A General Framework for Computing Block Accesses[†]

Brad T. Vander Zanden
Howard M. Taylor
Dina Bitton

TR 85-718
December 1985

Department of Computer Science
Cornell University
Ithaca, NY 14853

[†] This research was partially supported by the National Science Foundation under grant DCR - 8410889.

A GENERAL FRAMEWORK FOR COMPUTING BLOCK ACCESSES

Brad T. Vander Zanden*

Howard M. Taylor†

Dina Bitton*

ABSTRACT

A physical database system design should take account of skewed block access distributions, nonuniformly distributed attribute domains, and dependent attributes. In this paper we derive general formulas for the number of blocks accessed under these assumptions by considering a class of related occupancy problems. We then proceed to develop robust and accurate approximations for these formulas. We investigate three classes of approximation methods, respectively based on generating functions, Taylor series expansions, and majorization. These approximations are as simple to use and far more accurate than the cost estimate formulas generated by making independence and uniformity assumptions.

1. INTRODUCTION

A database system's throughput is strongly influenced by the match between the physical database design and the environment in which it operates. This environment includes such characteristics as the mix of queries and the structure and storage of the data. Database designers typically make simplifying assumptions about this environment in order to obtain a tractable model. For example, they

*Computer Science Department; Cornell University; Ithaca, New York 14853

†Operations Research Department; Cornell University; Ithaca, New York 14853

assume that attributes are independent, that the values in an attribute domain are duplicated equiproportionally and that the blocks in a file are accessed equiproportionally. These assumptions considerably simplify the formulas that are employed to estimate the cost of a query (measured by the number of block accesses required to execute the query). However, several recent studies have shown that these simplified models also severely compromise the accuracy of the cost estimates [Christodoulakis 1982 and 1984; Montgomery 1984]. Indeed, these assumptions can lead designers to construct access configurations (sets of data structures and access paths) and query optimizers that severely degrade the performance of a DBMS.

Unfortunately, realistically modelling the database environment leads to cost formulas which are costly to calculate. This performance is intolerable since cost estimates play an integral role in both physical design and query optimization. In the physical design process, database designers attempt to find access configurations that minimize the total cost of a set of representative queries. Query optimizers then take the access configuration produced by this process and use it to select an access path which minimizes the number of block accesses required to satisfy a query. However, given the complexity of the formulas which currently arise from realistic models of the database environment, query optimizers and physical design might be more effective if they arbitrarily selected an access strategy. Ideally, though, they would calculate the cost of each strategy and then choose the cheapest one based on this analysis. The problem, then, is to design a set of cost formulas that can be efficiently and accurately computed. However, while there has been extensive research interest in finding approximations to these formulas [Luk 1983; Montgomery et al. 1983; Zahorjan et al. 1983; Christodoulakis 1984a], no generally accepted approximations have yet emerged.

In this paper we present a general framework for efficiently estimating the number of blocks accessed by a query, when realistic assumptions are made about the database state: records are not randomly placed on blocks, attribute values do not occur equifrequently, and attributes are not independent. We derive general formulas for the number of blocks accessed under these assumptions by considering a class of related occupancy problems [Feller 1968, Kotz and Johnson 1977]. We then proceed to develop robust and accurate approximations for these formulas. We investigate three classes of approximation methods, respectively based on generating functions, Taylor series expansions, and majorization. More specifically, the rest of the paper is organized as follows. In section 2 we present a database and query model and introduce a general formula for block accesses. In section 3, we estimate the number of blocks accessed in order to retrieve k records, under the assumption that records are not randomly placed. In section 4 we estimate the number of blocks accessed under the assumption that attribute values are non-uniformly distributed. In section 5 we combine both assumptions—non-random record placement and non-uniform attribute distributions. Finally in section 6, we summarize our results.

2. A DATABASE AND QUERY MODEL

In order to construct a set of cost formulas for a given database, we must make certain assumptions about the structure of the data, the way in which the data is stored, and the types of queries that users may pose. Taken together these assumptions constitute a model for the database system. In this section we first introduce the class of queries which we intend to consider and then present a deterministic process model which realistically models the data structure and storage in a database. We contrast this model with the random process model which is currently used in database physical design and query optimization.

2.1. Query Model

In this paper we consider two fundamental types of selection queries:

1. Queries that randomly retrieve k records from a set of blocks labelled $B_1, B_2, \dots, B_m$.
2. Queries that arbitrarily select one value from an attribute domain and which retrieve all records that contain that value.

We will refer to these queries as type 1 and type 2 queries respectively. Together they support a rich subset of the queries normally posed in a database system. Specifically, they support select-join-project queries of the form

$$\text{Retrieve}(t.a, u.b)$$

$$\text{where } t.c = u.c \text{ and } P(t) \text{ and } P(u)$$

where a , b , and c denote vectors of attributes, t and u denote relations, and $P(t)$ and $P(u)$ denote restriction predicates on relations t and u respectively. For example, suppose we are considering a join algorithm that does not employ sorting. Perhaps it uses a secondary index instead. In order to compute the number of block accesses required by this algorithm, we first estimate the record selectivity of each of the two predicates using one of the methods discussed in the literature [Merrett and Otoo 1979; Christodoulakis 1983b; Piatetsky-Shapiro and Connell 1984]. We will discuss these methods shortly. Without loss of generality, assume that $|P(t)| < |P(u)|$ where $|P(t)|$ and $|P(u)|$ refer to the number of records satisfying the two predicates. Then we can estimate the number of block accesses required for relation t by treating the join as a type 1 query that randomly retrieves $|P(t)|$ records from relation t . Further we can estimate the number of block accesses required for relation u by treating the join as a series of $|P(t)|$ type 2 queries on the restricted relation u . Thus we can estimate the number of block accesses required by the join algorithm using a combination of type 1 and type 2 queries.

In general, type 1 queries are most useful in query optimization settings since the optimizer is handling a specific instance of a query. For example, the optimizer might handle a query of the form "Retrieve all records with department = accounting". On the other hand, type 2 queries are better suited to physical design where the designer must deal with a general query rather than a particular instantiation. Continuing the previous example, the designer might be faced with the query, "Retrieve all records with department = *name*" where *name* is randomly chosen from the departments within the organization. In this case, an average cost must be calculated.

2.2. Deterministic Process Model

In this subsection we present a model which examines the underlying conditions that shape a database environment and that models this environment in a realistic manner. We call this model the deterministic process model. The types of assumptions that it makes about the database environment include the following:

1. Attribute Structure: Attribute structure encompasses two areas, the distribution of values in the attribute domain and the relationships between attributes.

- a. Attribute Distribution: The attribute distribution describes how often each value in the attribute domain is duplicated in a relation. The deterministic process model assumes that some attribute values may occur more frequently than others. For example, the attribute YEARS OF SERVICE tends to be highly skewed toward fewer years within an organization.

- b. Attribute Independence: Attribute independence refers to whether or not various combinations of attributes are related. Formally we state that A and B are independent if $\Pr[A = a \text{ and } B = b] = \Pr[A = a] * \Pr[B = b]$ where $\Pr[X = x]$ represents the probability that a record contains the value x for attribute X. In practice many attributes do not satisfy this criteria. Functional and multivalued dependencies

introduce explicit attribute dependence. Implicit dependence arises through less restrictive conditions. For example, the number of years of employee service and the employee's position within a firm tend to be related. However, this dependence is not normally made explicit through either a functional or multivalued dependency. The deterministic process model takes such dependencies into account in its model of the database state.

2. Data Storage: Data storage encompasses two areas -- record placement and the file's blocking factor. The deterministic model accounts for non-random record placement, which results from clustering by attribute values or by frequency of access. It also accounts for variable blocking factors.

The primary advantage of the deterministic process model is that it provides a realistic view of a database environment. Unfortunately, the cost formulas it generates are iterative and require a great deal of information (see section 2.4). In practice these formulas cannot be computed rapidly and thus, the deterministic process model is not currently used in the design process.

Instead a much simpler model called the random process model is used by most database designers. In contrast with the deterministic model, it makes the following set of assumptions:

1. Attribute Structure

a. Attribute Distribution: The random process model assumes that each value in the attribute domain occurs equifrequently.

b. Attribute Independence: The random process model assumes that all attributes and combinations of attributes are independent.

2. Data Storage: The random process model assumes random record placement and a fixed blocking factor.

The cost formulas generated by this model are simple and easy to compute since equifrequency implies that only one attribute duplication rate or one blocking factor

must be considered and independent attributes imply that each field may be considered independently of other fields. However, as noted in the introduction, the random process model makes assumptions which do not apply in many database environments and thus it may mislead designers into selecting inefficient access configurations.

2.3. Review of the Literature

The shortcomings of the random process model have encouraged researchers to examine methods for estimating the record and block selectivities of queries under the assumptions of the deterministic process model. This research has provided several promising techniques for rapidly estimating the record selectivity of a query (fraction of records retrieved by a query) [Merrett and Otoo 1979; Kerschberg 1982; Christodoulakis 1983; Piatetsky-Shapiro and Connell 1984; Kamel and King 1985]. Such advances are important for several reasons. First, they help query optimizers estimate the size of intermediate and final results. Second, they allow us to decompose the queries discussed in section 2.1 into type 1 and 2 queries. Finally, they hint at directions that might be followed in estimating block selectivities, an area where results have been less forthcoming. One of the more promising approaches involves dividing the empirical distribution of an attribute or group of attributes into unequal sized cells using a technique such as texture analysis [Kamel and King 1985]. By segregating the data in this manner, the designer is able to take account of clustering in the empirical distribution and obtain cells which contain homogeneously distributed data. Estimates of the number of records involved in relational operations such as selections, projections and joins can then be calculated using this condensed distribution. An idea similar to this one will be discussed later in the context of block selectivities.

Another promising approach involves estimating the data distribution using parametric techniques [Christodoulakis 1983b]. In this situation a multivariate

theoretical distribution is fit to the data and the record selectivity is estimated by integrating the probability density function over the appropriate ranges of attribute values. If the process which created the data follows some known probabilistic pattern, then this approach may be applicable.

The problem of estimating the block selectivity of a query (fraction of blocks retrieved by a query) has been much more intractable. Several different approaches have been suggested including simulation [Siler 1976], analytical modeling [Demolombe 1980; Luk 1983; Montgomery et al. 1983, Zahorjan et al. 1983], and multivariate distributions [Christodoulakis 1984a]. For example, Luk [1983] examined the feasibility of deriving approximate cost estimates assuming that blocks had equal probabilities of being accessed. However, the results were somewhat discouraging in that the error was often greater than 50% and often approached 100%. Taking another tact, Christodoulakis [1984a] extended his work with multivariate distributions by treating the location of a record in secondary storage as an additional attribute value. This innovation allowed him to develop lower and upper bounds on the number of block accesses required by a query when the records of a file are nonrandomly placed in secondary storage. This approach tends to work well when the joint distribution of the attributes (including the record's location) 1) exhibits a good deal of correlation and symmetry; and 2) can be modelled by a parametric distribution.

To date, most of the approaches for estimating block selectivities have either required parametric distributions or else detailed information about the empirical distributions of the data. However, no approaches have yet been advanced for estimating block selectivities which are both efficient and which are sufficiently general to handle most cases of practical interest. We believe that the approach we have crafted in this paper satisfies these criteria.

2.4 Precise Cost Formulas

The class of block access formulas introduced by the deterministic process model can be derived by considering a class of related occupancy problems [Feller 1968; Kotz and Johnson 1977]. Occupancy problems provide an elegant, unifying framework for deriving the cost expressions we are interested in and therefore we will discuss the results of this paper in that context. In fact, many problems related to record distribution and storage can be approached in this framework. For example, in a previous paper [Bitton and DeWitt 1983] we have also modelled duplicate record elimination as an occupancy problem.

The fundamental problem that underlies both type 1 and type 2 queries can be stated as follows: A query randomly retrieves k records from a file which is divided into m blocks and which contains n records. Let $\mathbf{p} = \langle p_1, \dots, p_m \rangle$ denote the block access distribution of this query where p_i denotes the probability that a query accesses the i th block. We assume a tuple may be retrieved more than once (this form of retrieval is termed sampling with replacement—although most queries retrieve records no more than once, Yao [1977] demonstrated that in most database systems, the results obtained for sampling with replacement are, for practical purposes, identical to those obtained for sampling without replacement). How many blocks must be accessed?

The equivalent occupancy problem can be stated in the following manner. A set of k balls is randomly assigned to a group of m urns which can hold a total of n balls. In addition, the i th urn has a p_i probability of being assigned a ball on each toss. How many urns are occupied? We can transform our block access problem to this occupancy problem by associating tuples with balls and blocks with urns. The expression for the mean number of occupied urns is [see, for example, Kotz and Johnson 1977]

$$E[X_k] = \sum_{i=1}^m [1 - (1 - p_i)^k] \quad (2.1)$$

where X_k = number of occupied urns when k balls are thrown

$E[X_k]$ = expected number of occupied urns when k balls are thrown

$1 - p_i$ = probability that the i^{th} urn is not assigned a ball on each toss

$(1 - p_i)^k$ = probability that the i^{th} urn is not assigned a ball on any of the k tosses

$1 - (1 - p_i)^k$ = probability that the i^{th} urn is assigned at least one ball in k tosses

Type 2 queries can be incorporated into this model by introducing nonuniform attribute value distributions. Specifically, let $k = \langle k_1, \dots, k_a \rangle$ denote the attribute value distribution of an attribute where k_i represents the number of times the i^{th} attribute value is duplicated and a denotes the number of values in the attribute's domain. Also, for concreteness, suppose we are given a type 2 selection query of the form "retrieve $t.b$ where $t.c = v$ " and v is allowed to range randomly over all values in the domain of attribute c . How many blocks must be accessed? This problem can be reduced to an occupancy problem by assuming that the number of balls thrown is randomly chosen from the vector k . From expression (2.1) we obtain the following analytical estimate for the number of accessed blocks,

$$E[X_{ak}] = (1/a) \sum_{i=1}^a \sum_{j=1}^m [1 - (1 - p_j)^{k_i}] \quad (2.2)$$

For random placement, $p_j = 1/m$, and equifrequent attribute values, $k_i = n/a$, expressions (2.1) and (2.2) reduce to the well known formulas [Cardenas 1975]

$$E[X_k] = m[1 - (1 - 1/m)^k]$$

and

$$E[X_{ak}] = m[1 - (1 - 1/m)^{n/a}]$$

Expressions (2.1) and (2.2) also satisfy a useful property known as Schur concavity. In order to introduce this concept, we must first define the concept of majorization.

Definition. Let $y = \langle y_1, \dots, y_l \rangle$ and $z = \langle z_1, \dots, z_l \rangle$ denote two vectors of nonnegative, nonincreasing (i.e., $y_1 \geq y_2 \geq \dots \geq y_{l-1} \geq y_l$ and $z_1 \geq z_2 \geq \dots \geq z_{l-1} \geq z_l$) real numbers. We say that vector y *majorizes* vector z (written $z < y$) if the following set of inequalities hold [Marshall and Olkin 1979]

$$\sum_{i=1}^k y_i \geq \sum_{i=1}^k z_i \text{ for all } k \leq l-1 \text{ and } \sum_{i=1}^l y_i = \sum_{i=1}^l z_i \text{ for } k = l$$

Majorization is a measure of non-uniformity in the sense that if y majorizes z , then the distribution described by y is more skewed than the distribution described by z .

Definition. A real-valued function $f(x_1, \dots, x_l)$ defined over the set of non-negative real vectors $(R_+)^l$ is said to be *Schur concave* if

$$z < y \Rightarrow f(z) \geq f(y)$$

In other words, as the skewness of a vector of real numbers increases, the value of the function f decreases. The following definition provides a practical tool for determining whether a function is Schur concave—a function $f(x_1, \dots, x_l)$ of l real arguments is *Schur concave* if for every pair i, j , $(x_i - x_j)(\partial f / \partial x_i - \partial f / \partial x_j) \leq 0$ [Marshall and Olkin 1979].

It has been shown that expressions (2.1) and (2.2) both satisfy this latter definition [Christodoulakis 1984b]. In the context of cost expressions, this result indicates that as the skewness of a distribution increases, the number of expected blocks accessed by a query decreases. This result also demonstrates why the random process model, with its assumptions of equifrequency, leads to pessimistic cost estimates when its assumptions do not hold. In the following three sections, we will utilize the properties of majorization and Schur concavity to obtain upper bounds for

expressions (2.1) and (2.2). These upper bounds can be computed using closed formulas rather than the iterative formulas of expressions (2.1) and (2.2).

3. Type 1 Query – Nonrandom Placement

The first problem we consider is that of estimating the number of blocks retrieved by a type 1 query when the blocks have unequal probabilities of being accessed. Several conditions could give rise to this type of situation. For example, a clustering index might exist on the queried attribute, or the queried attribute may be related to the clustering attribute. As another example, the tuples may be clustered according to their frequency of access. In each of these cases, the probability of accessing a given block is highly dependent on which records the query requests.

In section 2.4 we derived the following precise analytical estimate for the number of block accesses

$$E[X_k] = \sum_{i=1}^m [1 - (1 - p_i)^k] \quad (3.1)$$

where X_k represents the number of blocks accessed by a type 1 query that retrieves k records, $E[X_k]$ represents the expected value of X_k , and m represents the number of blocks holding the relation. Expression (3.1) requires $O(m)$ flops which may be unacceptable in many query optimization settings. In the special case that k is small ($k \ll m$), the expected number of block accesses is roughly equal to k (this result can be obtained by noting that for small k , $(1 - p_i)^k$ is roughly equal to $1 - p_i k$ and simplifying formula (3.1) in the appropriate manner). However, in the remaining cases, more powerful approximation tools are required. We have investigated three approximation methods—generating functions, Taylor series expansions, and majorization—which lead to acceptable closed form expressions for equation (3.1). The majorization technique is the most powerful since it works well for all vectors $\mathbf{p}$. However, the generating function and Taylor series techniques work somewhat better on vectors which contain many duplicate elements.

3.1. Generating Functions

The method of generating functions can be divided into three steps:

1. Transform expression (3.1) into a form that resembles the generating function of a general probability distribution.
2. Approximate the empirical distribution of the p_i 's with a parametric distribution.
3. Use the (hopefully) closed form expression for this distribution's generating function as the estimate for the number of blocks retrieved by a type 1 query.

The transformation step proceeds by first rewriting expression (3.1) as

$$E[X_k] = \sum_{i=1}^m [1 - (1 - \alpha_i/m)^k] \quad (3.2)$$

where $p_i = \alpha_i/m$ and α_i is a nonnegative real number. Reorganizing the terms in the sum we obtain

$$E[X_k] = m \sum_{j=1}^{pnum} (m_j/m) [1 - (1 - \alpha_j/m)^k]$$

where $pnum$ = the number of unique values in the vector $\mathbf{p}$ and m_j = the number of times the j th distinct value occurs in the vector $\mathbf{p}$. This formula reorganizes the original expression by aggregating blocks with equal probabilities of being accessed and representing the size of each such group by m_j . We now set $k = \lambda m$ (λ can be viewed as a proportionality constant that links m and k) and employ the approximation $(1 - \alpha_j/m)^{\lambda m} \approx e^{-\lambda \alpha_j}$ (recall that $k = \lambda m$) to transform this expression to the following form

$$\begin{aligned}
 E[X_k] &= m \sum_{j=1}^{pnum} (m_j/m)(1 - e^{-\lambda a_i}) \\
 &= m - m \sum_{j=1}^{pnum} (m_j/m) e^{-\lambda a_i} \quad (3.3)
 \end{aligned}$$

The sum in (3.3) represents the generating function $g(s) = \sum \pi(i)s^i$ of a discrete random variable ζ which takes the value i with probability (m_i/m) . In this instance the generating function is being evaluated at the point $s = e^{-\lambda}$. Thus, if a discrete theoretical distribution could be fitted to the a_i 's, there is a strong possibility that (3.3) could be reduced to a simple closed form. Unfortunately, discrete parametric distributions cannot be employed in the present problem since the a_i 's are real numbers rather than integers. This obstacle can be circumvented in two ways. First, if the designer is determined to use a discrete distribution, then instead of writing p_i as a_i/m , we write it as a_i/m' where a_i is an integer and m' is a suitably large integer such that $a_i/m' \approx a_i/m$. It is now permissible to associate a discrete distribution with the a_i 's. For example, if the a_i 's are binomially distributed with parameters $amax$ and p , then the closed form expression for $E[X_k]$ is

$$E[X_k] = m[1 - (1 - p + pe^{-k/m'})^{amax}]$$

where $(1 - p + pe^{-k/m'})^{amax}$ is the generating function for this binomial distribution evaluated at the point $e^{-k/m'}$. For notational convenience we have employed k rather than λm .

Alternatively if we wish to stick with real numbers, we can utilize a continuous parametric distribution. If $\pi(a)$ is a probability density function that accurately models the distribution of the a_i 's, we can write

$$E[X_k] \approx m \left[1 - \int_0^{amax} e^{-\lambda a} \pi(a) da \right]$$

To satisfy the restriction $\sum m_i a_i = m$ (alternatively $\sum \pi(i) a_i = 1$), we require that $\pi(a)$ have a mean of 1. In other words, we require that $\int \pi(a) a da = 1$. The expression $\int e^{-$

$\lambda \pi(\alpha) d\alpha$ represents a Laplace transform of the density function $\pi(\alpha)$, or alternatively, the generating function of this distribution evaluated at $e^{-\lambda}$.

Many empirical distributions that arise in practice lend themselves to approximation by parametric distributions. For example, Fedorowicz [1982, 1984] successfully applied regression analysis to fit a Zipf distribution to the keywords of a medical database. More generally, Zipf distributions have been found to accurately model many text and name distributions [Zipf 1949; Booth 1967; Siler 1976]. When a parametric distribution is used to model an empirical distribution, the amount of error in the block estimate approximation is determined solely by the goodness of the fit between the two distributions; the generating function itself contributes no error since it is precise.

3.2. Taylor Series Expansion

If it is necessary to use the empirical distribution of the block access probabilities, p_i , then the following three step Taylor series expansion can be employed to obtain an approximation for $E[X_k]$:

1. Rewrite X_k as a function of a random variable p , where p represents the probability of accessing an arbitrary block. X_k can now be written as follows,

$$X_k = f(p) = m[1 - (1 - p)^k]$$

2. Expand the function as a Taylor series about the mean and truncate the series after the third term. The general formula will look as follows

$$X_k = f(p) \approx f(u) + (p - u)f'(u) + 1/2(p - u)^2 f''(u)$$

where u denotes the mean of the random variable p . In the case at hand, u represents the average probability of accessing a block. Thus $u = 1/m$.

3. Take the expected value of both sides

$$\begin{aligned} E[X_k] &= E[f(p)] = f(u) + E[p - u]f'(u) + 1/2E[(p - u)^2]f''(u) \\ &= f(u) + 1/2\sigma^2 f''(u) \end{aligned} \tag{3.4}$$

where σ^2 denotes the variance of the random variable p . Since $f(p) = m[1 - (1 - p)^k]$, we have $f'(p) = -mk(k - 1)(1 - p)^{k-2}$. Plugging this information into expression (3.4), we obtain the approximation,

$$E[X_k] \approx m \left[1 - (1 - 1/m)^k - k(k-1)(1 - 1/m)^{k-2} \sigma^2 / 2 \right] \quad (3.5)$$

Expression (3.5) can be simplified by noting that $(1-1/m)^k = (1-1/m)^{\lambda m}$ and employing the approximation $(1-1/m)^{\lambda m} \approx e^{-\lambda}$. Thus we obtain

$$\begin{aligned} E[X_k] &\approx m \left[1 - e^{-\lambda} - \lambda^2 m(m - 1/\lambda) e^{-\lambda + 2/m(\sigma^2)/2} \right] \\ &\approx m \left[1 - e^{-\lambda} [1 + \lambda^2 m^2 \sigma^2 / 2] \right] \end{aligned}$$

This last expression demonstrates how the proportionality constant λ and the variance σ^2 affect $E[X_k]$. For small λ , the correction factor $e^{-\lambda} \lambda^2 m^2 \sigma^2 / 2$ has very little impact, but its influence increases as λ increases. These approximations work well when the random variable p has a small third moment (i.e., $E[(p - u)^3]$ is small) or the value of the third derivative evaluated at u tails off rapidly. These requirements demand either that the vector $\mathbf{p}$ have relatively few unique values (leading to a small third moment) or that k not be overly large (leading to a rapid tailing off of the third derivative). If each block has an equal chance of being accessed, then the variance is zero and this expression reduces to the well known formula

$$E[X_k] \approx m \left[1 - (1 - 1/m)^k \right]$$

Figures 3.1 and 3.2 provide numerical examples which illustrate the use of Taylor series approximations. Each of these figures assumes that a type 1 query is directed at a relation which contains 25,000 tuples spread over 500 blocks. In figure 3.1, we have employed expression (3.1) to plot the block access curves of four different distributions, and in figure 3.2 we have plotted the percentage differences between the estimates produced in figure 3.1 by the analytic formula and the estimates produced by the Taylor series estimates. Several conclusions may be

Figure 3.1: Expected number of blocks accessed by a type 1 query that retrieves k records when the block accesses have the following distributions: 1) equipfrequent; 2) uniform; 3) normal; and 4) exponential

Figure 3.2: Percentage difference between the Taylor series estimates and the analytic estimates of the number of blocks accessed when a type 1 query retrieves k records and when the block accesses have the following distributions: 1) uniform; 2) normal; and 3) exponential

gleaned from these figures. First, the cost estimates produced by each of the four assumptions behave approximately linearly for $k < 100$ and $k > 1000$, and logarithmically for $100 < k < 1000$. Or, put another way, the cost estimates grow linearly for $\lambda < .2$, logarithmically for $.2 < \lambda < 2$, and linearly again for $\lambda > 2$. Second, figure 3.1 demonstrates that the equipfrequency assumption provides reasonably accurate cost estimates when the block access distribution is either uniformly or normally distributed. In each of these cases, the error remains under 5% for $k < 100$, then rises somewhat rapidly to a peak of 20% for values of k around 1000 before slowly declining as the number of block accesses approaches 500. On the other hand, the equipfrequency assumption performs quite poorly for exponential block access distributions. In this instance, the error rises above 10% before k reaches 100 and rises rapidly, hitting 50% for values of k around 1000 and declining very slowly to 37% for large k (in our experiment with the exponential distribution, 135 blocks had zero probability of containing a record; thus for large k the appropriate cost estimate was 365 blocks and not 500 blocks). These results provide empirical justification for the majorization results discussed in section 2—as the block access distribution becomes increasingly skewed, the expected number of blocks accessed begins to decline dramatically.

Figure 3.2 also exhibits several interesting characteristics. First, it indicates that the Taylor series expansion is fairly reliable for $\lambda < 2$ ($k < 1000$). Beyond this point, the reliability of the approximation depends on the skewness of the distribution. No matter what the block access distribution looks like, the Taylor series approximation eventually reaches 500. On the other hand, a skewed distribution is likely to assign zero probabilities to at least several blocks. Thus accurate cost estimates will always be less than 500 blocks. This point is illustrated by figure 3.1. As the skewness of the distribution increases, the number of blocks

that will never be accessed also increases and thus the Taylor series expansion becomes increasingly unreliable.

3.3 Majorization

Whereas the methods of generating functions and Taylor series work well only when the duplication rate in the $\mathbf{p}$ vector is relatively high, the method that we introduce in this section works well regardless of the duplication rate in the $\mathbf{p}$ vector. This method attempts to compute a tight upper bound on the analytical estimate of the number of accessed blocks by replacing the original $\mathbf{p}$ vector with a vector that contains very few distinct values (i.e., a vector with many duplicate values) and that is majorized by the original $\mathbf{p}$ vector. The properties of Schur concavity and majorization that we defined in section 2.4 guarantee that this computed estimate will be an upper bound for expression (2.1).

The first step of the majorization method involves permuting the elements of $\mathbf{p}$ so that they form a decreasing sequence $\mathbf{p} = \langle p_1', \dots, p_a' \rangle$. We can approximate the distribution of these permuted elements using a histogram with unequal intervals (see figure 3.3) [Piatestky-Shapiro 1984]. The height of each box in the histogram is equal to the average block access probabilities of the elements which comprise it. In effect we are approximating the nonuniform attribute distribution with a series of uniform distributions. We now replace the elements in each interval with a like number of equal-sized elements whose block access probability is equal to the average block access probability of the original elements. The original elements of each interval represent a sub-vector of $\mathbf{p}$ while the new elements represent a sub-vector of equal-sized components. Since a vector $\mathbf{y} = \langle y_1, \dots, y_a \rangle$ with equal-sized elements ($y_1 = y_2 = \dots = y_a$) is majorized by any vector $\mathbf{z} = \langle z_1, \dots, z_a \rangle$ with nonincreasing non-negative reals satisfying the restriction $\sum z_i = \sum y_i$, the equal-sized sub-vectors are majorized by the sub-vectors of $\mathbf{p}$. Thus by concatenating these equal-sized vectors, we obtain a vector which is majorized by $\mathbf{p}$. However, since this

Figure 3.3: Histogram approximation for the distribution of a sample $\mathbf{p}$ vector.

new vector contains many fewer distinct elements than $\mathbf{p}$, the desired cost formula can be quickly approximated using this vector. For example, the approximation for expression (3.1) can be written as follows

$$E[X_k] \approx \sum_{i=1}^{m'} u_i [1 - (1 - p_i'')^k]$$

where m' = number of intervals in the histogram, u_i = number of elements in the i th interval of the histogram, and p_i'' = average block access probability of the elements in the i th interval of the histogram. Since expression (3.1) is Schur concave, we are guaranteed that this approximation is at least as large as the estimate produced by (3.1).

In practice only 2-5 intervals are required to achieve extremely tight upper bounds on this estimate. Figures 3.4 and 3.5 demonstrate this assertion for a relation that consists of 25,000 tuples distributed over 500 blocks. In figure 3.4, we

Figure 3.4: Expected number of blocks accessed by a type 1 query that retrieves k records. The block access distribution of this query is Zipf distributed with decay factor z .

Figure 3.5: Percentage difference between the block estimates produced by the majorization technique with two and five intervals and expression (3.1). The block access distribution of this type 1 query is Zipf distributed with decay factor z .

have employed expression (3.1) to plot the variations in block accesses that occur when the block access distribution is Zipf distributed ($p_i = h/i^z$ where $h = 1/\sum(1/i^z)$ and z represents a factor of decay), and in figure 3.5 we have plotted the percentage differences between the estimates produced in figure 3.4 by the analytic formula and the estimates produced by the majorization technique. Figure 3.4 uses a Zipf distribution to illustrate the impact that increasingly skewed block access distributions have on the estimates of the number of blocks retrieved by a type 1 query. Intuitively, the decay parameter z can be thought of as an indication of the amount of correlation that exists between the queried attribute and the clustered attribute. As the correlation increases, the z parameter increases and the block access distribution becomes increasingly skewed. The results plotted in figure 3.4 strongly suggest that the designer should not assume that blocks have equal probabilities of being accessed when a fairly strong relationship exists between two attributes.

Figure 3.5 demonstrates the power of the majorization technique—in some instances a judicious choice of two intervals achieves improvements of more than 2000% as compared with the equiprequent (one interval) estimate. If the query optimizer is willing to incur the added cost of a few microseconds, it can achieve improvements of 6000% or better by employing five intervals. Further, the majorization technique will not produce a block access estimate that causes the query optimizer to make the wrong decision. The same cannot be said of the equipfrequency approximation. For example, when $z=2$ and $k=1000$, the equiprequent estimate would lead a query optimizer to adopt a strategy that scans the entire relation. The proper strategy is an index lookup, a strategy indicated by the majorization technique.

4. TYPE 2 QUERY – NONUNIFORM ATTRIBUTE DISTRIBUTIONS

In this section we will begin to consider the impact that nonuniform attribute distributions have on the average number of blocks retrieved by a type 2 query. Specifically we examine selection queries of the form "retrieve t.all where t.A = v" and v can range over all values in the domain of A.

Assuming random placement of records, we can obtain an analytical estimate for the number of accessed blocks by setting $p_i = 1/m$ in expression (2.2) (recall that p_i represents the probability that a query accesses the i th block and m denotes the number of blocks in the file)

$$E[X_a] = (m/a) \sum_{i=1}^a [1 - (1 - 1/m)^{k_i}] \quad (4.1)$$

In this expression, a represents the number of unique values in the attribute domain and k_i denotes the number of times the i th attribute value is duplicated. This formula requires $O(a)$ flops and $O(a)$ space. Such performance may be unacceptable in both query optimization and physical design settings. Therefore we will examine alternative methods for approximating this expression.

In two special cases the approximations are quite simple. If each of the values in an attribute domain is duplicated relatively infrequently (i.e., each value appears in roughly n/a tuples where n is the number of tuples in the relation) and the average duplication rate is much less than the number of blocks ($n/a \ll m$), then the average number of blocks accessed by a type 2 query is n/a . The following simple analysis demonstrates this result

$$\begin{aligned} E[X_a] &= (m/a) \sum_{i=1}^a [1 - (1 - 1/m)^{k_i}] \\ &\approx (m/a) \sum_{i=1}^a [1 - (1 - k_i/m)] \\ &\approx (1/a) \sum_{i=1}^a k_i = n/a \end{aligned}$$

Alternatively if the duplication rate of most of the attribute values greatly exceeds the number of blocks, then the average number of blocks retrieved is roughly m . This result can be obtained by noting that $(1 - 1/m)^{k_i} \approx 0$ for $k_i \gg m$ and therefore each such term contributes m/a blocks to the final result. By our assumptions $k_i \gg m$ for most i so most terms will contribute the full amount m/a . The relatively few terms that remain will remove at most a few blocks from the final answer.

The approximation techniques that may be utilized in the remaining cases depend on the amount of duplication in the k vector. If there is a large number of duplicate elements in the k vector then the generating function and Taylor series expansions work well, otherwise the technique of majorization works best.

4.1 Generating Functions

If the k vector contains a high proportion of duplicate elements, then it is reasonable to assume that the relation contains a large number of distinct attribute values. When this condition holds, most of the attribute values will be duplicated relatively infrequently although a few will be duplicated fairly often. Thus a probability distribution can be meaningfully associated with the attribute frequency rate. Formally, let k represent the number of times an attribute value is duplicated and let $\pi(k)$ represent its associated probability function. $\pi(i)$ denotes the probability that an attribute value appears in i tuples of a relation. We again modify the occupancy problem introduced in section 2.4 by rewriting the vector k as an integer sequence $k = \langle 1, 2, \dots, d_{\max} \rangle$ where $d_{\max} = \max(k_i)$ is the maximum number of times an attribute value occurs. The number of balls assigned is chosen from this vector in such a manner that the probability of drawing the number i is $\pi(i)$. Thus the appropriate formula for the expected number of occupied urns, or, equivalently the number of block accesses, is

$$E[X_a] = m \left[1 - \sum_{i=1}^{d_{\max}} \pi(i) \left(1 - \frac{1}{m} \right)^i \right]$$

where m = number of blocks holding the relation.

If we associate a discrete distribution with $\pi(i)$, then the previous expression for $E[X_a]$ can be approximated as

$$E[X_a] \approx m \left[1 - \sum_{i=1}^{d_{\max}} \pi(i) e^{-i/m} \right] \quad (4.2)$$

This expression employs the approximation $(1 - 1/m)^i \approx e^{-i/m}$. The sum in expression (4.2) represents the generating function $g(s) = \sum \pi(i) s^i$ of a discrete random variable ζ that takes the value i with probability $\pi(i)$. In this instance the generating function is being evaluated at the point $s = e^{-1/m}$. Since discrete parametric distributions often have closed formulas for their generating functions, expression (4.2) will often collapse to a simple form.

If a continuous parametric distribution approximates the distribution of the k_i 's more closely than a discrete distribution (this assumption is not unreasonable since $d_{\max}$ is moderately large), then $E[X_a]$ is approximated by the expression

$$E[X_a] \approx m \left[1 - \int_1^{d_{\max}} \pi(x) e^{-x/m} dx \right] \quad (4.3)$$

In this expression $\pi(x)$ represents the probability function of the continuous distribution. Note that the expression also uses the approximation $(1 - 1/m)^x \approx e^{-x/m}$. To satisfy the restriction $\sum \pi(i) = n/a$, we require that our continuous distribution have a mean of n/a . Put in words, we require that the mean of the continuous distribution equal the average duplication rate of an attribute value. The integral in expression (4.3) represents a Laplace transform of the continuous distribution's probability function, or equivalently, its generating function evaluated at $e^{-1/m}$. Thus there is a good possibility that this expression will collapse to a simpler form.

4.2 Taylor Series

If database designers wish to utilize the empirical distribution of the attribute value duplication rate (i.e., the distribution of the k_i 's), they can obtain an approximate expression for $E[X_a]$ in the following manner. First, note that

$$E[X_a] = m[1 - E[(1 - 1/m)^K]]$$

K is a random variable representing the number of times an attribute value is duplicated. Next apply a truncated Taylor series expansion to $(1 - 1/m)^K$ using the technique outlined in section 3.2. The mean of K is the average attribute duplication rate, n/a , and the second derivative of the function $(1 - 1/m)^K$ is $(1 - 1/m)^K(\log(1 - 1/m))^2$. Thus, using the formula presented in section 3.2, we obtain

$$\begin{aligned} E[X_a] &\approx m \left[1 - (1 - 1/m)^{n/a} - \sigma^2 (1 - 1/m)^{n/a} (\log(1 - 1/m))^2 / 2 \right] \\ &= m \left[1 - (1 - 1/m)^{n/a} (1 + \sigma^2 (\log(1 - 1/m))^2 / 2) \right] \\ &\approx m \left[1 - (1 - 1/m)^{n/a} (1 + \sigma^2 / 2m^2) \right] \end{aligned} \quad (4.4)$$

Expression (4.4) demonstrates how the variance of a distribution affects the value of $E[X_a]$. Distributions that are nearly uniform will produce almost no correction to the term $(1 - 1/m)^{n/a}$ whereas highly skewed distributions will generate a large correction. In the extreme case where each attribute value is duplicated equifrequently, the formula reduces to the well-known form

$$E[X_a] \approx m \left[1 - (1 - 1/m)^{n/a} \right]$$

If many values in the k vector have the same duplication rate, expression (4.4) provides a good approximation for $E[X_a]$, even for highly skewed distributions. However, our experiments have shown that when the distribution is symmetric, the equifrequency assumption also leads to good approximations. Figures 4.1 and 4.2 illustrate these points for one specific experiment. Each of these figures assumes that we are issuing a type 2 query against a relation that contains 1) 25,000 tuples

Figure 4.1: Expected number of blocks accessed by a type 2 query when the attribute domain has the following distributions: 1) equifrequent; 2) uniform; and 3) normal:

Figure 4.2: Percentage difference between the Taylor series estimates and the analytic estimates of the number of blocks accessed by a type 2 query when the attribute domain has the following distributions: 1) uniform; and 2) normal

distributed over 500 blocks; and 2) 500 unique attribute values but only 20 different duplication rates. In figure 4.1, we use expression (4.1) to plot the block access curves that result when attribute domains are uniformly or normally distributed while in figure 4.2, the percentage differences between the block access estimates produced by the analytic formula (expression (4.1)) and the Taylor series approximation are plotted. Note that as the variance of the distributions increases, the number of block accesses decreases slowly. However, as figure 4.2 illustrates, at no point does the error introduced by the equiprequent assumption exceed 8%.

The reason the equiprequent assumption performs so well is that each duplication rate in a symmetric distribution has a "mirror" image on the other side of the mean. If we average the number of blocks required to execute a query that specifies an attribute value with a fixed duplication rate and a query that specifies a value with a "mirror" image duplication rate, we find that this number is approximately equal to the number of blocks accessed if the query specifies an attribute value with the mean duplication rate. For example, in an experiment that employed a normal distribution with mean 250 and variance 75 (i.e., each attribute value is duplicated an average of 250 times with a variance of 75), we had 7 attribute values that were duplicated 158 times and 7 that were duplicated 342 times. The average number of blocks accessed by a query which selected one of these 14 values is 192 blocks whereas the average number of blocks retrieved by a query that selects 250 records is 197 blocks, a difference of 2.5%. Thus the equiprequent assumption provides good cost estimates when the block access distribution is symmetric. For nonsymmetric distributions or symmetric distributions with very large variances this good performance degrades. This point will be demonstrated in the next section.

4.3 Majorization

When the amount of duplication in the k vector is fairly low (i.e., most attribute values have different duplication rates), then the best approximation technique involves majorization. The same method that was outlined in section 3.3 works here as well so we will proceed directly to numerical examples. Figure 4.3 uses expression (4.1) to plot the expected number of blocks that must be accessed to satisfy a type 2 query when the duplication rate of the attribute values follows a Zipf distribution (i.e., $k_i = h / i^z$ where $h = 1 / \sum 1/i^z$ and z is the factor of decay). Figure 4.4, on the other hand, plots the percentage differences between the estimates produced in figure 4.3 by the analytic formula and the estimates produced by the majorization technique. We assume that the relation contains 25,000 tuples distributed over 500 blocks with 20 unique attribute values. Figure 4.3 demonstrates that as the z parameter increases from 0 to 3 (i.e., as the skewness of the distribution increases), the number of blocks retrieved decreases by a factor of approximately 5. Thus making the equifrequency assumption could lead to serious errors if the factor of decay is somewhat high. On the other hand, if the factor of decay is somewhat high, then all but a handful of values will be duplicated relatively infrequently with the remaining values duplicated extremely often. In this case, it might be helpful to consider the query distribution (the frequency with which various values are accessed) to determine which values are accessed most frequently. For example, if the infrequently duplicated values are accessed most often, it might be helpful to remove the most frequently duplicated values from the calculation since they skew the block estimates upward.

Figure 4.4 demonstrates that the majorization approximation yields excellent results, even when only two intervals are used. Specifically, the error introduced by using two intervals remains under 5% for $z < 1$, increases to 10% for

Figure 4.3: Expected number of blocks accessed by a type 2 query when the attribute domain is Zipf distributed with decay factor z .

Figure 4.4: Percentage difference between the block estimates produced by the majorization technique with two and five intervals and expression (4.1). The type 2 query is being directed against a Zipf distributed attribute domain.

values of z between 1 and 2 and eventually reaches 20% for z equal 3. For five intervals this performance is even better—the relative error never exceeds 3%. This reliability contrasts sharply with the reliability of the cost estimates under the equifrequent assumption. Under the equifrequent assumption, the relative error increases to 20% for z around 1, then rises sharply to a maximum error of 350% for z equal to 3.

Another, perhaps more reliable, way to compare the majorization and equifrequent approaches is to compare the actions a designer might choose under each of these two assumptions. For lower z , the designer would probably omit an index under both assumptions. However, for larger values of z , the majorization approach would lead the designer to place an index on this field while the equifrequent assumption would mislead the designer into omitting an index.

5. TYPE 2 QUERY – NONUNIFORM ATTRIBUTE DISTRIBUTION AND NON-RANDOM RECORD PLACEMENT

In this section we derive a cost estimate for a type 2 query when the attribute values are nonuniformly distributed and the records are nonrandomly placed in secondary storage. Thus we are considering a generalization of the queries in sections 3 and 4. Actually two cost estimates are derived. In the first instance we assume that queries on different attribute values share the same probability distribution for block accesses, while in the second instance we assume they have different distributions. In other words, the first case assumes that each attribute value shares the same vector $\mathbf{p}$ while the second case assumes that each attribute value has a different vector $\mathbf{p}$.

5.1 Fixed Block Access Distribution

We will first consider the case where the attribute values share a common $\mathbf{p}$ vector. In section 2.4 we derived the following analytical expression for the number

of block accesses

$$E[X_a] = 1/a \sum_{i=1}^a \sum_{j=1}^m [1 - (1 - p_j)^{k_i}] \quad (5.1)$$

In this expression, X_a = number of blocks retrieved by a type 2 query on an attribute domain that contains a unique attribute values, m = number of blocks, p_j = probability that the query accesses the j^{th} block, k_i = number of tuples which contain the i^{th} attribute value. The $O(am)$ floating point operations required to compute this expression may be prohibitive in many design settings (the $O(a+m)$ space required to store the k and p vectors may also be prohibitive), so we again need to evaluate various alternatives for approximating this expression.

Each of the approximation techniques we consider proceeds in two steps. We first hold one of the indices, either i or j , constant and approximate the resulting expression using one of the techniques outlined in sections 3.1–3.3 and 4.1–4.3. We then use the majorization technique to shrink the remaining vector and use this vector to calculate our final answer.

As an example, suppose we hold i constant (i.e., fix an attribute duplication rate) and use the generating function approach to approximate the resulting type 1 query. We first write $p_j = \alpha_j/m$ and treat α as a random variable (see section 3 for details). From section 3.1 we obtain

$$E[X_a] = m - (m/a) \sum_{i=1}^a E[e^{-\alpha k_i/m}] \quad (5.2)$$

where $E[e^{-\alpha k_i/m}]$ represents the generating function of the blocking factor distribution evaluated at $e^{-k_i/m}$. We next employ the majorization technique discussed in section 3.3 to construct a vector which is majorized by the vector k . A good approximation for $E[X_a]$ can be obtained by plugging this vector into expression (5.2) and rewriting it as

$$E[X_a] = m - (m/a) \sum_{i=1}^a u_i E[e^{-\alpha k'_i/m}]$$

where a' = number of distinct values in the constructed vector (number of intervals in the histogram), u_i = number of times the i th distinct value occurs in the constructed vector (number of elements in the i th interval), and k_i' = the value of the i th distinct element in the constructed vector (average value of the elements in the i th interval of the histogram).

The process outlined above can also be used if the majorization or Taylor series procedures are chosen as the approximation method in the first step.

5.2 Varying Block Access Distributions

We now consider the case where the probability distribution of block accesses varies depending on which attribute value is specified. This problem can be reduced to an occupancy problem by assuming that the probability of assigning a ball to the i th urn on each toss is a function of the value selected from the k vector. The expression for the number of expected block accesses becomes

$$E[X_a] = \frac{1}{a} \sum_{i=1}^a \sum_{j=1}^m [1 - (1 - p_{ij})^{k_i}]$$

where a = number of distinct attribute values, m = number of blocks, p_{ij} = probability that the j th block is accessed by a query which retrieves the i th attribute value, and k_i = the number of times the i th attribute value is duplicated.

We can no longer use our "divide and conquer" strategy since it works by considering only one variable (and thus one vector) at a time and our new expression requires us to consider two variables (and thus two vectors) at a time. Thus instead of directly finding an approximation for $E[X_a]$, we must compromise and find approximate lower and upper bounds. We begin by imagining each of the a block access distributions as a vector of m components $\langle p_{i1}, p_{i2}, \dots, p_{im} \rangle$. These vectors can be ordered so that their components form a decreasing sequence. We denote these permuted components by p_{ij}' . Consider the attribute value with the largest number

of replications (for the sake of convenience let it be the first attribute value). The tuples which contain this value probably will be spread over more blocks than the tuples associated with any of the other values and thus its block access distribution will be more "uniform" than the distributions for the remaining values. Put another way, the first few components of its permuted vector probably will be slightly smaller than the first few components of the other vectors and its last components a bit larger. Thus there is a good possibility that this vector is majorized by each of the remaining vectors. If this majorization holds, then the property of Schur concavity ensures that the following inequality holds

$$\frac{1}{a} \sum_{i=1}^a \sum_{j=1}^m (1 - (1 - p_{ij})^k) \geq \frac{1}{a} \sum_{i=1}^a \sum_{j=1}^m [1 - (1 - p_{ij})^k] = E[X_a] \quad (5.3)$$

The expression on the left side of this inequality represents an upper bound for $E[X_a]$ and it can be evaluated using the approximations introduced for the first case in this section.

We can obtain a lower bound by arguing in the same manner as above. The attribute value with the smallest number of replications (for convenience let it be the a^{th} attribute value) will probably have a more highly skewed distribution for block accesses than the remaining attribute values. Thus there is a reasonable probability that its permuted vector majorizes the vectors of the remaining attribute values. Using the previous inequality, we obtain the following lower bound for $E[X_a]$

$$\frac{1}{a} \sum_{i=1}^a \sum_{j=1}^m (1 - (1 - p_{aj})^k) \leq \frac{1}{a} \sum_{i=1}^a \sum_{j=1}^m [1 - (1 - p_{ij})^k] = E[X_a] \quad (5.4)$$

Again this lower bound can be evaluated using the techniques described for the first case of this section.

Even if the permuted vector of the most frequently occurring attribute value is not majorized by the block access vectors associated with the other attribute

Figure 5.1: Expected number of blocks accessed by a type 2 query when 1) the attribute domain is Zipf distributed with decay factor z ; and 2) the attribute is related to the clustered attribute.

values, it is probably less skewed than most of these vectors and thus expression (5.3) should be a very accurate approximation for the true upper bound. The same argument holds for the lower bound in expression (5.4). The error introduced by employing these bounds as an approximation for $E[X_a]$ depends on how closely the block access distributions for the least and most frequently occurring attribute values match the distributions for the remaining values. As an example, we performed several experiments with a relation that contained 25,000 tuples distributed over 500 blocks and 20 unique attribute values. We assumed that the attribute values were Zipf distributed with a factor of decay that varied between 0 and 2. We also assumed that the queried attribute was strongly dependent on the clustered attribute. This assumption implied that the tuples corresponding to a given attribute value were somewhat contiguous in secondary memory. To model this situation, we utilized a Zipf distribution whose factor of decay parameter (z) varied between 0 and 3. Higher values of z were associated with attribute values that are relatively infrequent. The effect of this assignment was to increase the number of blocks occupied by an attribute value as its duplication rate increased. Figure 5.1 plots the expected number of blocks accessed by a type 2 query under these assumptions, as well as the approximations obtained using lower and upper bounds. The results again demonstrate the strong impact that correlation has on the number of blocks retrieved. The equifrequency assumption would indicate that an index on the queried attribute is unnecessary. However, the majorization technique correctly predicts that an index on this field is in fact quite beneficial (although sole reliance on the upper bound for higher factors of decay might mislead a designer into omitting the index).

6. Conclusion

Most database systems assume that attributes are independent and uniformly distributed and that blocks are accessed equifrequently. In this paper we have demonstrated that such assumptions can cause a physical design or a query optimizer to overestimate the block cost of a query by more than 6000% (see section 3). Thus we have developed simple, accurate formulas that estimate the cost of a query in a variety of database settings. Specifically, we have shown how the cost of selection-project-join queries can be estimated when 1) attributes are dependent; 2) attribute values are nonuniformly distributed; and 3) records are nonrandomly placed on disks. The approximation methods introduced in this paper – generating functions, Taylor series expansions, and majorization – also can be used to simplify the formulas that we obtain when variable blocking factors and nonuniform query distributions are incorporated into a database model. Used in tandem, these cost estimate expressions provide a powerful tool for designers who wish to realistically model data distributions and access patterns in physical database design and query optimization.

REFERENCES

- Bitton, D. and DeWitt, D. 1983. Duplicate Record Elimination in Large Data Files. *Trans. Database Syst.*, 8, 2 (June), 255-265.
- Booth, A.D. 1967. A "Law" of Occurrences for Words of Low Frequency. *Information and Control*, 10, 386-393.
- Cardenas, A.F. 1975. Analysis and Performance of Inverted Database Structures. *Commun. ACM*, 18, 5 (May), 253-263.
- Christodoulakis, S. 1981. Estimating Selectivities in Data Bases. *Technical Report CSRG-136*, (Dec.), University of Toronto.
- Christodoulakis, S. 1983a. Estimating Block Transfers and Join Sizes. In *Proceedings SIGMOD 1983 Conference*, ACM, 40-50.

- Christodoulakis, S.** 1983b. Estimating Record Selectivities. *Information Systems*, 8, 2, 105-115.
- Christodoulakis, S.** 1984a. Estimating Block Selectivities. *Information Systems*, 9, 1, 69-79.
- Christodoulakis, S.** 1984b. Implications of Certain Assumptions in Database Performance Evaluation, *Trans. Database Syst.*, 9, 2 (June), 163-186.
- Conte, S.D. and deBoor, C.** 1980. *Elementary Numerical Analysis: An Algorithmic Approach*. New York, McGraw-Hill.
- Demolombe, R.** 1980. Estimation of the Number of Tuples Satisfying a Query Expressed in Predicate Calculus Language. In *Proceedings of the 6th International Conference on Very Large Data Bases* (Montreal, Oct. 1-3). IEEE, New York, pp. 55-63.
- Fedorowicz, J.E.** 1982. A Zipfian Model of an Automatic Bibliographic System: An Application to MEDLINE. *Journal of the American Society for Information Science*, 33, 4 (July), 223-232.
- Fedorowicz, J.** 1984. Database Evaluation Using Multiple Regression Techniques. In *Proceedings SIGMOD 1984 Conference*, (Boston, Mass., June 18-21). ACM, 70-77.
- Feller, W.** 1968. *An Introduction to Probability Theory and its Applications*. Vol 1. 3rd Ed. New York, Wiley.
- Johnston, J.** 1972. *Econometric Methods*, 2nd Ed. McGraw-Hill, New York.
- Kamel, N. and King, R.** 1985. A Model of Data Distribution Based on Texture Analysis. In *Proceedings SIGMOD 1985 Conference*, (Austin, Texas, May 28-31). ACM, 319-325.
- Kershberg, L., Ting, P.L. and Yao S.B.** 1982. Query Optimization in Star Computer Networks. *Trans. Database Syst.*, 7, 4 (Dec), 678-711.
- Kotz, S. and Johnson, N.L.** 1977. *Urn Models and Their Application*, New York, John Wiley and Sons, 1977.
- Luk, W.S.** 1983. On Estimating Block Accesses in Database Organizations. *Commun. ACM*, 26, 11 (Nov.), 945-947.
- Merrett, T.H. and Otoo, E.** 1979. Distribution Models of Relations. In *Proceedings of the 5th International Conference on Very Large Data Bases* (Rio de Janeiro, Brazil). IEEE, New York, 418-425.
- Montgomery, A.I., D'Souza, D.J., and Lee, S.B.** 1983. The Cost of Relational Algebraic Operations in Skewed Data: Estimates and Experiments. In *Information Processing 83*. Elsevier North-Holland, New York, 235-241.
- Palvia, P. and March, S.T.** 1984. Approximating Block Accesses in Database Organizations. *Information Processing Letters*, 19, (Aug.), 75-79.

Piatestsky-Shapiro, G. and Connell, C. 1984. Accurate Estimation of the Number of Tuples Satisfying a Condition. In *Proceedings SIGMOD 1984 Conference*, (Boston, Mass., June 18-21). ACM, 256-276.

Rothnie, J.B., Jr. and Lozano, T. 1974. Attribute Based File Organization in a Paged Memory Environment. *Commun. ACM* 17, 2 (Feb.), 63-69.

Siler, K.F. 1976. A Stochastic Evaluation Model for Database Organizations in Data Retrieval Systems. *Commun. ACM* 19, 2 (Feb.), 84-95.

Waters, S.J. 1967. Hit Ratios. *Computer Journal*, 19, 1, (Feb.), 21-24.

Whang, K-Y., Wiederhold, G., and Sagalowicz, D. 1983. Estimating Block Accesses in Database Organizations: A Closed Noniterative Formula. *Commun. ACM*, 26, 11 (Nov.), 940-944.

Yao, S.B. 1977. Approximating Block Accesses in Database Organizations. *Commun. ACM* 20, 4 (April), 260-261.

Yu, C.T., Siu, M.K., Lam, K., and Ozsoyoglu, M. 1979. Performance Analysis of Three Related Assignment Problems. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data* (Boston, Mass., May 30-June 1). ACM, New York, 82-92.

Zahorjan, J., Bell, B.J., and Sevcik, K.C. 1983. Estimating Block Transfers when Record Access Probabilities are Non-Uniform. *Information Processing Letters*, 16, 5 (June), 249-252.

Zipf, G.K. 1949. *Human Behavior and the Principle of Least Effort*. New York, Addison Wesley.