A DATABASE TEST AND REPAIR FACILITY

BY DUANE SOUDER

A Database Structure Test and Repair Facility

Databases may be corrupted in many ways. However, the most common way for a database to become corrupted is for a System Failure to occur during a DBPUT or a DBDELETE. IMAGE does not cause system failures by itself. The operating system will interrupt the execution of an IMAGE intrinsic for other processes (usually system processes). During the interruption, the operating system can encounter a situation from which it cannot recover. The result of course, is a system failure. There is a possibility that DBPUT or DBDELETE was doing internal pointer manipulation. If internal pointers were being changed, you have a physically corrupted database.

There is no current way to tell if the internal pointers were being changed at the time of the system failure. When the machine is restarted, the database is in an unknown state. The user has only a few choices available:

A) Just continue running and hope that there was no damage to the internal pointers, or

B) Use a program from the contributed library like DBCHECK (or DBGROOM, etc.) to try and find any problems, or

C) Do a DBUNLOAD followed by a DBLOAD.

Today, databases are being built larger and larger to meet the increasing demands of data processing. DBUNLOAD and DBLOAD

are not reasonable solutions unless the database is damaged beyond the "Point of No Return". This "Point of No Return" has been a gray area for sometime. This new tool (or utility) called DBTEST, will attempt to address this gray area and help to define its boundaries by recommending DBUNLOAD/DBLOAD only if the problems are severe enough to warrant this action.

DBTEST was designed to examine the internal structures of the database and if necessary, to repair any damage found. Database down time will be significantly reduced by not having to unload and then reload the database. The user does not need to know anything about IMAGE internals, how chains work or any of the internal pointers. The user needs only the schema of the database and some working knowledge of their applications. The user interface was made simple and there are helping routines throughout the program to explain situations and aid the user in making decisions. If internal damage is found, the user does not have to allow the program to repair the database. The user has the final word and the program will not modify any internal structures unless the ok has been obtained from the user.

IMAGE uses a simple data structure called a Doubly Linked List for building chains in DETAIL data sets and Synonym Chains in MASTER data sets. Synonym chains are formed in MASTER data sets when different key values hash to the same location. During the addition or deletion of data records, the doubly linked lists are modified to reflect the operation performed on the data. A

damaged database is the result of the modifications not being completed on the doubly linked lists. This is true for MASTER data sets as well as DETAIL data sets. DBTEST operates in maintenance mode like the other utility programs and it uses the IMAGE intrinsics along with privileged mode to examine and repair the internal chains.

DBTEST's internal pointer checking is simple and straight forward. For example, when you request to check a detail data set, DBTEST will examine two (2) things:

First, DBTEST will scan what is called the free record list. This list is a singly linked list of records that were deleted from the detail data set using DBDELETE. This free record list is the garbage collection mechanism that IMAGE uses during a call to DBPUT. The bit map is tested and the record itself is checked to be sure that it truly is a free record. Any necessary repairs to this list are attempted without user intervention. If a record is found that does not look like a free record, it will be displayed to the user with an appropriate message so that the user may recover any data in the record by examining the appropriate search items. This record will be delinked from the free chain list (so DBPUT will not use this record). If the free chain list cannot be repaired, an appropriate message will be displayed and the user will be given some alternatives based on the severity of the problem encountered.

Secondly, DBTEST will then ask the user to input the search item name and its value (also called the argument). DBINFO is then called to find out if the search item is valid. Assuming it is vaild, DBFIND will verify the argument and return the first and last record number in the chain along with the number of entries in the chain. DBTEST will then find out which master set and internal path is associated with that search item through a series of calls to DBINFO. (The record in the master set is obtained with DBGET to verify the internal pointers and the record number is kept for later access if necessary.) Using calls to DBGET, the chain is examined by making sure that the current record points back to the previous record that was examined. This process is continued until the end of the chain is reached or until a hole is found. (A hole is nothing more than a pointer to an invalid record.) While the links are being examined, a counter is incremented to compare against the master's entry count. The bit map is tested and the key is compared against the users input to make sure that a record with a different key value has not been linked into this chain. If a hole is found, DBTEST will then start from the bottom of the chain making the same checks above to insure consistency. The chain will be examined until another hole is found or the last record is reached while going forward. (In this case, the user will be told what has happened and will be asked if the chain should be repaired.) The counter will be compared against the masters record of the number

of entries and if they do not match, the user will be asked if this should be repaired also. If there was a hole, the user will be told how many entries were lost/gained based on the master's record of the number of entries in the chain.

When a MASTER data set is to be examined, DBTEST needs only to access that data set. The doubly linked lists are used only for synonym chain construction as described earlier. The master data set is then read serially until a record is found that qualifies as a potential primary entry. The count, beginning and end of the synonym chain are remembered and the synonym chain is then checked for consistency. (This examination includes bit map checking, a running count of the entries and backward/forward pointer checking.) If there are any holes, the user will be told of the problem and DBTEST will automatically start from the end of the synonym chain making all consistency checks as described above until another hole is found or the last record going forward is reached. Then the user will be asked to allow DBTEST to repair any damage after an explanation has been given. DBTEST's synonym count is then compared to the primary's count of synonyms. The result of this comparision will be displayed to the user.

These descriptions are the methods used in verification of DETAIL chains and MASTER synonym chains in IMAGE databases by DBTEST. The checking is simple and the resultant patches are seen to be both useful and powerful. DBTEST was originally

designed to examine internal chain structures and provide useful reports for further investigation. When it found internal problems, it was simple to implement code to fix the broken chains. This is how DBTEST evolved into a repair facility. With minor knowledge of IMAGE and a database schema, the user has a powerful tool which will help repair the database with minimum down time. This down time will of course be dependent on the size of the database and the extent of the user's knowledge of the applications.

There are some slides which show the useage of DBTEST and its user interface. All error conditions are not presented here but we feel that a reasonable amount are present to provide the user with a good idea how this tool may be used to help them should the situation ever arise.

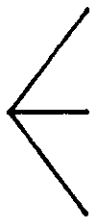
THIS IS

DBTEST

A.03.00 MON, APR. 27, 1981, 1:00 PM

DATABASE STRUCTURE (CHECK/REPAIR) TOOL

HELP(H) OR WHICH BASE OR 'CR' ? MFG

DATA BASE MFG  APPEARS TO BE OK.
WAS BEING ERASED.
WAS BEING MODIFIED WITH
OUTPUT DEFERRED !

OUTPUT TO TERMINAL (Y/N) ? Y

HELP(H) OR LOOK(L) OR 'CR' ? L

HELP(H) OR WHICH DATA SET OR 'CR' ? SUP

EXAMINATION OF DETAIL DATA SETS

HELP(H) OR SERIAL(S) OR CHAINED(C)
READS ON THIS DETAIL ? C

HELP(H) OR SEARCH ITEM NAME ? ACCT

HELP(H) OR ARGUMENT ? 153405

PRINT OUT ENTRIES IN CHAIN (Y/N) ? Y

WE COULD NOT GET THE N'TH FORWARD ENTRY IN THE CHAIN.
WE HAVE ENCOUNTERED WHAT APPEARS TO BE A BROKEN
DETAIL CHAIN WHILE USING THE FORWARD POINTERS. WE
SHALL ATTEMPT TO RUN THE BACKWARD POINTERS TO
RETRIEVE ANY OTHER ENTRIES IN THE CHAIN.

WE COULD NOT GET THE N'TH BACKWARD ENTRY IN THE CHAIN.
THE DETAIL CHAIN IS BROKEN IN BOTH DIRECTIONS.

NUMBER OF ENTRIES IN CHAIN ACCORDING TO MASTER = 10

HELP OR PATCH ENTRY OR DO NOTHING (H/P/N) ? P

YOU HAVE TWO (2) OPTIONS:

1) DO NOTHING AND CONTINUE SEARCHING
2) PATCH

   NOTE: THIS PATCH WILL DO THE FOLLOWING:

   a) PATCH ENTRY
      FORWARD = LAST GOOD ENTRY (GOING BACKWARD)
   b) PATCH ENTRY
      BACKWARD = LAST GOOD ENTRY (GOING FORWARD)
   c) PATCH PRIMARY
      ENTRY COUNT = CURRENT NUMBER OF ENTRIES

WHICH OPTION ? 2

ARE YOU SURE ? Y

ENTRY COUNT KEPT IN PRIMARY IS CORRECT.

OR ENTRY COUNT KEPT IN PRIMARY IS BAD.

NUMBER OF ENTRIES $\begin{matrix}\text{GAINED}\\\text{LOST}\end{matrix}$ = 1

EXAMINATION OF MASTER DATA SETS

HELP(H) OR CHECK A PARTICULAR ENTRY(PE)
OR CHECK ALL SYNONYM CHAINS(SC)
OR PERFORM SERIAL READ(SR) ? PE

PRINT OUT PRIMARY (Y/N) ? Y

PRINT OUT ALL SYNONYMS (Y/N) ? Y