APPROXIMATING BLOCK ACCESSES IN RANDOM FILES: THE CASE OF BLOCKING FACTORS LOWER THAN ONE

By: Prashant C. Palvia

Abstract:

Expressions available in the current literature to estimate the number of blocks accessed in a random file fail to work when the blocking factor is lower than one. A new expression is developed in this article to estimate the number of blocks accessed; this expression is valid for blocking factors that are higher as well as lower than one. It is shown using simulation experiments that this expression is quite accurate in most situations.
Key words: Databases, database design, performance evaluation, random tiles, blocking factors, block accesses

Article:

INTRODUCTION

It is well known that input/output (I/O) activity is among the slowest operations in a computer system, and as such has a significant impact on performance. A major determinant of I/O activity is the number of block accesses from the disk. Consequently, many analytical works in database and file management need to determine the number of blocks accessed for a given query, and generally use some kind of estimation method (*e.g.* in [1-3, 5, 8, 9, 11]). Many expressions have been developed for estimating the number of block accesses in order to retrieve a given number of batched records from a random file. For example, expressions reported in [2, 4, 7] are approximations, while expressions in [10, 11] are exact expressions. In addition, the effect of buffer size on block accesses has been studied [6].

All of the above expressions, although do not explicitly state so, assume that the blocking factor (defined as the number of records per block) is greater than one. This does remain a valid assumption in a majority of data processing and number-crunching applications as the block size is typically very large and accommodates several records. However, in several existing arid many emerging applications, each record can be very large and may span more than one block. To name a few, examples of extremely large records can be found in legal database applications, text-processing applications, engineering applications (CAD/CAM), graphical-digitizing applications and archival databases. We also believe that there are many traditional data processing applications containing extremely large records.

In this paper, we address the case when the blocking factor is lower than one, which arises when a record spans several blocks, In the next section, it is shown that the current expressions for estimating the number of block accesses are no longer valid for blocking factors of lower than one. In order to address this shortcoming of the current expressions, we develop a general expression which is applicable for blocking factors of both higher and lower than one. The accuracy of the new expression is evaluated by conducting a simulation, and comparing it with the simulation results.

THE CURRENT EXPRESSIONS

There are two widely used approximate expressions. The one reported by Cardenas [2] is the following:

$$B = m \cdot [1 - (1 - 1/m)^k]. \quad (1)$$

The expression reported by Palvia and March [7] is the following:

$$B = m \cdot [1 - (1-k/n)^p].$$

An exact expression reported by Yao [10] is the following:

$$B = m \cdot \left(1 - \sum_{i=1}^{k} \frac{n \cdot d - i + 1}{n - i + 1}\right), \quad \text{where } d = 1 - 1/m \qquad (3)$$

In all of the above expressions, B is the number of blocks accessed, n is the number of records in the random file, k is the number of records to be retrieved, m is the total number of blocks used for storing all of the records in the file, and p is the blocking factor so that $p = n/m$.

In order to show that these expressions fail to yield correct results for blocking factors smaller than one, let us take a simple case. Let $n = 300$ records in the file, $k = 2$ records to be retrieved, $p = 0.5$ (i.e. one record takes two blocks). Then, in $m = n/p = 600$ blocks. It is obvious that in this situation, four block accesses will be required. Evaluating the Cardenas expression gives a result of 1.998 accesses, the Palvia—March expression gives a result of 2.003 accesses, and the Yao expression gives a result of 2.002 accesses. With the same parameters, and $k = 16$ records to be retrieved, the three expressions estimate the number of blocks as 15.8, 16.22 and 16.205, respectively. Again, these are incorrect as the number of blocks will be 32.

THE NEW EXPRESSION

The above examples suggest that the appropriate expression for estimating the number of block accesses when the blocking factor is lower than one, may be very trivial, It seems that the number of blocks accessed will be simply given by:

$$B = k/p. \qquad (4)$$

Certainly, this expression works in the above two examples. But does it work in all cases? Let us examine.

The k/p expression works in the above examples because each record requires an exact number of full blocks, i.e. $1/p$ is an integer number. But what if a record does not completely fill a block. As an example, let $p = 0.4$. This means that a single block holds four-tenths of a record, or that a record occupies two-and-a-half blocks ($1/p = 2.5$). Now if $k = 1$, then three blocks need to be accessed to retrieve one record, while the k/p expression suggests 2.5. In order to access two records, six blocks need to be accessed if the two records are placed at a distance from each other (a very likely possibility in random files). The k/p expression estimates five blocks; this will occur only if the two records are placed contiguously. Obviously, we need a better expression which will give good estimates in all cases.

The following expression is developed which works well for blocking factors that are higher as well as lower than one:

$$\text{let } Q = 1/p.$$

Let $q = \lfloor Q \rfloor$, where $\lfloor\ \rfloor$ is the lower ceiling function, and gives the highest integer less than or equal to Q:

$$\text{let } r = Q - q.$$

The parameters q and r separate the number of blocks occupied by a record into complete blocks and partial blocks. Now the *"k/p"* expression can be used for the complete blocks, and any of the three equations (1)—(3) can be used for the partial blocks. We use equation (2), as it has shown to be both computationally simple and generally accurate [7].

The final expression for the number of blocks accessed is:

$$B = k \cdot q + (n \cdot Q - k \cdot q) \cdot \left[1 - \left(1 - \frac{r \cdot k}{n \cdot Q - k \cdot q}\right)^{1/r}\right]. \qquad (5)$$

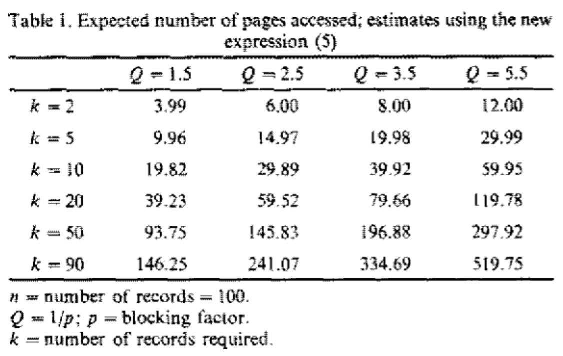
Table 1. Expected number of pages accessed; estimates using the new expression (5)

	$Q = 1.5$	$Q = 2.5$	$Q = 3.5$	$Q = 5.5$
$k = 2$	3.99	6.00	8.00	12.00
$k = 5$	9.96	14.97	19.98	29.99
$k = 10$	19.82	29.89	39.92	59.95
$k = 20$	39.23	59.52	79.66	119.78
$k = 50$	93.75	145.83	196.88	297.92
$k = 90$	146.25	241.07	334.69	519.75

n = number of records = 100.
$Q = 1/p$; p = blocking factor.
k = number of records required.

Note that the first $k \cdot q$ term in the above expression corresponds to the k/p expression of equation (4). The second term corresponds to equation (2). In the second term, $(n \cdot Q - k \cdot q)$ represents the remaining blocks which have not been accounted for by the first term.

When p is greater than one, equation (5) reverts completely to equation (2). This is because, Q is less than one, q is zero, and $r = Q = 1/p$. When Q is an integer (i.e. a record occupies complete blocks), then equation (5) reverts to equation (4). This is because $q = Q = 1/p$ and $r = 0^1$; and when $r = 0$, the second term becomes zero.

DISCUSSION

Using expression (5), the number of blocks accessed have been computed for different blocking factors and k values, and are shown in Table 1. For illustrative purpose, the total number of records in the file, for these computations and in subsequent sections, is 100 (similar results are obtained for larger files). Results for blocking factors higher than one have been reported before [7, 10]; therefore, we only show results for blocking factors lower than one. Q Represents the reciprocal of the blocking factor. The table only uses non-integer values of Q, as for integer values we have already shown that the number of blocks accessed is simply $k \cdot Q$ (or k/p). We make the following observations from Table 1:

(A) When the number of records required, k, is small, then the number of block accesses is closer to $k \cdot |\bar{Q}|$ (where $|\overline{\ }|$ is the upper ceiling function giving the lowest integer higher than or equal to Q). This is readily seen in the data for k -values of 2, 5 or 10.

(B) When k is large, this is when batching of records really starts to pay off. This can be seen in the data, when $k = 50$ and $k = 90$. In both cases, the number of blocks assessed is less than the theoretical maximum of $k \cdot |\bar{Q}|$ Actually, the minimum for the number of blocks assessed is $k \cdot |\bar{Q}|$, and the maximum is the lower of $k \cdot |\bar{Q}|$ and $n \cdot |\bar{Q}|$.

(C) If Q is an integer, then there are no advantages of batching the request for the required records. Again this is because, then the number of required blocks is $k \cdot Q$, the same if each record was individually retrieved. It is only when Q is non-integer, and k is high (compared to n, the total number of records in the file) that advantages due to batching are achieved.

EVALUATION OF THE EXPRESSION

As stated earlier, equation (5) is an estimate. It, therefore, needs to be evaluated for its correctness over a wide range of k and Q values. Since, we do not have an exact expression to evaluate equation (5), we wrote a computer simulation program to compute the number of block accesses for different k and Q values. The simulation program was general enough so that it could handle different n values (number of records in the file) as well as different Q values (integer, non-integer, less than one, greater than one). The inputs to the simulation program were n, k and Q; the output was the number of block accesses.

Table 2. Expected number of pages accessed; estimates using simulation with contiguous placement of blocks of a record

	$Q = 1.5$	$Q = 2.5$	$Q = 3.5$	$Q = 5.5$
$k = 2$	4.00	6.00	8.00	12.00
	0	0	0	0
$k = 5$	10.00	14.97	19.97	29.90
	0	0.17	0.17	0.30
$k = 10$	19.27	29.60	39.53	59.63
	0.74	0.62	0.63	0.56
$k = 20$	38.10	58.67	77.90	118.67
	1.18	1.03	1.30	1.03
$k = 50$	87.03	137.37	187.67	287.47
	1.63	1.75	1.58	2.05
$k = 90$	139.63	229.57	319.40	499.63
	0.56	0.68	0.77	0.57

Note: In each cell, the top number is the average and the bottom number is the standard deviation.
$Q = 1/p$; p = blocking factor.
k = number of records required.

Table 3. Expected number of pages accessed; estimates using simulation with random placement of blocks of a record

	$Q = 1.5$	$Q = 2.5$	$Q = 3.5$	$Q = 5.5$
$k = 2$	4.00	6.00	8.0	12.00
	0	0	0	0
$k = 5$	9.83	14.97	19.90	29.97
	0.46	0.17	0.30	0.17
$k = 10$	19.77	29.80	40.00	59.90
	0.44	0.41	0	0.30
$k = 20$	38.70	59.03	79.33	119.37
	0.95	0.93	0.75	0.67
$k = 50$	91.33	140.30	194.83	297.27
	1.70	1.58	1.74	1.49
$k = 90$	147.64	244.63	338.43	523.23
	1.75	2.13	2.52	2.61

Note: In each cell, the top number is the average and the bottom number is the standard deviation.
$Q = 1/p$; p = blocking factor.
k = number of records required.

Two simulation programs were developed. The first program places records sequentially and contiguously in the blocks and the required records are randomly selected from the collection of all records in the file. This is analogous to retrieving records randomly from an indexed sequential file. The results of this simulation (for the same *k* and *Q* values as in Table 1) are shown in Table 2. Note that for each case, the simulation was run 30 times and an average and a standard deviation was obtained.

Comparing Tables l's results with those of Table 2, expression (5) seems to work fairly well. The expression is very accurate for small values of *k;* however, it tends to overestimate for larger values of *k.* In any case, the expression provides a tight upper bound on the number of blocks accessed. The probable reason for the overestimation by equation (5) is that in a sequential arrangement of records, there is a greater likelihood of previously accessed blocks containing parts of other required records. Thus, the simulation will generate fewer block accesses than that a truly randomized condition will predict, as does equation (5).

The second simulation program places the records randomly in the blocks, and the required records are still randomly selected. This situation represents the case of truly randomized files. The simulation results for this case are presented in Table 3. In the truly randomized case, expression (5) results are in fact very close to the simulation results. The differences in the two sets of results (Tables 1 and 3) are very small and there really does not seem to be any pattern in the differences (i.e. for $k = 2$, the results are the same; then the expression seems to be slightly overestimating for $k = 5, 10, 20, 30$; but then it slightly underestimates for $k = 90$).

In essence, then, the expression (5) is quite accurate for randomized files. For indexed sequential files with random accesses, it is again quite accurate when a small number of records are required and provides a close upper bound when a large number of records are required. Note that while all of the results are reported for a file size of 100 records (i.e., $n = 100$), similar results are experienced with larger files.

One final comment. Our simulation results clearly show that the standard deviation for the number of blocks accessed. is quite low. This confirms the observation made earlier in [6]. It also validates and justifies the use of approximate expressions (such as in [2, 4, 6, 7, 10]) in analytical studies.

CONCLUSION

This paper has reported a new expression which gives an estimate of the number of blocks required to access a given number of records in a random file when the blocking factor is lower than one. Previous expressions reported in the literature assume that the blocking factor is never lower than one, and produce erroneous results when the blocking factor is in fact lower than one, Moreover, the expression developed in this paper is valid for all values of the blocking factor, lower as well as higher than one.

Notes:

[1] Even though it is easily seen that the expression (5) reverts to *B =kip* when *Q* is an integer; computer codes of the expression (5) are likely to give a run-time error. This is because, the expression would involve division by zero. It is recommended that a computer implementation should simply use the first term of the expression when *Q* is an integer.

REFERENCES

IQ D. S. Batory and C. C. Gotlieb, A unifying model of physical databases. *ACM Trans, Database Syst.* 7, 509-539 (1982),
(2] A, F. Cardenas. Analysis and performance of inverted database structures. *Commun. ACM.* 18, 253-263 (1975). [31 J. V. Carlis and S. T. March. A computer aided database design methodology. *Comput. Perform. Dec.* 4, 198-214 (1983).
[41 T. Y. Cheung. Estimating block accesses and number of records in file management. *Commun. ACM* 25, 484-487 (1982).
[51 H. K. Jain, A comprehensive model for the storage structure design of CODASYL databases. *Information Systems* 9, 59-67 (1984).
(61 P. Palvia. The effect of buffer size on pages accessed in random files. *Information Systems* 13, 151-161 (1988),
[71 P, Palvia and S. T. March. Approximating block accesses in database organizations. *information Process, Lett.* 19, 75-79 (1984).
[81 B. Schneiderman and V. Goodman. Batched searching of sequential and tree structure files. ACM *Trans, Database Syst,* 1, 268-275 (1976).
(91 S. B. Yao. An attribute based model for database access cost analysis, *ACM Trans. Database Syst.* 2, 45-67 0977). [101 S. B. Yao. Approximating block accesses in database organizations. *COMMIRI. ACM* 20, 260-261 0977).
[111 P. C. Yue and C. K. Wong. Storage cost considerations in secondary index selection, *Mt. J. Comput. information Sci,*
4, 136-148 (1975).