QBAL

A Professional Sample Balancing Program

User's Guide

Revision: 21.1.5

QBAL is the sample balancing component of the QUIP System

QUIP System software is available from:

www.quipsoftware.com

QBAL documentation by

Jan Werner Data Processing

TABLE OF CONTENTS

INTRODUCTION

The QUIP System is a software package that allows almost any data set to be tabulated and the results presented as fully annotated tables. It includes programs to handle data manipulation and conversions, validity checking, marginal counts and weighting, as well as cross-tabulation.

The QUIP System was designed to meet the needs of data processing professionals. Programs are script-driven using a common structure. This allows for batch processing or for users to create their own custom environments, using QUIP System programs as processing engines.

The programs in the QUIP System share a common specification language that allows users to describe extremely complex combinations of logical conditions and numeric expressions. This language is compiled in memory at run-time, so all programs run very quickly on large data files.

This guide provides an introduction to QBAL, the sample balancing component of the QUIP System. QBAL computes weight values used to match a sample to the overall population on known characteristics by means of a statistical technique known as iterative proportional fitting.

The following conventions are used throughout this document:

Angle brackets:	<...>	A mandatory item to be specified.
Square brackets:	[...]	An optional item to be specified.
Vertical bar:	..|..	Only one of the listed items should be specified.

QBAL keywords are in uppercase and the minimum abbreviation for each in bold uppercase, lowercase letters are used where a number should be entered (e.g., **WID**TH nnn).

1 BASIC CONCEPTS

When a sample is drawn from a population, sub-groups are frequently represented in different proportions than in the entire population, whether by design or by accident. When analyzing the information obtained from a sample, a researcher will often seek to eliminate the bias caused by this imbalance or to control the effect of sub-groups whose proportions in the overall population are known from other sources.

1.1 Weighting Samples

The adjustment of a sample to match a population on specific sub-groups, sometimes called “post-stratification”, is accomplished by multiplying the count in each sub-group by a number called a **weight**, defined as the ratio of its proportion of the total population to its proportion of the sample. In order to apply the adjustment to all measurements derived from a data set, the weight value is attached to each individual case.

Computing weights on a single dimension is a simple procedure: One obtains marginal counts for each sub-group to determine its proportion in the sample, then divides this into its proportion for the overall population. The values obtained in this manner are called the **rim weights** of the dimension for which they were computed.

When a sample needs to be adjusted on multiple dimensions, the entire sample must be divided into mutually exclusive cells corresponding to the possible combinations of sub-groups from each dimension, then weights for each cell may be computed as the ratio of the proportion of each cell in the population to its proportion in the sample. These are called **cell weights**.

1.2 Sample Balancing

Using cell ratios to weight a sample works well when the number of sub-groups to be adjusted is small, but when many dimensions are involved, the number of cells required quickly grows too large to compute weights using that method. For example, using cell ratios to adjust a sample on 8 age groups, 6 income groups and 4 geographic regions would entail computing 8x6x4=192 different cell weights. Furthermore, some of those sample cells may be empty, which would necessitate combining sub-groups in ways that might negatively affect the analysis.

Unfortunately, one cannot weight on multiple dimensions by simply computing the rim weights for each dimension and combining them, because applying rim weights to a dimension changes the proportions on each other dimension. Methods for adjusting sample sub-groups on multiple dimensions simultaneously are called **sample balancing**.

Iterative proportional fitting or **IPF** is a widely accepted sample balancing technique originally developed by W. Edwards Deming and Frederick F. Stephan to adjust samples taken for economic and social surveys on selected demographic characteristics against data obtained from the U.S. Census. The theory behind IPF is explained in Deming's book STATISTICAL ADJUSTMENT OF DATA (1943), available in reprint from Dover Publications. Details on the Deming-Stephan method are spelled out in Chapter VII: "Adjusting to Marginal Totals."

IPF uses least-squares curve fitting algorithms to obtain a unique weight for each case that minimizes the root mean square error (**RMSE**) across multiple dimensions simultaneously. Then it applies these weights to the data and repeats the procedure using the newly obtained marginal counts to obtain yet another set of weights. This process is repeated for a specified number of iterations or until the difference in the RMSE between successive steps becomes less than a specific minimum value.

1.3 Using QBAL to Balance Samples

QBAL uses an optimized version of Deming's IPF methodology to balance large data sets on many variables simultaneously, quickly and accurately. QBAL takes advantage of the flexibility of the QUIP specification language to allow complex cell specifications without having to recode the data and performs all computations internally in double-precision floating point to virtually eliminate rounding errors in the output.

QBAL accepts previously weighted data and can project back to the original sample size or to a specified population count. The weighting process can be restricted to a subset of the data so that subsamples do not need to be extracted for balancing.

A note of caution: While QBAL makes it extremely easy to balance samples on many variables simultaneously, it cannot compensate for structural problems such as collinearity in the data. Attempting to balance in too many dimensions at once may result in a "negative progression" error message, indicating that there is a dependency condition between the variables defined. This can usually be solved by eliminating one or more variables from the specifications.

Sections 2-5 of this User's Guide describe how to use QBAL to balance samples. Section 6 gives a detailed description of the QUIP specification language, and sections 7-8 describe some advanced features useful for integrating QBAL into automated data processing projects.

2 THE SPEC FILE

QBAL instructions are called specifications, or "specs" for short. QBAL reads the specs from an ASCII text file called a spec file. The spec file contains all the information the program needs to process the data and produce the desired output.

Specs must be written using a text editor that can write files that do not contain any embedded control codes or tab characters. Case is ignored for specs and keywords, but is preserved in annotation, tags, dictionary definitions and literal strings within quotes.

Position on the text line is significant in QBAL variable definitions, where a specific column is used to separate cell specifications from label text. Spaces are used as separators between keywords, but the number of spaces does not matter. Tab characters are not allowed in specs and are treated as non-blank characters in label annotation.

Any line that begins with an asterisk (*) in column 1 is treated as a comment. It will print in the spec listing but otherwise be ignored by the program, as will any completely blank line.

QBAL compiles the spec file in a single pass so forward references are not allowed. Dictionary entries must be defined before they can be referenced elsewhere in the spec file.

After a spec file has been written, QBAL may be run from a command prompt by entering:

```
QBAL <filename> [command line directives|parameters]
```

QBAL will read the spec file named on the command line and compile it, checking for syntax errors and writing a listing file back to disk as it proceeds. If no fatal errors are found, QBAL will read the input data file(s), compute the sample balancing weights and add statistical information about the process to the listing file, then copy the input file(s) to an output file with the weights added in the location specified.

2.1 Elements of the Spec File

The following types of specifications can be used in a QBAL spec file:

DATA DIRECTIVES:	Instructions that describe the input, output and listing files.
ADDRESS MODE:	Instructions that describe how the data will be addressed.
RUN OPTIONS:	Overall parameters for the sample balancing process.
BASE: (Optional)	Applies a filter to the input file describing which cases to include in the sample balancing process.
WEIGHT: (Optional)	Specifies a weight value to be applied to input data before the sample balancing process.
NUMBERSETS	Sets of numbers that determine the marginal proportions for each dimension to balance on.
VARIABLE DEFINITIONS:	Sets of instructions that define the contents of each cell for a given dimension and provide text labels for them.

Numbersets and variables are considered in pairs and should be placed together. Regardless of their names, the first numberset will apply to the first variable, the second numberset to the second variable, and so on.

Other specifications, such as dictionaries and includes, may appear anywhere in the spec file.

3 DATA DIRECTIVES

Data directives identify the input data files and the output files created by QBAL. Each data directive begins with a keyword followed by the file name.

3.1 File Names

QBAL accepts the following data directives:

INTAP[n]	<[path\]filename.ext (lrecl> [V]	Input data file(s).
LISTING	<[path\]filename.ext>	Job spec listing file.
OUTAP1	<[path\]filename.ext (lrecl> [V|C]	Output data file.

If no explicit path is given, QBAL always reads from and writes to the current directory. File names must be in DOS format (8.3). If no LISTING file is specified, QBAL will create one in the current directory, generating a unique name for it.

The INTAP and OUTAP1 directives name data files followed by a left parenthesis and the record length in bytes. Unless explicitly specified as variable or constant, the record length is assumed fixed, meaning that each consecutive block of bytes of the specified record length is considered a separate record.

The letter "V" after the record length indicates variable length records, each of which must be terminated by an end-of-line marker (Carriage Return-Line Feed). For variable length records, the specified record length must be equal or greater to that of the longest record in the file, not counting the 2 bytes for the CR-LF.

The letter "C" after the record length in OUTAP directives indicates constant length records, which are padded with blanks to the indicated record length, then terminated with a CR-LF pair. QGEN looks for the CR-LF when reading files specified as either variable or constant, so while a "C" may be used in input data directives without causing an error, the result is identical to "V".

Any number of INTAP files may be named, (usually numbered INTAP1, INTAP2, etc.). Data files are processed in the order they appear in the spec file, regardless of the numbering used for the INTAP keywords. All input data files should have the same record length.

Only a single OUTAP1 file can be specified If the output record length is shorter than for input, records will be truncated; if longer, records will be padded with blanks to the specified length. Note that “blanks” are space characters in ASCII data and nulls in column binary data.

3.2 Command Line Data Directives

Data directives may optionally be specified on the command line using the following switches:

/I[n]= <[path\]filename.ext>	INTAP file name (n=1,2,3,4)
/IL[n]= <nnn>	INTAP record length (n=1,2,3,4)
/O1= <[path\]filename.ext>	OUTAP1 file name
/OL1= <nnn>	OUTAP1 record length
/L=<[path\]filename.ext>	LISTING file name
/S[n]=<[path\]filename.ext>	SPEC file name (n=1,2,3,4)

Multiple data directives are separated by spaces (e.g., /X=xx /Y=yy /Z=zz). The total length of the command line (including program invocation) may not exceed 127 characters. Multiple spec files listed on the command line are read in numeric sequence and treated as a single spec file. Data directives contained in the spec files always override those provided on the command line.

Variable length record files cannot be specified in command line data directives.

4 GLOBAL OPTIONS

Global options apply to an entire QBAL run. Most are specified on a RUNOPS line which must appear immediately after the data directives at the beginning of the spec file. Most run options will default to a specific setting unless they are explicitly defined otherwise.

4.1 Address Mode

QBAL requires a data address mode to be defined before any specifications can be processed.

The SETUP ADDRMODE statement determines whether specs refer to 1-byte ASCII characters or the 2-byte column-binary format used mostly in marketing research. It also specifies whether the data are addressed using card/column or direct offset notation.

The concept of the address mode is fundamental to understanding how QBAL processes data: The address mode determines how the program interprets the specification of data locations and their contents in the remainder of the spec file. For more on how ADDRMODE affects data addressing, see section 6.1.1.

To set the address mode, use one of the following statements at the beginning of the spec file:

SETUP ADDRMODE B	Column-binary data addressed by card and column.
SETUP ADDRMODE C	Character data addressed by position in record.
SETUP ADDRMODE D	Character data addressed by card and column.
SETUP ADDRMODE E	Column-binary data addressed by position in record.

If address mode is not explicitly stated, QBAL defaults to ADDRMODE B (column-binary data, card/column addressing).

4.2 Run Options

Run options are specified on a line beginning with the keyword RUNOPS. Options are invoked by keywords followed by one or more parameters. The available options are:

PASS <passes,error>	Sets the maximum number of iterations, and the minimum error (RMSRE) required to end the QBAL run. When omitted, the number of passes is set to 10 and the error threshold to 0.1.
POP <nnn>	Project to population size ‘nnn’. If omitted, the output weights will project to the input sample size or the weighted input sample size if an input weight was specified (see section 4.4)
RESULT <field[.decimals]>	Specifies the location and format for the weights resulting from the sample balancing process to be placed in the output file. Any data in the location specified will be overwritten for all cases weighted. Must specify a valid numeric field format (see section 6.1.2).
CONSTRAINTS <max[,min]>	Specifies the maximum and, optionally, the minimum final weight values to be written to the output. If omitted, these default to the largest and smallest numeric values that can be written in the format specified for the RESULT field. 'min' can only be specified if 'max' is specified first. May only be used with ASCII or column-binary result fields.

The RESULT field is the only parameter that must be specified for every QBAL run. Unless integer weight values are desired, the number of decimal places must be specified and one position reserved for the period when weights are written as ASCII or column-binary numbers. For example, if the RESULT field were specified as 121C7.3, a weight value of 1.65 would be written as 001.650 in the 7 bytes starting at position 121.

When requesting integer weights, as required for use with software that does not allow decimal values in ASCII or column-binary data, the POP option should be used to multiply the sample size by the appropriate factor of 10 to preserve adequate precision.

If a final weight value is too large to be written in the RESULT field and no explicit constraints are provided, QBAL will substitute the largest value that can be written in the format specified. Likewise, if a final weight value is so small that it would round to zero, QBAL will substitute the smallest value that can be written to the field.

The CONSTRAINTS keyword allows the user to set a narrower range for the maximum and minimum weight values than allowed by the RESULTS field format.

For example, if the RESULT field were specified as 151C6.3, the largest value that could be written would be 99.999 and the smallest value 00.001. Adding CONSTRAINTS 85,0.01 would change any final weight greater than 85 to 85.000, and any weight smaller than 0.01 to 00.010 in the output file.

Whenever QBAL substitutes a maximum or minimum value for a computed weight, it also displays a message on screen and in the listing file, showing the original and adjusted values and the sequence number within the file of the case for which the adjustment was made.

The PASS option is used to limit the number of iterations. QBAL balances samples through a series of least squares approximations that successively narrow in on the best fit. For each pass, the root mean square relative error (RMSRE) for the current solution is computed and compared to the error threshold. Essentially, the RMSRE measures how much improvement each successive pass adds to the fit. The run ends when either the number of passes specified has been reached or the RMSRE goes below the error threshold level, whichever comes first.

The PASS option can usually be omitted as the default values of 10 passes and a 0.01 error threshold are nearly always sufficient and will be reached within seconds in most situations.

4.3 Base Definition

QBAL allows a base definition that determines which cases in the input file will be used in the sample balancing process. A base definition is indicated by a line beginning with the keyword BASE in column one and, after one or more spaces, a name up to 8 characters long. The next line should contain a logical condition written in the QUIP specification language (see section 6), followed by an END statement.

The format of the base definition is:

```
BASE   <name>
<condition>
END
```

QBAL will use only one base definition in a spec file. If multiple bases are defined, only the last one will be used to filter the input data. Any cases that do not meet the condition specified in the base definition will not be used in the sample balancing process and will be written to the output file unmodified, with no weights placed in the RESULT field.

4.4 Weight Definition

A weight definition instructs QBAL to apply pre-existing weights to the input data for the sample balancing process. The definition begins with the keyword WEIGHT or WT, followed by a valid numeric field, constant or numeric expression (see section 6.1.4):

WEIGHT <expression>

When the value specified consists of a numeric field in the data, this value can be adjusted to account for implicit decimal places by adding a period and the number of decimals by which the field value should be adjusted.

QBAL will stop processing a case and skip to the next one if a weight is specified that contains non-numeric values, so care should be taken that weights always evaluate to valid numbers.

QBAL will use only one weight definition in a spec file. If multiple weights are defined, only the last one will be applied to the input data.

Some examples of weight definitions:

WEIGHT 171CE3.2	The weight is the value found in the 3 character field beginning in position 171, read as having 2 decimal places implied (x.xx) and with any invalid values set to zero
WEIGHT 171ce3/k100	Defines the same weight as above
WT k2.5	The weight is a constant value of 2.5

5 NUMBERSETS AND VARIABLE DEFINITIONS

Sample balancing requires two basic types of information for each dimension, the sample must be divided into mutually exclusive sub-groups along that dimension and the desired proportions for each sub-group in the output must be specified.

QBAL uses a variable definition to specify the sub-groups for each dimension, and a numberset to specify the desired proportions of each sub-group in the weighted data. Each pair of a numberset and a variable in the spec file defines a dimension for the sample balancing process.

QBAL will always match the first numberset with the first variable, the second numberset with the second variable, and so forth, regardless of where they are placed in the spec file, but it is generally a good idea to keep each pair together and give them the same descriptive name.

5.1 Numbersets

A numberset is simply a list of constant numeric values. It is specified with the keyword **NS** in column 1, followed by a name, and a series of numbers separated by blanks. A backslash (\) may be used to indicate that the list of numbers continues on the following line.

The numbers in each numberset are used to specify the desired marginal proportions for the corresponding categories in the associated variable. Each numberset must contain exactly one more entry than there are categories defined in the variable. The first number will be applied to the first category, the second to the second category, and so on, with the final number used for the “Other” category, that is, all cases that do not fall into any of the categories explicitly defined in the variable.

QBAL recomputes the actual proportions based on the sum of the values in each numberset, so proportions, percentages or projected counts can be used.

5.2 Variables

A variable is a set of instructions that defines how a balancing dimension will be broken into mutually exclusive categories (called classes). Each QBAL variable begins with a VAR statement and closes with an END statement. The lines in between specify the classes.

The **VAR** statement begins with the keyword VAR followed, after one or more spaces, by a name up to 8 characters long, the LABEL keyword (which may be abbreviated as L) and the location of the spacer column prefixed by the letter "S". The letter "G" appended to the spacer column definition tells QBAL to use the actual specification for each class as its label.

```
VAR <name>  LABEL  Snn[G]
```

Each spec line of a variable is divided into two distinct parts separated by the spacer column. The left-hand side is used to specify the condition that must be met for a case to be included in a class, using the QUIP specification language (see section 6). The right-hand side is used to provide a text label up to 24 characters long for the class. Unless labels are generated from specifications (by appending "G" to the spacer column in the VAR statement), a label must be provided for each class specified or QBAL will stop with spec errors.

Each individual specification begins in column 1 of a line and may extend up to, but not into, the spacer column. Specifications too long to fit in the available space may be broken and wrapped to the next line with a backslash (\). Embedded blanks and tabs are not allowed.

Users of QTAB, the QUIP System cross-tabulation component, will recognize QBAL variables as simplified QTAB stub variables without the formatting, subtotal and volumetric features.

5.3 Using Numbersets and Variables

Each dimension for sample balancing purposes is defined by a numberset and variable pair in the spec file. Each class defined in the variable must correspond to a value in the numberset.

Cases can be excluded altogether from sample balancing by means of a BASE definition (see section 4.3), but cannot be excluded from a single dimension of a QBAL run. Classes must be mutually exclusive and must not be empty, since it is not possible to weight a zero value to a non-zero value. Likewise, a numberset should have no zero values matched to classes defined in the variable. The only exception to this is the "Other" class automatically added to the end of each variable in which cases that are not counted in any of the specified classes are counted. In general, one should define classes so as to account for all cases in the data file and end the numberset with a zero to match the "Other" class.

In the following example, a variable defining age groups in a sample of adults only is matched to percentages of the total population from a Census publication. Because QBAL recomputes the proportion of each class based on the sum of the values in the numberset, the weighted results for each class will actually reflect the proportion of adults, rather than of the total population.

```
NS      AGE      25.4  35.4  12.7  0
VAR     AGE            L S24
121c2r'18;34'           Ages 18 to 34
121c2r'35;49'           Ages 35 to 49
121c2g'50'              Ages 50 and over
END
```

6 THE QUIP SPECIFICATION LANGUAGE

The QUIP specification language was designed to allow nearly any condition to be described in nearly any kind of data. QBAL reads data sequentially, applying the instructions in the spec file to each record and counting the cases that meet each condition specified.

6.1 Data Specifications

This section describes the basic elements of the QUIP specification language and how they relate to the contents of a data record. Section 6.2 will show how to use these elements to specify test conditions in QBAL specifications.

6.1.1 Addressing Data

QBAL addresses data using absolute column locations within each record. The location may be specified directly as the number of columns offset from the beginning of each record or using the card/column notation favored by many statisticians and marketing researchers. Data may also be defined in character mode (e.g., ASCII), where each column occupies a single byte, or in column-binary mode, where each column occupies two adjacent bytes.

QBAL provides four addressing modes to handle the possible combinations (see section 4.1).

In address modes C and D, a column occupies a single byte, while in address modes B and E, a column occupies two bytes. Address modes C and E use direct offset notation, so in address mode C, column x is byte x of the record, while in address mode E, column x consists of the two consecutive bytes 2x-1 and 2x.

In card/column notation, the column number within the card is a two-digit number ranging from 01 to 80 and must be preceded by a card number. The first 80 columns of a record are written as 101 through 180, the next 80 columns as 201 through 280, and so on. The data in columns 180 and 201 would be addressed as columns 80 and 81 in direct offset notation.

QBAL interprets data locations according to the current address mode automatically. Any data type can be used in any address mode but even-numbered bytes cannot be addressed directly in column-binary address modes since addresses correspond to 2-byte column locations.

6.1.2 Fields and Numbers

A data address followed by one of the codes listed below will be treated as a field or a number. QBAL recognizes the following field or numeric codes (either upper or lower case):

An Column number or field, 'n' digits or columns long, where column depends on address mode. This translates to Bn if the address mode is B or E, and to Cn if the address mode is C or D.

Cn Character number or field, 'n' digits or characters long

Bn Column-binary number or field, 'n' digits or columns long

Xn Extended column-binary number, 'n' digits long, where an 'X' punch repeated 'n' times represents 10 to the power n (e.g. for X2, 'XX' = 100).

H Short, signed integer (sometimes called a 'halfword')

I Short, unsigned integer

F Long, signed integer (sometimes called a 'fullword')

G Long, unsigned integer

Q Single-byte, signed integer (sometimes called a 'quarterword')

R Single-byte, unsigned integer

E Single-precision floating point (occupies 4 bytes)

D Double-precision floating point (occupies 8 bytes)

Pn Packed decimal number 'n' digits long; 'n' is an odd number between 1 and 15

Fields (codes A, B, C, X) may be from 1 to 15 characters long for numbers or field comparisons. Column-binary fields (A, C) may be up to 80 columns long and character fields (A, C) may be up to 250 bytes long when testing for their being blank or non-blank.

Leading blanks are allowed in numeric fields, embedded blanks and non-numeric characters are not, except for a leading minus sign to indicate a negative value. To handle situations when numeric fields may contain non-numeric values, codes A, B, C and X may be followed by a suffix indicating how to handle special conditions: "B" indicates that blanks are to be given the value zero, and "E" indicates that any non-numeric value be given the value zero (e.g., 101be2 represents a 2 column column-binary field in which any non-numeric entry would translate to 0).

Numeric fields of data type A may have an additional suffix "X" that works like the data type X (extended column-binary) for either column-binary of ASCII data. In ASCII data, an X-punches are represented in data by minus signs is used in data for X-punches, so for a field defined as 21AEX3, an entry of "---" would translate to 1000, "--1" to zero, and "-10" to minus 10.

6.1.3 Columns and Punches

An address with no field code identifies a single column that contains "punch" data and may contain only the values that correspond to the punches in a column of a Hollerith (or IBM) card. The punches are 1 through 9, 0, X and Y, in that order. In ASCII modes, the X punch refers to a minus sign (-) and the Y punch to an ampersand (&) in the actual data.

In column-binary data, a column may contain multiple punches in any combination. This allows as many as 12 different codes to be stored in a single 2-byte column in column-binary data, as compared to the 12 bytes that would be required to store the same information as ASCII. For this reason, column-binary is often used for data sets containing many multiple response items.

6.1.4 Literals, Constants and Numeric Expressions

Literals are constant values used in comparisons and usually follow a logical operator. They are indicated in specs by single quotes. QBAL treats literals differently depending on the context in which they appear. In column tests, they are punch lists; in numeric tests, literals are numeric values and in field comparisons they are character strings. In some situations, multiple literals may be specified within a pair of single quotes (see section 6.2).

Constants are values used in numeric expressions and are indicated by prefixing a number by the letter "K", thus K132 represents the constant value 132.

Numeric expressions are built from numeric fields and constants using the four standard arithmetic operators: **+**, **-**, ***** and **/** for addition, subtraction, multiplication and division. Parentheses **( )** may be used to provide precedence among operators (up to 15 levels).

QBAL converts all numeric values internally to double-precision for evaluation or computation, so spec writers do not need to be concerned with mixing data types or multiplying values to preserve their precision.

6.2 Counting Cases

QBAL reads data sequentially, one record at a time, into a work area and applies the specs to the contents. A case is counted by QBAL when specified conditions test true. Field and numeric tests require an operator. Punch tests apply to single columns and no operator is used. Complex conditions may be created by using Boolean operators to combine simple tests.

6.2.1 Field and Numeric Tests

The following operators test a field for the conditions indicated:

B	Test field for being Blank (blank means that the field contains all spaces for ASCII data, all nulls for column-binary data).
P	Test field for being Packable (the field contains a valid numeric value).
Q	Test the field for being either Packable or Blank.
U	Test the field for being Unpackable (the field is not a valid numeric value).

The following operators test the numeric value of a field against one or more number literals:

='nn[;mm]	Equal to a numeric value. A maximum of 4 numeric values may be specified for the literal, separated by semi-colons.
G'nn'	Greater than or equal to a numeric value.
L'nn'	Less than or equal to a numeric value.
>'nn'	Greater than a numeric value.
<'nn'	Less than a numeric value.
R'nn;mm'	Falls within a numeric Range (from;to).

The following operator compares the contents of an ASCII field to one or more string literals:

@[-n]'mm'	Character string compare [optionally for "n" contiguous fields] (used with ASCII data only). Accepts up to 4 literals separated by semi-colons, all of which must be of the same length as the field being compared.

The following operators test the value of an ASCII field against the ASCII string value of literals (the relative value of an ASCII string is determined by its sort sequence):

@**G**'aa'	Greater than or equal to an ASCII string value.
@**L**'aa'	Less than or equal to an ASCII string value.
@**>**'aa'	Greater than an ASCII string value.
@**<**'aa'	Less than an ASCII string value.
@**R**'aa;bb'	Falls within a Range of ASCII string values (from; to).

The sense of all field and numeric tests may be reversed (negated) by the "Not" operator **N**. For field and numeric tests, the N is placed immediately in front of the operator. For the field compare test, the N is placed immediately before the literal, outside the quote. For string value tests, the N is placed between the @ sign and the comparison operator.

6.2.2 Punch Tests

Punch tests are specified by a column address followed immediately by a literal containing a list of punches to be tested for. The list of punches is called a punch mask and the test is satisfied if any of the punches in it are present in the column. A range of consecutive punches may be specified using a hyphen (-), so '1-Y' is equivalent to '1234567890XY' and '1-39-X' is the same as '12390X'. Note the order of punches in the mask (see section 6.1.3). Punch tests may test for the absence of punches by prefixing either the literal or the punch mask itself with an N.

For column-binary data only, a column may be tested for the number of punches it contains by using the tally operator:

T<op><n>'pp'	Where <op> is a numeric comparison operator (=,<,>), n is a number from 0 to 12 and 'pp' is a list of the punches to be counted in the tally.

6.2.3 Boolean Operators and Compound Tests

The following Boolean operators are used to combine conditions into compound tests:

&	Logical AND: The result is true if, and only if, both conditions are true. (Ampersand)
!	Logical OR: The result is true if either condition, or both, are true. (Exclamation point)
( )	Parentheses: The entire expression within a matching pair is treated as a single condition or value in a Boolean or arithmetic expression.
{ }	Braces: Allows an arithmetic expression to be used in a test as if it were a single numeric field in a test condition.

The maximum length of a QBAL specification is 500 bytes, and expressions may be nested in parentheses up to 15 levels deep, so very complicated conditions may be specified. There is no Boolean "not" operator, so compound expressions cannot be negated as a whole.

Any numeric field or arithmetic expression that is not generate a valid number will cause the entire specification in which it appears to immediately evaluate as false.

6.2.4 Examples of Test Conditions

The following examples illustrate data tests using the QUIP specification language. Note that tests on single column fields and punch tests can almost always be used interchangeably, but the punch tests are often easier to specify and are more readable:

11'125'	Test for punches 1, 2 or 5 in column 11
11C1R'1;2'!11C1='5'	Same result, using value tests
11C1R'1;5'&11N'3;4'	Same result, different logic
21C5='125'	Test a 5 byte field for the numeric value 125 (the test will be true for 125, 0125, 00125)
21C5@'00125'	Test a 5 byte field for the string "00125" (the test will be false for " 125" or " 0125")
21C3B	Test for a 3-column field being blank
21C3P	Test for a valid number in a 3-column field
101'1-8'	Test for the presence of punches 1 through 8 in column (any combination will satisfy this in column-binary data)
101N'1-8'	Test for the absence of punches 1 through 8 in column (any punches other than 1-8 may be present)
101'N1-8'	Same as the preceding
101T=1'1-8'	Test for the presence of exactly 1 punch in the range 1-8 in column (column-binary data only)
101T>2'1-8'	Test for the presence of more than 2 punches in the range 1-8 in column (column-binary data only)
{21C3+24C3}='100'	Test that the sum of the values in the 3-column fields beginning in 21 and 24 is equal to 100
{21CB3+24CB3}='100'	Same as above, but set the value to a blank field to zero so that it does not cause the entire specification to fail.
21C3@-3'100'	Test for the character string "100" beginning in columns 21, 24 or 27
21C1@R'A;Z'	Test for any uppercase letter in a 1 column field
121X2='100'	Test for a value of 100 in a 2 column field in column-binary data (represented by an X punch in both columns)
121AX2='100'	Same as above, but for either column-binary or ASCII data, depending on the address mode

7 DICTIONARIES

A Dictionary is a lookup table by which labels are assigned definitions that will be substituted for them whenever referenced in the spec file. Dictionary substitutions may be used anywhere in the spec file, including in subsequent dictionary definitions. A QBAL spec file may contain any number of dictionaries anywhere before the table definition section.

7.1 Dictionary Labels and Definitions

A dictionary begins with a DICT statement and ends with an END statement. Every line between them, except for comments and blank lines, must be a dictionary entry beginning with a label in column 1, followed by two definitions separated by spaces and enclosed by double quotes ("). All text after the second definition is treated as a comment and ignored, although it will appear in the listing file. The format is:

DICT
<LABEL> <"Definition 1"> <"Definition 2">
...
END

For each entry, the contents of the first definition will be substituted whenever the label appears in the specification area of a variable or weight definition. The contents of the second definition will be substituted for the label whenever it appears anywhere else in the spec file.

The contents of a definition are only checked if and when a substitution is made in the spec file. A dictionary definition may refer to another dictionary item previously defined in the spec file.

Labels are case sensitive and may be of any length, terminated by a space. Labels are read sequentially from left to right and each label must be unique in a spec file. QBAL checks labels for uniqueness as it compiles the spec file: if ABC is a label, then ABCD will be rejected, because it will be recognized as ABC after the first three characters have been read, whereas ABD will be accepted, because AB was not recognized as a label.

7.2 Using Dictionary Substitutions

Once a dictionary entry has been defined, it may be used anywhere in the spec file by entering its label, prefixed by a double pound sign (##), exactly where the substitution is desired.

Substitutions take place immediately, as the spec file is compiled, and work exactly as if the contents of the definition had been entered where the label appears in the specs. The following example illustrates how dictionary entries are defined and substitutions are called for in specs:

```
DICT
STATE      "62"     "GEOGRAPHIC LOCATION"
EAST       "01"     "EASTERN REGION"
SOUTH      "456"    "SOUTHERN REGION"
CENTRAL    "23"     "CENTRAL REGION"
WEST       "78"     "WESTERN REGION"
END

NS  REGION    19.14    23.27    35.31    22.29    0
VAR REGION             L S24
##STATE'##EAST'         ##EAST
##STATE'##SOUTH'        ##SOUTH
##STATE'##CENTRAL'      ##CENTRAL
##STATE'##WEST'         ##WEST
END
```

The result of the preceding would be exactly as if the variable REGION had been written as:

```
VAR REGION            L S24
62’01’                 EASTERN REGION
62’456’                SOUTHERN REGION
62’23’                 CENTRAL REGION
62’78’                 WESTERN REGION
END
```

Note how the first definition is always used when a substitution is made on the specification side of a variable definition line, while the second definition is always used in the label area.

QBAL parses the text immediately following the ## keyword for a valid label, and continues until it finds the longest one so far in its dictionary as it processes the spec file. This allows labels to be concatenated with additional text, or even other labels, to determine which dictionary entry will be used for a substitution.

7.3 Built-in Dictionary Labels

QBAL provides several built-in dictionary labels for which the definitions are generated by the program when it runs. These are:

##SYS_DATE	The current date at the time of execution, as it appears at the top of the listing, in the format: Mth dd yyyy (e.g., Dec 16 1998)
##SYS_TIME	The time at which QBAL began execution, as it appears at the top of the listing, in the format: hh:mm (e.g., 15:30)

These built-in dictionary entries have the same contents for both spec and text definitions.

7.4 Command Line Parameters

QBAL provides for pairs of parameters and their arguments to be passed to the program on the command line. Each parameter becomes a dictionary entry label, and its argument is used as the contents of both spec and text definitions for that dictionary entry. The format is:

QBAL specfile **/# Parameter1 Argument1 Parameter2 Argument2 .../#**

The following example demonstrates how a command line parameter can be used with a table of dictionary entries to select the output format of a QBAL run:

```
QBALD QBDEMO.QBS  /# VERSION A /#              <--- Command line
...
DICT
PROJ_A  ""  " 25 25 25 25 0”
PROJ_B  ""  " 40 30 20 10 0”
END
...
NS  PROJECT  ##PROJ_##VERSION
...
```

In this example, the argument for the command line parameter "VERSION" is "A" which will be substituted for ##VERSION in "##PROJ_##VERSION " in the numberset definition. QBAL will parse this to recognize the dictionary label PROJ_A and use the text definition for that dictionary to complete the numberset definition. When using command line parameters in this manner, care must be taken that any argument supplied is accounted for in the specs, otherwise the reference will not resolve, causing a fatal error at run-time.

8 FLOW CONTROL AND BATCH PROCESSING

QBAL is strictly script-driven, as are most of the programs in the QUIP System. It is designed for a production environment and optimized for repetitive processing tasks. QBAL can also be used as a tabulation engine for "user friendly" systems built with environments or languages that can generate text instructions and run external processes, such as FoxPro, Delphi, Visual Basic, and most standard programming languages.

Through judicious use of dictionaries and templates, it is possible to design "generic" spec files that can be used for a number of similar projects, needing only minimal information to be added or modified for each individual set of tabulations. QBAL provides flow control features to assist with batch processing and process automation situations, and allow building modular libraries of spec files that can be selected and included into other files at run-time.

8.1 Includes

The include statement allows a spec file to call another spec file. QBAL processes an included spec file as if its entire contents appeared in the calling file beginning at the line containing the include statement, and then returns to processing the calling file at the next line. Included spec files can in turn call other spec files, but the flow of specs will always return through all nested levels to the original file.

An include statement begins with the keyword #INCLUDE in column 1, followed by the name of the spec file to be included. The format is:

#INCLUDE <[path]specfile.ext>

An include statement must always follow an END statement and any spec file that will be called by an include statement must also be terminated by an END statement, so an include cannot appear within a variable definition or a dictionary.

8.2 Stopping and Starting Specs and Listings

QBAL provides special statements that allow the flow of spec processing to be turned on or off when reading a spec file, and to suppress the output of those specs processed to the listing file. The flow control statements all consist of a single beginning in column 1 with a keyword and, if allowed, its parameter, with no other text on the line. They are:

STOPSPEC	Stop processing the spec file from this point until a STARTSPEC statement is read.
STARTSPEC	Resume processing the spec file if it has been stopped by a STOPSPEC statement.
STARTSPEC IF <##X=Y>	Resume processing the spec file if it has been stopped by a STOPSPEC statement and if the definition of dictionary entry X matches the character string "Y".

STOPLIST	Stop sending any output to the listing file from this point until a STARTLIST statement is read.
STARTLIST	Resume sending output to the listing file if it has been stopped by a STOPLIST statement.

All of these statements must appear after an END statement and take effect immediately and unconditionally, except for STARTSPEC IF, which applies only if the condition evaluates true. STARTSPEC IF may be used with any dictionary entry, however it is most effective when used with command line parameters (see section 7.4) to control inclusion or exclusion of sections of spec files at run time.

In the following example, one or the other of two spec files (or neither) will be included in the calling spec file depending on the value of the dictionary entry labeled RUNCODE:

```
STOPSPEC
STARTSPEC IF ##RUNCODE=A
#INCLUDE subfile1.qbs
END
STOPSPEC
STARTSPEC IF ##RUNCODE=B
#INCLUDE subfile2.qbs
END
STARTSPEC
```

9 WHERE TO GET ADDITIONAL INFORMATION

For information about QBAL and other QUIP System programs, contact:

Jan Werner Data Processing

www.jwdp.com

10 SAMPLE QBAL LISTING

```
TCA QBAL Version 1.50 Release 99068 S/N 00002
Copyright (C) Trewhella, Cohen & Arbuckle, Inc., 1987-1999.
All rights reserved.  Reproduction in whole or in part without written
permission of the copyright owner is prohibited.
C:\BIN\QBAL.EXE qbaldem1.qbs
Start Time: 00:20:57 Sat Apr  3 1999
Processing Spec File: qbaldem1.qbs

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* QBTEST01.QBS - 03/28/99 - REV. 00 -- QBAL Test - 01
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

INTAP    qt32demo.dat  ( 80 v
OUTAP1   qbaldem1.bal  ( 80 v
LISTING  qbaldem1.lst

**** Dictionary definitions used later in specifications
DICT
PERIOD     "02a3"   "PERIOD"
CURRENT    "712"    "DECEMBER, 1997"
PRIOR      "706"    "JUNE, 1997"
STATE      "62"     "GEOGRAPHIC LOCATION"
EAST       "01"     "EASTERN REGION"
SOUTH      "456"    "SOUTHERN REGION"
CENTRAL    "23"     "CENTRAL REGION"
WEST       "78"     "WESTERN REGION"
END

SETUP   ADDRMODE C
RUNOPS  RESULT 071C5.3

**** Balance only those cases for which state code is answered
BASE    STATE          L B24
62c1nb                  Answered State

NS  REGION    19.14    23.27    35.31    22.29    0
VAR REGION             L S24
##STATE'##EAST'         ##EAST
##STATE'##SOUTH'        ##SOUTH
##STATE'##CENTRAL'      ##CENTRAL
##STATE'##WEST'         ##WEST
END

NS  VERSION   50  50   0
VAR VERSION            L S24
1'1'                    Group 1
1'2'                    Group 2

NS  PERIOD    75  25   0
VAR PERIOD             L S24
##PERIOD='##CURRENT'    ##CURRENT
##PERIOD='##PRIOR'      ##PRIOR
```

```
END
```

```
Input File Record Length: 80
Run Type: Normal
Case Count: 1062
Total Weight:          1062.000000
Projected Weight:      1062.000000
Pass #:   1   RMSRE:   43.456%
Pass #:   2   RMSRE:    2.455%
Pass #:   3   RMSRE:    0.115%
Pass #:   4   RMSRE:    0.001%
```

Question: 1

		CONTROLS				
	INPUT				RIM	OUTPUT
CLASS	FREQUENCY	INPUT	ADJUSTED	PROJECTED	WEIGHT	FREQUENCY
EASTERN REGION	254.000000	19.140000	203.246475	203.246475	0.800276	203.270000
SOUTHERN REGION	291.000000	23.270000	247.102690	247.102690	0.849000	247.059000
CENTRAL REGION	232.000000	35.310000	374.954705	374.954705	1.616086	374.932000
WESTERN REGION	285.000000	22.290000	236.696130	236.696130	0.830646	236.734000
OTHER	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
	1062.000000	100.010000	1062.000000	1062.000000		1061.995000

Question: 2

		CONTROLS				
	INPUT				RIM	OUTPUT
CLASS	FREQUENCY	INPUT	ADJUSTED	PROJECTED	WEIGHT	FREQUENCY
Group 1	586.000000	50.000000	531.000000	531.000000	0.906169	531.015000
Group 2	476.000000	50.000000	531.000000	531.000000	1.115504	530.980000
OTHER	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
	1062.000000	100.000000	1062.000000	1062.000000		1061.995000

Question: 3

		CONTROLS				
	INPUT				RIM	OUTPUT
CLASS	FREQUENCY	INPUT	ADJUSTED	PROJECTED	WEIGHT	FREQUENCY
DECEMBER, 1997	528.000000	75.000000	796.500000	796.500000	1.508542	796.510000
JUNE, 1997	534.000000	25.000000	265.500000	265.500000	0.497163	265.485000
OTHER	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
	1062.000000	100.000000	1062.000000	1062.000000		1061.995000

```
ROOT-MEAN-SQUARE RELATIVE ERROR:      0.001%
SQUARE-ROOT OF DESIGN EFFECT:          1.20
STATISTICAL EFFICIENCY:                70.0%

MAXIMUM BALANCING WEIGHT:            2.9090
MINIMUM BALANCING WEIGHT:            0.3370
RATIO:                                 8.63

MAXIMUM WEIGHT:                    2.909000
MINIMUM WEIGHT:                    0.337000
RATIO:                                 8.63
Stop Time: 00:20:57 Sat Apr  3 1999
Elapsed Time:  0 Minutes    0 Seconds
QBAL End Of Run
```