An Evaluation of Data Base Performance Tools

Thomas Gosnell

Infocentre, Montreal, Canada

Overview

Fourth generation languages have become a common toolset, if not the toolset of choice, for most HP3000 development shops. The increase in use of fourth generation language and the large amounts of systems which have been built have resulted in the need for not only more functionality but greater performance.

During the early years of fourth generation language use, the need of data processing management was to increase programmer's productivity, release systems to the users faster and reduce the backlog of application systems. At the time, the driving force for leading fourth generation language vendors was to provide greater functionality and more productive means. Thus, succeeding in helping data processing departments deliver more application systems faster, more attention has been focused on optimising the runtime performance.

Leading fourth generation language vendors have in the past, and continue to address, the need for improved system performance by investing research and development resources in advanced alternate processing technologies. Two methodologies which have had success in improving Turbo Image../ Image throughput and performance are high speed indexing and fast serial read technology.

Turbo Image/3000, hereafter called just Image, has proven itself to be a reliable, flexible, efficient, and in many ways a very fast data base management system. Used by the majority of HP3000 application system users, Image access is a fundamental part of most fourth generation language's and can be arguably attributed to the success of many applications on the HP3000.

Image however has limitations and bottlenecks which limit fourth generation language processes from achieving maximum performance in both interactive and batch processing. Image's speed in interactive access can be excellent by using hashed master files to quickly retrieve data entries with complete key values. Yet, limitations exist preventing the rapid retrieval of data by secondary key value in the master sets, partial key retrieval in the master/detail sets, and relational conditions. Such constraints are addressed by building indexes on top of the existing Image structure to provide increased functionality and higher speed of retrieval.

Batch processing typically involves reading data sets serially, extracting the records meeting various selection criterias and subsequently processing the valid entries. This procedure is often restricted to the speed at which the records are accessed and read from the data base. There are parameters in both Image and MPE which may be changed to tune performance. These include buffspecs, blocksize, cache block size; all can make a significant improvement in performance or degradation but cannot overcome the overhead associated with Image serial read performance. However, fast serial read technology addresses the problem with impressive results. The present paper will review these complementary technologies in a generic fashion, providing the reader with some general guidelines for the best fit scenarios.

Capability sets and performance will differ between vendors of similar products. It is the purpose of this paper to illustrate the comparative advantage of the different technologies and not provide product comparisons or definitive statements on performance. As such not all listed capabilities will exist in similar products, nor will performance figures be the optimal available. All tests were executed on a HP3000/930 running MPE/XL A.01.20 in compatibility mode.

Indexing Systems

There exists many types of indexing techniques and technologies which increases the speed in which data may be accessed. Every viable algorithms possesses characteristics which assists in defining performance and functionality versus resources consumed . Hashing techniques, such as used by Image master files are arguably the fastest least consumptive methodologies available, but require that key items will hash evenly and will be suitable for retrieval by their full key value.

In the real world of data processing, many situations arise whereby user retrieval requirements dictates a data model which does not correspond to a proper fit to the Image data base management system technology. There are without questions, techniques and tricks which have been used to force fit Image with application systems. Such methods are not always efficient or easy to program using a fourth generation language.

An alternative data indexing technique to hashed master files is the binary tree concept, used with KSAM on the HP3000 and HPIMAGE on the 9000/8XX machines. B-Trees possess many characteristics which make them a very good alternative and an excellent complement to hashed master files.

B-Trees, unlike hashed techniques keep key values in sorted order with index blocks and pointers to quickly locate key items with partial as was well as full key values. Such characteristics allow B-Trees indexes to serve as foundation to full feature data indexing and enhanced data base management system functionality.

Implementation of the technology differs for enhanced Image/fourth generation language data base access, but typically they consist of various programs and intrinsics which support the complete development and runtime environment. Such routines allow the definition, installation, maintenance, programming and runtime support needed for external access.

Capabilities

Data entry retrieval for indexed data items are possible by:

- item value;
- partial item value;
- key word;
- partial key word;
- multiple item;
- multiple item, multiple values;
- relational conditions between items;
- multi set retrieval;
- field grouping

Implementations

Index data structures are built using vendor specific techniques, but typically:

- contain a B-Tree structure for partial value indexed item retrieval;
- build data structures within standard Image data sets;
- create an index root file to describe indexing characteristics;
- provide transparent and non transparent support from fourth generation languages;
- installation programs for configuring and reconfiguring indexing requirements;
- maintenance programs to rebuild indexes on new or old data bases;

Operational characteristics

- Automatic recognition of indexed data sets by fourth generation language processor.

- During inquiry or modification mode, indexed data items are prompted for, at which point any combination of the retrieval capabilities can be used.
- Entries are qualified from indexes and not retrieved from data sets during inquiry processing producing a very responsive interface.
- Qualified entries can be listed data using relational conditions, displayed and modified if required.
- Qualified entries can be passed to reports for printing or complicated outputs.
- Indexing cannot easily span data bases without the use of regular image keys.

Programming Effort

- Indexed data sets are automatically recognized and require no effort to maintain from a programming stand point.
- Advanced reporting on multiset displays typically have to be coded in the reporting language.
- Existing screen displays will typically benefit from retrieval capabilities with no modifications.

Performance Characteristics

-Retrieval from Data Base

- Qualifies entries very quickly; example: Master data set 8192 entries, 18013 capacity;

single item value	160 records	< 1 second
three values using "or"	370 records	< 1 second
single partial value	212 records	~ 1 second

- Multiple items from single data set;

single item	160 records	< 1 second
and		
second item	160 records	< 1 second
	TOTAL TIME	< 1.5 second

- Qualification and Retrieval

single item value	160 records	4.5 seconds (1.8 seconds)
multiple item value	370 records	10 seconds (4 seconds)
- Average Rate of Retrieval	37 records/ seconds	91 records/ CPU second

- Adding to Data Base

Indexed data items require significant amount of time to be placed into a data base. The number of index data sets which need to be updated for each new entry can be as little as one and as many as four for each indexed item! Therefore, the performance of adding to a data set can vary and be significant. Typically in on line entry with four or five items indexed, the time to add to the data base will be less than 1.5 seconds. This amount of time is very often acceptable for low volume of fairly static information. In high volume data entry applications, the time to add may be too great to sustain an acceptable level of performance. In general, each item indexed will have a cost similar if not slightly greater then adding a chained item in a detail data set.

In addition to the performance cost of adding records there exists a cost in disk space. The data base in which the performance data was extrated was originally contained within approximately 34,000 sectors. After a moderate amount of indexing, 4 items totalling 50 bytes in a master of 8930 records and one 82 byte field in a detail data set of 62,190 records the data base required an additional 7100 sectors of disk space or approximately 25% more. It should be noted however, if the application did not use indexing, it may not have been possible to implement with a fourth generation language or Image.

Using the Technology

Index systems offer functionality and performance which is very appealing. The ability "to find the blue needle in the hay stack" in under a second is very attractive. Care must be taken however, not to try and replace all Image search items with indexed data items or index indiscriminantly.

Image search item structure are efficient and provide excellent performance, where full key values are known or when the data base is being accessed programmatically.

Index data structures are less efficient in CPU and disk space usage, but provide a more flexible and improved responsive user interface. These characteristics indicate that using indexes should be viewed as a complement to Image search structures and not a replacement.

Good candidates for using index data items are master type files where data is relatively stable and textual information is present; two examples of such files are the customer master and part master.

In both of these data sets, numbers and/or codes are

typically assigned to each record. These represent a unique occurrence of data and represent an item with textual information.

Without indexing an interactive user would normally be required to know the unique record identified, be it part number or customer number to access stored information. Using indexes, users may query the data base with more meaningful data items such as names, cities, or part descriptions and interactively retrieve the information required without knowledge of codes or numbers.

Poor example of using index data items would be high volume transactions data sets, such as an order detail or inventory transaction file. While their may very well be data that would benefit from indexing, performance of maintaining index data's structures, in high volume applications, may slow data base performance to unacceptable limits.

Reporting on data sets which have been indexed may benefit from using the indexes, but care must be taken as the maximum throughput of reading data with indexes is less than with other techniques. Maximum rates qualifications and retrieval are about 70 records/second. Other techniques, such as fast serial read can be processed at a rate of over 2000 records/second. Indexing however, does not require that the entire data set be read and can therefore offer superior performance for small subsets of data.

High Speed Serial Read

High speed serial read technology enhances data base access from fourth generation languages by increasing the rate at which data can be extracted. Using a set of decision rules, the language processor will invoke special data base access routines when possible to dramatically increase the speed at which data can be read from a master of detail data set.

Requiring no prior configuration or set up, high speed serial read is very easy to use and to benefit from. Maximum optimization of the technology requires that the user clearly comprehends both performance characteristics affecting the data as well as the data itself.

Implementation

- High speed data access intrinsincs use privileged mode to open data sets.
- Data access intrinsincs are typically contained within the system segmented library, thereby eliminating the need for privileged mode compatibility on the fourth generation language

processor.
- Language processor has knowledge of processing context and with built in decision rules can activate high speed routines if needed and available.
- Access intrinsics routines use multi record NoBuff IO.
- Access routines may be disabled by system manager or programmers.

Operational Characteristics

- Forward serial read only (DBGET mode 2);
- Data cannot be updated while being read;
- Verifies and respects Image security;
- Bypasses Image control blocks to reduce overhead and increase speed;
- Requires a minimum of 4,000 words of stack space.

Performance

- As no data modifications must be performed, there is no effect on adding data to the data base.
- Indirectly using fast read can reduce the need of search items, or indexed data items by providing alternate means to access data.
- Forward serial read is approximately 6 items faster than a regular serial read.

Set Type	Capacity	Entry Size	Reading Technique	Elapsed Time	CPU Time	Records/ Second	
Master	8930/18013	101	Image	29	20	307	
Master	8930/18013	101	Fast Read	42	3.9	2126	(6.7 times)
Detail	62190/91008	51	Image	126	124	493	
Detail	62190/91008	51	Fast Read	22	18.5	2826	(5.7 times)

Using the Technology

The raw speed of fast read technology is impressive and useful by itself, but to gain maximum benefit, some thought must be given to reading algorithms for the given structure a data base.

To maximise the gain of using fast read technology, data sets which require lengthy processing should be reviewed so as to decide whether serial read may be implemented. An example follows which illustrates three different approaches, with and without fast read to help comprehend the performance characteristics in reading a master and detail data set.

- Data base Layout (subset):

Set	Type	Capacity	Entries	Seconda -ries	Average chain
M-Product	Master	18013	8930	21	1.25
D-Description	Detail	91008	62190	-	7

The processing objective is to read each master record and its related detail. In the first two cases, for each master record a chained read is performed on the detail data set. In cases 3 and 4, the detail data set is read serially and for each record, a calculated read is performed on the masted data set. In tests 5 and 6, the detail data set is again read serially , but a calculated read is performed when the current master record does not have the same key value as the current detail.

Cases	Elapsed Time	CPU Time	% Case 1	% Case 2
1 Read master chain detail	186	178	100	105
2 Fast read master chain detail	177	174	95	100
3 Read detail-calculated read master	318	315	171	179
4 Fast read detail-calculated read master	225	220	121	127
5 Read detail-calculated read master-break	213	211	115	120
6 Fast read detail-calculated read master-break	109	108	59	61

The best results illustrate the potential of using alternate processing methods. Case 6 in which the detail data set was read serially with fast read and the master data set read for each new key value resulted in a performance increase of 67% over case 2 where we read the master serially with fast read and we chained to the data set.

Their exists a particular data characteristics which had an effect on the results that is, the packing of the detail data set. This one fact resulted in the optimization of cases 5 and 6 by eliminating over 53,000 calculated reads. This same fact also greatly optimises cases 1 and 2, by producing the nearly optimal chain read environment.

While the ideal packing skewed the results, it should be kept in mind that all data bases have particular data characteristics. The knowledge of data particularities and of performance profiles of the various access techniques are mandatory for achieving maximum performance.

By retrieving the results, the following performances profiles can be extracted:

Fast serial read	= 2100	records/second for master data set at 50% capacity
	= 2800	records/second for detail set
Regular serial read	= 307	records/second for master data set at 50% capacity
	= 493	records/second for detail set
Calculated read	= 320	records/second for master data set at 50% capacity
Chained read	= 350	records/second for detail set
Logic process for conditional check	= 1000	records/second

These results should not be taken to suggest that detail sets should be read first but rather that details can be read first and thus eliminate the need for a detail search path and still be faster.

The replacement of detail chains used for reporting purposes by run time access routines have the potential of not only increasing the speed of reading data but also decrease the overhead associated with adding records to the data base.

In summary, fast read routines will improve the rate at which data can be read from the data base. The passive use of this technology will provide small gains in performance. More conscious use however, will provide the analyst with the possibility of gaining significant run time reductions and the opportunity to decrease data base index structures.

Conclusion

Fourth generation language vendors are supplying alternative ways of addressing the need for greater input/output performance from Image data bases. Easy to use interfaces and, in some cases fully transparent interfaces, may use these techniques very easily. However, the numerous performance parameters and the greater number of processing options require that additional analysis be performed. Informed use will increase system throughput and provide more productive systems. Negligent use can result in no improvements and even possible performance degradation.

With the maturing of fourth generation language products, data processing system production is less time consuming, however, to fully exploit their capabilities requires very detailed analysis and excellent understanding of the development tools.