Programmer's Guide to the TBL Facility

A Facility for Manipulating Tables in a Relational Database

David E. Beecher

Mallinckrodt Institute of Radiology
Electronic Radiology Laboratory
510 South Kingshighway Boulevard
St. Louis, Missouri 63110
314/362-6965 (Voice)
314/362-6971 (FAX)

Version 2.10.0

August 3, 1998

---

1 Introduction

The TBL routines provide a structured access mechanism for tables defined using pre-existing database products. These routines currently support a locally engineered database system, and the commercially available Sybase line of database products. This library includes routines to open and close individual tables (TBL_Open and TBL_Close), as well as insertion (TBL_Insert), deletion (TBL_Delete), selection (TBL_Select), and modification (TBL_Update). Additionally, a layout routine (TBL_Layout) is provided to allow the user to determine the number and type of columns present in a particular table.

This library was designed primarily for the support of other facilities (e.g. IDB facility), but is also used directly by various other applications.Data Structures

2 Data Structures

tbl.h is the primary include file for applications wishing to use the facility. There are 3 primary data structures defined which are of use to the developer. They are the field, criteria, and update structures (TBL_FIELD, TBL_CRITERIA, and TBL_UPDATE). These structures are used to specify field lists, criteria lists, and value lists for insertions, deletions, updates, etc. These structures are defined as follows:

```
typedef struct {
    char *FieldName;
    TBL_OPERATOR Operator;
    TBL_VALUE Value;
} TBL_CRITERIA;

typedef struct {
    char *FieldName;
    TBL_VALUE Value;
} TBL_FIELD;

typedef struct {
    char *FieldName;
    TBL_FUNCTION Function;
    TBL_VALUE Value;
} TBL_UPDATE;
```

TBL_FUNCTION and TBL_OPERATOR are typedefed as follows:

```
typedef enum {
 TBL_NULL,TBL_NOT_NULL,TBL_EQUAL,
 TBL_NOT_EQUAL,TBL_GREATER,
 TBL_GREATER_EQUAL,TBL_LESS,
 TBL_LESS_EQUAL,TBL_LIKE,TBL_NOP
} TBL_OPERATOR
```

```
typedef enum {
  TBL_SET,TBL_INCREMENT,TBL_DECREMENT,
  TBL_ZERO,TBL_ADD,TBL_SUBTRACT
} TBL_FUNCTION
```

In general, arrays of fields, criteria, and updates are passed to the routines to either provide information or as a place holder for information to be returned from that function. These arrays are always terminated by supplying a null string in the FieldName element of the structure. Since a general mechanism to deal with any data type was desired, the TBL_VALUE structure was defined as follows:

```
typedef struct {
    TBL_DATATYPE Type;
    int AllocatedSize;
    int Size;
    int IsNull;
    union {
        void *Other;
        short *Signed2;
        int *Signed4;
        unsigned short *Unsigned2;
        unsigned int *Unsigned4;
        float *Float4;
        double *Float8;
        char *String;
        char *Text;
        void *BinaryData;
    } Value;
} TBL_VALUE;
```

Notice that it is the users responsibility to allocate the proper amount of storage for the data item to be stored, as well as assigning a pointer to that storage to the proper data element of this structure. Finally, TBL_DATATYPE is defined as an enum with the following code:

```
typedef enum {
  TBL_OTHER,TBL_UNSIGNED2,TBL_UNSIGNED4,
  TBL_SIGNED2,TBL_SIGNED4,TBL_FLOAT4,
  TBL_FLOAT8,TBL_STRING,TBL_TEXT,
  TBL_BINARYDATA
} TBL_DATATYPE;
```

3 Include Files

Any applications needing to use this facility should include the following files:

```
#include “dicom.h”
#include “tbl.h”.
```

In addition, Sybase users need to set at least one environment variable in order to use this facility. Sybase uses the environment variable DSQUERY to determine which server to use. The server names available to Sybase are listed in the interfaces file located in the home directory of the sybase user. This facility first looks at the environment variable CTN_SYBASE_SERVER to determine the server to use, then at the variable DSQUERY. One of these two variables must be correctly set in order to use this facility.

4 Return Values

The following returns are defined from the TBL routines:

TBL_NORMAL	Operation completed successfully
TBL_UNIMPLEMENTED	The operation attempted is currently unimplemented
TBL_ALREADYOPENED	The table/database pair is already opened
TB__DBNOEXIST	Specified database does not exist
TBL_TBLNOEXIST	Specified table does not exist
TBL_NOMEMORY	There is no more memory available via malloc
TBL_CLOSERROR	The handle identified in the close does not exist
TBL_BADHANDLE	The handle passed is invalid
TBL_NOFIELDLIST	The null pointer was passed for the field list
TBL_SELECTFAILED	The select operation failed from bad specifications
TBL_EARLYEXIT	The select callback routine returned something other than TBL_NORMAL and caused an early termination of this select
TBL_DELETEFAILED	The delete operation failed from bad specifications
TBL_INSERTFAILED	The insert operation failed from bad specifications
TBL_UPDATEFAILED	The update operation failed from bad specifications
TBL_DBINITFAILED	The initial database open operation failed
TBL_NOCOLUMNS	The specified table contains no columns
TBL_NOCALLBACK	No callback function was specified in the TBL_Layout routine

5 TBL Routines

This section provides detailed documentation for each TBL facility routine.

TBL_Close

Name

TBL_Close - close the specified table in the named database

Synopsis

CONDITION TBL_Close(TBL_HANDLE **handle)

handle The point to the database/table pair to be closed..

Description

Locates the handle specified in the call and removes that entry from the internal list maintained by this facility.

Notes

None.

Return Values

```
TBL_NORMAL
TBL_CLOSERROR
```

TBL_Debug

Name

TBL_Debug -This function controls the printing of Sybase error messages

Synopsis

CONDITION TBL_Debug(BOOLEAN flag);

flag The variable that controls whether or not error messages from sybase are printed.

Description

If *flag* is TRUE, error messages from Sybase will be printed, if FALSE, error messages from Sybase will be suppressed.

Notes

Return Values

TBL_NORMAL

TBL_Delete

Name

TBL_Delete -This function deletes the specified records from the specified table.

Synopsis

CONDITION TBL_Delete(TBL_HANDLE **handle, TBL_CRITERIA *criteriaList)

handle	The pointer for the database/table pair to be accessed for deletion. This table must be open.
criteriaList	Contains a list of the criteria to use when deleting records from the specified table. A null list implies that all records will be deleted.

Description

The records selected by the *criteriaList* are removed from the database/table indicated by handle.

Notes

None.

Return Values

```
TBL_NORMAL
BL_BADHANDLE
TBL_DBNOEXIST
TBL_DELETEFAILED
```

TBL_Insert

Name

TBL_Insert -This function inserts records into the named table.

Synopsis

CONDITION TBL_Insert(TBL_HANDLE **handle, TBL_FIELD *fieldList)

handle	The pointer for the database/table pair to be accessed for insertion. This table must be open.
fieldList	Contains a list of the keyword/value pairs to be inserted into the specified table.

Description

The table values contained in *fieldList* are added to the database and table specified by *handle*. Each call inserts exactly 1 (one) record. It is the users responsibility to ensure that the correct number of values are supplied for the particular table, and that any values which need to be unique (i.e. for the unique key field in a table), are in fact unique.

Notes

None.

Return Values

```
TBL_NORMAL
TBL_BADHANDLE
TBL_DBNOEXIST
TBL_NOFIELDLIST
TBL_INSERTFAILED
```

TBL_Layout

Name

TBL_Layout -This function returns the columns and their types of a particular table specified by handle.

Synopsis

CONDITION TBL_Layout (char *databaseName, char *tableName, CONDITION (*callback)(), void *ctx)

databaseName	The name of the database to use tableName The name of the table to use.
callback	The callback function invoked whenever a new record is retrieved from the database. It is invoked with parameters as described below.
ctx	Ancillary data passed through to the callback function and untouched by this routine.

Description

As each column is retrieved from the specified table, the callback function is invoked as follows:

callback(TBL_FIELD *fieldList, void *ctx);

fieldList contains the field name and the type of the column from the table specified. *ctx* contains any additional information the user originally passed to the layout function. If *callback* returns any value other than TBL_NORMAL, it is assumed that this function should terminate and return an abnormal termination message (TBL_EARLYEXIT), to the routine which originally invoked TBL_LAYOUT.

Notes

None.

Return Values

```
TBL_NORMAL
TBL_BADHANDLE
TBL_NOCALLBACK
TBL_DBNOEXIST
TBL_SELECTFAILED
TBL_TBLNOEXIST
TBL_NOCOLUMNS
TBL_EARLYEXIT
```

TBL_NextUnique

Name

TBL_NextUnique -this function is a simple unique number generator

Synopsis

CONDITION TBL_NextUnique(TBL_HANDLE **handle, char *name, int *unique)

handle	The handle of the previously opened database/table name pair.
name	The name of the unique number to retrieve
unique	The unique number is stored here.

Description

TBL_NextUnique retrieves the next unique number for the given *name*. The name must be a valid unique number name as specifed in the appropriate table definition. After the current number is retrieved and placed in *unique*, this value is incremented and replaces the old value in the database. This function will generate approixmately 2,147,483,647 unique numbers before repeating.

Notes

None.

Return Values

```
TBL_NORMAL
TBL_BADHANDLE
TBL_DBNOEXIST
TBL_SELECTFAILED
TBL_UPDATEFAILED
```

TBL_Open

Name

TBL_Open -this function opens the specified table in the specified database. It creates a new handle for this particular table and passes that identifier back to the user.

Synopsis

CONDITION TBL_Open(char *databasename, char *tablename, TBL_HANDLE **handle)

databasename	The name of the database to open tablename The name of the table to open which is contained in the aforementioned database
handle	The pointer for the new identifier created for this database/table pair is returned through handle

Description

The first time *TBL_Open* is invoked, database specific routines may be called to allocate the communication structures needed for subsequent operations. If the database/table pair has already been opened, the caller is returned a reference to the already opened table. A unique handle (address) is then created for this pair and returned to the user for subsequent table operations.

Notes

TBL_Open used to return an error when the caller tried to open a table that was already open. The function has been modified to allow the user to open the table multiple times. The user needs to call *DB_Close* one time for each time that a table is opened.

Return Values

```
TBL_NORMAL
TBL_DBINITFAILED
TBL_DBNOEXIST
TBL_TBLNOEXIST
TBL_NOMEMORY
```

TBL_Select

Name

TBL_Select -This function selects some number of records (possibly zero) that match the criteria specifications given in the input parameter criteriaList

Synopsis

CONDITION TBL_Select(TBL_HANDLE **handle, TBL_CRITERIA *criteriaList,
TBL_FIELD *fieldList, int *count, CONDITION (*callback)(), void *ctx)

handle	The pointer to the database/table pair to be accessed. This table must be open.
criteriaList	Contains a list of the criteria to use when selecting records from the specified table
fieldList	Contains a list of the fields to be retrieved from each record that matches the criteria specification. It is an error to specify a null fieldList.
count	Contains a number that represents the total number of records retrieved by this particular select. If this parameter is null, then an internal counter is used and the final count will not be returned when the select finishes.
ctx	Ancillary data passed through directly to the callback function and untouched by this routine.

Description

As each record is retrieved from the database, the fields requested by the user (contained in fieldList), are filled with the information retrieved from the database and a pointer to the list is passed to the callback routine designated by the input parameter callback. The callback routine is invoked as follows:

callback(TBL_FIELD *fieldList, long count, void *ctx);

Count contains the number of records retrieved to the point. *ctx* contains any additional information the user originally passed to this select function. If *callback* returns any value other than TBL_NORMAL, it is assumed that this function should terminate (i.e. cancel the current db operation), and return an abnormal termination message (TBL_EARLYEXIT) to the routines which originally invoked the select.

Notes

None.

Return Values

```
TBL_NORMAL
TBL_BADHANDLE
TBL_DBNOEXIST
TBL_DBINITFAILED
TBL_NOFIELDLIST
TBL_SELECTFAILED
TBL_EARLYEXIT
```

TBL_Update

Name

TBL_Update -This function updates existing records in the named table.

Synopsis

CONDITION TBL_Update(TBL_HANDLE **handle, TBL_CRITERIA *criteriaList,
TBL_FIELD *fieldList);

handle	The pointer to the database/table pair to be accessed for the modification. This table must be open.
criteriaList	Contains a list of the criteria to use when selecting records from the specified table
fieldList	Contains a list of the keyword/value pairs to be updated in the specified table.

Description

The records which match the (ANDED) criteria in *criteriaList* are retrieved and updated with the information contained in *fieldList*. Only the fields contained in *fieldList* will be updated with this call.

Notes

None.

Return Values

```
TBL_NORMAL
TBL_BADHANDLE
TBL_DBNOEXIST
TBL_NOFIELDLIST
TBL_INSERTFAILED
```