Worldwide Response Center

HP 3000 APPLICATION NOTE #54

Improving Database Performance

HEWLETT PACKARD

April 01, 1989
Document P/N #5959-9251

RESPONSE CENTER APPLICATION NOTES

HP 3000 APPLICATION NOTES *are published by the Worldwide Response Center twice a month and are distributed with the Software Status Bulletin. These notes address topics where the volume of calls received at the Center indicates a need for addition to or consolidation of information available through HP support services. Following this publication you will find a list of previously published notes and a Reader Comment Sheet. You may use the Reader Comment Sheet to comment on the note, suggest improvements or future topics, or to order back issues. We encourage you to return this form; we'd like to hear from you.*

IMPROVING DATABASE PERFORMANCE

DATABASE PERFORMANCE ISSUES

Many questions are received by the Response Center on TurboIMAGE performance.

"How long is the DBUNLOAD/DBLOAD going to take?" "My program takes a long time to update the database now. It used to be fast. What happened?" "I am designing a program/schema now. What can I do to have the best design for performance?" These are just a few examples, there are many more. Let us approach performance from three sides. First, design features for optimal performance will be discussed, then maintenance functions to help achieve better performance, and finally programmatic procedures for best results.

A. DATABASE DESIGN CONSIDERATIONS OR WHAT THE DATABASE ADMINISTRATOR CAN DO

How may features be used in the DESIGN PHASE to obtain optimal performance from a database? The effects of poor database design may be most obvious on a DBLOAD. TurboIMAGE is performing massive DBPUTs as data is loaded back into the database. If this process seems to take too long, then the time has come to look at the database design factors discussed here.

Many factors may affect the response time of a program that is accessing a database. How can a database be designed that will result in faster response to the particular demands of a particular set of users? By first examining the database concepts using schema structure, the optimal structure for performance may be determined.

BLOCKSIZE

The $CONTROL statement has the option of altering the blocksize through the use of the BLOCKMAX=nnnn. The blocksize chosen determines the size of the buffers in the extra data segment known as the DBB (DATABASE BUFFERS) which is created at the time of the first DBOPEN of a database by the TurboIMAGE intrinsics. These buffers will all be a uniform size, a "one size fits all" concept. This extra data segment, normally resident in memory, acts as traffic cop for input and output to the database. If updates to the database will occur using batch applications at a time when memory contention is low, say at night, then the blocksize should be large. On the other hand, if updates occur through application programs when many users are active and memory contention is high, then block size should be smaller. Disc caching is a mitigating factor in this process which will be discussed in a later section.

How applications access data is an important consideration. If chained reads to details or frequent key item reads to a master are used, then smaller blocks are more effective since there will not be a high read hit in the blocks read into the buffers of the DBB. On the other hand, if the database is accessed using serial reads, then large block sizes will lead to a better read hit ratio. Another consideration to block size might be the largest record size in the database. If the records in the database are large, then use a larger block size.

PASSWORDS/SECURITY

Passwords and security are also a performance consideration. Using set level security will cause less overhead than item level access, because of the additional security checking that must be done on the item level. This overhead is actually CPU time plus the additional time involved in queuing for the CPU. Password security has the same overhead.

ITEM LIST ORDER

Put items into the schema list in the order they will be used. In TurboIMAGE items are hashed and loaded into a table in the root file. This table allows faster access of items so at the time of the first DBOPEN they are loaded into the DBG (DATABASE GLOBALS DATA SEGMENT) for faster access. Secondary chains may exist in this area of the root file. So to make certain the most used items are primary entries, they are added first and therefore chasing synonym chains is avoided.

As in security checking, the overhead is CPU and queuing time.

SET USAGE - MASTERS AND DETAILS

The possible uses of data sets themselves in a data processing environment are important in the performance arena.

Why make a set a detail or master?

Consider the record information. Is it summary type or detail? If the grouping of the data in this data set will be of summary type and will have a limited number of data items that are static, then this should be a master. On the other hand, if the set will contain a larger number of data items, there are lengthy fields, the data is constantly changing, the fields describe data that is unique, or the set contains an item that is related to other members of the set, this is a detail data set.

Stand alone masters are useful for table lookups, while stand alone details are useful for organizing data with possible security needs associated with them. In this way TurboIMAGE security may be used to apply to data rather than requiring a program to be written to create security.

Automatic masters are easier for users but involve more processing behind the scenes by TurboIMAGE intrinsics. The TurboImage intrinsics will be performing all the updates to chain information as well as necessary additions or deletions to the automaster.

PATH USAGE

Now consider the linking of masters to details. Using masters linked to details requires updating chain information in the chain head when adding and deleting from associated details. The chain head, present in the master entry, contains chain count as well as forward and backward pointers.

First consider if each path is really needed. Consider carefully whether or not this particular search item will be used often enough to make it a search item. If paths are to be used infrequently by application(s) or the search item will have only a few possible values, such as "Y" or "N", then use as a path should be avoided.

Because of the overhead involved with each chain update, it is best to go easy on multiple associations of details to masters. Or, in other words, it is better to use more details and fewer masters due to the overhead associated with chain updates. (Another problem with multiple relationships is the additional disc space required in the masters due to the space needed for each path.)

It is also important to define the path most often used between a master and a detail as its primary path. The default is the first path defined in the schema. (To change from the default include an " ! " prior to the path in the schema, and this will become the primary path.) Know the data. What item should be the key into the detail? Chained access is direct, while serial access is time consuming.

SORTED PATHS

Sorted paths can have a major impact on performance. In placement of records into a sorted chain, the chain is read backwards and comparisons are made between the previous entry read and the current entry to find the correct logical position for the entry that is being added. The LENGTH of the chain is important. The reason being that to place a new entry the entire chain will be read until the sort item which will - according to the collating sequence - preceed, and the record which will follow the newest entry is found. If the chains are long, then more records must be read. Long sorted chains are a surefire performance problem.

The PLACEMENT of the sort item within the record also will have great impact on performance. A chain sort is a collating/sort process. If an identical sort item value is found, the sort will continue into the next field(s) until the collation of records determines which should maintain fore and aft positions. If the sort item is at the beginning of a record, this extended sort might continue to the end the record. This will have a magnified effect if the chain is long and the sort item is at the beginning of a record. For example, if all records in a data set were in the same sorted chain and/or the sort item was poorly placed-such as in the beginning of the record- either adding or deleting a record would become a noticeably time consuming process.

The number of sorted chains may also become a problem. This is a geometric problem: length of chain, placement of sort item and number of sorted chains are all factors that determine how many I/O's are to be needed to add and delete items in a data set. These all decrease performance in a geometric fashion.

SYNONYMS

TurboIMAGE intrinsics determine where to place records in master data sets by "hashing". This will not be a discussion of how hashing is done but where database design can impact performance and how. For a discussion of hashing refer to Chapter 10 of the TurboIMAGE Manual (Part No. 32215-90050). Hashing is actually a method of finding a unique position within a range of values between 1 and the capacity for a master entry. The search item together with the capacity of the data set is used in a mathematical formula to generate this unique position which will be used to place the record physically in the data set. If a record position generated for a new record is identical to one already generated and is still in use by that record, the newest record is placed nearby and becomes a SECONDARY or SYNONYM of the original record. Then the chain information is updated to keep track of these similar records. These are called SYNONYM CHAINS.

When new records hash to the same location, synonym chain information must be updated to include new members. When the synonym chain head is deleted, the first member of the chain will "migrate" or physically move to the location occupied by the old chain head and the chain will be updated to reflect its new head. A migration will also occur if a member of a synonym chain (a secondary member instead of the chain head) is in a physical location to which a new record hashes. The secondary record will migrate to a new location.

This overhead of moving records and chain overhead can be limited by a proper use of search items and capacity. Since the point is to create a unique number then the two factors that are controlled by the designer of the database: search item and capacity, should be optimized.

The best search item to use to generate the most unique numbers is a item defined as a type U, X, S, or P. The reason for this is these types are used in the hash routine in a different manner than types I, J, K, or R. Types U, X, S, or P use the whole search item value no matter what its length to obtain a positive doubleword value while types I, J, K, or R use the low order 31 bits to form a doubleword value. In other words more bits are used in the case of a U, X, S, or P type and therefore a better randomization occurs.

However, if the key MUST be an integer, then keep the range of values for the search item within the value assigned to the capacity. Every key will hash to a unique location, with NO synonyms! With this hashing setup the data set can run up to full capacity without performance degradation. The capacity is used in the formula to find a remainder from the doubleword value discussed above. One school of thought suggests that prime numbers for capacity cause a better distribution than even numbers. The reasoning behind this is a prime number would generate less even number remainders and therefore create more odd number results resulting in more unique numbers needed to prevent the synonyms and the problems discussed above.

AUTO DEFER

When the time has come for batch processing, performance times may be enhanced by the setting of the AUTODEFER flag through DBUTIL. This sets a flag in the root file informing TurboIMAGE extra data segments that large blocks of data only will be coming. These large blocks of data will fill the buffers of these extra data segments, then, and only then, will they be posted. The default is that with each intrinsic level transaction the buffers are posted. A word of caution, since the file labels and buffers are not posted with either ILR nor ROLLBACK is compatible with this method of updating. The database should be backed up prior to setting this flag and processing.

B. MAINTENANCE PRACTICES FOR GOOD PERFORMANCE OR WHAT THE SYSTEM MANAGER CAN DO

If the database is already designed and loaded with data, but response times are getting longer and longer, what can be done to improve response?

MIXES OF BATCH AND REAL TIME APPLICATION

Take care mixing online user applications with batch application processes. Consider users and choose non peak times to run batch applications which do many updates of a database or databases.

DATA SET LOADING

A utility such as DICTIONARY/3000's DICTDBA (Database Audit Utility) or Query/3000 and some calculation, may be used to determine the fill percentage of master data sets. A rule of thumb on this percentage is not to allow a master to become more than 80% full. The reason being that beyond that range the likelyhood becomes very high that the synonym chains already existing will become a more serious performance problem. The problems arise in adding, searching blocks for an empty record in which to place the new arrival and in deleting the records with the resultant migration of synonyms. In other words it is simply an amplification of an existing problem which in actuality becomes geometric in terms of performance as the data base fills.

Detail data sets may begin to have performance problems at this point as well, especially if the detail set has been volatile. A detail set with frequent deletes and adds will lead to a lengthy delete chain. A delete chain is the TurboIMAGE mechanism to reuse space occupied by previously deleted records in a detail set. This is chain information within the detail set which is modified with each delete and add for that set. Once again this is simply a revelation of a pre-existing problem which appears when data sets are nearing capacity. Another issue with volatility of details is the chains that previously occupied a single block, therefore contiguous on disc, begin to scatter to different blocks, spread out on the discs. This results in the overhead that is most costly: I/O on discs. This may be somewhat offset by disc caching which will be discussed.

CLUSTERS

Clusters may occur in a master data set regardless of fill rate. Clusters occur when the search item and capacity result in many values for the search item hashing to the same location. Since changing the capacity of the master data set alters the formula for hashing, a change in capacity is used to redistribute the records of a cluster.

REDUCING DISC CONTENTION

The most volatile data sets should be separated across discs to reduce disc contention. Physical I/O is the most time costly activity on the system. If queuing occurs as well, say for the same disc, performance suffers. Masters and related details should be spread across discs to achieve good results. The master should be placed on one disc and its related detail on a different disc. DBUTIL's MOVE command can be used as well as STORE/RESTORE with a "DISC=" option to achieve this result.

DISC CACHING

DISC CACHING is an MPE subsystem which utilizes excess memory and the CPU to keep certain frequently used areas of disc in memory. The idea being that disc I/O is slower than memory access, so disc caching creates a buffer storage area for frequently used disc files.

When a process requests a read, if the area is already in main memory the read will take a fraction of the time required for the disc I/O to be completed to retrieve the requested data.

When a process requests a write, if the area is in main memory queuing must be done to wait for the previous write to post. If it is not in memory only the changed area will be written and the process does not have to wait for completion to continue processing. This is handled by disc caching.

OVERHEAD OF LOGGING AND ILR

Logging and ILR both cause overhead. Prior to a change in the database, the log file and the ILR files must be posted with new information which includes before and after images of record, file label, and user label. Log files may be either disc or tape so we incur the expense of I/O for each change to the database. ILR is a disc file so the expense is once again the I/O of disc update. The advantages of these activities must be weighed against the cost of the I/O activities involved here.

DBUNLOAD/DBLOAD - CHAINED

Perhaps a long time has gone by since the last DBUNLOAD/DBLOAD. It might be of great value to perform this function to achieve better performance. For detail sets the delete chain will be removed and data compacted, no matter which option is taken: Chained or Serial. For masters unless the capacity is changed (use a prime number) the location of synonyms will remain the same.

If the DBUNLOAD is done serially then broken chains will be rebuilt along all paths. If performance is the objective then a chained DBUNLOAD will rebuild along the primary path AND place members of the same chains in the detail set physically next to one another to allow faster access especially with DISC CACHING.

Remember this will only improve access time in detail sets since chain members are residents there but a change to the capacity might generate fewer synonyms as each master record is rehashed.

C. PROGRAMMATIC DESIGN OR WHAT THE PROGRAMMER CAN DO

The database is in place and loaded. What is the best way to access it and have the fastest response times? Programming practices will have great impact on what the user perceives as performance.

MISCELLANEOUS

Consider the data type and language used. For example, avoid use of FORTRAN if the database is mostly of character data since FORTRAN is designed for scientific and mathematical computing. Similarly avoid use of COBOL if the database contains many REAL data types since COBOL will not be able to contend with REAL formatted data.

Keep code localized. Try to keep procedures that call one another within the same segment.

Trap for errors. This will avoid time consuming debugging when problems do occur.

Use Query as an aid to debugging. If the program is written and in testing phase, Query may be used to see how the program is deals with data.

Remember changes to a sort item involve doing a DBDELETE and DBPUT instead of a DBUPDATE.

If possible, sort the data before adding it to sorted chains.

Avoid the use of lengthy chains in chained reads. Keep chained reads short.

In programs having tables that are frequently used, think of using a dedicated database for lookups. Static "lookup" type information such as territory information or product categories is perfect for database use. This information could be updated easily if needed and most important, during processing time, would be loaded into TurboIMAGE buffers in main memory allowing for cheap (do not use any user stack space), easy access.

USE OF LIST parameter

When designing a program which accesses a database, a list parameter must be used to indicate what data items are to be accessed. There are several ways of programmatically completing this list. Use of item names, item numbers, "@" (all), and "*" (current) are the possible designations. Table A below shows the activities required for satisfying each type:

TABLE A

"LIST" OVERHEAD FUNCTIONS PERFORMED ON YOUR BEHALF

Procedure	Item Name	Item Number	@	*
DBGET	ADBJEG	ABJEG	ACBJEG	AJF
DBPUT	ADBIJ	ABIJ	ACBIJ	AJF
DBUPDATE	ADBHJ	ABHJ	ACBHJ	AJF

A	- MOVE LIST TO TRAILER AREA OF TURBOIMAGE-USER EXTRA DATA SEGMENT
B	- CHECK ACCESS SECURITY
C	- REPLACE "@" WITH NUMERIC LIST
D	- CONVERT NAME TO ITEM NUMBER
E	- MOVE ITEM TO GLOBAL TURBOIMAGE EXTRA DATA SEGMENT
F	- MOVE ITEM FROM GLOBAL TURBOIMAGE EXTRA DATA SEGMENT TO USER STACK
G	- MOVE TRAILER AREA OF TURBOIMAGE-USER EXTRA DATA SEGMENT TO USER STACK
H	- MOVE ITEM FROM USER STACK TO TURBOIMAGE-USER EXTRA DATA SEGMENT
I	- KEY/SEARCH ITEM CONFIRMATION
J	- INTRINSIC DOES OTHER THINGS

NOTE

The " @ " parameter shows the first use. Each additional use is the same as " * ".

The important thing here is the number of tasks in each type of list parameter. Notice that " * " has the least while item name has the most. So in the range of most efficient to least efficient the parameters should be used in this order: " * ", item numbers, " @ ", followed by item name.

LOCKING

Locking has more impact on perceived performance than any other factor. Locking is a necessity and cannot be tossed aside if there is any concern about the validity of data. Wisely used, locking can be transparent to the user and need not impact performance.

First there are a few important things to remember about locking. If a process is using MR (multiple rin) capability, to avoid a deadlock - which is the ultimate performance block - lock in a predefined sequence and unlock in the reverse order. Back out the way you came in. Also remember locking is unnecessary if DBOPENing in modes 3, 7, and 8. If using chained or calculated, reads, lock at the entry level, but if using serial or directed reads lock at the set or base level.

The golden rule of locking is: "The longer the lock will be held the lower the level it should be." In other words do not program a base level lock, which is meant to allow users to enter multiple changes across sets (maybe go to lunch), then unlock. (This is poor data security as well.) Lengthy and/or complex transactions should be locked at the entry level. It is a bit more programming, however it results in infinitely happier users. Set level locks are acceptable when doing approximately 8 TurboIMAGE intrinsics. Another way to minimize locking problems, is to lock at the same level throughout all process that share a resource. Do not allow one process to lock at the set level while another locks at the item level. This results in bottlenecks. A process must have an "all clear" or green light before proceeding through the various levels to achieve the desired level of lock.

The last rule of locking is that when locking at the entry level it is important to always lock on the same item in any one data set. If this rule is ignored then TurboIMAGE intrinsics will generate a set level lock and the extra overhead of the entry level lock will be incurred. An example of locking on multiple items in a set: attempting to lock on the item DOG=MASTIFF while a different user tries to lock in the same set on the item SIZE=BIG. Both locks cannot be granted at one time since DOG=MASTIFF is a subset of SIZE=BIG records, so some of the records meeting criteria might be identical for both users. If however one item had been chosen as the most appropriate, say SIZE=BIG then one lock at a time can be processed.

BACK ISSUE INFORMATION

Following is a list of the Application Notes published to date. If you would like to order single copies of back issues please use the *Reader Comment Sheet* attached and indicate the number(s) of the note(s) you need.

Note #	*Published*	*Topic*
1	*2/21/85*	*Printer Configuration Guide (superseded by note #4)*
2	*10/15/85*	*Terminal types for HP 3000 HPIB Computers (superseded by note #13)*
3	*4/01/86*	*Plotter Configuration Guide*
4	*4/15/86*	*Printer Configuration Guide - Revised*
5	*5/01/86*	*MPE System Logfile Record Formats*
6	*5/15/86*	*Stack Operation*
7	*6/01/86*	*COBOL II/3000 Programs: Tracing Illegal Data*
8	*6/15/86*	*KSAM Topics: COBOL's Index I/O; File Data Integrity*
9	*7/01/86*	*Port Failures, Terminal Hangs, TERMDSM*
10	*7/15/86*	*Serial Printers - Configuration, Cabling, Muxes*
11	*8/01/86*	*System Configuration or System Table Related Errors*
12	*8/15/86*	*Pascal/3000 - Using Dynamic Variables*
13	*9/01/86*	*Terminal Types for HP 3000 HPIB Computers - Revised*
14	*9/15/86*	*Laser Printers - A Software and Hardware Overview*
15	*10/01/86*	*FORTRAN Language Considerations - A Guide to Common Problems*
16	*10/15/86*	*IMAGE: Updating to TurboIMAGE & Improving Database Loads*
17	*11/01/86*	*Optimizing VPLUS Utilization*
18	*11/15/86*	*The Case of the Suspect Track for 792X Disc Drives*
19	*12/01/86*	*Stack Overflows: Causes & Cures for COBOL II Programs*
20	*1/01/87*	*Output Spooling*
21	*1/15/87*	*COBOLII and MPE Intrinsics*
22	*2/15/87*	*Asynchronous Modems*
23	*3/01/87*	*VFC Files*
24	*3/15/87*	*Private Volumes*
25	*4/01/87*	*TurboIMAGE: Transaction Logging*
26	*4/15/87*	*HP 2680A, 2688A Error Trailers*
27	*5/01/87*	*HPTrend: An Installation and Problem Solving Guide*
28	*5/15/87*	*The Startup State Configurator*
29	*6/01/87*	*A Programmer's Guide to VPLUS/3000*
30	*6/15/87*	*Disc Cache*
31	*7/01/87*	*Calling the CREATEPROCESS Intrinsic*
32	*7/15/87*	*Configuring Terminal Buffers*
33	*8/15/87*	*Printer Configuration Guide*
34	*9/01/87*	*RIN Management (Using COBOLII Examples) (A)*
34	*10/01/87*	*Process Handling (Using COBOLII Examples) (B)*
35	*10/15/87*	*HPDESK IV (Script files, FSC, and Installation Considerations)*
34	*11/01/87*	*Extra Data Segments (Using COBOLII Examples) (C)*
36	*12/01/87*	*Tips for the DESK IV Administrators*
37	*12/15/87*	*AUTOINST: Trouble-free Updates*
38	*1/01/88*	*Store/Restore Errors*
39	*1/15/88*	*MRJE Emulates a HASP Workstation*
40	*2/01/88*	*HP 250 / 260 to HP 3000 Communications Guidelines*
41	*4/01/88*	*MPE File Label Revealed - Revised 6/15/88*
42	*7/15/88*	*System Interrupts*
43	*7/15/88*	*Run Time Aborts*
44	*8/01/88*	*HPPA Pathing Conventions For HP3000 900 Series Processors*
45	*8/15/88*	*Vplus & Multiplexers*

READER COMMENT SHEET

Worldwide Response Center Support
HP 3000 Application Note 54: Improving Database Performance
(April 01, 1989)

We welcome your evaluation of this Application Note. Your comments and suggestions help us to improve our publications. Please explain your answers under Comments, below, and use additional pages if necessary.

Is this Application Note technically accurate?	☐ Yes	☐ No
Are the concepts and wording easy to understand?	☐ Yes	☐ No
Is the format of this Application Note convenient in size, arrangement and readability?	☐ Yes	☐ No

Comments and/or suggestions for future Application Notes:

This form requires no postage stamp if mailed in the U.S. For locations outside the U.S., your local HP representative will ensure that your comments are forwarded.

FROM: **Date** ______

Name ______________________

Company ______________________

Address ______________________

____

____

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 95, MT. VIEW, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Application Note Comments
Hewlett-Packard Manufacturing Specifications
690 E. Middlefield Road
Mail Stop 30-0
Attention: AN ORDERS
Mt. View, CA 94043

FOLD FOLD