NAME

gdb \- NMRWish TCL Generic Database Tools

HIERARCHY

```
                                  . (root)
                    dBase1 ................ dBaseN
             table1 ...... tableN            text
    text variable... entry...
         cell...     cell...
```

KEY

() type of data returned by a function.
[] optional argument.
"" keyword.
| separates mutually exclusive options.
{} a single item or proper TCL list.

Elements such as names and keywords are case−sensitive.

TABLE FORMAT

Keywords in a table file consist of the following:

```
          #        comment
          REMARK   comment
          DATA     [information]
VARS    [variableNameList]
FORMAT  [variableFormatList]
```

Empty lines are ignored.

All other lines are treated as table entries, regardless of where they are in the file.

Each entry should have as many elements as there are variable names listed in the VARS line.

Currently, spaces are not allowed within an item (i.e., within a text string or format); this will be changed in future versions.

Variable names must begin with a letter or underscore, and contain only letters, numerals, or underscores.

INITIALIZE THE DATABASE SYSTEM

(handle) **gdbInit**

CHECK THE INTEGRITY OF THE DATABASE

(int) **gdbCheck** [**−verb**]
Checks all database info; returns error count.

(int) **gdbCheck** [*handleList*]
Returns valid handles or zeros for bad handles.

CREATE AN OBJECT

(handle) **gdbCreate** "dBase" | "table" | "variable" | "entry"
[**–name** *objectName*]
[**–parent** *parentHandle*]
[**–result** *resultHandle*]
–varType *variableType*
–varFmt *variableFormat*

Arguments **–varType** and **–varFmt** are required for creation of variables, and are ignored in other cases.

READ A TABLE OR DATABASE

(handle) **gdbRead** "dBase" | "table"
–in *fileOrDirName*
–name *objectName*
[**–parent** *parentHandle*]
[**–result** *resultHandle*]
[**–pdb**]

Argument **–in** should be a directory name when reading a database (dBase), and a file name when reading a table.

Option **–pdb** is only valid when reading a PDB format file as a table.

WRITE A TABLE OR DATABASE

(boolean) **gdbWrite** *dBaseHandle* | *tableHandle*
[*outName*]

The *outName* should be a directory name when writing a database, and a file name when writining a table. If the *outName* argument is omitted, the data will be written to standard error.

DESTROY AN OBJECT

Returns count of items destroyed.
(int) **gdbDestroy** *handleList*
[**–verb**]

APPEND TABLES

Appends rows of source to destination.
Returns the handle to the result table.

(handle) **gdbAppend**
[**–dest** *destHandle*]
–src *srcTableHandleList*

If the **–dest** argument is a table, this table will modified by the appending the source table to it.

If the **–dest** arguement is a dBase, a new table will be created in this dBase, and the source table copied to it.

JOIN TABLES

Appends columns of source to destination.
Returns the handle to the result table.

(handle) **gdbJoin**
[**−dest** *destHandle*]
−src *srcTableHandleList*

If the **−dest** argument is a table, this table will
modified by the appending the source table to it.

If the **−dest** arguement is a dBase, a new table will
be created in this dBase, and the source table copied to it.

QUERY TABLES

(result) **gdbSelect**
−from *tableList*
[**−inPlace**]
[**−tcl** | **−M**]
[**−cond** *condition*]
[**−vars** *varsSelectedList*]
[**−using** *varsLoadedList*]
[**−destSuffix** [*dSufixList*]]
[**−querySuffix** [*qSuffixList*]]
[**−suffix** [*suffixList*]]
[**−name** *tableName*]
[**−result** *resultVar*]
[**−parent** *dBaseHandle*]

Selects TCL (default) or M format query.
Result is list of entries from input.
Default: all entries.
Default: all variables.
Default: variables in unsubstituted condition.
Maps input vars to output vars.
Maps input vars to query vars.
Sets both suffix lists.
Sets name of result table.
Tcl variable; destroys previous object.
Desired dBase for result, if different from first input table.

If no suffix list is given, _0, _1 etc
is used. Keyword 'None' in a suffix list
means no suffix will be applied.

SORT A TABLE

(boolean) **gdbSort** [<|>]*varNameOrHandleList*
[**−proc** *tclSortProc*]

The TCL sort procedure should return −1, 0, or 1
when two entries are compared:

```
proc sortProc {entry1ValList entry2ValList} ...
```

FIND AN OBJECT BY NAME

(handleList) **gdbFind** *parentHandle name*
[**−all**]

Use . for root handle
Older command gdbFindByName is equivalent to gdbFind.

GET FIRST/LAST ITEM OF AN OBJECT

(infoList) **gdbFirst** *handle*
(infoList) **gdbLast** *handle*

For root, same as: gdbGetInfo -parms dBase
For dBase, same as: gdbGetInfo -parms table
For table, same as: gdbGetInfo -parms entry

GET INFORMATION ABOUT AN OBJECT

(infoList) **gdbGetInfo** *handle*
[**−array** *tclArrayName*]
[**−parms** *parmNameList*]

Types: dBase table variable entry text cell

dBase:	text:	root:	table:
type	type	type	type
handle	handle	handle	handle
name	next	dBase	name
next	prev	dBaseN	varCount
prev	parent	{dBases}	entryCount
table	value		var
tableN			entry
tabCount			varN
text			entryN
textN			next
parent			prev
{tables}			parent
			text
			textN
			{vars}
			{varNames}
			{entries}

variable:	entry:	cell:
type	type	type
handle	handle	handle
name	next	north
varFmt	prev	south
varType	cell	east
next	cellN	west
prev	parent	value
cell	{cells}	empty
cellN	{values}	var
parent		entry
{cells}		

{values}

SET PROPERTIES OF AN OBJECT

(boolean) **gdbSetInfo** *handle*
[**–name** *objectName*]
[**–varFmt** *varFmt*] (variable only)
[**–value** {*val*}] (cell or text only)

GET VALUES FROM TABLES

(string) **gdbGet** *entry varNameOrHandleList*
[**–notrim**]

(string) **gdbGetVal** *cellList* [**–notrim**]

TEST FOR DEFINED VALUES IN TABLES

(string) **gdbTest** *entry varNameOrHandleList*
(string) **gdbTestVal** *cellList*

SET VALUES IN TABLES

(int) **gdbSet** *entry* {*varNameList*} {*valueList*}
(int) **gdbSetVal** {*cellList*} {*valueList*}

Older command gdbSetValByVar is equivalent to gdbSet.

GET THE NTH ITEM

Returns the next, previous, or Nth item in a linked list.

(handleList) **gdbNext** {*handleList*} [**n** | "last"]
(handleList) **gdbPrev** {*handleList*} [**n** | "last"]

(handleList) **gdbEast** {*cellList*} [**n** | "last"]
(handleList) **gdbWest** {*cellList*} [**n** | "last"]
(handleList) **gdbNorth** {*cellList*} [**n** | "last"]
(handleList) **gdbSouth** {*cellList*} [**n** | "last"]

SWAP TWO ITEMS

(boolean) **gdbSwap** *handle1 handle2*
Swaps two items in a linked list;
Not implemented yet.

REVERSE THE ORDER

(handle) **gdbReverse** *listBase*
Reverses the order of items in a linked list;
Not implemented yet.

MANIPULATE ANNOTATIONS

(boolean) **gdbDeleteAnnotation**
dbaHandle | *tableHandle*

(boolean) **gdbAddAnnotation**

dbaHandle | *tableHandle* {textValue}

PDB TABLES

(float) **pdbCoords** *entry*
Returns X,Y,Z coords of an ATOM entry.

(float) **pdbDist** *entry1 entry2*
Returns distance between ATOMs 1, and 2.

(float) **pdbAngle** *entry1 entry2 entry3*
Returns angle formed by ATOMs 1, 2, and 3

(float) **pdbAngle** *entry1 entry2 entry3*
Returns dihedral formed by ATOMs 1, 2, 3, and 4.