WORKING DRAFT WORKING DRAFT WORKING DRAFT

RDB: a Relational Database Management System

Walter V. Hobbs

$Id: RDB.er,v 1.33 1992/04/14 15:04:45 hobbs Exp $

I. INTRODUCTION

A good question one could ask is "With all the relational database management systems available today, why do we need another one?" There are five reasons. They are:

1. RDB is easy to use by non-computer people. The concept is straight forward and logical. To select rows of data, the ’row’ operator is used; to select columns of data, the ’column’ operator is used.
2. The data is highly portable to and from other types of machines, like Macintoshes or MSDOS computers.
3. The system will run on any UNIX machine (that has the PERL Programming Language).
4. The system can work on intermediate data, which will later be put into a commercial RDBMS, like INGRES.
5. RDB essentially has no arbitrary limits, and can work where INGRES can’t. For example there is no limit on data field size, the number of columns, or file size.

A more through discussion of why this type of relational database structure makes sense is found in the book, "UNIX Relational Database Management", Reference #2.

It is assumed that the reader has at least a minimum knowledge of the UNIX Operating System, including knowledge of Input/Outout redirection (e.g., STDIN, STDOUT, pipes).

The RDB system was implemented in the PERL programming language on a SUN Sparc I computer.

This document presents information in the following order: The DATA section describes the structure of the data, with examples. There is a general discussion about operators in the section on OPERATORS, followed by several sub-sections, one for each operator in alphabetic order. Each has detailed instructions for use, and examples. There are sections describing selection of information using multiple operators, producing reports, and generating new rdbtables (data files in RDB format).

II. DATA (RDBTABLE)

The data is contained in regular UNIX, ASCII files and so can be manipulated by regular UNIX utilities, e.g. ls, wc, mv, cp, cat, more, less, head, editors like the RAND editor ’e’, RCS, etc. A good way to view the data of course, would be to use the RDB operator ’ptbl’.

The relation, or table structure is achieved by separating the columns with ASCII TAB characters, and terminating the rows with ASCII NEWLINE character. That is, all rows in a file of data contain data separated by TAB characters and terminated with NEWLINE characters. Therefore the data must not contain TAB characters.

The header of the structure, the first two rows of the file, contain structure definition information used by the operators. The rest of the file, called the body, contains the actual data values. A file of data, structured as defined above, is said to be an ’rdbtable’.

The fields in the first row of the header contain the names of each column, and the fields in the second row contain the data definitions and optional documentation for each column. The column names are case sensitive, i.e. ’COUNT’ is different from ’Count’.

The guideline for which characters should be used in column names is that alphabetic, numeric, and non-alphanumeric characters that are not special to the UNIX shell are good choices. Column names should include at least one alphabetic character.

Non-alphanumeric characters that work well in column names are the underscore (_) percent sign (%) colon (:) at sign (@) equals (=) comma (,) dot (.) dash(-). The sharp sign (#) is allright if it is not the first character in a column name.

The TAB character is never to be used in column names, nor should internal spaces or UNIX I/O redirection characters (<,>,|) be used.

The data definitions include column width, data type, and justification. The column width must be specified; the others are optional. The data definitions take the form of adjacent characters in a single word. Note that it is not a serious error if some data in a column is wider than the defined width. A listing produced with ’ptbl’ might be slightly out of line however.

The width of each field is specified by a numeric count.

The type of data is "string", "numeric", or "month". The types are specified by an ’S’, ’N’, or ’M’ respectively. Default is type string.

Printout justification, for use by the ’ptbl’ operator, is ’left’, or ’right’, and is specified by an ’<’ or ’>’ character respectively. If not specified, types string and month will be left justified and type numeric will be right justified.

The optional documentation for each column follows the data definition word. There must be one or more space characters after the data definition and before the documentation. For more detail and an example of column documentation see "COLUMN DOCUMENTATION" in section V. "GENERATING NEW RDBTABLES".

A sample rdbtable (named sample) that will be used in several later examples is shown in Table 1. The picture in Table 1 is for illustrative purposes; what the file would actually look like is shown in Table 2, where a TAB character is represented by ’<T>’ and a RETURN character is represented by ’<R>’. It is important to note that only actual data is stored in the data fields, with no leading or trailing space characters. This fact can have a major effect on the size of the resulting files.

Table 1

RDBTABLE (SAMPLE)

NAME	COUNT	TYP	AMT	OTHER	RIGHT
6	5N	3	5N	8	8>
Bush	44	A	133	Another	This
Hansen	44	A	23	One	Is
Jones	77	X	77	Here	On
Perry	77	B	244	And	The
Hart	77	D	1111	So	Right
Holmes	65	D	1111	On	Edge

Table 2

RDBTABLE (SAMPLE) ACTUAL CONTENT

```
NAME<T>COUNT<T>TYP<T>AMT<T>OTHER<T>RIGHT<R>
6<T>5N<T>3<T>5N<T>8<T>8><R>
Bush<T>44<T>A<T>133<T>Another<T>This<R>
Hansen<T>44<T>A<T>23<T>One<T>Is<R>
Jones<T>77<T>X<T>77<T>Here<T>On<R>
Perry<T>77<T>B<T>244<T>And<T>The<R>
Hart<T>77<T>D<T>1111<T>So<T>Right<R>
Holmes<T>65<T>D<T>1111<T>On<T>Edge<R>
```

III. OPERATORS

The operators are separate program modules that each perform a unique function on the data. They can be grouped into data movers, report generators, and utilities.

The data movers are operators that extract or rearrange the data in some way. They each read an rdbtable via STDIN and write a rdbtable via STDOUT and so are frequently connected using the UNIX pipe function to form a larger task. Each operator in such a "pipeline" style of operation gets its input from the output of the previous operator in the "pipeline". The data movers include:

- row - Selects rows based on arbitrary expressions.
- column - Selects columns by name, outputs columns in listed order.
- sorttbl - Sorts a datafile by one or more columns.
- jointbl - Natural or "Master/Detail" join of two datafiles.
- compute - Computes an arbitrary expression using column names.
- subtotal - Lists subtotals of specified columns.
- uniqtbl - Makes an rdbtable unique on specified columns.

The report generators each read an rdbtable via STDIN and produce a report on STDOUT, so when they are in a "pipeline" of operators they will be the operator at the end. The report generators are:

- ptbl - Quick and easy printing of output formatted from information in the header.
- reporttbl - Best form of output, with definable format.
- summ - Summary information about columns of an rdbtable.

The utilities are used for manipulating the structure and content of rdbtables and are generally used as separate tasks. The utilities are:

- headchg - Generates and replaces (or removes) the header of an rdbtable.
- dataent - An interactive capability for entering data into an rdbtable.
- etbl - Uses an editor to allow modifications to an rdbtable.
- valid - Verifies the structure of an rdbtable.
- repair - Attempts to repair candidate RDB datafiles.

All operators take a '-help' option to show details of operation online. Following is a sub-section for each operator, in alphabetic order.

COLUMN

Usage: column [options] list

Selects columns by name (and order) and outputs an rdbtable with these columns. Can effectively select, order, add, delete, or duplicate columns.

The value 'list' is normally a list of column names. If 'list' contains a triplicate of the form '-c NAME NEW' then column name 'NAME' will be changed to 'NEW'. If 'list' contains a triplicate of the form '-a NAME DEFN' then a new (null) column is added, at that point in the list of column names, with name 'NAME' and definition 'DEFN'.

This RDB operator reads an rdbtable from STDIN and writes an rdbtable to STDOUT. Options may be abbreviated.

Options:

-edit	Edit option. Used by etbl.
-help	Print this help information.
-v	Inverse option. Selects all columns except those named.

As an example using the sample rdbtable from the DATA section (named sample), to select columns named ’NAME’ and ’COUNT’ the command would be:

```
column  NAME  COUNT  <  sample
```

To select all columns except column ’NAME’ the command would be:

```
column  -v  NAME  <  sample
```

To add a new column named ’LENGTH’ with a size of 10 the command would be:

```
column  -v  -a  LENGTH  10  <  sample
```

Note that to include documentation with the new column definition the command would be:

```
column  -v  -a  LENGTH  ’10 length in meters’  <  sample
```

The ’10 length in meters’ must be quoted so that it will be treated as a single token.

COMPUTE

Usage: compute [options] [statements]

Computes values for data fields based on arbitrary statements using column names. Any characters that are special to the UNIX shell must be quoted.

Comparison operators may be of the form: gt, ge, lt, le, eq, ne. For example ’NAME eq Hobbs’. Logical constructors ’or’ and ’and’ may be used; as well as ’null’ to indicate an empty data value. The supplied statements may be essentially any valid PERL statements.

All of the Comparison operators and Logical constructors are reserved and should not be used as column names (they are all lower case and four characters or less).

Options:

-help Print this help information.

-fXXX The statements are in the file 'XXX', instead of on the command line. The advantage in this case is that no quoting of characters that might be special to the UNIX shell is necessary.

This operator reads a rdbtable via STDIN and writes a rdbtable via STDOUT. Options may be abbreviated.

If a file is used to contain the statements any line in the file that starts with a sharp sign (#) is treated as a comment and ignored. Also if there is a sharp sign preceded by a space character anywhere on the line the rest of the line is also treated as a comment.

Since column names and reserved words are parsed by the program, do not put the entire expression in a single pair of quotes as that will prevent the parsing. Also note that column names and reserved words need to be surrounded by blank spaces if they are not individually quoted. For example either form below is fine:

```
row   NAME    eq   "L Brown"  <  sample
row  "NAME"  "eq"  "L Brown"  <  sample
```

but do not use this form:

```
row  "NAME  eq  L Brown"  <  sample
```

Example rdbtable (named cfile):

```
name    count   type    amt
6       5N      4       5N
Bush    3       A       133
Hansen  39      A       23
Newton  8       E       8
Hobbs   42      B       144
Hart    2       C       55
Jones   4       B       244
Smith   5       D       1111
```

The command:

```
compute  count  +=  100  if  type  lt  D  <  cfile  |  ptbl
```

gives the output:

```
name    count  type    amt
------  -----  ----  -----
Bush      103  A       133
Hansen    139  A        23
Newton      8  E         8
Hobbs     142  B       144
Hart      102  C        55
Jones     104  B       244
Smith       5  D      1111
```

Example file of commands named ’XXX’:

```
if( type eq A ){
    name = NEW ;
    amt = count * 2 ;
    type = ’AAA’ ;
}
else{
    name = OLD ;
    amt = count + 1000 ;
    type = ’ZZZ’ ;
}
```

Output from command:

```
compute  -fXXX  <  cfile  |  ptbl
```

would be:

```
name    count  type    amt
------  -----  ----  -----
NEW         3  AAA       6
NEW        39  AAA      78
OLD         8  ZZZ    1008
OLD        42  ZZZ    1042
OLD         2  ZZZ    1002
OLD         4  ZZZ    1004
OLD         5  ZZZ    1005
```

DATAENT

Usage: dataent [options] rdbtable

or: dataent [options] -init template

This utility provides an interactive capability for entering data into an rdbtable. The user is prompted by column name for data values. Options may be abbreviated.

Options:

- -help Print this help info.
- -init Initiate a new rdbtable.
- -nbr Do NOT remove leading and trailing blank space from data values.
- -prev Use data values from previous row as defaults for current row.

In the first case of usage above, new rows of data are added at the end of an existing rdbtable. In the second case, a template file is used to generate a new rdbtable and add rows of data to it.

At each column name prompt the user may enter data followed by a <RET> or just a <RET> to retain the current value, which is initially null. Normally leading and trailing blank space is removed from data values. If it is desired to prevent this enter a backslash (\) as the first character (the backslash will be removed) or use the '-nbr' option. In order to replace an existing data value with a null value enter a backslash (\) and a <RET>.

At any column name prompt, if a single space character is entered, followed by a <RET>, control is transferred to the end of row action prompt. This is useful if not all data values need to be entered on all rows. If two space characters followed by a <RET> is entered control is transferred to the previous column name prompt. This is useful if an error was made entering data; it can be corrected immediately.

After all the column name prompts for a row have been responded to, the user is asked for the next action. The default is to save the current row of data to the rdbtable and go on to enter data for the next row. Other options available are to go back and check each value of the current row; to quit, saving or not saving the current row; to list the data values for the entire row; to delete the current row and start a new one; to jump back to a specified column name prompt for the current row and continue from there; or to produce a help listing. Also the setting of the '-nbr' and '-prev' options may be toggled on or off.

At any time an INTERRUPT signal (^C or DEL) may be entered to abort the program. In this case all rows of data except the current one will be saved.

Uses RDB operator: headchg.

ETBL

Usage: etbl [options] rdbtable [col_spec] [line_spec] [pat_spec]

This utility calls an editor to allow the editing of selected lines and/or columns of (or the entire) rdbtable. Options may be abbreviated.

Options:

- -eNAME   Use the editor 'NAME'.
- -help   Print this help info.
- -l[N]   Use "list" format for editing instead of "column" format. If N is given it is the line length to use (min is 50).
- -npp   No postprocessing after return from the editor. The edited file is saved with a name of the form "etbl.t.nnnn" where "nnnn" is the process number.
- -RCS   Force checkout of the rdbtable from RCS.

A "col_spec" is a list of column names.

A "line_spec" is a list of line numbers, of increasing value, optionally separated by a dash to specify a range, e.g. "10-20". The form "N-" means from line N to end of file. The header is always included, so do not specify lines 1 or 2 (except as the first part of a larger group, e.g. "1-10").

A "pat_spec" is a single pattern (of the form: /pat/) optionally followed by one or more column names, and may be preceded with the reserved word 'ne' to negate the meaning (e.g. the pattern should NOT match).

The order of "Col_spec", "line_spec", and "pat_spec" is significant only to the extent that "col_spec" must precede "pat_spec" in the command line if both are given.

If none of "col_spec", "line_spec", or "pat_spec" are given then the entire rdbtable will be edited. If one or more of the three above options are given then the selected subset of the rdbtable will be edited. The option "col_spec" identifies which columns of the rdbtable are to be edited, and options "line_spec" and "pat_spec" determine which lines will be selected for editing, either by direct reference ("line_spec" given "col_spec" not given) or by pattern matching ("col_spec" given "line_spec" not given). If both "line_spec" and "pat_spec" are given then only lines within the bounds of "line_spec" will be considered for selection by pattern matching.

If "pat_spec" does not include column names then the pattern (any PERL regular expression) is matched against each entire row; a row is selected if there is a match anywhere in the row. If column names are included the pattern is matched against only the specified columns. In this case a row is selected if a match is found in any specified column. If the "ne" option precedes the "pat_spec" without column names then an entire row is selected if the pattern does not match anywhere in the row, and if column names are given then the row is selected if the pattern does not match in any specified column.

The form of the file to be edited is either "column" with visible column delimiters (the default) or "list" format where the column names are on the left and the data is on the right. The default editor is specified by the environment variable EDITOR if set, otherwise the editor ’e’ is used.

In either form of editing the delimiter is a "pipe" symbol (|). Care should be taken when editing not to use any "pipe" symbols in the data, or to delete any existing pipe symbols in the file. Also, in the case of "list" form, one or more blank lines must separate each record.

The rbdtable may be an existing file, or it may be automatically checked out from RCS. In the latter case it will be checked back into RCS after the editing is complete.

The default action is that if the rdbtable does not exist an attempt will be made to find the rdbtable under RCS (the '-RCS' option may be used to force the use of an RCS file).

Afterward, except in the RCS case, the original contents of the rdbtable will be left in a file of the same name preceded with a comma, e.g. "sample" will be ",sample".

Uses RDB operators: column, ptbl, mktbl, tbl2lst, lst2tbl.

WARNING: If line_spec is given the number of columns must not be changed by editing, or if col_spec and/or "pat_spec" is given the number of lines must not be changed by editing, otherwise the results may be unpredictable.

An example command to edit the rdbtable (named sample) from the DATA section would be:

```
etbl  sample
```

which would edit the entire rdbtable. The file as it is ready to edit is shown in Table 3. The pipe character '|' must not be removed during the editing process, although it may be moved left or right if necessary and the spaces around the pipe character may be deleted if desired.

This form of editing is fine if the rdbtable is not large. If it is large then editing only those parts that need changes is faster and less error prone. To edit only columns 'NAME', 'COUNT', and 'AMT', the command would be:

```
etbl  sample  NAME  COUNT  AMT
```

To edit only lines five thru seven the command would be:

```
etbl  sample  5-7
```

Table 3

RDBTABLE (SAMPLE) READY TO EDIT, COLUMN FORM

NAME	COUNT	TYP	AMT	OTHER	RIGHT
6	5N	4	5N	8	8>
Bush	44	A	133	Another	This
Hansen	44	A	23	One	Is
Jones	77	X	77	Here	On
Perry	77	B	244	And	The
Hart	77	D	1111	So	Right
Holmes	65	D	1111	On	Edge

To edit only lines five thru seven of only columns 'NAME', 'COUNT', and 'AMT' the command would be:

```
etbl  sample  NAME  COUNT  AMT  5-7
```

and the file to edit would look like:

```
..>>> 1 2 CONTROL LINE, DO NOT TOUCH <<<
NAME   | COUNT |   AMT
6      |    5N |    5N
..>>> 5 3 CONTROL LINE, DO NOT TOUCH <<<
Jones  |    77 |    77
Perry  |    77 |   244
Hart   |    77 |  1111
```

Note that whenever a line_spec is given, control lines (starting with '..>>>' are inserted into the file to edit. They must not be modified during the editing process. They are used to reconstruct the rdbtable after editing.

If the rdbtable has data fields that are long, i.e. longer than convenient to edit in the column form shown above, the 'list' form is the preferred method. The usage of line_spec and col_spec are unchanged but the form of the file to edit is different. For example consider an rdbtable (named sample3) which is shown in Table 4, where the TAB characters are represented by '<T>' and the newline characters are represented by <R>. This small rdbtable looks very incoherent in raw form, and a file of any real size with long data fields is even more so. The command to edit the file sample3 in 'list' form would be:

Table 4

RDBTABLE (SAMPLE3) ACTUAL CONTENT

```
name<T>datatype<T>agencysrc<T>dbms<T>contact<T>contents<T>notes<R>
46<T>15<T>60<T>15<T>21<T>530<T>600<R>
ACAS (Air Combat Assessment)<T>BDA<T>Bigplace AFB<T>File<T>Starr<T>Air
Combat Assessment BDA data.  Duplicates data under ACAS (Air Combat
Assessment) BDA Sorties, diskettes nr 1,2,3.<T>On two 3.5 inch
diskettes.<R>
ACAS (Air Combat Assessment) BDA Sorties<T>BDA<T>Sawyer AFB<T>File<T>
Hobbs/Emerson<T>85 files, 2 per day containing 12 and 24 hour reports.
This data is different from that under ACAS (Air Combat Assessment) BDA
Data, from diskettes 1,2,3.<T>Received 5/6/91.<R>
ATO (Air Tasking Orders) Original<T>ATO<T>HQ USAF, Universal AFB<T>
File<T>Marshall<T>Original ATO messages.  Both sets are incomplete.<T>
To be joined into single file and edited.  Missing sections not yet
ordered.  May be parsed completely, or only for key comments. Much data
to be processed.<R>
ABC Original<T>ABC<T>HQ USAF, Universal AFB<T>File<T>Marshall<T>Original
ATO messages.  Both sets are incomplete.<T>To be joined into single file
and edited.  Missing sections not yet ordered. May be parsed completely,
or only for key comments. Much data to be processed.<R>
```

```
etbl  -list  sample3
```

which would produce a file to edit as shown in Table 5. Note that each section holds information relating to one row in the rdbtable and that the first section holds information relating to the header of the rdbtable. Also note that each section is separated by a blank line (it could be any number of blank lines).

Each row in a section relates to a single data value. The pipe character ’|’ must not be removed during the editing process, although it may be moved left or right if necessary. Only one pipe character is to be in the information relating to one data value, although that information may be physically on more than one line in the section if the data value is long.

The spaces on both sides of the pipe character as well as the spaces around the column names are only for readability; they may be moved or even deleted if desired.

Table 5

RDBTABLE (SAMPLE3) READY TO EDIT, LIST FORM

```
     name | 46
 datatype | 15
agencysrc | 60
     dbms | 15
  contact | 21
 contents | 530
    notes | 600

     name | ACAS (Air Combat Assessment)
 datatype | BDA
agencysrc | Bigplace AFB
     dbms | File
  contact | Starr
 contents | Air Combat Assessment BDA data.  Duplicates data under ACAS (Air
            Combat Assessment) BDA Sorties, diskettes nr 1,2,3.
    notes | On two 3.5 inch diskettes.

     name | ACAS (Air Combat Assessment) BDA Sorties
 datatype | BDA
agencysrc | Sawyer AFB
     dbms | File
  contact | Hobbs/Emerson
 contents | 85 files, 2 per day containing 12 and 24 hour reports. This data
            is different from that under ACAS (Air Combat Assessment) BDA
            Data, from diskettes 1,2,3.
    notes | Received 5/6/91.

     name | ATO (Air Tasking Orders) Original
 datatype | ATO
agencysrc | HQ USAF, Universal AFB
     dbms | File
  contact | Marshall
 contents | Original ATO messages.  Both sets are incomplete.
    notes | To be joined into single file and edited.  Missing sections not
            yet ordered.  May be parsed completely, or only for key
            comments. Much data to be processed.

     name | ABC Original
 datatype | ABC
agencysrc | HQ USAF, Universal AFB
     dbms | File
  contact | Marshall
 contents | Original ATO messages.  Both sets are incomplete.
    notes | To be joined into single file and edited.  Missing sections not
            yet ordered.  May be parsed completely, or only for key
            comments. Much data to be processed.
```

The advantage of this form of edit file is that even with very large data values most, if not all, of the information from each row of an rdbtable will be visible on the screen at once.

HEADCHG

Usage: headchg [options] file.tpl

Replaces the header (first two rows) of an rdbtable with a header generated from information in the template file 'file.tpl'. Options are available to add, copy, or delete the header, or to generate a template file from an existing rdbtable.

Each line of the Template file contains info about a column, in order. The lines contain: (optional) index number (starting at 0 or 1), column name, definition, and (optional) comments or documentation, white space separated. If column name contains spaces it must be enclosed in double quotes. Names containing space characters are not recommended, however, as it is generally troublesome and error prone. A good substitute is the underscore character (_).

Lines that start with a sharp character '#' are skipped, as are blank lines. To start a column name with a sharp character '#' the name must be enclosed in double quotes. (but this is not recommended).

The number of columns in the header is normally reported on STDERR.

This operator reads an rdbtable via STDIN and writes an rdbtable via STDOUT. Options may be abbreviated. This operator uses the RDB operator: valid.

Options:

- -add   Add the header to an rdbtable instead of replacing it.
- -copy   Copies the header from 'file.tpl' instead of generating it. In this case 'file.tpl' is (at least a header of) an rdbtable, NOT a template file.
- -del   Delete the rdbtable header instead of replacing it.
- -gen   Generate header only, no rdbtable read.
- -help   Print this help information.
- -ndoc   Documentation in template file is NOT to be included in the second

line of the header (included by default).

-quiet   No messages printed on STDERR.

-rdb   Treat 'file.tpl' as an rdbtable, use data in columns two and three to make the header.

-templ   Generate a template file from the header of the Table, on STDOUT.

As an example, to generate a template file named 'new.tpl' from the rdbtable (named sample) from the DATA section, the command would be:

```
headchg  -templ  <  sample  >  new.tpl
```

The contents of file 'new.tpl' would then be:

```
0                 NAME  6
1                COUNT  5N
2                  TYP  4
3                  AMT  5N
4                OTHER  8
5                RIGHT  8>
```

To change the header of rdbtable 'sample', the procedure is to edit the file 'new.tpl', and then run 'headchg' using the modified file. For example, to change the names so that only the first letters are upper case and to make column 'OTHER' numeric, edit file 'new.tpl' so it looks like the following:

```
0                 Name  6      All names are first letter upper case.
1                Count  5N
2                  Typ  4
3                  Amt  5N
4                Other  8N     Now numeric.
5                Right  8>
```

Note the index in the zeroth column and the documentation in the fourth column, both of which are optional, but recommended. The command to change the header of rdbtable 'sample' and make a new rdbtable called 'new.sample' would be:

```
headchg  new.tpl  <  sample  >  new.sample
```

JOINTBL

Usage: jointbl [options] col.name[=col.name_2] rdbtable_2 < rdbtable_1

Does a join of two rdbtables on the column(s) specified. The default is a "natural" join, with optional "Master/Detail" or cartesian (cross-product) type joins. Options may be abbreviated.

Options:

-c	Do a cartesian (cross-product) join.
-help	Print this help information.
-md	Do a "Master/Detail" join rather than a natural join. The Table from STDIN is the master.

A natural join produces a new rdbtable that contains only rows from the input rdbtables that match on the specified columns (key columns). A master-detail join produces a new rdbtable that contains all rows from the master rdbtable and those rows from the secondary rdbtable that match. A cartesian join produces an rdbtable that contains all rows of both input rdbtables.

Each item in the list of column(s) may specify column names that are different in the two rdbtables, i.e. ’=column_2’, if given, refers to a name in rdbtable_2 that corresponds to ’column’ in rdbtable_1. If ’=column_2’ is not given it means that the corresponding column name in both rdbtables is the same.

If different column names are specified, the name of the join columns in the output rdbtable will be from rdbtable_1.

Note that the two rdbtables must be sorted on the columns specified in order for a join operation to function correctly.

The order of columns in the output rdbtable will be: first the join columns, then the other columns from rdbtable_1, then the other columns from rdbtable_2.

This operator reads an rdbtable via STDIN and writes an rdbtable via STDOUT.

If we have the rdbtable (named samplej) here:

name	nr	typ	amt
6	2	4	4
Bush	1	A	133
Bush	2	A	134
Hansen	3	A	143
Hobbs	4	B	144
Hobbs	5	B	144
Jones	6	C	155
Perry	7	D	244
Perry	8	D	311

and the rdbtable (named samplej2) here:

name	cnt	typ	amt
6	5N	4	5N
Hobbs	41	A	141
Hobbs	42	BB	142
Hobbs	51	BB	144
Hobbs	43	CC	143

then the command to do a natural join of samplej and samplej2 on column name is:

```
        jointbl  name  samplej2  <  samplej
```

and the results is shown in Table 6. The command to do a "master-detail" join of the same two rdbtables on column name is:

```
        jointbl  -md  name  samplej2  <  samplej
```

Table 6

NATURAL JOIN OF RDBTABLES SAMPLEJ AND SAMPLEJ2

name	nr	typ	amt	cnt	typ	amt
6	2	4	4	5N	4	5N
Hobbs	4	B	144	41	A	141
Hobbs	4	B	144	42	BB	142
Hobbs	4	B	144	51	BB	144
Hobbs	4	B	144	43	CC	143
Hobbs	5	B	144	41	A	141
Hobbs	5	B	144	42	BB	142
Hobbs	5	B	144	51	BB	144
Hobbs	5	B	144	43	CC	143

Table 7

MASTER-DETAIL JOIN OF RDBTABLES SAMPLEJ AND SAMPLEJ2

name	nr	typ	amt	cnt	typ	amt
6	2	4	4	5N	4	5N
Bush	1	A	133			
Bush	2	A	134			
Hansen	3	A	143			
Hobbs	4	B	144	41	A	141
Hobbs	4	B	144	42	BB	142
Hobbs	4	B	144	51	BB	144
Hobbs	4	B	144	43	CC	143
Hobbs	5	B	144	41	A	141
Hobbs	5	B	144	42	BB	142
Hobbs	5	B	144	51	BB	144
Hobbs	5	B	144	43	CC	143
Jones	6	C	155			
Perry	7	D	244			
Perry	8	D	311			

and the results is shown in Table 7.

LST2TBL

Usage: lst2tbl [options]

Converts a file in "list" format to an rdbtable. Long data fields may be folded. This operator is mainly used by other operators. Options may be abbreviated.

Options:

-edit   Edit option. Used by etbl.

-help   Print this help information.

This RDB operator reads an rdbtable from STDIN and writes an rdbtable to STDOUT.

MKTBL

Usage: mktbl [options]

Makes a file of data in columns (with visible column delimiters) into an rdbtable. The column delimiter is the pipe symbol (|). This operator is mainly used by other operators.

This operator reads a file via STDIN and writes an rdbtable via STDOUT. Options may be abbreviated.

Options:

-edit   Edit form of output, used primarily by ’etbl’.

-help   Print this help information.

PTBL

Usage: ptbl [options]

This operator used for quick and easy printing of an rdbtable, in a simple but useful form. It prints an rdbtable using formatting information from the header.

The printing of each row of data will be on one line if possible, but when multiple lines are necessary the second and later lines are indented for readability. Also when multiple lines are necessary a simple space availability algorithm is used to minimize the number of lines printed for each row of data. This may result in the order of some data values being rearranged from their order in the rdbtable. The ’-b0’ option can override this algorithm and force the same printing order as in the rdbtable.

This RDB operator reads an rdbtable from STDIN and writes a formatted report on STDOUT. Options may be abbreviated.

Options:

```
-b0       By default, when a multi-line record of output for each row
          is necessary (due to the width of the current window or terminal)
          the program will try to fill space at the end of lines that
          would otherwise be wasted by moving some columns.  This option
          prevents the moving of any columns.
-b[N]     This option attempts a "best fit" by rearranging columns (widest
          columns first).  If 'N' is given the first N columns of the first
          line will not be moved. The default condition is '-b'.
-Bigf     Handle very large data fields, e.g. over 1000 characters. This
          option takes longer but it works for any size data fields.
-edit     Edit form of output, used primarily by 'etbl'.
-fold     Fold long data fields into multi line data based on the defined
          field width.
          May be used with the '-t' option to limit the field width.  Only a
          single line record of output is produced with this option.
-help     Print this help information.
-iN       Indent size of N spaces on 2nd and later lines of a multi-line
          record of output. Default is 4 spaces.
-lN       Line length of N characters for output. Default is the width of
          the current window or terminal.
-pN       Page size in of N lines. Default is the height of the current
          window or terminal.  A value of zero '-p0' will turn paging off.
-PX[stg]  Page headings and settings for printing.  A two line heading is
          put onto each page: page number, current date, and an optional
          string (stg).  Sets page length (in lines) and line length (in
          characters) according to the value of 'X' as follows.
                 X: P  page: 60  line:   80     (default font size)
                 X: R  page: 47  line:  116     (rotated default)
                 X: A  page: 51  line:  125     (rotated 10 point font)
                 X: 8  page: 63  line:  144     (rotated  8 point font)
                 X: 6  page: 82  line:  192     (rotated  6 point font)
                 X: W  page: and line: from current window size.
          Other desired page and/or line size options may be set after
          this in the option list.
```

-sK Separator 'K' (which may be multi character) placed between columns. Default is two spaces.

-t[N] Truncate data to the width defined in the header. If N is given the width of printed fields will be further limited to N characters.

-window List as many columns as possible in single line records that will fit in the current window or terminal width.

As an example using the sample rdbtable from the DATA section (named sample), the command to view this rdbtable would be:

```
ptbl < sample
```

which would produce the output shown in Table 8. The same command with a page heading for printing:

```
ptbl  -PP  <  sample
```

produces the output as shown in Table 9. Using an rdbtable (named

Table 8

PRINTING RDBTABLE (SAMPLE) USING PTBL

```
NAME    COUNT  TYP     AMT  OTHER        RIGHT
------  -----  ----  -----  --------  --------
Bush       44  A       133  Another       This
Hansen     44  A        23  One             Is
Jones      77  X        77  Here            On
Perry      77  B       244  And            The
Hart       77  D      1111  So           Right
Holmes     65  D      1111  On            Edge
```

Table 9

PRINTING RDBTABLE (SAMPLE) WITH PAGE HEADING USING PTBL

```
Page   1   Mon Dec  2 16:56:43 PST 1991

NAME    COUNT  TYP     AMT  OTHER        RIGHT
------  -----  ----  -----  --------  --------
Bush       44  A       133  Another       This
Hansen     44  A        23  One             Is
Jones      77  X        77  Here            On
Perry      77  B       244  And            The
Hart       77  D      1111  So           Right
Holmes     65  D      1111  On            Edge
```

Table 10

RDBTABLE WITH LONG DATA VALUES (SAMPLE4) ACTUAL CONTENT

```
name<T>type<T>contact<T>contents<R>
10<T>4<T>21<T>20<R>
Hansen<T>AAA<T>R. Starr at the UCLA & USC<T>Duplicate data under
processing order number 55-7.<R>
Hart<T>CCC<T>Hobbs/Emerson at RAND Corporation<T>85 files, 2 per
day containing 12 and 24 hour reports.<R>
Hobbs<T>EEE<T>Marshall at Universal AFB<T>Original PAF messages.
Both sets are incomplete.<R>
Bush<T>KKK<T>General USAF personnel<T>Duplicate ATO messages,
incomplete.<R>
Lender<T>RRR<T>Army base in Nevada<T>Nothing.<R>
Emerson<T>UUU<T>Navy at Washington DC<T>More than we thought at
first.<R>
```

sample4) that has long data values, shown in Table 10, the command to print the rdbtable using the truncate option is:

```
ptbl  -t  <  sample4
```

which will produce output with the data values truncated to the defined column width as in Table 11. Using the same rdbtable with the fold option:

```
ptbl  -fold  <  sample4
```

produces output with the long data values 'folded' within their defined column widths as shown in Table 12. Note that each line is repeated until the entire data value for each column is completely shown. This makes this type of output variable length.

Table 11

PRINTING RDBTABLE (SAMPLE4) WITH PTBL -TRUNC OPTION

```
name        type  contact                contents
----------  ----  ---------------------  --------------------
Hansen      AAA   R. Starr at the UCLA   Duplicate data under
Hart        CCC   Hobbs/Emerson at RAND  85 files, 2 per day
Hobbs       EEE   Marshall at Universal  Original PAF message
Bush        KKK   General USAF personne  Duplicate ATO messag
Lender      RRR   Army base in Nevada    Nothing.
Emerson     UUU   Navy at Washington DC  More than we thought
```

Table 12

PRINTING RDBTABLE (SAMPLE4) WITH PTBL -FOLD OPTION

```
name        type  contact                 contents
----------  ----  ---------------------   --------------------
Hansen      AAA   R. Starr at the UCLA    Duplicate data under
                  & USC                   processing order
                                          number 55-7.
Hart        CCC   Hobbs/Emerson at RAND   85 files, 2 per day
                  Corporation             containing 12 and 24
                                          hour reports.
Hobbs       EEE   Marshall at Universal   Original PAF
                  AFB                     messages.  Both sets
                                          are incomplete.
Bush        KKK   General USAF            Duplicate ATO
                  personnel               messages,
                                          incomplete.
Lender      RRR   Army base in Nevada     Nothing.
Emerson     UUU   Navy at Washington DC   More than we thought
                                          at first.
```

If you need a quick and easy way to look at the data in an rdbtable use the -win option. This option will cause ptbl to list as many columns as possible in single line records that will fit in the current window or terminal width. Note that you do not have to type the column names (or even know them) to use this option.

It may be combined with the -t option to increase the number of columns of data shown on each line at the expense of some column width.

For example the command ’ptbl < d11c’ on an 80 character wide window or terminal produces the following:

```
name    count            type  amt    n1                        n3
------  ---------------  ----  -----  ------------------------  ------------
    n2                        n4                        n5
    ------------------------  ------------------------  ------------
    n6            n7
    ------------  ------------
Bush    3                A     133    alpha22.307               117722
    baker                     DDBBx17                   other
    124567        8GGXXH17
Hansen  39               A     23     beta222.307               117723
    charlie                   DDBBx18                   data
    1239870       GGXXH17
Newton  8                E     8      gama22.333                117724
    dog                       DDBBx19                   exists
    1239870       GGXXH17
Hobbs   42               B     144    delta3.3.118              117725
    echo                      DDBBx20                   here
    1239870       GGXXH17
Hart    2                C     55     epslion33.118             117726
    foxtrot                   DDBBx21                   also
    1239870       GGXXH17
```

This is readable, but not very nice to look at, and even worse if there are more columns. The command 'ptbl -win < d11c' produces:

```
name   count           type amt   n1                       n3
------ --------------- ---- ----- ------------------------ ------------
Bush   3               A    133   alpha22.307              117722
Hansen 39              A    23    beta222.307              117723
Newton 8               E    8     gama22.333               117724
Hobbs  42              B    144   delta3.3.118             117725
Hart   2               C    55    epslion33.118            117726
```

Not all the data is listed, but the first few columns (sometimes the most important) are easier to view. The command 'ptbl -win -t6 < d11c' shows even more of the data, at the expense of some column width:

```
name   count  type amt   n1     n2     n3     n4     n5     n6     n7
------ ------ ---- ----- ------ ------ ------ ------ ------ ------ ------
Bush   3      A    133   alpha2 baker  117722 DDBBx1 other  124567 8GGXXH
Hansen 39     A    23    beta22 charli 117723 DDBBx1 data   123987 GGXXH1
Newton 8      E    8     gama22 dog    117724 DDBBx1 exists 123987 GGXXH1
Hobbs  42     B    144   delta3 echo   117725 DDBBx2 here   123987 GGXXH1
Hart   2      C    55    epslio foxtro 117726 DDBBx2 also   123987 GGXXH1
```

REPAIR

Usage: repair [options] file ...

Attempts to repair candidate RDB datafiles, e.g. files that have been ported from a MacIntosh or PC (MSDOS computer) in spreadsheet form but that do not yet have valid rdbtable structure. Generates definition lines (second line of header). The width of all data values is checked and the maximum width for a column is used as the column width in the definition line for that Table.

It also works with existing rdbtables ('-exist' option) and is convenient for removing leading and trailing space characters from data values (-blank option).

Adds fields as necessary to rows (null), or to header (DUM1, DUM2, ...) to make the Table structure valid.

The new rdbtables will be in the current directory (even if the input files are not) and will have the suffix changed (or added) to '.rdb' by default.

Options may be abbreviated.

Options:

-blank	Remove leading and trailing blank characters from data fields.
-dN	Deletes the first N lines of each input file. Used to remove extra lines before the actual header.
-exist	The file(s) are existing rdbtables. Instead of generating new definition lines the current ones are used as starting values.
-Exist	Just like '-exist' except the data width check is not done.
-help	Print this help information.
-kWORD	The first line of each input file containing "WORD" will be considered the line containing the column names Works after the '-d' option, if given.
-rFILE	Use the template file 'FILE' to replace the existing header.
-sXXX	Suffix on new files of 'XXX' instead of '.rdb'.
-tFILE	Use the template file 'FILE' for header and definition data instead of scaning the input files.

REPORTTBL

Usage: reporttbl [options] file.frm

Formats and prints an arbitrary style report, with the format specified in the file "file.frm". A page header may be specified.

This RDB operator reads an rdbtable from STDIN and writes a formatted report on STDOUT. Options may be abbreviated.

Options:

-help Print this help information.

-pN Page size in of N lines. Default is 60 lines. A value of zero ’-p0’ will turn paging off.

The "file.frm" file (or form file) shows pictorially one ’record’ of output, which will contain data from one row of an rdbtable. An optional page header may be defined as well.

The form file contains regular text, picture fields, and associated column names. Regular text prints just as given. Picture fields define the width and justification for printing a data value from a column. The names of the associated columns are listed on the line following the picture fields and in the same order.

Picture fields start with either ’@’ or ’^’ and are followed by one of three primary characters to define the width of the field. The three characters are ’<’, ’>’, or ’|’ to specify left, right, or center justification respectively. There is also an alternate right justification character for printing numeric data, with optional decimal point. The character is the sharp sign ’#’, and a period specifies the decimal point placement, as in ’@########.##’.

A numeric picture field has the following features:

Data is lined up on the decimal point (if any)

Automatic rounding of data

Automatic conversion of data in scientific notation

Numeric Data may be in the form of integers, fixed point, or scientific notation’ e.g. 12345, 4567.345, or 1.678E17.

Normally picture fields start with the '@' character. That means to put the referenced data value into the defined picture field, or as much of the data as will fit into the field, if the data is larger than the field. If the field starts with the '^' character it means to repeat the field on as many lines as necessary in order to print the entire data value. This is useful for large data fields, such as comments or free text.

Instead of a column name there are some special names that can be used to have other information inserted. This are especially useful if there is a page header. The special names and what they mean are:

- _pgnr_ - current page number
- _date_ - current date
- _rcnr_ - current record number (row number)
- _`cmd arg1 ... argN`_ - the UNIX command is executed once, and its output is put into the associated picture field. Note that they are BACKTICKS (grave accents) not single quotes.

An example of a form file for use with rdbtable 'sample' is shown in Table 13.

The first and last lines (that start with 'format' or a single period) define the pictorial records and must be as shown. The first record defines the header and is optional. If this form file (named sample.frm) were used in the command:

Table 13

FORM FILE FOR RDBTABLE (SAMPLE)

```
format top =
Page @>,            The Page Header     @<<<<<<<<<<<<<<<<<<<<<<<<<<<
   _pgnr_                                       _date_

.
format =
  Name:   @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<          Type:  @>>>>
               NAME                          TYP
                  Total: @<<<<<<<<     Other:  @<<<<<<<<<<<<<<<
                         AMT                        OTHER
```

.

```
        reporttbl  sample.frm  <  sample
```

it would produce the one page report as in Table 14.

For another example, one might want to have a date on a report in other than the standard date output format, and an idea of who executed the program, and have a reference count of the records being produced. The form file might be as in Table 15. which could produce:

Table 14

PRINTING RDBTABLE (SAMPLE) WITH REPORTTBL

```
Page  1,            The Page Header     Mon Dec  2 16:21:18 PST 1991

  Name:   Bush                                         Type:      A
                  Total: 133           Other:  Another
  Name:   Hansen                                       Type:      A
                  Total: 23            Other:  One
  Name:   Jones                                        Type:      X
                  Total: 77            Other:  Here
  Name:   Perry                                        Type:      B
                  Total: 244           Other:  And
  Name:   Hart                                         Type:      D
                  Total: 1111          Other:  So
  Name:   Holmes                                       Type:      D
                  Total: 1111          Other:  On
```

Table 15

ANOTHER FORM FILE

```
format top =
Run By: @<<<<<<<        The Date/Time is    @<<<<<<<<<<<<<<<<<<<<<<<<
        _‘whoami‘_                           _‘date "+%m/%d/%y %H:%M"‘_
.
format =
    RecordNr:  @>>      @<<<<<<<<<<<<<<<<<<<<
              _rcnr_    OTHER
.
```

```
Run By: hobbs           The Date/Time is    10/15/91 09:43

    RecordNr:    1      Other data here
    RecordNr:    2      and here,
    RecordNr:    3      and so on.
    ...         ...     ...
```

Another example shows how longer data values can be handled. If the form file (named sample2.frm) looks like Table 16, and if the following command is used:

```
        reporttbl  sample2.frm  <  sample2
```

then the output would be as shown in Table 17.

Note that since there were two picture fields that started with the '^' character on one line the length of output records varies according to the length of the two associated data values.

Table 16

FORM FILE FOR RDBTABLE (SAMPLE2)

```
format top =
Page @>,                Page Header here                @<<<<<<<<<<<<<<
_pgnr_                                       _`date "+%m/%d/%y %H:%M"`_
                Executed by @<<<<<<< on: @<<<<<<<<<
                            `whoami`     _`hostname`_

format =
    Name:   @<<<<<<<<<<<<<<<<  Other: @<<<<<<<<<<<        Type: @<<<<<<<<
            NAME               OTHER                             TYP
 Comment:   ^<<<<<<<<<<<<<<<    Long: ^<<<<<<<<<<<<<<<    Right: @>>>>>>>
            COMMENT                     LONG                     RIGHT
=======================================================================
```

Table 17

PRINTING RDBTABLE (SAMPLE2) WITH REPORTTBL

```
Page  1,               Page Header here              12/02/91 16:32
                Executed by hobbs    on: id

    Name:   Bush              Other: Another            Type: A
 Comment:   A comment          Long: This a long        Right:     This
            here.                    message for
                                     test.
=======================================================================
    Name:   Hansen            Other: One                Type: A
 Comment:   A longer           Long: This a long        Right:       Is
            comment here.            message for
                                     test.
=======================================================================
    Name:   Jones             Other: Here               Type: X
 Comment:   A longer,          Long: Short test.        Right:       On
            longer comment
            here.
=======================================================================
    Name:   Perry             Other: And                Type: B
 Comment:   A short comment    Long: This a long        Right:      The
            here.                    message for
                                     test.
=======================================================================
    Name:   Hart              Other: So                 Type: D
 Comment:   Little here.       Long: Here too.          Right:    Right
=======================================================================
    Name:   Holmes            Other: On                 Type: D
 Comment:   A comment here     Long: A short            Right:     Edge
            that is a                message.
            little ongoing,
            so to speak.
=======================================================================
```

ROW

Usage: row [options] expression

Options:

```
    -help    Print this help information.
    -fXXX    The expression is in the file 'XXX', instead of on the
             command line. The advantage in this case is that no quoting
             of chars that might be special to the UNIX shell is necessary.
```

Selects rows from the input rdbtable based on an arbitrary expression using column names. Characters that are special to the UNIX shell must be quoted.

Comparison operators may be of the form: gt, ge, lt, le, eq, ne. E.g 'name eq Hobbs'. Logical constructors 'or' and 'and' may be used; as well as 'null' to indicate an empty data value.

All of the Comparison operators and Logical constructors are reserved and should not be used as column names (they are all lower case and four characters or less).

Since column names and reserved words are parsed by the program, do not put the entire expression in a single pair of quotes as that will prevent the parsing. Also note that column names and reserved words need to be surrounded by blank spaces if they are not individually quoted. For example either form below is fine:

```
row   NAME    eq   "L Brown"  <  sample
row  "NAME"  "eq"  "L Brown"  <  sample
```

but do not use this form:

```
row  "NAME  eq  L Brown"  <  sample
```

This operator reads an rdbtable via STDIN and writes an rdbtable via STDOUT. Options may be abbreviated.

As an example using the sample rdbtable from the DATA section (named sample), to select rows that have the NAME column equal to 'Hansen' the command would be:

```
row  NAME  eq  Hansen  <  sample
```

which would produce:

NAME	COUNT	TYP	AMT	OTHER	RIGHT
6	5N	4	5N	8	8>
Hansen	44	A	23	One	Is

to select rows that have the TYP column equal to 'A' or that have the AMT column greater than 1000 the command would be:

```
row  TYP  eq  A  or  AMT  gt  1000  <  sample
```

producing:

NAME	COUNT	TYP	AMT	OTHER	RIGHT
6	5N	4	5N	8	8>
Bush	44	A	133	Another	This
Hansen	44	A	23	One	Is
Hart	77	D	1111	So	Right
Holmes	65	D	1111	On	Edge

SORTTBL

Usage: sorttbl [options] [-r] column [[-r] column] ...

Sorts an rdbtable on one or more columns. Each column may be sorted in normal (ascending) or reverse (descending) order. Also a column of monthnames (Jan, Apr, ...) in any case letters, may be sorted.

This operator reads an rdbtable via STDIN and writes an rdbtable via STDOUT. Options may be abbreviated. Uses the UNIX 'sort' routine.

Options:

- -c   Check that the rdbtable is sorted on the selected columns.
- -help   Print this help information.
- -r   Reverse order. Applies to the following column only.
- -u   Make rows unique on selected columns.

For example, using the sample data file from the DATA section (named sample) in the following command:

```
sorttbl  COUNT  TYP  <  sample
```

would produce:

NAME	COUNT	TYP	AMT	OTHER	RIGHT
6	5N	4	5N	8	8>
Bush	44	A	133	Another	This
Hansen	44	A	23	One	Is
Holmes	65	D	1111	On	Edge
Perry	77	B	244	And	The
Hart	77	D	1111	So	Right
Jones	77	X	77	Here	On

Of course it would look better if it was piped through ’ptbl’.

The command:

```
sorttbl  COUNT  -r  AMT  <  sample
```

would produce:

NAME	COUNT	TYP	AMT	OTHER	RIGHT
6	5N	4	5N	8	8>
Bush	44	A	133	Another	This
Hansen	44	A	23	One	Is
Holmes	65	D	1111	On	Edge
Hart	77	D	1111	So	Right
Perry	77	B	244	And	The
Jones	77	X	77	Here	On

SUBTOTAL

Usage: subtotal [options] B_column ... -s column ...

This operator lists subtotals of specified column(s) whenever the value of specified break columns(s) (B_column(s)) changes. A single break column will produce a sub-total of all specified columns on each line. If there is more than one break column given then in addition whenever the value of the first break column changes an additional line will be printed showing the sub-total for that group.

If no break column is given the first column is used; if no sub-total column is given then all columns of type numeric are sub-totaled.

This operator reads an rdbtable via STDIN and writes an rdbtable via STDOUT. Options may be abbreviated.

Options:

-help   Print this help information.

Example rdbtable (named small):

```
name      amt    typ   count  n
6         5N     4     5N     2
Hansen    39     A     23     3
Hansen    9      A     3      3
Hansen    9      B     3      4
Jones     42     B     144    5
Jones     4      B     14     5
Hart      9      C     3      5
Hart      2      C     55     6
Hart      2      D     55     6
Hobbs     57     X     7      4
Hobbs     5      X     57     4
```

The output from the command:

```
subtotal  name  -s  amt  <  small  |  ptbl
```

would be:

```
name      amt
------  -----
Hansen     57
Jones      46
Hart       13
Hobbs      62
```

The output from the command:

```
subtotal  name  typ  -s  amt  count  <  small  |  ptbl
```

is shown in Table 18.

Table 18

OUTPUT FROM THE SUBTOTAL OPERATOR

name	typ	amt	count
Hansen	A	48	26
Hansen	B	9	3
		57	29
Jones	B	46	158
		46	158
Hart	C	11	58
Hart	D	2	55
		13	113
Hobbs	X	62	64
		62	64

SUMM

Usage: summ [options] [column ...]

Produces "summary" information about the rdbtable. If no columns are given then information about all columns is produced. A Count of the data rows is always shown.

This operator reads an rdbtable via STDIN and writes a summary report via STDOUT. Options may be abbreviated.

Options:

- -cu A Count of the unique values for each column given.
- -cun Like option ’-cu’ but also shows counts of null (empty) and blank values (have only space chars), if either exist.
- -cuu A Count of each unique value for each column given.
- -help Print this help information.
- -m The min, average, max, for each column given.
- -v Inverse option. Selects all columns except those named.

TBL2LST

Usage: tbl2lst [options]

Converts an rdbtable to "list" format. Long data fields are folded. This operator is mainly used by other operators.

This RDB operator reads an rdbtable from STDIN and writes an rdbtable to STDOUT. Options may be abbreviated.

Options:

-edit	Edit option. Used by etbl.
-help	Print this help information.
-lN	Line length of N is to be used.

UNIQTBL

Usage: uniqtbl [options] column ...

Reads the input rdbtable and compares adjacent rows. The second and succeeding copies of repeated rows, considering only the selected columns, are removed. That is, adjacent rows are considered equal if the data values in all of the selected columns are equal. The remaining rows are written to the output rdbtable.

Note that repeated rows must be adjacent in order to be found. Normally this means that the input rdbtable should be sorted on the selected columns for this capability to work properly.

Options:

-D	Diagnostic output. Prints number of rows removed on STDERR.
-help	Print this help info.
-v	Inverse option. Selects all columns except those named.

This RDB operator reads an rdbtable from STDIN and writes an rdbtable to STDOUT. Options may be abbreviated.

VALID

Usage: valid [options] [file ...]

Validates the structure of one or more rdbtables. Checks number of data fields per line, max width of column names and data values, and checks numeric data type values. Reports errors by line number and column name.

Reads from STDIN if filenames are not given. Writes diagnostic information on STDOUT. Options may be abbreviated.

Options:

```
    -help    Print this help information.
    -l[x]    List exact data values with visible delimiters, using 'x' as the
             delimiter. The value of 'x' may be multi-character, default is "|".
    -nw      No check of the width of the data.
    -size    Report max size of actual data in each column.
    -templ   Generate a template file from the header of the table, on STDOUT.
             Does NOT check the body of the table.
```

If there is more than one file given each file will be identified on the output.

The '-size' option has proven very useful as it shows the actual size of the largest data value for each column, in addition to the template information. The command:

```
        valid  -size  sample
```

shows the following output:

```
            0                NAME  6     6
            1               COUNT  5N    2
            2                 TYP  4     1
            3                 AMT  5N    4
            4               OTHER  8     7
            5               RIGHT  8>    5
        Columns: 6, Rows: 6, File format valid  sample
```

The last two columns above show the defined size of each column in the rdbtable, and the actual maximum size of the data values for each column in the rdbtable.

IV. EXTRACTING INFORMATION FROM RDBTABLES

GENERAL

The following shows some examples of how the system is usually used, which involves a combinations of operators. Using the rdbtable named 'sample' the command:

```
column NAME OTHER TYP AMT < sample | sorttbl TYP AMT | ptbl
```

gives the output:

```
NAME    OTHER     TYP    AMT
------  --------  ----  -----
Hansen  One       A        23
Bush    Another   A       133
Perry   And       B       244
Hart    So        D      1111
Holmes  On        D      1111
Jones   Here      X        77
```

Note that columns COUNT and RIGHT were excluded by the 'column' oper, and that the order of the selected columns was changed from that in the rdbtable. Of course to save the output in a file, (redirection of STDOUT into a file) something like the following is used:

```
column ... < sample | sorttbl ... | ptbl > file.out
```

An example using the operator 'row' on the rdbtable sample is:

```
row AMT lt 900 < sample | column NAME AMT TYP RIGHT |\
sorttbl l NAME | ptbl
```

Note that the '\' character at the end of the first line of the above command is the signal to the UNIX shell that the command is continued on the next line. Here we select some rows using 'row', select some columns using 'column', sort what we have with 'sorttbl', and print with 'ptbl'. The output is:

```
NAME      AMT  TYP      RIGHT
------  -----  ----  --------
Bush      133  A         This
Hansen     23  A           Is
Jones      77  X           On
Perry     244  B          The
```

A REAL WORLD PROBLEM

The following shell script shows how the RDB operators and other UNIX utilities can be fitted together to solve a real world problem. The problem was to find out if the rows in a large rdbtable were unique over four columns. Since ’summ’ will tell us whether the rows of an rdbtable are unique on a single column, we need to construct a temporary tdbtable.

To illustrate the solution on a small rdbtable, the script below works on an rdbtable like ’sample’ but with some rows repeated. First the script selects four columns and adds a dummy column named ’uniq’ (using ’column’). It then puts the combined values of the four columns into the dummy column (using ’compute’). Next it examines the value of the dummy column ’uniq’ for uniquness (using ’summ’) and then uses the UNIX command ’egrep’ to show only the lines of interest, e.g. those lines that start with something other than a ’1’.

```
column < table.rdb  Unit  Day  Time  MSN  -a uniq 12 |\
compute uniq = Unit . Day . Time . MSN |\
summ  -cuu  uniq  |\
egrep -v  ’^  *1’
```

The output was like the following:

```
Rows: 9
Unique values for uniq: 6
       2 Jones77X77
       3 Perry77B244
```

meaning that there were (in this example) two rows that had duplicates over the four columns with one set of values, and three rows that had duplicate with another set of values.

ANOTHER REAL WORLD PROBLEM

This next one is a bit more complicated although very useful, and it does demonstrate the use of 'compute' using a newly created column. The idea is to make a summary of the data in rdbtable sample3 (from the section on 'etbl'). The command is:

```
        column name datatype -a nr 2 < sample3 |\
        compute nr = '++$x' |\
        sorttbl datatype name | reporttbl sum.frm
```

Note that 'column' selects the two columns on which to make the summary and adds a new (null) column 'nr'. Then 'compute' puts data into the new column by using the PERL expression shown. The expression '++$x' merely increments itself by one each time it is evaluated, an easy way to get an increasing number. Finally 'sorttbl' sorts the newly created rdbtable and it is then printed with 'reporttbl' using the form file 'sum.frm'.

Table 19

FORM FILE (SUM.FRM)

```
format top =
                                              @<<<<<<<<<<<<<<<<<<<<<<<<<<<
                                                _date_
            Sample Three Database Summary by Datatype

       Datatype  Nr  Name
       --------  --  -----------------------------------------
.
format =
       @>>>>>>>  @>  @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
       datatype  nr  name
.
```

Table 20

DATABASE SUMMARY EXAMPLE

```
                                            Wed Dec  4 21:23:20 PST 1991
     Sample Three Database Summary by Datatype

Datatype  Nr  Name
--------  --  -----------------------------------------
     ABC   5  ABC Duplicate
     ABC   4  ABC Original
     ABC   8  LAF-S Air Floating Model
     ATO   3  ATO (Air Tasking Orders) Original
     ATO   6  ATO Number 2222-11
     BDA   1  ACAS (Air Combat Assessment)
     BDA   2  ACAS (Air Combat Assessment) BDA Sorties
     BDA   7  BDA Supplement number 17
```

The form file is shown in Table 19 and the output in Table 20.

V. GENERATING NEW RDBTABLES

GENERAL

Any editor can be used to construct an rdbtable, since it is a regular UNIX file. However, it may be easier to first generate a template file with an editor, then use ’dataent -init’ to enter the rows of data.

To modify an rdbtable at any time use ’etbl’ in either column or list form. The choice of form to use depends somewhat on the structure of the rdbtable. If the rdbtable has several columns of relatively narrow data (that will all fit in the width of the current window or terminal) and also several very wide columns (none of which will fit) and changes need to be made to some of the narrow columns, then it makes sense to use ’etbl’ on the desired narrow columns in ’column’ form. If changes need to be made to some of the wide columns then use ’etbl’ on the wide columns, plus any key columns necessary for control, in ’list’ form.

After editing an rdbtable it is always a good idea to use the operator ’valid’ to re-confirm the structure of the rdbtable.

COLUMN DOCUMENTATION

The RDB system provides the capability to include a small amount of documentation about each column in an rdbtable. The documentation stays with the rdbtable, and may be viewed at any time. This feature has proven very worthwhile in some current work.

The documentation is placed in the template file (which is used to define, in list form, the header of an rdbtable). For example the following header template file contains the usual elements: index number, column name, column definition, and in addition some documentation about each column.

```
0       Name  24S  Name of item
1       Type   1S  Type: 1,2,3,7,9,A,B,X
2      Count   3N  Number of items
3          K   1   Constant modifier
4        SS7   2   Special status for type 7
5       Size  12N  In K. $Id: RDB.er,v 1.33 1992/04/14 15:04:45 hobbs Exp $
```

Note that the index number, column name, and column definition all consist of contiguous characters, each forming a word separated by whitespace. Also note that there is one or more whitespace characters after the column definition and before any documentation. That is, the documentation starts with the fourth word on the line.

This is a very good way to locate an RCS Identifier, as in line 5 above. This way, when checking out an rdbtable from RCS the identification of the current revision will be placed in the actual data file. This is a better (and more efficient) solution than using a row or column for the same purpose.

When the operator 'headchg' uses the template file to generate the header it will keep the documentation with the header. At any time the entire header, including documentation, can be viewed by using the command 'valid -templ < rdbtable' (or 'headchg -templ < rdbtable). The output from either command will be essentially like the above example.

CONCATENATING RDBTABLES

The need to concatenate rdbtables comes up every so often and although it is simple to do it may not be obvious. The UNIX 'cat' command can not be used as it would result in duplicating the header and thus an invalid rdbtable. And of course, only rdbtables with the same header should be concatenated, otherwise an invalid rdbtable would result (in this case it could be a gross inconsistency if the number of columns were different). If we have two rdbtables, TABA and TABB, then to concatenate TABB onto the end of TABA we use the command:

```
headchg -del < TABB >> TABA
```

Note that this avoids duplicating the header. Note also that in this case the operator ’headchg’ does not use a template file.

VI. CONVERSION OF EXTERNAL DATA INTO RDBTABLES

The best method I have found to convert data in files that were received from an external source is to use the PERL Programming Language. A typical PERL script for such a conversion from a file of fixed column data is shown in Table 21. The last two lines do most of the work. Note that this process converts the data into an rdbtable body only; the rdbtable header must still be generated manually.

The easiest and best way to accomplish the above is to first construct a template file from known or observed information in the external data file, then generate the header using ’headchg -gen’. Then run the PERL script appending the output to the header file. Then run ’valid’ to make sure everything went allright.

Table 21

PERL SCRIPT FOR DATA CONVERSION

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#! /usr/bin/perl
$templ = "A6 A12 x A5 x6 A18 A17 x A38 x4 A2 x A x2 A15 x3 A6 A4 x2" .
    " A2 x4 A5 x A3 A3 A4 x26 A12 A12 A12 A29 x7 A3 A3 x6 A6 A6 A8" .
    " A4 A3 x3 A3 x3 A8 x10 A2 x4 A12 x138 A6 x66 A24 A36 A12 x6" .
    " A6 A6 x6 A6 A24 A12 x50 A42 A42 A42 x34 A12 A12 x186 A6" ;
$0 =~ s-.*/-- ;
$HelpInfo = <<EOH ;

    Strip out and reform an 'external' data file into an rdbtable.

Usage:  $0  [options]  file

Options:
    -help    Print this help information.

Strips out the first 46 fields from an 'external' data file and
reformats it into 'rdbtable' format (TAB delimited with NEWLINE at end).
Output is on STDOUT.
EOH
while ( $ARGV[0] =~ /^-/ ) {                            # Get args
    $_ = shift ;
    if( /-h.*/ ){ die $HelpInfo ; }
    die  "Bad arg: $_\n", $HelpInfo ;
}
while(<>){
    @a = unpack( $templ, $_ );
    print join( "\t", @a), "\n" ;
}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

VII. DATA ACCESS CONTROL

Since rdbtables are regular UNIX files, we have so far found no need to implement, and have the associated overhead of, data access controls in RDB. Setting the UNIX permissions on each file or directory of files has proven very useful. This is another example of how the RDB system works "WITH" UNIX, not in addition to it, e.g. not duplicating UNIX functions.

The Revision Control System (RCS) is one of the best configuration management tools available and is in use here at RAND for version control of many types of files, including rdbtables. The operator 'etbl' will automatically check out an rdbtable for editing, and then check the new version back into RCS. Other operators can utilize rdbtables that are under RCS control by using commands like:

```
co  -p  sample  |  row  ...  |  column  ...  |  ptbl
```

VIII. INDEXING, FAST ACCESS METHODS

These topics are under consideration and have not yet been implemented.

IX. LIMITS, A FEW MINOR ONES

The following limits apply.

- There must not be any ASCII TAB characters in the data. This is the primary limit as the ASCII TAB character is the delimiter in rdbtables.
- There must not be any pipe characters '|' used or entered as data when using the operator 'etbl'.
- All of the Compare operators and Logical constructors are reserved and should not be used as column names (they are all lower case and four characters or less). They are: 'gt', 'ge', 'lt', 'le', 'eq', 'ne', 'or', 'and', and 'null'.

X. REFERENCES

1. "Unix Review" magazine, March, 1991, page 24, "A 4GL Language".

2. "UNIX Relational Database Management", R. Manis, E. Schaffer, R. Jorgensen, 1988, Prentice Hall.

3. "Programming PERL", L. Wall, R. Schwartz, 1991, O’Reilly & Associates.

Appendix A

EXAMPLES OF PERL EXPRESSIONS AND STATEMENTS

Following are some examples of PERL expressions and statements of the type that might be used with RDB operators, and their meaning. Note that the operator 'row' takes a PERL expression while the operator 'compute' takes a complete PERL statement.

Expressions:

```
    COLA =~ /XXX/
                    -- column COLA contains the pattern 'XXX'.
    COLA !~ /XXX/
                    -- column COLA does NOT contain the pattern 'XXX'.
    COLA =~ /^XXX/
                    -- column COLA starts with the pattern 'XXX'.
    COLA =~ /XXX$/
                    -- column COLA ends with the pattern 'XXX'.
    COLA ne null
                    -- column COLA is not null (but it could contain blanks).
    COLA =~ /^\s*$/
                    -- column COLA is null or contains only blank space.
    COLA eq 'YYY'
                    -- column COLA equals the literal 'YYY'.
    COLA =~ /X..Y/
                    -- column COLA contains the pattern 'X..Y', which means
                    'X', followed by any two characters, then 'Y'.
    COLA =~ /X.*Y/
                    -- column COLA contains the pattern 'X.*Y', which means
                    'X', followed by any number of (including zero)
                    characters, then 'Y'.
    NUMC eq 12
                    -- column NUMC equals 12.
```

```
    COLA ne null && COLB ne null
                    -- column COLA and column COLB are not null (empty).
    COLA eq 'ABC' || COLA eq 'BCD'
                    -- column COLA equals the literal 'ABC' or column COLB
                    equals the literal 'BCD'
```

Statements:

```
    COLA = COLB ;
                    -- set the value of column COLA to that of COLB.
    NUMC = NUMC - 7 ;
                    -- decrement the value of column NUMC by 7.
    NUMC -= 7 ;
                    -- (same as above).
    NUMC = NUMC / 4 ;
                    -- divide the value of column NUMC by 4.
    NUMC *= 2.3 ;
                    -- multiply the value of column NUMC by 2.3.
    $abc++ ;
                    -- increment the value of variable $abc by 1.
    ++$abc ;
                    -- (same as above).
    COLA = 'WORDS' ;
                    -- set the value of column COLA to the literal 'WORDS'.
    NUMC = 12 ;
                    -- set the value of column NUMC to 12.
    if( COLA =~ /XXX/ ){ COLA .= 'YYY' ; }
                    -- If column COLA contains the pattern 'XXX' then add
                    the literal 'YYY' to the end.
    COLA .= 'YYY' if COLA =~ /XXX/ ;
                    -- (same as above).
    if( COLA eq 'ABC' || COLA eq 'BCD' ){ COLA = 'XXX' ; }
                    -- If column COLA equals 'ABC' or 'BCD' set the value
                    of COLA to 'XXX'.
```

PREFACE

This working draft describes, and provides instructions for the use of, the RDB system as it currently exists at RAND. The RDB system is currently in use at RAND, however it is not complete and there are plans to make enhancements and extensions as required. Accordingly, revisions of this document will be produced.

The preparation of this research document was sponsored through RAND’s three federally funded research and development centers - the National Defense Research Institute (sponsored by the Office of the Secretary of Defense and the Joint Staff), the Arroyo Center (sponsored by the U.S. Army), and Project AIR FORCE (sponsored by the U.S. Air Force).

It should be noted that the development of the computer code comprising the RDB system was not done under any RAND contract and was initially accomplished on non-RAND time as a personal computer science project of the author.

SUMMARY

RDB is a fast, portable, relational database management system without arbitrary limits, (other than memory and processor speed) that runs under, and interacts with, the UNIX Operating system.

It uses the Operator/Stream DBMS paradigm described in "Unix Review", March, 1991, page 24, entitled "A 4GL Language". There are a number of "operators" that each perform a unique function on the data. The "stream" is suplied by the UNIX Input/Output redirection mechanism. Therefore each operator processes some data and then passes it along to the next operator via the UNIX pipe function. This is very efficient as UNIX pipes are implemented in memory (at least in versions of UNIX at RAND, including SunOS).

The data is contained in regular UNIX ACSII files, and so can be manipulated by regular UNIX utilities, e.g. ls, wc, mv, cp, cat, more, less, editors like the RAND editor 'e', head, RCS, etc.

The form of each file of data is that of a relation, or table, with rows and columns of information.

To extract information, a file of data is fed to one or more "operators" via the UNIX Input/Output redirection mechanism.

There are also programs to generate reports, and to generate, modify, and validate the data.

ACKNOWLEDGEMENTS

I would like to thank the following people for their ideas and suggestions on the implementation and improvement of the RDB system, as well as some needed checkout of obscure bugs in the code:

Chuck Bush

Don Emerson

Judy Lender

Roy Gates

Rae Starr

CONTENTS

TABLES