AN EMPIRICAL COMPARISON OF HISTOGRAM-LIKE TECHNIQUES FOR QUERY OPTIMIZATION

B. John Oommen[1*] and Luis G. Rueda[2†]

[1] *Senior Member, IEEE. School of Computer Science, Carleton University.*
1125 Colonel By Dr., Ottawa, ON, K1S 5B6, Canada.
oommen@scs.carleton.ca

[2] *School of Computer Science, Carleton University.*
1125 Colonel By Dr., Ottawa, ON, K1S 5B6, Canada.
lrueda@scs.carleton.ca

Abstract

We consider the problem of Query Optimization which consists of a database system choosing, among many different Query Evaluation Plans (QEP), the most economical one for a given query. Since the number of QEPs increases exponentially with the number of relations involving the query, query optimization is a very complex problem. Many estimation techniques have been developed in order to approximate the cost of a QEP, including the *Equi-width* [1, 2], the *Equi-Depth* [3, 4], the *Rectangular Attribute Cardinality Map (R-ACM)* [5, 6], and the *Trapezoidal Attribute Cardinality Map (T-ACM)* [5]. For our analysis we have developed a formal, rigorous prototype model used to analyze these methods on random databases. We show that as the error of an estimation technique increases, the probability of choosing the optimal QEP decreases. We also formally show how the estimates can be obtained in joined relations. The research also contains a detailed description of the rigorous prototype model and its implementation, and an empirical comparison of these histogram schemes.

Keywords: Query Optimization, Histogram Methods, Query Evaluation Plans.

1 Introduction

Query optimization in relational and object oriented database systems has been studied for many years. It is well known that the most important operation between tables is the join on a particular attribute. When a user requests a query, it can be performed by means of different intermediate (join) operations. A simple sequence of join operations that leads to the same final result is called a *Query Evaluation Plan* (QEP). Each QEP has an associated internal cost, which depends on the number of operations performed in the intermediate joins. The problem of choosing the best QEP is a combinatorially explosive optimization problem. This problem is currently solved by estimating the query result sizes of the intermediate relations and selecting the most efficient access QEP.

Since the analysis of selecting the best QEP must be done in "real" time, it is not possible to inspect the real data in this phase. Consequently, query result sizes are usually estimated using statistical information about the structures and the data maintained in the database catalogue. This information is used to approximate the distribution of the attribute values in a particular relation. Hence the problem of selecting the best QEP depends on how well that distribution is approximated.

In [7], it has been shown that errors in query

*Partially supported by NSERC, the Natural Science and Engineering Research Council of Canada.

†Partially supported by Departamento de Informática, Universidad Nacional de San Juan, Argentina.

result size estimates may increase exponentially with the number of joins. Since current databases and the associated queries increase in complexity, numerous efforts have being made to devise more efficient techniques that solve the query optimization problem.

Several techniques have been proposed to estimate query result sizes, including histograms, sampling, and parametric techniques [2, 3, 4, 8, 9, 10, 11]. Of these techniques, histograms are the most commonly used form of statistical information. They are incorporated in most of the commercial database systems such as Oracle, Microsoft SQL Server, Teradata, and DB2, all of which primarily use the Equi-depth histogram.

In this paper, we focus on methods that use non-parametric statistics, i.e. histogram-based techniques. First of all, we have developed a formal model for random databases with random attributes and random queries. Using this rigorous model, we have developed a prototype that tests the four prominent models of histograms known in the literature: *Equi-width* [1, 2], *Equi-Depth* [3, 4], the *Rectangular Attribute Cardinality Map (R-ACM)* [5, 6], and the *Trapezoidal Attribute Cardinality Map (T-ACM)* [5, 12]. Our empirical results demonstrate that the R-ACM chooses a superior QEP more than three times as often as the Equi-width and Equi-depth. In the case of the T-ACM the ratio is even greater than four. Indeed, in the most general scenario, we analytically prove that under certain models, the better the accuracy of an estimation technique, the greater the probability of choosing the most efficient QEP.

2 Histogram Techniques

Histogram based techniques are the most used methods among the non-parametric estimates. These techniques approximate the distribution of an attribute value by a pre-computed histogram. Basically, the attribute value set is divided in *intervals* or *buckets*, each of them with an arbitrary number of values. These data values are later used for approximating the individual counter for each value. The approximation is achieved by interpolation.

2.1 The Equi-Width Histogram

The concept of the Equi-width histogram was used in statistical literature and in statistical pattern recognition for many decades. However, its use in databases was predominantly advocated by Piatetsky-Shapiro and Connel [3]. The basic idea of this technique is to divide the set of attribute values in intervals or buckets that have, *as approximately as possible*, the same width. The value of each attribute value in the bucket is approximated by the height of the bucket. The height is calculated as the summation of the frequencies divided by the number of values in the bucket. The frequency for each attribute value is the number of tuples in the relation that have this attribute value. This scheme is formally given in [3, 5, 13].

2.2 The Equi-Depth Histogram

The structure of the Equi-depth histogram has also been extensively studied both by statisticians and by researches in pattern recognition. However, its use in databases was also primarily motivated by the work of Piatetsky-Shapiro and Connel [3]. Basically, this technique consists of dividing the set of attribute values in buckets that have *as approximately as possible* the same population[1]. The frequency of a value is calculated (approximated) by dividing the population in the bucket by the number of attribute values that belong to the bucket. This is formalized in [3, 5, 13].

2.3 The Rectangular Attribute Cardinality Map

The mathematical foundation for the R-ACM is the previous work due to Oommen and his student Nguyen, in which they proposed a general strategy for obtaining *Moment-Preserving Piecewise Linear Approximations* of functions [14]. This technique approximates a function by preserving a finite number of geometric moments which are related to the transform domain parameters of a function.

The Rectangular Attribute Cardinality Map (R-ACM) histogram of a given attribute, in its simplest form, is a one-dimensional integer array that

[1] The population is the number of tuples in which the attribute has any of the values in the range.

stores the count of the tuples of a relation corresponding to that attribute, and for some subdivisions for the range of values assumed by that attribute [5, 6]. The R-ACM is, in fact, a modified form of the histogram. But unlike the two major forms of histograms, namely, the Equi-width histogram (in which all the sector widths are equal) and the Equi-depth histogram (in which the number of tuples in each histogram bucket is equal) the R-ACM has a variable sector width, and has varying number of tuples in each sector. The sector widths or subdivisions of the R-ACM are generated according to a rule that aims at minimizing the estimation error within each subdivision. In this paper, we shall only deal with the one-dimensional case of ordinal numeric values.

Definition 1. *A One dimensional Rectangular ACM*: Let $\mathcal{V} = \{v_1, \ldots, v_m\}$ be the set of values of an attribute X in relation R. Let the value set $\mathcal{V}$ be subdivided into s number of sector widths according to the range partitioning rule described below. Then the Rectangular Attribute Cardinality Map histogram of attribute X is an integer array in which the j^{th} index maps the number of tuples in the j^{th} value range of the set $\mathcal{V}$ for all j, $1 < j \leq s$.

Rule 1. *Range Partitioning Rule*: Given a desired tolerance value τ for the R-ACM histogram, the sector widths, $l_1, \ldots, l_s$, of the R-ACM histogram should be chosen such that for any attribute value v_i, its frequency x_i does not differ from the running mean of the frequency of the sector by more than the tolerance value τ, where the running mean is the mean of the frequency values examined so far in the current sector.

The properties of the R-ACM are found in [5].

2.4 The Trapezoidal Attribute Cardinality Map

The Trapezoidal Attribute Cardinality Map (T-ACM) histogram of a given attribute is a one-dimensional integer array that stores the count of the tuples of a relation corresponding to that attribute, and for some equal subdivisions for the range of values assumed by that attribute. Unlike the Rectangular Attribute Cardinality Map (R-ACM) discussed above and the two major forms of histograms, namely Equi-width histogram [2] and the Equi-depth histogram [3], where the histogram buckets are rectangular cells, the sectors in the T-ACM are trapezoidal cells. In this paper we consider the one-dimensional T-ACM, described formally below.

Definition 2. *A One dimensional Trapezoidal ACM*: Let $\mathcal{V} = \{v_1, \ldots, v_m\}$, where $v_i < v_j$ when $i < j$, be the set of values of an attribute X in relation R. Let the value set $\mathcal{V}$ be subdivided into s Equi-width sectors, each having sector width l. We approximate each Equi-width sector by a trapezoid in which the j^{th} trapezoid is obtained by connecting the starting value, a_j, to the terminal value, b_j, where the quantities a_j and b_j satisfy:

(a) The starting value a_1 is a user-defined parameter.

(b) For all $j > 1$, the starting value of the j^{th} trapezoid, a_j, is the terminal value of the $(j-1)^{th}$ trapezoid, $b_{(j-1)}$.

(c) The area of the j^{th} trapezoid exactly equals the area of the j^{th} Equi-width sector from which the exact computation of the quantity, b_j, is possible.

Then the Trapezoidal Attribute Cardinality Map of attribute X with initial attribute value x_1 and width l is the set $\{(a_i, b_i) \mid 1 \leq i \leq s\}$.

The major properties of the T-ACM, and their proofs can be found in [5, 12].

3 Efficiency vs. Accuracy of Various Histogram Methods

It has been shown that the R-ACM and the T-ACM are more accurate than the Equi-width and the Equi-depth histograms, with regard to the approximation to the real underlying attribute-value histograms. In this section, we analyze the efficiency of these methods from another point of view. Consider the equi-join operation of several relations. We shall now quantify the quality of a method by measuring the number of times this method achieves the right decision, i.e. it chooses the optimal QEP.

For the analysis done below, we work with the model that the error function is doubly exponential [13].

Although we only consider the analysis for any two histogram schemes, referred to by M_1 and M_2 (which could be the R-ACM and the Equi-width schemes), the result we claim can be extended to the T-ACM and the Equi-depth as well.

Consider two query-size estimation methods, M_1 and M_2. The probability of choosing a cost value of a particular QEP by M_1 and that of choosing a cost value by M_2 are represented by two independent random variables.

Without loss of generality, if the mean cost of the optimal QEP is μ, by shifting the origin by μ, we can work with the assumption that the cost of the best QEP is 0, which is the mean of these two random variables. The cost of the second best QEP is given by another two random variables (one for M_1 and the other one for M_2) whose mean, $c > 0$, is the same for both variables. Then, the following result is true.

Theorem 1. Suppose that:

- M_1 and M_2 are two query result size estimation methods.
- X_1 and X_2 are two d-exp random variables that represent the cost values of the *optimal* QEP obtained by M_1 and M_2 respectively.
- X_1' and X_2' are another two random variables representing the cost value of a *non-optimal* QEP obtained by M_1 and M_2 respectively.
- $0 = E[X_1] = E[X_2] < E[X_1'] = E[X_2'] = c$.

Let p_1 and p_2 be the probabilities that M_1 and M_2 respectively make the wrong decision. Then, if

$$\mathrm{Var}[X_1] = \mathrm{Var}[X_1'] < \mathrm{Var}[X_2] = \mathrm{Var}[X_2'],\ p_1 < p_2 .$$

Proof. The proof of the theorem is quite involved and found in [13]. □

4 The Database Model

In this section, we describe the formal model that we have developed to test the efficiency of the methods (or for that matter, any query optimization strategy) discussed above. Since the estimation of the accuracy of these methods is done under *synthetic* data distributions, it requires a formal relational database model in which the relations are populated randomly according to some statistical distributions, and the queries also obey user-defined query distributions.

The database has n relations, called $R_1, R_2, \ldots, R_n$. Each relation has n attributes, called A_{ij}, such that A_{ij} is the j^{th} attribute of the relation R_i. It will be presently clear that the attribute A_{ij} symbolically represents the joining attribute for R_i and R_j, and the attribute A_{ii} represents the data inherent to relation R_i alone.

Among the various *join* operations available in relational database systems the model incorporates the equi-join. Clearly, this can be extended easily to other join operations.

Definition 3. The equi-join operation, denoted $R_1 \bowtie_{X=Y} R_2$, combines all the tuples in the relations R_1 and R_2 whenever the value of the attribute X from relation R_1 is equal to the value of the attribute Y from relation R_2.

We represent a string query as an ordered sequence of integers as a permutation of the integers $[1 \ldots n]$. Each number in the list represents the relation number.

Definition 4. A query is represented by the string $Q = i_1 i_2 \ldots i_n$, such that $1 \le i_j \le n$ and $1 \le i_k \le n$, for all $i_j,\ i_k$ satisfying $j, k = 1, \ldots, n$, $i_j \ne i_k$. The query join resulting from the string $i_1, i_2, \ldots, i_n$ is the following:

$$J = R_{i_1} \bowtie_{A_{i_1 i_2} = A_{i_2 i_1}} \ldots \bowtie_{A_{i_{n-1} i_n} = A_{i_n i_{n-1}}} R_{i_n} . \tag{1}$$

This string is called *query string*. For the sake of simplicity, the query join given in (1) is referred below to as $R_{i_1} \bowtie \ldots \bowtie R_{i_n}$ or simply as J.

The details of how queries can be specified and implemented are given in [13].

We have described how to perform a query evaluation plan given a string query. Note that the QEP is also represented by a string, and for n relations, the length of the join order string is $n-1$. Since all the indexes in the string are different, there are $(n-1)!$ different strings representing the $(n-1)!$ QEPs. All we need now is to consider how to perform the join for the $(n-1)!$ different join order strings, calculate the number of operations for each of them, and choose the string that leads to the minimum cost. This effectively models the query optimization problem completely. We use the estimation methods discussed above to *approximate* the size of the query.

5 Join Estimation

The aim of the query optimization problem is to minimize two crucial quantities: the *response time*, which means how quickly the query is executed, and the *resource consumption*, i.e. the computer resources (memory, CPU, etc.), used to execute the query. We first deal with some aspects in the calculation of the cost of a query.

The cost of a query, in terms of the response time, is given by the *query result sizes*, the *query result distributions*, the *disk access costs*, and other non-relevant factors. We consider only the number of read/write disk operations, since they are the dominant factors [5].

Since the read operations can be done in insignificant time, we calculate the cost of an equi-join query as the size of the resulting relation, $\mathcal{C} = |(R_1 \bowtie_X R_2)|$. Our problem now is one of calculating the cost of an equi-join that includes more than two relations. Using Algorithm **Join_Order_Selection**, we can calculate the total number of operations for performing the join $J = R_{i_1} \bowtie \ldots \bowtie R_{i_n}$ subject to the join order $S = j_1 j_2 \ldots j_{n-1}$.

To estimate the total number of operations needed to perform the join $J = R_{i_1} \bowtie \ldots \bowtie R_{i_n}$ subject to the join order $S = j_1 j_2 \ldots j_{n-1}$, we add all the costs of *all* the intermediate relations in a straightforward manner.

Since, at each step, two relations of J are merged into a new one, the relations in J can be of two types, referred to as primitive or derivative. In all the possible scenarios encountered we shall show that the result of Theorem 2 (stated below) can be recursively invoked so as to calculate the cost at every step and obtain the total cost.

Theorem 2. Consider the join $R_a \bowtie_{A_{jk}=A_{kj}} R_b$, in which R_a and R_b can be either primitive or derivative relations, where a is a string of integers whose *last* symbol is j, and b is a string of integers whose *first* symbol is k.

If A_{jk} and A_{kj} are independent of R_a and R_b respectively, the size of $R_a \bowtie R_b$, $|\widehat{R}_{ab}|$, is estimated as:

$$|\widehat{R}_{ab}| = \frac{|\widehat{R}_a||\widehat{R}_b|}{|R_j||R_k|} \sum_{i=1}^{m} \widehat{x}_i \widehat{y}_i \ , \qquad (2)$$

where $\widehat{x}_i$ and $\widehat{y}_i$ are the estimated frequencies for the i^{th} value of A_{jk} and A_{kj} respectively.

Proof. This result can be proven by considering the three mutually exclusive and collectively exhaustive cases of R_a and R_b being primitive or derivative. The details are found in [13]. $\square$

In our model (and in all database query systems worldwide), we have assumed independence between attributes and relations. The question of how attribute dependency can be considered is also discussed in [13].

6 Database Generation

In this section, we discuss the statistical distributions used in the generation of the databases. We have carried out two sets of experiments for the methods discussed above. In one set of experiments, the databases were generated using the uniform distribution modified under a multi-fractal decomposition. These databases were used to compare the R-ACM with the Equi-width and the Equi-depth methods. In the other set of experiments, we have tested the T-ACM with the Equi-width and the Equi-depth with distributions which are more suitable for linear modeling of attribute values.

6.1 Database Generation for R-ACM Evaluation

The databases used to test the accuracy of R-ACM are generated using the uniform distribution modified by the multi-fractal decomposition. In order to populate the database we used a probability list, P, for each attribute value, where if $P = \{p_1, \ldots, p_m\}$, it implies that v_i is randomly generated with probability p_i with $E(x_i) = Np_i$.

In order to generate a database, we first ensure that the distribution of the probability set is uniform. We do this by assigning the probability values by setting the value p_i to $\frac{1}{m}$, for $i = 1, \ldots, m$.

Hence the expected number of occurrences of a value in a table is given by $E(x_i) = \frac{N}{m}$.

The next step of the database generation consists of transforming the distribution of P to a non-uniform multi-fractal distribution. In order to achieve this, we used the multi-fractal decomposition described in [15].

In our model, we have further augmented the multi-fractal decomposition by a sliding index, k. The output list of the previous recursive scaling, $\{p_1, p_2 \ldots, p_m\}$, is transformed into a new normalized probability list, $\{p_k, p_{k+1}, \ldots, p_m, p_1, p_2, \ldots, p_{k-1}\}$, where k is a random integer such that $1 \leq k \leq m$.

6.2 Database Generation for T-ACM Evaluation

The databases used in the experiments for testing the T-ACM and comparing it with the traditional methods, used a particular probability distribution that we called the *noisy* TD, which is based an underlying exact TD distribution.

In the exact TD distribution, the probability set of a given attribute, P, is divided in k sub-sets, $P = \{P_1, \ldots, P_k\}$. Each sub-set has the same range values as the corresponding range in the Equi-width histogram. The probability sub-set P_1 is generated by choosing two random numbers, a and b; $p_j \in P_1$ is calculated as $p_j = (v_j - v_1)\frac{b-a}{|P_j|}$, where $|P_j|$ is the number of values of P_j. The next probability sub-sets are generated by setting a for the new range as the 'b' of the previous range, and choosing a new random number b for the subsequent range. Once all the probability sub-sets are generated, P is normalized.

The database is generated in such a way that all the pairs of attributes to be joined have the same distribution. In other words, the attribute A_{ij} has the same distribution as the attribute A_{ji}, for $i, j = 1, \ldots, n$, $i \neq j$.

The random values for each attribute for the noisy TD are generated using the exact TD distribution with some noise κ. The expected number of occurrences of v_i in the relation is $N(1 - o_i)p_i$, where o_i is a randomly generated real number such that $-\kappa \leq o_i \leq \kappa$, $\sum_{i=1}^{m} o_i = 0$, and N is the number of tuples in the relation.

One of the problems encountered in the implementation is the following. It can be seen that if the number of tuples in the relations is small, the histograms are not so good and the estimations are poor. However, if the number of tuples is large, the estimates are good, but the equi-joins can not be achieved due to the sizes of the joined relations. This makes the problem difficult to implement, simulate, and study.

7 Simulation Results

In this section, we discuss the empirical results obtained from some simulations with randomly generated databases. We have conducted two independent experiments, one of them is to compare the efficiency of R-ACM with the traditional methods, and the other one is related to the T-ACM.

In each experiment, we performed four independent runs. In each run, 100 random databases were generated. Each database was composed of six relations, each of them having six attributes. Each relation was populated with 100 tuples.

7.1 Simulation for R-ACM and Traditional Methods

In order to compare the efficiency of the R-ACM with that of the Equi-width and the Equi-depth, we used 50 values per attribute. We set the number of bins for the Equi-width to be 11 and the number of bins of the Equi-depth to be as close to 11 as possible. The number of bins for the R-ACM is the smallest integer, s, such that $s \geq 11$ and τ is an integer; s is automatically calculated by the prototype by performing a sequential search over all possible values of τ.

The simulation results obtained from 400 independent runs, used to compare the efficiency of the R-ACM and the Equi-width, are given in Table 1. As in the subsequent tables, the column labeled "R-ACM" is the number of times that R-ACM is better than Equi-depth. The column labeled "Equi-width" indicates the number of times that Equi-width is superior to R-ACM. As we can observe from the last row, the R-ACM chooses the superior solution more than *three* times as often as the Equi-width. Table 2 shows the results obtained after comparing the R-ACM with the Equi-depth. Again, we observe the superiority of the R-ACM over the Equi-depth – the R-ACM chooses a superior solution almost *four* times as often.

Simulation	R-ACM	Equi-width
1	41	10
2	26	10
3	37	6
4	33	16
Total	**137**	**42**

Table 1: Simulation results for R-ACM and Equi-width, after optimizing a query on 400 randomly generated databases.

Simulation	R-ACM	Equi-depth
1	50	14
2	44	7
3	38	10
4	44	15
Total	**176**	**46**

Table 2: Results of 400 runs for query optimization used to test the efficiency of R-ACM and Equi-depth.

7.2 Simulation for T-ACM and Traditional Methods

In this section we discuss the results obtained from the simulations performed to compare the efficiency of the T-ACM, the Equi-width, and the Equi-depth. In our experiments, we have used the noisy TD distribution to generate random databases with a noise level of 20%. Hence, the samples were generated with $\kappa = .2$, and $-.2 \leq o_i \leq .2$, for $i = 1, \ldots, m$. We set the number of attribute values to be 25, and the buckets to be 4.

The experimental results obtained after performing query optimization on 400 random databases are shown in Table 3. We observe the superiority of the T-ACM over the Equi-width – it chooses a superior solution more than *four* times as often as the Equi-width. In Table 4, we show the results of another set of runs used to compare the T-ACM and the Equi-depth. Again, the T-ACM chooses a superior solution more than *four* times as often as the Equi-depth. The power of T-ACM is obvious!

Simulation	T-ACM	Equi-width
1	26	3
2	25	7
3	24	8
4	18	5
Total	**93**	**23**

Table 3: Simulation results of 400 runs used to test the efficiency of T-ACM and Equi-width.

Simulation	T-ACM	Equi-depth
1	23	3
2	34	9
3	33	8
4	24	6
Total	**114**	**26**

Table 4: Experimental results obtained by performing query optimization on 400 randomly generated databases used to test the efficiency of T-ACM and Equi-depth.

7.3 User Interface

The user interface has been built using the state-of-art Java programming language in a Visual Development Environment. The main options allows the user to perform the following facilities: to initialize parameters, to generate a random database database, to create the various histograms, to estimate the queries, and to perform a simulation with many random databases. The details of this user interface are included in [13].

8 Conclusions

In this paper we have discussed the efficiency of the traditional histogram-based estimation methods for Database Query Optimization, namely the Equi-width and the Equi-depth, as well as the new methods proposed in [5, 6] and [5], the R-ACM and the T-ACM respectively. We have presented a prototype that was developed to benchmark these four histogram-based estimation methods.

We have shown analytically (using a reasonable model of accuracy, namely the doubly exponential distribution for errors) that as the accuracy of an estimation method increases, the probabilities of it leading to a superior QEP also increases. Our empirical results show that the R-ACM chooses superior QEPs *three* times more often than the Equi-width and the Equi-depth. This ratio is more than *four* times when comparing the T-ACM with the Equi-width and the Equi-depth.

The question of analyzing the accuracy/speed problem for other distributions (for example, Gaussian) remains open, but is far from trivial.

References

[1] S. Christodoulakis, "Estimating selectivities in data bases," in *Technical Report CSRG-136*, (Computer Science Dept, University of Toronto), 1981.

[2] R. P. Kooi, *The optimization of queries in relational databases.* PhD thesis, Case Western Reserve University, 1980.

[3] G. Piatetsky-Shapiro and C. Connell, "Accurate estimation of the number of tuples satisfying a condition," in *Proceedings of ACM-SIGMOD Conference*, pp. 256–276, 1984.

[4] M. Muralikrishna and D. J. Dewitt, "Equi-depth histograms for estimating selectivity factors for multi-dimensional queries," in *Proceedings of ACM-SIGMOD Conference*, pp. 28–36, 1988.

[5] M. Thiyagarajah, *Attribute Cardinality Maps: New Query Result Size Estimation Techniques for Database Systems.* PhD thesis, Carleton Univeristy, Canada, 1999.

[6] B. J. Oommen and M. Thiyagarajah, "The Rectangular Attribute Cardinality Map: A New Histogram-like Technique for Query Optimization," in *International Database Engineering and Applications Symposium, IDEAS'99*, (Montreal, Canada), August 1999.

[7] Y. Ioannidis and S. Christodoulakis, "On the propagation of errors in the size of join results," in *Proceedings of the ACM-SIGMOD Conference*, pp. 268–277, 1991.

[8] S. Christodoulakis, "Estimating block transfers and join sizes," in *ACM SIGMOD 83, Proceedings of the Annual Meeting*, (San Jose, CA), pp. 40–54, 1983.

[9] S. Christodoulakis, "Estimating record selectivities," in *Information Systems*, vol. 8, 1983.

[10] M. Mannino, P. Chu, and T. Sager, "Statistical profile estimation in database systems," in *ACM Computing Surveys*, vol. 20, pp. 192–221, 1988.

[11] P. Selinger, D. C. M.M. Astrahan, R. Lorie, and T. Price, "Access Path Selection in a Relational Database Management System," in *Proceedings of ACM-SIGMOD Conference*, 1979.

[12] B. J. Oommen and M. Thiyagarajah, "The Trapezoidal Attribute Cardinality Map: A New Histogram-like Technique for Query Optimization," Tech. Rep. TR-99-04, School of Computer Science, Carleton University, Ottawa, Canada, February 1999.

[13] B. J. Oommen and L. G. Rueda, "The Efficiency of Modern-day Histogram-like Techniques for Query Optimization," *Submitted for Publication.*

[14] T. B. Nguyen and B. J. Oommen, "Moment-Preserving Piecewise Linear Approximations of Signals and Images," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, 1997.

[15] C. Faloutsos, Y. Matias, and A. Silberschatz, "Modeling skewed distributions using multifractals and the 80-20 law," in *Technical Report*, (Dept. of Computer Science, University of Maryland), 1996.