ADD NEW CAPABILITIES TO IMAGE/3000
BY CAPTURING TRANSACTIONS

ROBERT M. GREEN

ROBELLE CONSULTING LTD.
#130-5421 10th Avenue
Delta, B.C. V4M 3T9
Canada (604) 943-8021

ABSTRACT: This paper is addressed to the manager or designer of IMAGE/3000 on-line applications. I raise questions regarding throughput, integrity, response time, design ease, maintainability and failure recovery.

I suggest the use of an interface between the application programs and the database system to answer some of these problems. One such interface that I have developed is called SNAPSHOT, and is currently in use at a test site.

QUESTIONS FOR THE DATABASE MANAGER:

Q: Why should my on-line users have to wait at the terminal for non-urgent DBPUTs and DBDELETEs to complete processing?

Why should I have to artificially simplify my database design (by deleting paths) to provide on-line users with fast data-entry response times?

A: Let SNAPSHOT capture and defer instead.

WHAT IS DIRECT TRANSACTION PROCESSING?

Perform transactions on the database as they are collected from the on-line users. Since the update is done by the collection program, the user waits.

Presumably, other on-line users must be able to check this data at any time.

Lock records during updates for integrity.

Immediate update yields the slowest response time, but is the "normal" method (used by QUERY/3000 and most applications).

You capture the data, IMAGE organizes it. Usually, the organizing (indexing, sorting, posting to the disc) uses more resources than the collection.

WHAT IS DEFERRED TRANSACTION PROCESSING?

Gather and check data from on-line users, but merely capture the database changes for later update by a background utility program. This means throughput is increased.

Gather all data for a "GROUP" of related transactions before committing to final database changes. This means database consistency.

"HOLD" existing records until their updates are completed (perhaps hours later), even if the system crashes.

Trade off timeliness against response time and control. Do I really need that data in the database right now?

SNAPSHOT: CAPTURE NOW, ORGANIZE AT LEISURE

SNAPSHOT/ROBELLE is a software interface to IMAGE/3000 that takes care of capturing IMAGE/3000 transactions for deferred update.

Private workfiles for each user hold transactions against multiple databases. Transactions can be direct or deferred.

Background "UPDATE" program finishes deferred updates, and cleans up (i.e., completes or backsout) transactions after a crash.

Since Puts/Deletes take much longer than Updates (about a factor of ten), why not do Updates directly (to modify on-line status), and let the new records (Puts) follow in a deferred manner?

QUESTIONS FOR THE DATABASE MANAGER:

Q: What happens to my application programs if I change the structure of my database?

What if I want to use "@" in my calls to IMAGE to improve performance, but I also want data-independence?

What if I expand the Customer-Number and forget to change the buffers in my application program?

A: Let SNAPSHOT bind your programs to your databases at run-time.

RUN-TIME BINDING

An initial call to SNAPSHOT binds your program and field-lists to the database. You specify field-name-lists and set-names in a subprogram (using recoverable memory space).

SNAPSHOT checks the field-list and passes it to IMAGE. This allows you to use "*"-field-list and set-numbers in the rest of your program.

SNAPSHOT checks that the buffer size required for each dataset has not changed.

SNAPSHOT substitutes "@" whenever your field-list contains all fields of the set in the proper order.

Initial data binding provides full data-independence, in spite of database changes, and most efficient use of IMAGE by the program.

QUESTIONS FOR THE DATABASE MANAGER:

Q: How far from "standard" IMAGE will my programs wander if I use an interface package?

Who will write all of this initial run-time binding code (specifying field-lists and set-names)?

A: SNAPSHOT calls are so close to standard IMAGE calls that most code can be converted automatically with a few EDIT/3000 commands. Where special calls are needed (as in the initialization routine), SNAPSHOT provides a program to generate the source code automatically. It is no longer necessary to calculate the length of field-list buffers by hand; SNAPSHOT does this mechanical programming for you.

Q: Why should my system have to be down for hours in order to :RESTORE and roll-forward after every operating system failure?

Why should I have to worry nights about an application aborting and leaving the database contaminated?

A: Let SNAPSHOT capture and backout instead.

TERMINATIONS: FROM AMICABLE TO PSYCHOTIC

Normal program end - (sigh of relief)

Program bug - (when and how will I detect it?)

Program abort - (what was it doing?!)

System problem - (i.e., disc failure, but no crash)

System crash - warmstart (same as multiple aborts)

System crash - reload (transactions lost)

WHAT IS TRANSACTION LOGGING?

Uses "after" pictures of entries to roll-forward from a good backup copy after a crash.

Time-consuming; usually several hours.

Assumes a good backup copy of the database(s) and all transactions until the present. May fail if a transaction is lost.

Should leave the databases clean (i.e., no partially completed "groups" of transactions).

Most common reasons for rolling-forward are catastrophic system failure and operator (or program) error (database damaged).

WHAT IS TRANSACTION BACKOUT?

Need "before" pictures of entries to undo the transactions.

Used after a soft crash or program abort, instead of a full roll-forward; gets the system on-line faster.

Backout can be done by an application program itself (if it finds an irrecoverable error part way through a set), or by a stand-alone utility.

Records changed by the set must be "held" away from other users until the backout is done and the records are once again consistent.

WHAT IF I HAVE MULTIPLE DATABASES?

During a roll-forward (or backout) recovery, do I want to update the customer's balance if his sales order is not updated (in another database)?

Do my logging/backout methods recognize that transactions on different databases may be critically related?

If the answer is no, can I afford to merge all of my data into one IMAGE/3000 database (255 items!)?

THE PROBLEM OF INTERVENING UPDATES

What has happened to this customer in the database since I retrieved him and displayed him on the screen for my operator to change?

By keeping the record (or database) locked, I can be sure that nothing happened; but it may stay locked for a long time.

Why couldn't the system tell me if anyone had changed my records since I retrieved them? If I can detect intervening changes to the entries, then I can cancel and redo my own.

FAIL-SOFT RECOVERY DESIGN

Eliminate heavy dependence on record numbers, both in masters (migrating secondaries) and details (lost transactions).

Eliminate heavy dependence on the time-sequence of updates. Within a transaction workfile, each "group" of transactions is unrelated to the others and can be completed even if the others fail.

"Hold" records to prevent updates by other users, even if the holding user aborts or the system crashes. Stronger than a lock.

After a crash, the databases and transaction files must be cleaned up before full processing can be allowed to continue.

SNAPSHOT SURVIVAL PACK

Backout of "direct" transactions via the workfile.

Archiving and auditing transactions.

You can lock a set of records even across system crashes.

Handles transactions for multiple databases as a single "group", not as separate "groups".

Intelligent crash recovery.

QEDIT: A SYSTEM LOAD ANALYSIS

ROBERT M. GREEN

ROBELLE CONSULTING LTD.
#130-5421 10th Avenue
Delta, B.C. V4M 3T9
Canada (604) 943-8021

ABSTRACT: The system load of any program on the HP 3000 is composed of cpu time consumed, disc accesses demanded, data stack size and code segment number, size and frequency of transfer. QEDIT is an attempt to make on-line programming more practical by reducing the cost and customizing editing features for programmers.

Many HP 3000 users underestimate the resources consumed by on-line programming. What does a programmer go through to accomplish a simple, one-line change to a program on the HP 3000?

> Using EDIT/3000, the programmer "texts" a copy of the program into the editor workspace, then modifies a line and "keeps" another copy so that it can be compiled.
>
> Assuming a 3200-line source file, this innocent-sounding cycle will take from 3 to 10 minutes (depending upon the system load) and will consume these resources:
>
> 76 c.p.u. seconds
>
> 1236 d-i-s-c transfers

What is the effect on the system response time while this program change (TEXT/KEEP) is occurring in the computer? Take a look at an actual HP 3000 in commercial usage:

> Before the program change begins, the response time on this 512K-byte system is immediate (less than a second) for inquiries into a database.
>
> During the time of the change, the response time on all terminals degrades to 2 seconds.
>
> Now put three programmers on the system and let them do their changes concurrently: response time stretches out to 6 seconds.

What options does the system manager have to restore response times to their former level?

> Force programmers to work nights and weekends?
>
> Upgrade to a larger computer?

Or, the system manager can try another method of on-line programming that eliminates the TEXT/KEEP cycle. With QEDIT, instead of using TEXT/KEEP, the programmer OPENs his "source" file (maintained in a special format) and modifies it directly. The same one-line program change (referred to on the previous page) can be done in seconds, consuming only 2 c.p.u. seconds and less than 10 disc transfers.

User response times are not degraded by on-line programming. Switching a typical installation to QEDIT releases the computer power to add more user terminals, run an extra batch job, or give every programmer a personal terminal.

In addition to QEDIT's economical use of c.p.u. time and disc transfers, here are examples of other measures of system load:

Using "OVERLORD", from the Contributed Library, you can compare the data stack sizes of QEDIT and EDIT/3000:

QEDIT: 3600 words

EDIT/3000: 6300 words

Using "PROGSTAT", from the Contributed Library, you can compare the code segment sizes of QEDIT and EDIT/3000:

QEDIT: 7 segments, 15600 words

EDIT/3000: 12 segments, 33400 words

QEDIT is about 1/2 the size.

An actual comparison of disc space using a 3200-line source file:

```
EDIT/3000:  1025 sectors for a "KEEP" file.
      PLUS  1800 sectors for a working copy.
            ------------
      TOTAL 2825 sectors for one program.

QEDIT:       390 sectors for a source file.
      PLUS     0 sectors for a working copy (not needed).
            -----------
      TOTAL  390 sectors of disc consumed.

   SAVINGS:  2435 disc sectors for ONE program.
```