

VARIANCE

CONVERT

CHARTOROWID

HEXTORAW

RAWTOHEX

ROWIDTOCHAR

TO_CHAR – Not needed by jBASE handled by dictionary Conversions.

TO_DATE – Not needed by jBASE handled by dictionary Conversions.

TO_NUMBER – Not needed by jBASE handled by dictionary Conversions.

DUMP

GREATEST

LEAST

UID

USERENV

INITCAP

LPAD

SELECT * AND SELECT COUNT(*)

Select * might not work as expected if some of the data is multi-valued. This is because multi-valued data is really a table within a table. Therefore, the multi-valued data (or inner table) is joined to the outer. This might not be what is intended. It is important to know the structure of the underlying data before using the * syntax of SQL. By contrast, SELECT COUNT(*) FROM fileName , returns the amount of items in filename, which is more likely what the user expects. For example, if there is a file CUSTOMER with no multi-values (tables within a table), then the following query would produce the expected results;

```
Jbase -->SQLSELECT * FROM CUSTOMERS WHERE FIRSTNAME = 'DONNAYA'

@ID         FIRSTNAME              LASTNAME            AGE
----------  ---------------------  ------------------  --------------
3           DONNAYA                HUNT                31

Selected 1 rows.
```

However, if you query the file CUSTOMERS in the samples directory (which does contain tables within tables) with the exact same query, you retrieve back 3888 rows(because there are multiple tables within tables in this file)! You can, however, retrieve the tables within tables in one column which can then be parsed programmatically. To do this, set the environment variable JQL_DONT_MAKE_ROWS as shown below;

```
>set JQL_DONT_MAKE_ROWS=1
>jsh
jsh -->SQLSELECT * FROM CUSTOMERS WHERE FIRSTNAME = 'DONNAYA'
@ID               FIRSTNAME                     LASTNAME                 AGE
STAR NUMBEERSPERBRAND                 NUMCALSPERBRAND
BRANDS                                ADDR1                               CITY
POSTCODE    COUNTRY PETS
--------------  -----------------------  --------------------  -------
----  ---------------------------  ---------------
--------------  --------------------------  --------------------
--------  -------------------- ---------- ----------
 --------------------
3               DONNAYA                       HUNT                     31
11   4]6                               200]150
             KRONENBURG]COORS                      105 DONNYA HUNT
RD.]106 MARCUS SAN FRAN.]SAN FRAN.] 91654]9165 US]US]US
 MADDOX\GLAVINE]CONE\

Selected 1 rows.
```

As well, the * syntax is now only supported for one and only one table in the FROM clause. Queries such as SELECT a.*, b.* FROM table1 a, table2 b WHERE a.id = b.id will not work. This is a limitation that will be fixed in a future release.

SQL Programatic Options

Some options need to be known such that the statement can be compiled. These options are passed to JQLCOMPILE like below.

```
Options = JQLOPT_USE_SQLSELECT + JQLOPT_DONT_MAKE_ROWS

SelCriteria =  "SELECT ":SelCriteria

ResultCode = JQLCOMPILE(Statement, SelCriteria,Options,ErrorString)
```

Available options	Description
JQLOPT_USE_SQLSELECT	Use the SQL engine instead of the jQL engine
JQLOPT_LIMIT_WHERE	Treat ANDs LIKE Ors when limiting (see section on limiting)
JQLOPT_DONT_LIMIT	Don't do any limiting at all
JQLOPT_DONT_MAKE_ROWS	Keep multi-values and subvalues as is without splitting them up into rows (most useful for PICK developers who want to handle processing multi-values and sub-values themselves)

Appendix A – SQL / jQL constants

Constant	Value	Used	Notes
JQLOPT_USE_SELECT	1	Yes	Used to set _useSelectList, this as a switch for various internal functions
JQLOPT_FETCH_ALL_VALUES	2	No	
JQLOPT_USE_SQLSELECT	4	Yes	Used to say that this is the SELECT function…
JQLOPT_LIMIT_WHERE	8	Yes	Sets limitDisplayWithWhere, also assocated with env var JQL_LIMIT_WHERE
JQLOPT_DONT_LIMIT	16	Yes	Sets dontLimit, oppisite of above, (Assocated with env var JQL_DONT_LIMMIT)
JQLOPT_DONT_MAKE_ROWS	32	Yes	Sets dontMakeRows, (see env var JQL_DONT_MAKE_ROWS), should force SQL to not split up all MV's
JQLOPT_TRANSLATE_DB2	64	Yes	Sets translateDecode, Different syntax
JQLOPT_TRANSLATE_SQL_SERVER	128	Yes	See TRANSLATE_DB"
JQLOPT_SYSTEM_QUERY	256	Yes	// Must be used with JQLOPT_USE_SQLSELECT and refers to a prop. Below
JQLOPT_FORCE_SELECT	512	Yes	// switch on trigers in no active select list (if file has triggers)
JQLOPT_USE_SQLDELETE	1024	Yes	// Delete, suports clear file, where but no sub queries
JQLOPT_USE_SQLINSERT	2048	Yes	// Insert, add new data
JQLOPT_USE_SQLUPDATE	4096	Yes	// Update, change existing data
JQLOPT_USE_SQLCREATETABLE	8192	Yes	// Simple create table command
JQLOPT_USE_SQLDROPTABLE	16384	Yes	// drop table command
JQLOPT_USE_SQLBEGINTRANS	32768	Yes	// BEGIN TRANSACTION
JQLOPT_USE_SQLCOMMITTRANS	65536	Yes	// COMMIT TRANSACTION
JQLOPT_USE_SQLROLLBACKTRANS	131072	Yes	// ROLLBACK TRANSACTION
JQLOPT_USE_SQLSAVETRANS	262144	Yes	// SAVE TRANSACTION
JQLOPT_USE_SQLPREPARE	524288	Yes	// Ran via SQL PREPARE/BIND ,
JQLOPT_USE_SQL	1047556	Yes	// We are running a command as jSQL
FIRST_STMT_PROPERTY	1000	Yes	Location in the array of the first item,
STMT_PROPERTY_HEADING	1000	Yes	Location in property array of JQL Display items
STMT_PROPERTY_FOOTING	1001	Yes	Location in property array of JQL Display items
STMT_PROPERTY_GRAND_TOTAL	1003	Yes	Location in property array of JQL Display items
STMT_PROPERTY_LPTR	1004	Yes	Location in property array of JQL Display items
STMT_PROPERTY_COL_HDR_SUPP	1005	Yes	Location in property array of JQL Display items
STMT_PROPERTY_COL_SPACES	1006	Yes	Location in property array of JQL Display items
STMT_PROPERTY_COL_SUPP	1007	Yes	Location in property array of JQL Display items
STMT_PROPERTY_COUNT_SUPP	1008	Yes	Location in property array of JQL Display items
STMT_PROPERTY_EXECUTE_COUNT	1009	Yes	Location in property array of JQL Display items
STMT_PROPERTY_DBL_SPACE	1010	Yes	Location in property array of JQL Display items
STMT_PROPERTY_DET_SUPP	1011	Yes	Location in property array of JQL Display items
STMT_PROPERTY_HDR_SUPP	1012	Yes	Location in property array of JQL Display items
STMT_PROPERTY_ID_SUPP	1013	Yes	Location in property array of JQL Display items
STMT_PROPERTY_MARGIN	1014	Yes	Location in property array of JQL Display items
STMT_PROPERTY_NOPAGE	1015	Yes	Location in property array of JQL Display items
STMT_PROPERTY_NOSPLIT	1016	Yes	Location in property array of JQL Display items
STMT_PROPERTY_ONLY	1017	Yes	Location in property array of JQL Display items
STMT_PROPERTY_OUTCOLS	1018	Yes	Location in property array of JQL Display items
STMT_PROPERTY_VERT	1019	Yes	Location in property array of JQL Display items
STMT_PROPERTY_TOTAL_WIDTH	1020	Yes	Location in property array of JQL Display items
STMT_PROPERTY_FILE_NAME	1021	Yes	Location in property array of JQL Display items

STMT_PROPERTY_BINARY_MODE	1022	Yes	Location in property array of JQL Display items
STMT_PROPERTY_ASCII	1023	Yes	Location in property array of JQL Display items
STMT_PROPERTY_EBCDIC	1024	Yes	Location in property array of JQL Display items
STMT_PROPERTY_TAPELABEL	1025	Yes	Location in property array of JQL Display items
STMT_PROPERTY_WITHIN	1026	Yes	Location in property array of JQL Display items
STMT_PROPERTY_UNIQUE	1027	Yes	Location in property array of JQL Display items
STMT_PROPERTY_NONULLS	1028	Yes	Location in property array of JQL Display items
STMT_PROPERTY_APPLY_OCONV	1029	Yes	Location in property array of JQL Display items
STMT_PROPERTY_APPLY_FORMAT	1030	Yes	Location in property array of JQL Display items
STMT_PROPERTY_SYSTEM_QUERY_TABLE_TYPES	1031	Yes	Location in property array of JQL Display items
STMT_PROPERTY_SYSTEM_QUERY_SCHEMAS	1032	Yes	Location in property array of JQL Display items
STMT_PROPERTY_SYSTEM_QUERY_TABLES	1033	Yes	Location in property array of JQL Display items
STMT_PROPERTY_SYSTEM_QUERY_COLUMNS	1034	Yes	Location in property array of JQL Display items
LAST_STMT_PROPERTY	1034	Yes	Used to check we don't overflow passed the last lement in the properties array….
FIRST_COL_PROPERTY	100	No	Equivalent to first statement prop…
COL_PROPERTY_HEADING	100	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_FORMATTED_HEADING	101	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_WIDTH	102	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_HEADING_WIDTH	103	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_VALUETYPE	104	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_MVGROUPNAME	105	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_SVGROUPNAME	106	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_DICT_IID	107	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_UPDATEABLE	108	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_AGGREGATE	109	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_AGGREGATE_SEPARATOR	110	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_VISIBLE	111	Yes	Used to say what this element is in outcolumn.cpp
COL_PROPERTY_JUSTIFATION	112	Yes	Used to say what this element is in outcolumn.cpp
LAST_COL_PROPERTY	111	Yes	Used to check we don't overflow passed the last element in the properties array….
BREAK_OPTIONS_B	1	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)
BREAK_OPTIONS_D	2	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)
BREAK_OPTIONS_L	4	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)
BREAK_OPTIONS_N	8	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)
BREAK_OPTIONS_O	16	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)
BREAK_OPTIONS_P	32	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)
BREAK_OPTIONS_R	64	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)
BREAK_OPTIONS_T	128	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)
BREAK_OPTIONS_U	256	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)
BREAK_OPTIONS_V	1024	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)
BREAK_OPTIONS_LR	2048	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)
BREAK_OPTIONS_SUP	4096	Yes	Used with breakDefnArray, sets break on types, (MIN/MAX/TOTAL etc…)

Comment Sheet

Please give page number and description for any errors found:

Page	Error

Please use the box below to describe any material you think is missing; describe any material which is not easily understood; enter any suggestions for improvement; provide any specific examples of how you use your system which you think would be useful to readers of this manual. Continue on a separate sheet if necessary.

Copy and paste this page to a word document and include your name address and telephone number. Email to documentation@temenos.com