July, 1996

Advisor Answers

FoxPro 2.x and Visual FoxPro

Q: I am having trouble with a multi-user app that adds a record to a table for each new transaction. It increments that table value by 1 with each pass and appends a new record with the new value. The problem I have encountered is when user X hits the last record at or near the same time as user Y, both users increment the value of the last record by 1 and the result is duplicate records (one generated by each user). I have tried FLOCK(), RLOCK(), with set reprocess to -1 and ON ERROR and even an IF RLOCK() = .F. loop. It seems that the trailing user will always hit the same record as the first unless SET EXCLUSIVE is set ON.

–Tony Frederick (via CompuServe)

A: The problem you're encountering is the reason that many people use a separate table to track unique id numbers. If you just grab the last value and increment, it's too hard to guarantee that several users don't do it at the same time.

Here's a better approach. One table (call it IdTable) can be used to keep track of id numbers for all the other tables in the application. The best technique is to put one record in the id table for each table that needs unique ids. The id table has two fields, one for the name of a table and one for the last used id in that table. It is indexed on the table name field.

A simple function called GetId then hands you a unique id whenever you ask. Here's the Visual FoxPro version. Just change LPARAMETERS to PARAMETERS and LOCAL to PRIVATE to use it in FoxPro 2.x.

```
PROCEDURE GetId
* Get a new unique id for the specified table.
* Assumes Ids are stored in a table called IdTable
* with a tag TableName on the TableName field.

LPARAMETERS cTable

LOCAL nOldArea,cOldOrder,lWasUsed,cRetValue

nOldArea=SELECT()

IF USED("IdTable")
   lWasUsed=.T.
   SELECT IdTable
   cOldOrder=ORDER()
ELSE
   lWasUsed=.F.
   SELECT 0
   USE IdTable
ENDIF

SET ORDER TO TableName

* Find this table
```

```
SEEK UPPER(cTable)

IF FOUND()
   * lock it, grab it and update it
   DO WHILE NOT RLOCK()
   ENDDO

   cRetValue=NextId

   REPLACE NextId WITH ;
           PADL(ALLTRIM(STR(VAL(cRetValue)+1, ;
                            LEN(NextId),0)), ;
                LEN(NextId),"0")

   UNLOCK RECORD RECNO()
ELSE
   * Add a record for this table
   INSERT INTO IdTable ;
       (TableName, NextID ) ;
       VALUES cTable, REPL("0",LEN(NextID)-1)+"1"
   cRetValue=REPL("0",LEN(NextID))
ENDIF

IF lWasUsed
   SET ORDER TO (cOldOrder)
ELSE
   USE IN IdTable
ENDIF

SELECT (nOldArea)

RETURN cRetValue
```

To get a new id number for a customer, you call it like this:

```
REPLACE CustId WITH GetId("Customer")
```

The routine assumes it will eventually be able to lock the right record in the id table. This is a fairly safe assumption if GetId is the only routine that's allowed to lock the records in the id table.

In Visual FoxPro, you can place a call to GetId as the Default value for a field and FoxPro automatically fills that field in whenever you add a new record. If you do so, it's wise to place GetId in the database's stored procedures so it can always be found. The TasTrade sample application that comes with Visual FoxPro uses a similar routine as the default value for a number of its id fields. Check out the Supplier table and the NewId routine.

This version of GetId assumes that all id fields in the system are the same length. If that's not a valid assumption, add a third field to the id table to hold the desired id length and modify GetId to trim returned ids to the appropriate length.

–Tamar