INDEX SEQUENTIAL ACCESS TO IMAGE DATA BASES

Wolfgang Matt
Industrieanlagen-Betriebsgesellschaft mbH,
Einsteinstrasse 20, D-8012 Ottobrunn, West Germany

Summary

IMAGE, though being a very successful data base system, has essential drawbacks: it does not allow access by generic keys and sorted chains show bad performance. Using KSAM in addition to IMAGE solves some problems, but means additional programming and extensive use of system resources. It is also not possible to log and recover IMAGE and KSAM consistently, unless you rewrite the logging software. This paper presents a software designed to enhance IMAGE for indexsequential access i.e. generic keys and sorted chains. The software uses only IMAGE and is itself IMAGE compatible (same database, intrinsics and calling sequences). It does not use privileqded mode.

The user can define indexsequential access paths for any item of master and detail data sets. These access paths are treated like IMAGE chains, but unlike IMAGE chains they can be defined and deleted without unloading the database. Many user applications (e.g. TRANSACT programs) can take advantage of generic keys and sorted chains without program changes, but programmers can take full advantage of the features by calling DBFIND and DBGET with mode parameters not used by IMAGE.

The paper compares the new access method with traditional access methods with regards to features, usability and performance.

Performance of Typical Database Accesses

We consider a typical data base consisting of a CUSTOMER master and an ORDER detail. Both are connected by the search item CUSTNR, the customer number. For the ORDER detail there exists a second search item, the article number ARTNR (connected to an automatic master).
We now consider some typical inquiries to this data base to see how IMAGE performs.

1. Retrieve name and address of a customer with a given CUSTNR.
   IMAGE does a hashed access which is very efficient.

2. Retrieve the address of a customer with a given name.
   A serial read has to be done which is very inefficient.

3. Retrieve all orders for a given customer number ordered by article number.
   If we have a sorted chain, retrieval is efficient, but adding and deleting is not. If no sorted chain is defined, an online sort must be programmed which is not efficient.

4. Retrieve all orders for given customer and article number.
   A chained read must be done which may be inefficient if many entries are discarded.

5. Retrieve all customers which buy a given article.
   A chained read on the second path is very efficient.

6. Retrieve all customers which buy a given group of articles identified by the first two characters of the article number. A serial read has to be done which is very inefficient.

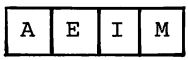
It can be seen from these examples that IMAGE performs well when the access is made using exactly one of the fields defined as IMAGE keys. Performance is bad when we need an access by a none key field of a master, by a combination of keys, or by a partial key. Since these questions exist, we have to look for an alternative method of accessing data in an IMAGE data base. This method should be able to retrieve long keys (i.e. the combination of several IMAGE fields) the same way as partial keys (e.g. the first character of a customer's name). This method should also provide the possibility of sorted retrieval without the overhead of sorted chains.

The B-Tree Method

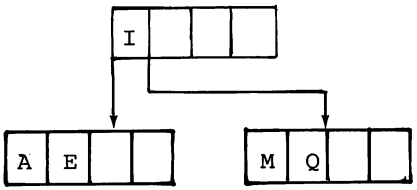
The method which allows partial key access and retrieval in ascending order is called index sequential access. One starts with a given index and reads from there in a logically sequential order. HP has implemented this access method in KSAM but not in IMAGE.

The method used by KSAM and tools of other manufacturers is based on the B-tree principle invented by R. Baier. Keys and data are stored in a special way which minimises disc read and allows fast access to any key. In order to make this method efficient, only pointers to the data are usually stored within the B-tree. This method is ideally applicaple to data stored in IMAGE data bases, since we only have to add a B-tree for the keys. A B-tree consists of blocks containing key values, the associated data pointers and pointers to other blocks.

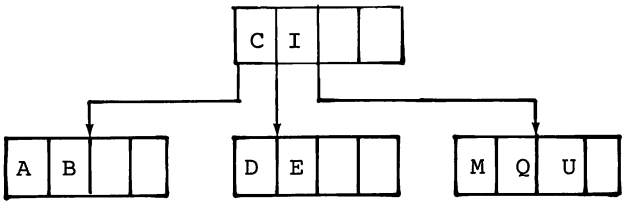
To illustrate the B-tree method we use very small dimensions: a key length of one character and assume that 4 keys fit into one block. We initially want to store the letters A E I M Q U. After storing the first four letters the first block is full



To add Q the block must be "splitted":



The new root block contains besides the key I the pointers to the two leave blocks. After adding the key U into the second leave block we want to add the remaining letters of the alphabet. We can add the keys B and C into the first leave block by shifting E but to add D we have to split again:



Using the B-tree we now demonstrate the retrieval of a key e.g. key E. First the root block is read. It does not contain key E, but since E is located in the alphabet between C and I (the two keys in the root block), the pointer is followed and the second leave block retrieved. In this block the key E is found and also the associated data pointer (not shown in the figures above).

Interfacing B-Trees with IMAGE

When we started to design an index sequential access method for IMAGE data bases, we decided not to use one of the following approaches:

1. We did not want to use KSAM. The reason was that standard recovery methods did not allow to keep KSAM and IMAGE files consistent in case of a system failure. Another disadvantage of KSAM is the extensive use of system resources.

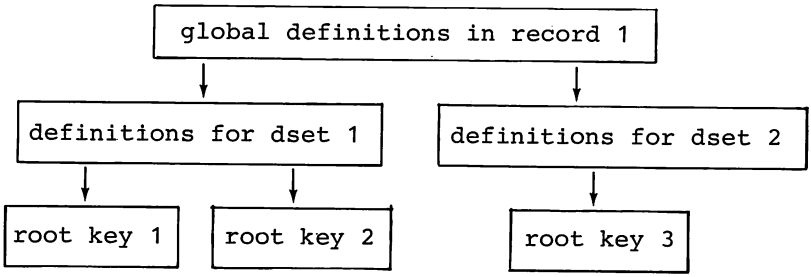
2. We did not want to change the IMAGE source code. This was done to be independent of HP enhancing IMAGE (e.g. TURBO IMAGE).

3. We did not want to use privileged mode. Access to IMAGE is done but solely using the documented IMAGE intrinsics. This makes the software independent of internal IMAGE changes and adds to system security.

For the interface software we defined the following requirements:

1. Indices must be covered by logging

   The B-tree must be stored in a data set within the data base to which it refers. Neither an MPE file nor a seperate data base can provide a consistency between data and keys in case of a system failure. A separate program to reconstruct the B-tree from the data may run for hours, while roll-back recovery can be done within a few minutes using TURBO IMAGE. One extra data set (stand alone detail) is sufficient to store the B-trees of all fields (or combinations of fields) for which index sequential access is defined. The data set also includes the definitions themselves. This leads to a logical structure like this:



2. Indices must be updated automatically

   It would be tedious and prone to error if a programmer had to code a normal DBPUT and then an intrinsic call to update the B-tree, code a DBBEGIN and DBEND around it and also include the index data set in the lock descriptor. We need a "super DBPUT" which does this all automatically. The same is true for DBDELETE and also for DBUPDATE, since we allow the new keys to be updated. Since DBDELETE does not have an argument buffer, the "super DBDELETE" must find out which keys have to be deleted according to the latest DBGET.

3. The calling sequence must be identical

   Though the "super DBPUT" does everything necessary, you do not want to change all programs replacing the call to DBPUT by a new call with additional parameters and pass new parameters from routine to routine. No - the "super DBPUT" must be called DBPUT and it must have identical parameters. Only this garantees existing programs to run without modification, recompilation or patching. The only restriction is that the new intrinsics must reside in an account SL, since they call the HP intrinsics in SL.PUB.SYS.

4. The logic of retrieval must be unchanged

Though partial key access is an addition to the standard IMAGE retrieval methods, it should be done by the same logic. You want do a DBFIND with the partial key as argument, and a series of DBGET's (mode 5) should retrieve all entries belonging to this partial key. An end-of-chain indicator (condition word 15) should also be returned. But somehow you have to tell the software, how long the partial key is. This is the only information not provided in the standard IMAGE calling sequence. We provide two methods:

- new values for the mode parameter allow to code the length of the partial key and also allow to specify wether the search should start at the lowest or the highest key which matches the partial key. In the latter case a backward read (mode 6) may be performed.

- software which cannot use anything but mode 1 for DBFIND (existing programs, RPG, 4GLs) can terminate the argument by a @ . All characters up to the @ are treated as partial key. This is especially convenient when an automatic master is replaced by index sequential access. You do not need to change any of your programs.
  This method also works for 4GLs like TRANSACT. But some 4GLs do not trust IMAGE. They compare the result of DBGET with the argument of DBFIND, which is of course not identical for partial key access.

Biography

Wolfgang Matt holds a PhD in physics. Since 1977 he works with IABG, a company with 1700 employees near Munich. He is head of a group of scientists, consulting HP 3000 users and developping individual software for them. He is the author of SI-IMAGE, a product for index sequential access to IMAGE data bases.