Benchmarking Attribute Cardinality Maps for Database Systems Using the TPC-D Specifications*

B. John Oommen† and **Murali Thiyagarajah**‡
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
{oommen, murali}@scs.carleton.ca

Key Words: Query Optimization, Query Result Size Estimation

Abstract

Benchmarking is an important phase in developing any new software technique because it helps to validate the underlying theory in the specific problem domain. But benchmarking of new software strategies is a very complex problem, because it is difficult (if not impossible) to test, validate and verify the results of the various schemes in completely different settings. This is even more true in the case of database systems because the benchmarking also depends on the *types* of queries presented to the databases used in the benchmarking experiments. Query optimization strategies in relational database systems rely on approximately estimating the query result sizes to minimize the response time for user-queries. Among the many query result size estimation techniques, the histogram-based techniques are by far the most commonly used ones in modern-day database systems. These techniques estimate the query result sizes by approximating the underlying data distributions, and, thus, are prone to estimation errors. In two recent works [16, 17] we proposed (and thoroughly analyzed) two new forms of histogram-like techniques called the Rectangular and Trapezoidal Attribute Cardinality Maps (ACM) respectively, that give much smaller estimation errors than the traditional equi-width and equi-depth histograms currently being used by many commercial database systems.

*TPC-D benchmark specifications have been proposed by the Transaction Processing Performance Council as a decision support benchmark. It comprises of a set of highly complex business oriented queries that are run on a scalable simulated real-world business database. *Patent applications have been filed to protect the rights for the results presented here.*

†Senior Member, IEEE. Partially supported by the Natural Sciences and Engineering Research Council of Canada.

‡Supported by the Natural Sciences and Engineering Research Council of Canada. Currently employed with *The ORACLE Corporation.*

This paper reports how the benchmarking of the Rectangular-ACM (R-ACM) and the Trapezoidal-ACM (T-ACM) for query optimization can be achieved. By conducting an extensive set of experiments using the acclaimed TPC-D benchmark queries and database [25], we demonstrate that these new ACM schemes are much more accurate than the traditional histograms for query result size estimation. Apart from demonstrating the power of the ACMs, this paper also shows how the TPC-D benchmarking can be achieved using a large synthetic database with many different patterns of synthetic queries, which are representative of a real-world business environment.

1 Introduction

Benchmarking is an important step in the developmental cycle of any new software strategy as it validates the functional properties of the new software in a typical real-world environment. It also enables the user to compare the performance of various similar methods against a single standard. But benchmarking is often a challenging and a very complex problem, because it is difficult (if not impossible) to test, validate and verify the results of the various schemes in completely different settings. This is even more true in the case of database systems because the benchmarking also depends on the *types* of queries presented to the databases used in the benchmarking experiments. In this paper, we study the benchmarking of two new query optimization strategies that we recently proposed in [16, 17].

1.1 Query Optimization in Database Systems

Modern-day database management systems (DBMS) provide competitive advantage to businesses by allowing quick determination of answers to business questions. Intensifying competition in the business marketplace continues to increase the sizes of databases as well as sophistication of queries against them. This has resulted in a greater focus (both academically and in the industrial world) to develop systems with superior DBMS functionalities that would, in turn, minimize the response time for business and other queries.

The problem of minimizing query response time is known as Query Optimization, which has been one of the most active research topics in the database and information system fields for the last two decades. Query optimization for relational database systems is a combinatorial optimization problem, which requires estimation of query result sizes to select the most efficient access plan for a query, based on the estimated costs of various query plans. As queries become more complex (such as found in modern-day business systems), the number of alternative query evaluation plans (QEPs) which have to be considered explodes exponentially. For example, for a join of 10 relations, the number of different QEPs is greater than 176 billion!. A typical inquiry that a bank officer runs many times a day, for example, to retrieve the current market value of a customer's mutual fund/stock portfolio, usually transparently carries out a join of many relations (i.e: behind the scene).

Query result sizes are usually estimated using a variety of statistics that are maintained

in the database catalogue for relations in the database. Since these statistics approximate the distribution of data values in the attributes of the relations, they represent an inaccurate picture of the **actual** contents of the database. It has been shown in [7] that errors in query result size estimates may increase exponentially with the number of joins. This result, in light of the complexity of present-day queries, shows the the critical importance of accurate result size estimation.

Several techniques have been proposed in the literature to estimate query result sizes, including histograms, sampling, and parametric techniques [3, 5, 10, 11, 13, 18, 21]. Of these, histograms are the most commonly used form of statistics, which are also used in commercial database systems[1] such as Microsoft SQL Server, Sybase, Ingres, Oracle and DB2.

We recently [16, 17] proposed two new catalogue based non-parametric statistical models called the *Rectangular Attribute Cardinality Map* (R-ACM) and *Trapezoidal Attribute Cardinality Map* (T-ACM), that can be used to obtain more accurate estimation results than the currently known estimation techniques. Since they are based on the philosophies of numerical integration, query result size estimations based on these models have been shown to be much more accurate than the traditional equi-width and equi-depth histograms. For a detailed mathematical treatment of these techniques, including their analytic properties, the reader is referred to [16, 17]. The goal of this paper is to demonstrate the power of the R-ACM and the T-ACM for query result size estimation by extensive experiments using the TPC-D benchmark queries and databases [25], and to justify the claim of our analytical results that the R-ACM and the T-ACM are superior to the current state-of-the-art techniques *that are being used in the commercial database systems.*

1.2 Aim of the Paper

The objective of the TPC-D benchmarking is to provide to industry users relevant objective performance data about any new system. The TPC-D benchmarking system is unique and different from other prototype validation environments because it has two distinct components, which are modeled based on their real-world counter-parts. First of all, it provides the "testing platform" with a simulated real-world database which is both scalable and yet easily modeled. Secondly, it provides the platform with an easy and convenient query generator, from which simulated real-world queries of "arbitrary" complexity can be generated and tested against the database. Thus, the platform works with a set of real-world like business queries designed to run on a carefully selected complex business database.

Due to its closeness to the real-world model, and its industry-wide acceptance as a standard database system benchmark, TPC-D becomes an ideal test-bed to validate our analytical results of the R-ACM and the T-ACM. In addition, unlike the stand-alone elementary relational operations that were used in [22, 23], the queries used in the TPC-D

[1] It should be noted that all the commercial systems utilize the equi-width or the equi-depth histogram methods in one form or the other. A Table listing the various commercial databases systems and the methods used is included in a later section.

benchmarking are highly complex. The queries are also designed in such a way so that they examine a large percentage of available data, but are implemented with sufficient degree of ease. These query patterns, which are catalogued in Table 4, consist of a wide range of relational operations and represent the type of queries that are usually posed by DBMS users in a business environment. This enables us to rigorously test the ACMs with various types of queries and confirm their superiority over the traditional histogram based estimation techniques. More over, the `DBGEN` program supplied by the TPC-D allows us to generate scalable databases reflective of a true business enterprise. This again permits us to conduct a set of robust experiments so that an industry acceptable fair comparison of our new structures (or for that matter, any new strategy) can be presented against the traditional histograms currently in use. Since the TPC-D queries and databases can be accurately generated according to the TPC-D specification document [25], the results presented in this paper can be easily duplicated and verified by interested researchers.

1.3 Benchmarking Versus Prototype Validation and Testing

Any new system or strategy that is proposed has to be validated and thoroughly tested before it can be effectively used in the industry. This, of course, involves three distinct phases: the validation of the prototype, the testing of the prototype, and more importantly, the benchmarking of the strategy against industry-acclaimed benchmark standards. Although these issues are significant for any software solution, this is particularly important when a new database solution (for example, a query processor) is in question. This, indeed, is because it not only involves the database itself. It also involves the queries that are put against the database. Consequently, although it is possible to test a new strategy against a "real-world" database, the question of testing against families of query patterns is far from trivial, since the query patterns are usually obtained from the users themselves.

In this regard, we would like to mention that the prototype validation and testing phases of the two schemes which we proposed (the R-ACM and the T-ACM) were earlier studied for the real-world databases, namely the U.S. CENSUS and NBA player statistics databases. However, the issue of testing the schemes against families of query patterns for a scalable database remained open. *This is exactly the contribution of this paper.*

To achieve this fundamental goal of this paper, we were posed with two issues. First of all we have to develop a database which was scalable and which represented a real-life scenario. Secondly, and more importantly, we were to be able to generate "real-life" queries without actually relying on a human interface. It is here that the TPC-D benchmarking platform has provided us with invaluable support.

As mentioned earlier, our previous experimental works related to the R-ACM and the T-ACM involved *prototype validation and testing* using two real-world databases, namely, the U.S. CENSUS database and the NBA player statistics database [22, 23]. Even though these databases were representative of the real-world data distributions, the scope of their use was rather limited to the specific attribute value domains (or data environments) under consideration. More over, the query types used for these specific databases were relatively

primitive and included only one of the elementary relational operations (such as equi-join, equi-select and range-select), at a time, in a given query. Consequently, the query types used in these previous experiments conducted in [22, 23] cannot be argued to represent any typical real-world user queries, and thus, we do not feel that it is fair for us to extrapolate our results of [22, 23] to the real-world query types.

In contrast to the above, the TPC-D benchmarking experiments presented in this paper consist of two distinct components. First of all, the databases used for the experiments are scalable simulated real-world data that are typically found in a business enterprise. Secondly, the query types used are representative of a large collection of business oriented queries with broad industry-wide relevance. The query types, as can be seen from Table 4, are usually complex and involve multiple relational operations. Hence we expect the experimental results with these TPC-D benchmark database/query types to closely represent the behavior of the ACMs in a real-world database environment. More detailed discussion on the TPC-D queries and databases are found in Section 4.

Since this paper involves the benchmarking of the R-ACM and the T-ACM, we briefly describe them in Section 3.

2 Previous Work in Data Representation

In the interest of brevity, it is impractical to give a good review of the field here. Such a detailed and comprehensive review is found in [24]. However, to present our results in the right perspective a brief survey is given. Rather than speak of our work in a competitive spirit, we shall compare it from the viewpoint of how it fits against the texture and lattice of the currently reported methods.

From a broad perspective, it is pertinent to mention that all histogram techniques essentially approximate the density function of the attribute that is being mapped. Thus, from a synthesis point of view, any efficient function approximation strategy can be considered as a starting point for a reasonably good histogram method.

The mathematical foundation for the Attribute Cardinality Maps (ACM) which we introduce 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, in turn, are related to its transform domain parameters. If this method is applied directly to yield a time/spatial domain approximation, it ensures that the approximated function is close to the actual function in both the time (space) domain and the transform domain in a well defined way. This solution, which reportedly is the only solution guaranteeing this dual property, in *both the time and frequency domains* has been used in image processing and pattern recognition. In the ACM's reported here, it is applied to query optimization. Indeed, the R-ACM can be viewed as a specific perspective of this method since it preserves (or minimizes the variation from) the *first* geometric moment of the actual histogram. It thus ensures that the variation of actual histograms (within any

specific bucket) is minimized by an accurate approximation in both the time and transform domains.

Equi-width histograms for single-attributes were considered by Christodoulakis [2] and Kooi [10]. Since these histograms traditionally have the same width, they produce highly erroneous estimates if the attribute values are not uniformly distributed. The problem of building equi-depth histograms on a single attribute was first proposed by Piatetsky-Shapiro and Connell [18]. This was later extended as multi-dimensional equi-depth histograms to represent multiple attribute values by Muralikrishna and Dewitt [13].

Ioannidis and Christodoulakis took a different approach by grouping attribute values based on their frequencies [7, 8]. In these serial histograms, the frequencies of attribute values associated with each bucket are either all greater or all less than the frequencies of the attribute values associated with any other bucket. They also considered optimal serial histograms that minimize the worst case error propagation in the size of join results [7, 8]. The serial histograms provide optimal results for equality join estimations, but less than optimal results for range queries. Faloutsos *et. al.* [5] proposed using a multi-fractal assumption for real-data distribution as opposed to the uniformity assumptions made within current histograms.

2.1 Comparative Open Work

Apart from the references mentioned above, a few significant references (which also objectively survey the field) and which present relatively new contributions deserve prominent mention. The first in this list is the paper due to Poosala *et. al.* [20], which specifically addresses the issue of designing histograms suitable for range predicates. In particular, the histogram design using the equi-sum, V-optimal, and spline-based methods are discussed. Of these, V-optimal-based histogram methods are of particular interest, because they group contiguous sets of frequencies into buckets so as to minimize the variance of the overall frequency approximation. Philosophically speaking, this resembles the principle of the R-ACM described presently, which attempts to minimize the variation around the mean in any given bucket, and consequently has the property of minimizing the variance inside the bucket. While, in the R-ACM, this is done along the attribute values, all the tuples within a bucket have almost identical *frequency* values. Thus, the R-ACM can be perceived to be analogous to clustering the attribute values by their frequencies, and yet maintaining the sequential nature of the attribute values. The similarities and differences between the actual histograms generated by these two families of methods are still being investigated. The paper [20] catalogues different families of V-optimal histograms based on the sort parameters and the source parameters, and contains a complete taxonomy of all resulting histograms.

Poosala and Ionnadis continued this work in 1997 [19], where the assumption of independence of the individual attributes within a relation was relaxed. While this work was directed towards certain specific histograms (and in particular, some of the histograms mentioned above), we believe that the exact same techniques can be used to develop multi-

dimensional ACMs. Indeed, we believe that the Hilbert numbering concept and the single value decomposition specified in [19] can be applied to the new histograms that we have introduced, for higher dimensions. This is also currently being investigated.

Jagadish *et. al.* have studied the problem of yielding optimal histograms which guarantee certain quality specifications [9]. These specifications can be either space-bounded or error-bounded. A formal scheme to design space-bounded V-optimal histograms (and a corresponding approximation algorithm for the same) have also been presented in [9]. Identical developments for the error bounded V-optimal histograms have also been included. As a matter of comparison with our work, we believe that these identical philosophies can be used to obtain space bounded and error bounded histograms for the R-ACM and the T-ACM. Because the R-ACM aims to minimize the variation around the first moment of the moment-based approximation, it is our conjecture that it is, in principle, an error bounded quality approximation. The space bounded approximation for the R-ACM is *probably* the approximation obtained by controlling and optimizing the value of "τ". All these issues lead to numerous open problems.

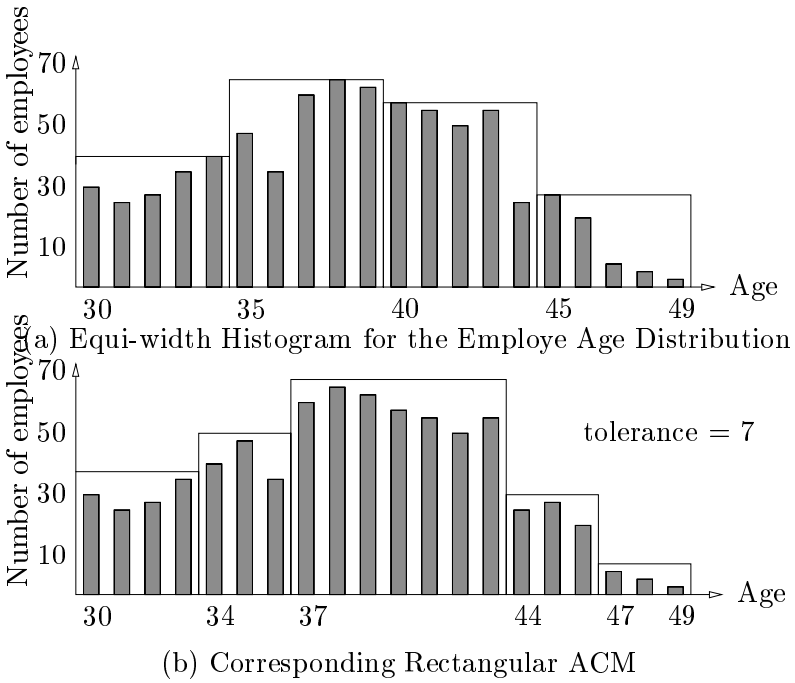
A completely new strategy for approximating density functions in database query optimization involves wavelet-based approximations. Such approximations have been used in signal processing, image processing, and, in general, in function approximation for almost two decades [1, 6, 12]. The application of these techniques to databases naturally leads to the approximation of the density function using the m-most significant wavelet coefficients. Informally speaking, due to the results in [14], we believe that these wavelet-based histograms are closely related to the R-ACM and the T-ACM because, the latter also optimize the linear approximations by maintaining the most significant moment coefficients, as opposed to wavelet coefficients. The issue of comparing our ACM technologies with such approximations is still open, and warrants investigation.

From the above discussion, it is clear that a lot of work has yet to be done to compare the ACMs with other histogram methods reported in the academic and commercial literature. A benchmarking exercise which compares the various schemes in terms of accuracy and query-evaluation plan efficiency, has to be undertaken before a conclusive ranking of the schemes can be specific. Such a tedious study for the TPC-family of benchmarks is currently underway. However, we believe the results presented here are extremely valuable, because they demonstrate how the ACMs compare with the two histogram methods that are used in *all the commercial database products.*

3 Attribute Cardinality Maps[2]

The Attribute Cardinality Maps (ACM) are histogram-like techniques for query optimization [16, 17, 24]. Since they are based on the philosophies of numerical integration, query

[2]This section is included here in the interest of completeness and for the sake of ease of readability. The details of the mathematical analysis of the properties of both the R-ACM and T-ACM structures, and the details of the schemes for implementing them can be found in [16, 17, 24].



(a) Equi-width Histogram for the Employe Age Distribution

(b) Corresponding Rectangular ACM

Figure 1: An Example Rectangular Attribute Cardinality Map

result size estimations based on these models have been analytically shown to be much more accurate than the traditional equi-width and equi-depth histograms. There are two types of ACMs, namely, the Rectangular ACM (R-ACM) and the Trapezoidal ACM (T-ACM). Since this paper is about the benchmarking of the R-ACM and the T-ACM, we briefly describe these structures below. Observe that since this paper is not intended to be of a theoretical flavor, we shall merely allude to the theoretical properties of the ACMs.

3.2 Rectangular Attribute Cardinality Map

The Rectangular Attribute Cardinality Map (R-ACM) of a given attribute, in its simplest form, is a one-dimensional integer array that 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. 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, where all the sector widths are equal, and the equi-depth histogram, where 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.

Definition 1 A One dimensional Rectangular ACM: *Let* $\mathcal{V} = \{v_i : 1 \leq i \leq |\mathcal{V}|\}$, *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 number of sector widths according to the range partitioning rule described below. Then the Rectangular Attribute Cardinality Map 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, the sector widths, $l_j, 1 \leq j \leq s$, of the R-ACM should be chosen such that for any attribute value X_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.*

For example, consider the frequency set $\{8, 6, 9, 7, 19, 21, 40\}$ corresponding to the attribute values $\{X_0, X_1, X_2, X_3, X_4, X_5, X_6\}$ of an attribute X. Using a tolerance value $\tau = 2$, the attribute value range will be partitioned into the three sectors, $\{8, 6, 9, 7\}, \{19, 21\}, \{40\}$ with sector widths of 4, 2, and 1 respectively. The formal algorithm for generating the R-ACM can be found in [16].

Since the ACM only stores the count of the tuples and not the actual data, it does not incur the usually high I/O cost of having to access the base relations from secondary storages. Secondly, unlike the histogram-based or other parametric and probabilistic counting estimation methods in use currently [11], ACM does not use sampling techniques to approximate the data distribution. Each cell of the ACM maintains the *actual* number of tuples that fall between the boundary values of that cell, and thus, although this leads to an approximation of the density function, there is no approximation of the number of tuples in the data distribution.

The one-dimensional R-ACM as defined above can be easily extended to a multi-dimensional one to map an entire multi-attribute relation. The multi-dimensional ACM, which can be used to store the multi-dimensional attributes that commonly occur in geographical, image, and design databases, is currently being investigated.

3.2.1 Properties of the R-ACM

In order to provide the right perspective for our experimental results, we give below a brief catalogue of the major properties of the R-ACM.

(1) The R-ACM is neither an equi-width nor an equi-depth histogram. This is due to the partitioning strategy based on the tolerance.

(2) The frequency distribution of any attribute value within an R-ACM sector obeys a Binomial distribution with mean $\mu = \frac{n}{l}$ and variance $V = \frac{n(l-1)}{l^2}$, where n is the number of tuples within the sector and l is the sector width.

(3) For a one-dimensional R-ACM, the maximum likelihood estimate of the number of tuples for a given value X_i of attribute X is given by,

$$\hat{x}_{ML} = \frac{n}{l}$$

where n is the number of tuples in the sector containing the value X_i and l is the width of that sector.

(4) For a one-dimensional R-ACM, the maximum likelihood estimate of the number of tuples for a given value X_i of attribute X falls within the range of,

$$\frac{(n+1)}{l} - 1 \leq \hat{x}_{ML} \leq \frac{(n+1)}{l},$$

where n is the number of tuples in the sector containing the value X_i and l is the width of that sector.

(5) The error in self-join estimation from the R-ACM is given by,

$$\epsilon = Var(ACM) + \sum_{j=1}^{s} \left\{ \sum_{k=1}^{l_j} x_k^2 - \frac{n_j^2 + n_j l_j - n_j}{l_j} \right\}$$

which is $O(Var(ACM))$, where self-join is the operation of joining the relation with itself and the $Var(ACM)$ is the variance of the entire R-ACM.

(6) The variance of the entire R-ACM is given by,

$$Var(ACM) = N - \sum_{j=1}^{s} \frac{n_j}{l_j},$$

where N is the total number of tuples mapped by the R-ACM, n_j is the number of tuples in the j^{th} sector, l_j is the j^{th} sector width, and s is the number of sectors in the R-ACM.

(7) If the attribute value X_i falls in the j^{th} sector of an R-ACM, then the number of occurrences of X_i is,

$$\frac{n_j}{l_j} - \left| \tau \left[\ln\left(\frac{l}{i-1}\right) - 1 \right] \right| \leq x_i \leq \frac{n_j}{l_j} + \left| \tau \left[\ln\left(\frac{l}{i-1}\right) - 1 \right] \right|$$

where n_j and l_j are the number of tuples and the sector width of the j^{th} sector and i is the location of the attribute value within the sector.

For example, consider an R-ACM sector of width 10 containing 124 tuples. Using a tolerance value $\tau = 3$, we see that the attribute value X_3 falls in the following range:

$$10.57 \leq x_3 \leq 14.23.$$

The power of the R-ACM is obvious when we observe that the corresponding range for the equi-width and the equi-depth histograms are,

$$0 \leq x_3 \leq 124.$$

(8) The average-case error in estimating the frequency of an attribute value is always bounded by 2τ.

The proofs of the above assertions can be found in [16, 24]. Also found in [16, 24] are various expressions for the join estimation error, average-case and worst-case errors for both equality-select and range-select operations using the R-ACM. The issue of how τ is chosen is an interesting problem in its own right, and it involves the issue of *optimizing the R-ACM.* This is discussed in depth in [24], and is currently being compiled for publication.

3.3 Trapezoidal Attribute Cardinality Map

A trapezoidal ACM is a modified form of the equi-width histogram where each histogram partition is a trapezoid instead of a rectangle. In fact, the trapezoidal ACM is obtained by replacing each of the rectangular sectors of the equi-width histogram by a trapezoid. The beginning and ending frequency values of each trapezoid sector is chosen so that the area of the resulting trapezoid will be equal to the area of the "rectangle" of the histogram it is replacing.

Definition 2 A One dimensional Trapezoidal ACM: *Let* $\mathcal{V} = \{v_i : 1 \leq i \leq |\mathcal{V}|\}$, *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:*

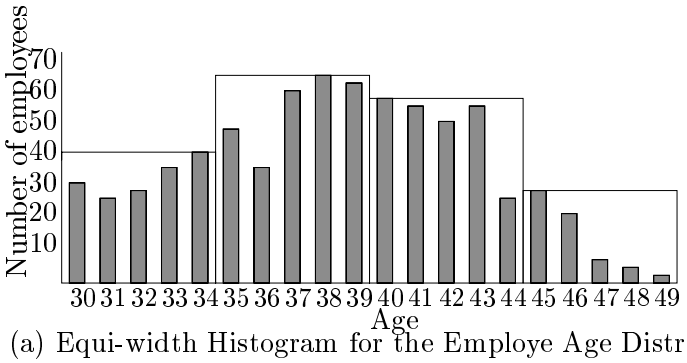
(a) The starting value a_1 *is a user-defined parameter.*

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

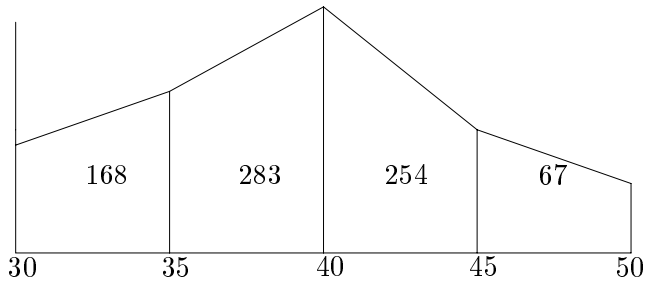
(c) The area of the j^{th} *trapezoid exactly equals the area of the* j^{th} *equi-width sector from which the exact computation of the quantity,* b_j, *is possible.*

Then the Trapezoidal Attribute Cardinality Map of attribute X *with initial attribute value* X_1 *and width* l *is the set* $\{(a_i, b_i) | 1 \leq i \leq s\}$.

Our motivation for proposing the trapezoidal ACM for query result size estimation originates from considering the various techniques used in numerical integration. Finding the result size of a selection query on a range-predicate can be considered as a discrete case of finding the area under a curve. Thus any numerical integration technique used to find



(a) Equi-width Histogram for the Employe Age Distribution



(b) Corresponding Trapezoidal ACM

Figure 2: An Example Trapezoidal Attribute Cardinality Map

the area under a curve will fit our purpose well. Though more accurate and sophisticated methods such as Simpson's Rule exist, since the trapezoidal method is relatively easy to use in a DBMS setting and is much superior to the traditional equi-width and equi-depth histograms currently in use, we have opted to use the trapezoidal method. In addition to providing more accurate result estimation on selection queries on range predicates, it also gives better results on equality-match predicates. The formal algorithm for generating the T-ACM can be found in [17].

3.3.1 Properties of the T-ACM

In order to provide the right perspective for our experimental results, we give below a brief catalogue of the major properties of the T-ACM.

(1) The T-ACM is based on the trapezoidal method of numerical integration. Thus it is more accurate than the right-end or left-end histogram estimation techniques.

(2) The probability, p_i, of the i^{th} attribute value in the j^{th} T-ACM sector is given by $p_i = \frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)}.i$, where a_j is the frequency of the first attribute value in the j^{th} sector, n_j is the number of tuples in the j^{th} sector and l is the sector-width.

(3) The frequency distribution of any attribute value, X_i, within an T-ACM sector obeys a Binomial distribution with mean $\mu = n_j p_i$ and variance $V = n_j p_i(1 - p_i)$ where n_j

is the number of tuples within the j^{th} sector, l is the sector width, and p_i is given as in (2) above.

(4) The error in self-join estimation from the T-ACM is given by,

$$\epsilon = \sum_{j=1}^{s} \left(\sum_{k=1}^{l} x_k^2 - \frac{a_j(l+1)(a_j l - 2n_j)}{3(l-1)} - \frac{2n_j^2(2l-1)}{3l(l-1)} \right)$$

where self-join is the operation of joining the relation with itself, s is the number of sectors in the T-ACM, x_k is the number of occurrence of the k^{th} attribute value in the sector, and a_j, n_j, and l are stated as above in (2).

(5) The worst-case error, ϵ, in estimating the result size of an equi-select operation, $\sigma_{X=X_i}(R)$, using a trapezoidal ACM is given by,

$$\epsilon = \begin{cases} a_j + \frac{2(n_j - a_j l)}{l(l-1)} i & \text{if } i < \frac{l(l-1)(n_j - 2a_j)}{4(n_j - a_j l)}, \\ n_j - a_j - \frac{2(n_j - a_j l)}{l(l-1)} i & \text{if } i \geq \frac{l(l-1)(n_j - 2a_j)}{4(n_j - a_j l)}, \end{cases}$$

where i is the location of the attribute value X_i within the sector and a_j, n_j and l are stated as above in (2).

(6) Assuming that the T-ACM sector has been obtained by processing a histogram bucket of size l with n_j tuples, the average error in estimating the frequency of an arbitrary attribute value X_i, obtained by averaging over *all* attribute values in this sector of the T-ACM is exactly zero.

(7) The upper bound of the average-case error, ϵ, in estimating the frequency of an arbitrary attribute value X_i in a T-ACM is,

$$\epsilon = a_j + \frac{2(n_j - a_j l)}{l(l-1)} . i - \frac{n_j}{l}.$$

The proofs of the above assertions are found in [17].

3.2.2 Generating T-ACMs for the Experiments[3]

Since the T-ACM is based on the trapezoidal rule of numerical integration, we expect it to be more accurate than the corresponding equi-width histogram of the same sector width. We shall now describe a process by which we can obtain a T-ACM that is much more accurate than the T-ACM generated by the algorithm given in [17].

[3] As mentioned earlier, in the interest of not being repetitive, we include here only the algorithm. This is done so that other interested researchers can independently duplicate our results.

Let us assume that T_A is the T-ACM derived from the equi-width histogram, H_A. Also assume that the frequencies of the starting attribute value of *every second* histogram sector are available. Observe that this can be computed in $O(s)$ time (as opposed to $O(L)$), where s is the number of sectors. Then we can generate a T-ACM which has a sector width that is *half* of the sector width of H_A and is much more superior than the initial T-ACM, T_A. The strategy to achieve this is given in the algorithm, `Implement_T-ACM`, below.

Since the trapezoidal rule of numerical integration is more accurate than the left-end or right-end rectangular rule of numerical integration, it is obvious that by virtue of the construction of the T-ACM, T_A is more accurate than the equi-width histogram H_B. We note that the area of a sector in T_A may not be exactly equal to the actual number of tuples falling in that sector. Hence, using the actual number of tuples within the sectors we can partition the sectors to obtain the T-ACM, T_B, where the sector areas represent the *actual* number of tuples more accurately. Using the same arguments, we obtain the T-ACM, T_C, from T_B. The T-ACM, T_C, can be expected to be more accurate than the T-ACM, T_B, as its boundary frequencies are more accurate estimates based on the *actual number* of tuples falling within the sectors.

Algorithm 1 Implement_T-ACM

```
Input:  (i) Equi-width histogram H_A with sector width l.
        (ii) Starting frequencies of every 2^nd sector.
Output:  T-ACM with sector width l/2.
begin
   Merge every two adjacent sectors of H_A to get H_B;
   Generate T-ACM, T_A from H_B.
   Estimate frequencies of T_A's middle attribute values.
   Generate T_B from T_A using frequencies obtained from the last step.
   Estimate frequencies of T_B's middle attribute values.
   Generate T_C from T_B using frequencies obtained from the last step.
end;
```

EndAlgorithm Implement_T-ACM

Thus, by invoking a small preprocessing step, we have generated a T-ACM that is much more accurate than the original T-ACM derived directly from the corresponding histogram. An example highlighting the steps taken by the above algorithm is shown in Figure 3. In this example, the input to the algorithm is an equi-width histogram with two sectors as shown in Figure 3(a). The number of tuples in the sectors are $n = 24$ and $n = 36$ respectively. Also the starting frequency value of the first sector is 9 and the terminal frequency of the second sector is 15. The first step of the algorithm merges these two sectors to create a single histogram sector shown in Figure 3(b). The next step generates a trapezoidal sector equivalent to this larger histogram sector. Since the trapezoidal sector is only an approximation of the number of tuples represented by the rectangular sector, its area may not reflect the actual number of tuples ($n = 60$) falling within that sector. Hence in the

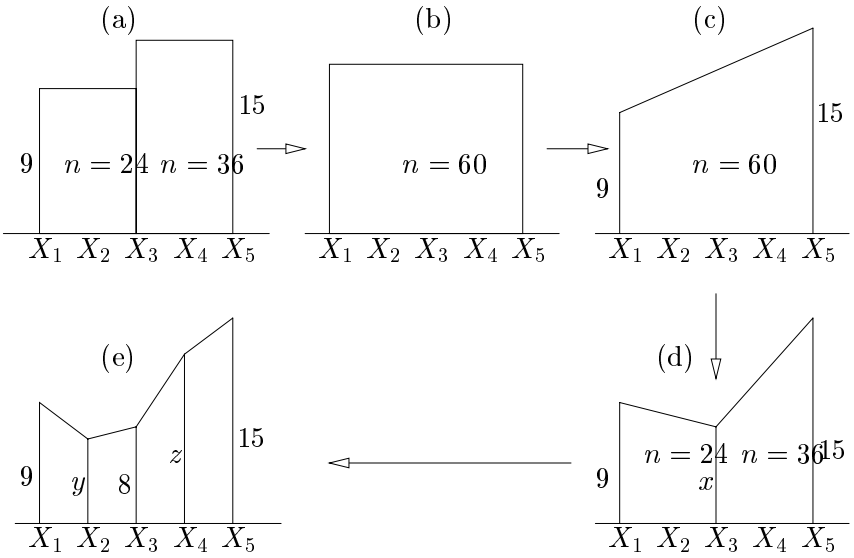


Figure 3: Construction of a T-ACM

next step of the algorithm, we *estimate* the middle frequency (i.e: $x_3 = 8$) of this sector by using the total number of tuples within the sector. The resulting T-ACM sectors are shown in Figure 3(d). Since the actual number of tuples contained in these two T-ACM sectors are already known (they are $n = 24$ and $n = 36$), the next step of the algorithm using this information, estimates the middle frequencies of these two sectors. The result is the T-ACM shown in Figure 3(e).

Before we conclude, we emphasize that the technique specified above for generating a T-ACM is more heuristic in nature. The problem has also been formally approached. Indeed, the issue of how the slope of a T-ACM bucket is determined is an interesting problem in its own right. In [24], this problem has been reduced to an optimization problem based on two error norms : the mean square norm and the absolute error norm. In each case, the properties of the optimal T-ACM have been derived. This topic is discussed in depth in [24], and is currently being prepared for publication.

3.4 Comparison of the ACMs to the Current State-of-the-Art

Since most of the current commercial DBMSs utilize non-parametric statistical techniques for query result size estimations, we have listed these techniques for some of the popular commercial DBMSs in Table 1. As can be seen from this table, most vendors use equi-depth histograms as it is more accurate than the equi-width histograms for both worst-case and average-case situations.

Having described the properties of the R-ACM and the T-ACM in the previous sections, we shall now provide a comparison of the worst-case and average-case errors of these new

Vendor	Product	Histogram Type
Oracle	Oracle	Equidepth
Sybase	System 11.xx	Equidepth
IBM	DB2-MVS	Equiwidth, Subclass of End-Biased(F,F)
Tandem	NonStop SQL/MP	Equidepth
Informix	Online Data Server	Equidepth
NCR	Teradata	Equidepth

Table 1: Histograms used in Some Popular Commercial DBMSs.

techniques to that of the traditional histograms. This is summarized in Table 2. It can be seen from this table that both the R-ACM and the T-ACM are much more accurate for query result size estimation than the traditional equi-width and equi-depth histograms.

Histogram	Worst-case Error	Average-case Error
Equi-width	$\max\left(n_j - \frac{n_j}{l}, \frac{n_j}{l}\right)$	$\max\left(n_j - \frac{n_j}{l}, \frac{n_j}{l}\right)$
Equi-depth	$\frac{2}{3l_j}$	$\frac{1}{2l_j}$
R-ACM	$\tau \left\lvert \ln\left(\frac{l_j}{i-1}\right) - 1 \right\rvert$	2τ
T-ACM	$\max\left(a_j + \frac{2(n_j - a_j l)}{l(l-1)} i, n_j - a_j - \frac{2(n_j - a_j l)}{l(l-1)} i\right)$	0

Table 2: Comparison of Histogram and ACM Errors. Note that τ is much smaller than l_j.

Note that the quantities a_j, n_j, i and l denote the frequency of the first attribute value in a sector, total number of tuples in the sector, location of the attribute value X_i within the sector and the sector width respectively.

4 TPC-D Benchmark Specification

The TPC Benchmark D (TPC-D) has been proposed by the Transaction Processing Performance Council (TPC) as a decision support benchmark. TPC-D models a decision support environment in which complex *ad hoc* business-oriented queries are submitted against a large database. The queries and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. We briefly describe the TPC-D benchmark queries and its database model below. A detail description about the complete benchmarking requirements, including auditing can be found

in the TPC-D specification document [25].

4.1 TPC-D Benchmark Queries

The TPC-D benchmark queries typically involve, multi-table joins, extensive sorting, grouping and aggregation and sequential scans. The purpose of TPC-D is to assess cost versus performance of a particular commercial system which supports the above queries. Since our objective is to study the performance of the ACMs in terms of their estimation accuracy, unlike a commercial DBMS, we do not require a fully functional DBMS for our experiments. Most of the TPC-D queries contain relational operators other than selection and equality join, and thus are not directly suitable for running on ACMs. Consequently, for our testing purposes, we shall use a slightly modified form of TPC-D queries by systematically eliminating the inapplicable operations such as grouping and aggregations etc., which are not supported by our current research work. Eliminating queries containing *views*, and modifying the remaining queries, we obtained a total of 11 query types from the original 17 query types specified in the TPC-D specification. These query types are given in Table 4.

For each of the relevant query types listed in Table 4, we construct an operator tree and estimate the result size of these operator trees in a bottom-up fashion. This bottom-up query execution strategy works as follows. First, we estimate the result sizes and distributions of leaf-level operators using the ACMs on the base relations. Then we construct[4] the ACMs on the result distributions and use these ACMs for estimating the result size and distributions of the operators in the next level of the tree. This process is repeated until the result size of the root node of the operator tree is estimated. We also compute the exact result size of the query by actually executing the simplified queries on the same data using a simple query processor which we implemented. Using these we compute the error from the result size estimation using the ACMs. For example consider the query `Q9` from Table 4:

```
select *
from supplier, partsupp, lineitem, order, nation
where partsupps.suppkey == lineitem.suppkey and
   order.orderkey == lineitem.orderkey and
   supplier.suppkey == lineitem.suppkey and
   supplier.nationkey == nation.nationkey
```

Observe that the query itself involves joining the four relations `supplier, partsupp, lineitem, order` and `nation` based on the equality predicates:

(i) `partsupp.suppkey = lineitem.suppkey`,

[4]The assumption we use is one of independence, in which the joint distribution at the higher level is taken as the product of the marginals at the lower level. The question of how this is accomplished in a real query optimizer is discussed in detail elsewhere [15]. [15] also catalogues the actual query processing advantages (as opposed to histogram accuracy advantages) of both the R-ACM and the T-ACM.

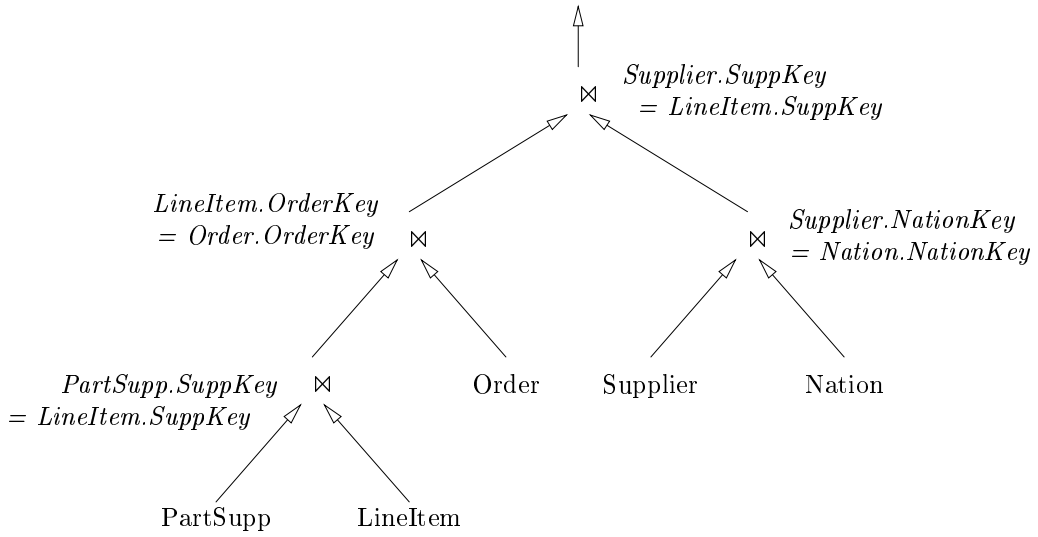


Figure 4: Operator Tree for Query `Q9`: Product Type Profit Measure Query

(ii) `order.orderkey = lineitem.orderkey`,

(iii) `supplier.suppkey = lineitem.suppkey` and

(iv) `supplier.nationkey = nation.nationkey`.

Each of them, in turn, is joined by an `and` clause which is finally used to select all the tuples from the resulting relation. Note that this join is actually a multi-table equi-join with a multi-clause predicate. An operator tree corresponding to the query, `Q9`, is given in Figure 4.

All of the TPC-D benchmark queries involve business oriented questions using standard SQL syntax, conforming to SQL-92 specification. Table 3 describes each query in plain English.

4.2 TPC-D Benchmark Database

The TPC-D database schema consists of eight tables detailed in the TPC-D specification. This schema models a database for a worldwide distributor that purchases parts from suppliers and sells them to customers. The two largest tables are the master-detail pair of `Order` and `LineItem`, which together constitute about 85% of the database. The other tables describe the business's parts, suppliers and customers. All tables except the two small `Nation` and `Region` tables scale linearly with the size of the database. The TPC-D database can be scaled upto a size of 1000GB in order to support high-end systems. For our experiments, we use a prototype version of this, namely, a test database with a size of approximately 100MB. The TPC-D database itself is generated using the data generation

Query No	Name	Description
Q2	Minimum Cost Supplier Query	Find which supplier should be selected to place an order for a given part in a given region
Q3	Shipping Priority Query	Retrieve the 10 unshiped orders with the highest value
Q5	Local Supplier Volume Query	List the revenue volume done through local suppliers
Q6	Forcasting Revenue Change Query	Quantify the amount of revenue increase in a given year
Q7	Volume Shipping Query	Determine the value of goods shipped between certain nations
Q9	Product Type Profit Measure Query	Determine how much profit is made on a given line of parts
Q10	Return Item Reporting Query	Identify customers who might be having problems with parts shipped to them
Q11	Identification of Stock Query	Find the most important subset of suppliers' stock in a given nation
Q12	Shipping Mode Query	Determine whether selecting less expensive modes of shipping affect priority orders
Q14	Promotion Effect Query	Monitor market response to a promotion such as a TV advertisement
Q17	Small-Quantity Order Query	Determine revenue lost if orders where not filled for small quantities of certain parts

Table 3: Description of the TPC-D Queries

utility, `DBGEN`, supplied by the TPC. The `DBGEN` program generates mainly uniformly distributed data. Since, we are interested in modeling skewed data distribution as well, we have opted to populate the database in three different ways listed below.

In the first method we simply use the data from the `DBGEN` program. This typically results in uniform data.

In the second method, we generate frequencies from a Zipf distribution with an appropriately chosen skew parameter, z, while retaining the original value domain and relation sizes. We combine the attribute values in various attributes in each relation randomly to generate the tuples for that relation, according to the frequencies obtained from the Zipf distribution.

It is claimed that the multi-fractals distributions occur more frequently in the real world data [5]. So in our third method, we generate frequencies using multi-fractal distributions with various bias values for the original value domain from the TPC-D database. The frequency values resulting from the multi-fractal distributions are combined with the values

Query Number	TPC-D Query
Q2	select s_acctbal, s_name, n_name from part, supplier, partsupp, nation, region where r_regionkey == 3 and n_regionkey == r_regionkey and s_nationkey == n_nationkey and p_size == 40 and ps_partkey == p.partkey and ps_suppkey == s_suppkey
Q3	select l_orderkey, o_shippriority from lineitem, order, customer where l_orderkey == o_orderkey and c_custkey == o_custkey
Q5	select n_name from order, lineitem, customer, supplier, nation, region where o_orderkey == l_orderkey and l_suppkey == s_suppkey and c_nationkey == s_nationkey and s_nationkey == n_nationkey and n_regionkey == r_regionkey and r_regionkey == 1
Q6	select * from lineitem where l_quantity $\leq$ 25
Q7	select supp_nation, cust_nation from lineitem, order, customer, supplier, nation where s_suppkey == l_suppkey and l_orderkey == o_orderkey and c_custkey == o_custkey and n_nationkey == c_nationkey and n_nationkey == 3
Q9	select * from supplier, partsupp, lineitem, order, nation where ps_suppkey == l_suppkey and o_orderkey == l_orderkey and s_suppkey == l_suppkey and s_nationkey == n_nationkey
Q10	select c_custkey, c_name, c_acctbal, n_name from lineitem, order, customer, nation where c_custkey == o_custkey and l_orderkey == o_orderkey and c_nationkey == n_nationkey
Q11	select ps_partkey from partsupp, supplier, nation where ps_suppkey == s_suppkey and s_nationkey == n_nationkey and n_nationkey == 3
Q12	select * from order, lineitem where o_orderkey = l_orderkey
Q14	select * from part, lineitem where p_partkey == l_partkey
Q17	select * from lineitem, part where p_partkey = l_partkey and p_brand = 'Brand4'

Table 4: Simplified TPC-D Benchmark Queries

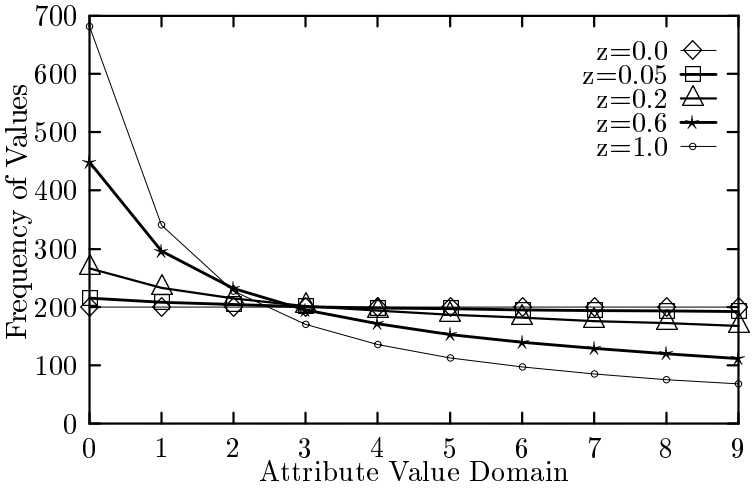


Figure 5: Zipf Distributions for Various z Parameters

from the TPC-D database to generate the tuples for the new relations.

Since the second and third methods of populating the databases of our experiments involve the Zipf and multi-fractal distributions, we give a brief overview of these distributions in the next section.

5 Overview of the Distributions Used in the Experiments

5.1 Zipf Distribution

G.K. Zipf first proposed a law, called the Zipf's law, which he observed to be approximately obeyed in many of the real-world domains, such as physics, biology, income distributions [26]. Zipf's law is essentially an algebraically decaying function describing the probability distribution of the empirical regularity. Zipf's law can be mathematically described in the context of our problem as follows.

For an attribute value X of size N with L distinct values, the frequencies generated by the Zipf distribution are given by,

$$f_i = N.\frac{1/i^z}{\sum_{i=1}^{L} 1/i^z}, \text{ for } 1 \leq i \leq L.$$

The skew of the Zipf distribution is a monotonically increasing function of the z parameter, starting from $z = 0$, which is the uniform distribution. We have plotted the frequency sets of several Zipf distributions with different z values in Figure 5. These frequency distributions were all generated for $N = 2000$ and $L = 10$.

One of the common claims in database literature is that many attributes in real-life databases contain a few attribute values with high frequencies and the rest with low fre-

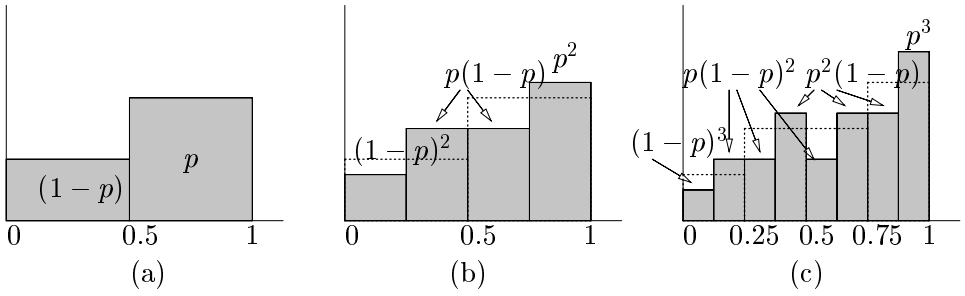


Figure 6: Generation of a Multi-fractal Distribution - First three steps

quencies [4], and hence can be modeled satisfactorily by Zipf distributions. Statistical literature also abounds with information on modeling real-life data by Zipf distributions. This is why we have also resorted to using Zipf distribution to generate frequencies for the value domains in the TPC-D benchmark databases.

5.2 Multi-fractal Distribution

The relationship of multi-fractals with the "80-20 law" is very close, and seem to appear often. Several real-world distributions follow a rule reminiscent of the 80-20 rule in databases. For example, photon distributions in physics, or commodities (such as gold, water, etc) distributions on earth etc., follow a rule like "*the first half of the region contains a fraction p of the gold, and so on, recursively, for each sub-region.*" Similarly, financial data and people's income distributions follow similar patters.

With the above rule, we assume that the attribute value domain is recursively decomposed at k levels; each decomposition halves the input interval into two. Thus, eventually we have 2^k sub-intervals of length 2^{-k}.

We consider the following distribution of probabilities, as illustrated in Figure 6. At the first level, the left half is chosen with probability $(1-p)$, while the right half is with p; the process continues recursively for k levels. Thus, the left half of the sectors will host $(1-p)$ of the probability mass, the left-most quarter will host $(1-p)^2$ etc.

For our experiments we use a *binomial multi-fractal* distribution with N tuples and parameters p and k, with 2^k possible attribute values. Note that when $p = 0.5$, we have the uniform distribution. For a binomial multi-fractal, we have

Count	Relative Frequency
C_0^k	p^k
C_1^k	$p^{(k-1)}(1-p)^1$
...	...
C_a^k	$p^{(k-a)}(1-p)^a$
...	...

In other words, out of the 2^k distinct attribute values, there are C_a^k distinct attribute values, each of which will occur $N * p^{(k-a)}(1-p)^a$ times. (For example, out of the 2^k distinct attribute values, there is $C_0^k = 1$ attribute value that occurs p^k times.)

6 Experiments with the TPC-D Benchmark Database

Unlike the prototype validating experiments which were discussed in [22, 23], the TPC-D benchmarking experiments involve arbitrarily large and complex query types. Consequently there are often thousands or more possible query evaluation plans (QEPs) for executing them. Since our objective is to compare the query result size estimation accuracy of various techniques as opposed to choosing the optimal QEP, we randomly selected an arbitrary QEP for a given query type and constructed the corresponding operator tree. The result size of this operator tree was then estimated in a bottom-up fashion from the leaves of the tree to the root of the tree. It is important to note that, unlike the case of a primitive relational operation, with these simulated "real-life" query types, it was necessary to construct the ACMs and histograms for all the resulting intermediate relations. What follows is the details of the results we obtained.

The experiments we conducted with the TPC-D benchmark database involved three distinct groups. In the first set of experiments, we used the uniform distribution data that was generated by the `DBGEN` program from TPC. We computed the estimation errors from the equi-width, equi-depth histograms, the R-ACM and the T-ACM for the set of TPC-D queries from Table 4. We conducted ten different set of experiments using the histograms and the ACMs with different build-parameters. The build-parameter for the equi-width histogram and the T-ACM is the sector width, the build-parameter for the equi-depth histogram is the number of tuples per sector and the build parameter for the R-ACM is the tolerance value, τ. For each attribute involved in the query, we generated the equi-width and equi-depth histograms with different sector-widths and different number of tuples/sector respectively. The corresponding T-ACMs were also generated with the same sector widths that were chosen for the equi-width histograms. The T-ACMs were generated using the `Implement_T-ACM` algorithm, which is a much improved version of the original `Generate_T-ACM` algorithm given in [17]. Similarly for each attribute involved in the query, we generated the R-ACM with a different value for the tolerance, τ. In order to obtain a fair comparison, the build-parameters for all different techniques were chosen such that the resulting storage requirements were the same, regardless of the method. The result sizes and consequently the computed estimation errors were averaged over these set of experiments. The results are shown in Table 5.

In the second set of experiments, we used Zipf data distributions with three different skew values ($z = 2, z = 4, z = 6$) on the value domains of the various attribute values of the TPC-D database and computed the estimation errors from the equi-width, equi-depth histograms, the R-ACM and the T-ACM. Again we conducted ten different set of experiments using different build-parameters for the equi-width, equi-depth histograms, the R-ACM and the

TPC-D Query	Actual Result Size	Error with Uniform TPC-D Database			
		Equi-width	Equi-depth	R-ACM	T-ACM
Q2	918	10.94%	9.60%	3.21%	3.72%
Q3	42190	13.62%	11.39%	4.37%	5.73%
Q5	209018	10.05%	9.68%	2.55%	3.61%
Q6	20706	12.33%	10.16%	4.38%	4.27%
Q7	2041	15.20%	10.75%	6.13%	5.99%
Q9	2.71e+06	13.83%	11.42%	5.48%	4.91%
Q10	44823	14.07%	12.93%	5.06%	6.15%
Q11	511	11.39%	10.40%	4.21%	5.03%
Q12	19	25.97%	22.73%	5.81%	5.62%
Q14	36724	14.66%	13.45%	4.77%	5.28%
Q17	523	20.35%	19.73%	5.81%	6.37%

Table 5: Estimation Error with Histograms, R-ACM and T-ACM on Uniform TPC-D Database

T-ACM. Due to the steep frequency variations of the Zipf distributions used, the R-ACM partitioning resulted in a very large number of sectors. In order to avoid this, we were forced to choose somewhat larger tolerance values for the R-ACM partitioning. As with the first set of experiments, in order to make a fair comparison of the various techniques, the build-parameters were chosen so that the resulting storage requirements were same for the histograms and the ACMs. The experiments were conducted for the set of TPC-D queries listed in Table 4. We averaged the results for the three Zipf distributions and computed the estimation errors. The results are given in Table 6.

The third set of experiments involve using the multi-fractal distributions. A single scan through the data generated by the `DBGEN` program returns the number of distinct attribute values for each of the attributes involved. We selected three different bias values ($p = 0.1, p = 0.3, p = 0.4$) that resulted in three completely different multi-fractal distributions. The frequencies from these multi-fractal distributions were applied randomly to the value domain of the attributes in the TPC-D database to generate a new set of relations. As before we conducted ten different set of experiments with different build-parameters for the equi-width, equi-depth histograms, the R-ACM and the T-ACM, using the same storage requirements. Again the experiments involved applying the TPC-D queries from Table 4. We averaged the results for the three multi-fractal distributions and computed the estimation errors. The results are given in Table 7.

TPC-D Query	Actual Result Size	Error with Zipf-TPC-D Database			
		Equi-width	Equi-depth	R-ACM	T-ACM
Q2	161	27.46%	24.30%	9.62%	11.83%
Q3	26428	29.75%	26.42%	12.87%	12.90%
Q5	38620	19.07%	17.29%	9.46%	10.94%
Q6	16271	17.20%	14.65%	9.33%	6.85%
Q7	263	26.71%	25.80%	13.42%	15.25%
Q9	2.8e+06	19.81%	20.11%	10.66%	11.17%
Q10	22670	26.28%	23.04%	12.37%	8.42%
Q11	395	21.44%	17.59%	9.28%	9.91%
Q12	38	37.46%	31.72%	14.36%	15.11%
Q14	26538	23.81%	21.25%	15.30%	14.65%
Q17	1932	31.63%	29.28%	11.92%	10.87%

Table 6: Estimation Error with Histograms, R-ACM and T-ACM on Multi-fractal TPC-D Database

7 Analysis of the Experimental Results

Our results from these three set of TPC-D benchmark experiments confirm our theoretical results, summarized in Table 2, and clearly demonstrate that the estimation accuracy of the R-ACM and the T-ACM structures is superior to that of the traditional equi-width and equi-depth histograms.

As we can observe, the estimation errors for the histograms and the ACMs are the lowest for the experiments with the original TPC-D database, which was generated by the TPC supplied `DBGEN` program. For example, for the TPC-D query $Q2$, the estimation errors for the original TPC-D database from Table 5 are 10.94%, 9.60%, 3.21% and 3.72% for the equi-width, equi-depth, R-ACM and T-ACM respectively. Whereas, as can be seen from Table 6, the estimation errors for query $Q2$ with the multi-fractal TPC-D database are 27.46%, 24.30%, 9.62%, and 11.83% in the same order. The estimation errors for the Zipf TPC-D database, given in Table 7, are even higher. The reason for the lowest error with the original TPC-D database is, that it is uniformly distributed. Similarly the reason for the largest estimation errors with the Zipf TPC-D database is obviously due to the steep frequency changes by virtue of the underlying Zipf distribution. Another reason why the estimation errors for the R-ACM are larger than for the first set of experiments is our choice of comparatively larger tolerance values to avoid generating R-ACMs with large number of sectors. The estimation accuracy of the R-ACM and that of the T-ACM are almost comparable except for range-select queries, where the T-ACM out-performs the R-ACM. This is due to the fact that the trapezoidal rule of numerical integration is more accurate than the right-end or the left-end rectangular rule of numerical integration for

TPC-D Query	Actual Result Size	Error with Multi-fractal-TPC-D Database			
		Equi-width	Equi-depth	R-ACM	T-ACM
Q2	528	30.49%	28.26%	15.42%	17.22%
Q3	26210	29.83%	24.68%	14.58%	16.26%
Q5	132450	40.63%	36.27%	12.72%	10.67%
Q6	19204	30.20%	26.11%	14.32%	11.71%
Q7	3096	34.26%	27.42%	12.33%	15.24%
Q9	3.6e+06	28.39%	25.75%	9.97%	9.29%
Q10	40022	26.92%	24.18%	11.92%	13.33%
Q11	212	28.21%	22.73%	12.07%	13.15%
Q12	86	26.42%	23.12%	10.25%	12.63%
Q14	49270	26.25%	21.49%	13.92%	12.84%
Q17	976	23.46%	21.22%	14.67%	15.90%

Table 7: Estimation Error with Histograms, R-ACM and T-ACM on Zipf TPC-D Database

computing the area under a curve.

The results from all three sets of experiments show that the estimation errors resulting from the R-ACM and the T-ACM are **consistently** much lower than the estimation errors from the equi-depth and equi-depth histograms. This is consequent to the fact the frequency distribution of an attribute value within an R-ACM is guaranteed to be close to the sector mean since the partitioning of the sectors is based on a user-specified tolerance value, and the deviation from the running mean. Similarly, the trapezoidal rule of the numerical integration technique is more accurate than the right-end or left-end histogram approximation techniques. Thus, these set of experiments with the TPC-D benchmark queries and databases demonstrate that the R-ACM and T-ACM strategies exhibit a distinctively superior performance over the traditional equi-width and equi-depth histograms. Such results are typical with both synthetic data and real-world data. The power of the ACM structures is obvious!

8 Conclusion

Benchmarking is an important and complex phase in the development of any new software strategy. In this paper, we have presented a set of benchmarking experiments and reported their respective results to validate the performance of two new histogram-like query result size estimation strategies, namely the Rectangular and Trapezoidal Attribute Cardinality Maps (R-ACM and T-ACM) [16, 17]. The set of benchmarking experiments we used were based on the industry-popular TPC-D benchmark database and query sets.

It was claimed in [16, 17] that, since the R-ACM and T-ACM techniques are based

on the philosophies of numerical integration, they are more accurate than the traditional histograms. Their analytical properties were also presented in [16, 17]. We also presented a set of prototype validating experiments and their results on real-world data in [22, 23]. But those experiments only included some synthetic queries, involving simple elementary relational operations. The benchmarking experiments we conducted in this work included both an arbitrarily large simulated "real-world" database and a set of highly complex simulated "real-world" query patterns, both satisfying strict industry related standards. These experimental results clearly demonstrate and validate our theoretical analysis of the R-ACM and the T-ACM and their superiority over the current state-of-the-art estimation techniques based on the traditional equi-width and equi-depth histograms. We hope that these new structures will become invaluable tools for query optimization in the future due to their superior estimation accuracy and low implementation overheads.

Although this paper has answered a few questions, it has also opened the door to a lot of new research avenues. It is not unwise to add that since the study of ACM's is in its infancy, many questions remain unanswered. In particular, we emphasize that the experimental comparison between the ACMs and many of the other reported histogram methods (other than the equi-width and equi-depth representations) has still to be done. The entire issue of how insertion and deletion operations can be achieved using the ACM's also remains unsolved. Although some of the work in these areas has been initiated, we invite collaborations and joint research endeavours.

A few commercial database vendors are currently investigating the possibility of using some of the concepts introduced in this work.

References

[1] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. Approximate Query Processing Using Wavelets. In *Proceedings of the 26th International Conference on Very Large Databases*, pages 111–122, Cairo, Egypt, September 2000.

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

[3] S. Christodoulakis. Estimating record selectivities. In *Information Systems*, volume 8, 1983.

[4] S. Christodoulakis. Implications of certain assumptions in database performance evaluation. In *ACM Transactions on Database Systems*, volume 9(2), pages 163–186, 1984.

[5] Christos Faloutsos, Yossi Matias, and Avi Silberschatz. Modeling skewed distributions using multifractals and the 80-20 law. In *Technical Report*, Dept. of Computer Science, University of Maryland, 1996.

[6] J. Goswami and A. Chan. *Fundamentals of Wavelets: Theory, Algorithms, and Applications*. Wiley Series-Microwave and Optical Engineering, 1999.

[7] Yannis Ioannidis and S. Christodoulakis. On the propagation of errors in the size of join results. In *Proceedings of the ACM SIGMOD Conference*, pages 268–277, 1991.

[8] Yannis Ioannidis and Stavros Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. In *ACM TODS*, 1993.

[9] H. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, and T. Suel. Optimal Histograms with Quality Guarantees. In *International Conference on Very Large Databases*, pages 275–286, 1998.

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

[11] M.V. Mannino, P. Chu, and T. Sager. Statistical profile estimation in database systems. In *ACM Computing Surveys*, volume 20, pages 192–221, 1988.

[12] Y. Matias, J. Vitter, and M. Wang. Wavelet-Bassed Histograms for Selectivity Estimation. In *ACM SIGMOD Conference on Management of Data*, pages 448–459, 1998.

[13] M. Muralikrishna and David J Dewitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *Proceedings of ACM SIGMOD Conference*, pages 28–36, 1988.

[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*, volume 19, pages 84–91, 1997.

[15] B. J. Oommen and L. Rueda. The Efficiency of Modern-day Histogram-like Techniques for Query Optimization. In *The Computer Journal*, 2002 - To Appear.

[16] B. J. Oommen and M. Thiyagarajah. Query result size estimation using a novel histogram-like technique : The rectangular attribute cardinality map. In *Proceedings of the 1999 International Database Engineering and Applications Symposium - IDEAS1999*, pages 3–15, Montreal, Canada, August 1999. Also available as a Technical report - TR-99-01, from School of Computer Science, Carleton University, Ottawa, Canada, January 1999.

[17] B. J. Oommen and M. Thiyagarajah. On the use of the trapezoidal attribute cardinality map for query result size estimation. In *Proceedings of the 2000 International Database Engineering and Applications Symposium - IDEAS2000*, pages 236–242, Yokohama, Japan, September 2000. Also available as a Technical report - TR-99-04, from School of Computer Science, Carleton University, Ottawa, Canada, Feb. 1999.

[18] Gregory Piatetsky-Shapiro and Charles Connell. Accurate estimation of the number of tuples satisfying a condition. In *Proceedings of ACM SIGMOD Conference*, pages 256–276, 1984.

[19] V. Poosala and Y. Ioannidis. Selectivity Estimation Without the Attribute Value Independence Assumption. In *23rd. International Conference on Very Large Databases*, pages 486–495, Athens, Greece, August 1997.

[20] V. Poosala, Y. Ioannidis, P. Haas, and E. Shekita. Improved Histograms for Selectivity Estimation of Range Queries. In *ACM SIGMOD Conference on Management of Data*, pages 294–305, 1996.

[21] P. Selinger, M.M. Astrahan, D.D. Chamberlin, R.A. Lorie, and T.G. Price. Access path selection in a relational database management system. In *Proceedings of ACM-SIGMOD Conference*, pages 23–34, 1979.

[22] M. Thiyagarajah and B. J. Oommen. Prototype Validation of the Rectangular Attribute Cardinality Map for Query Optimization in Database Systems. In *International Conference on Business Information Systems - BIS'99*, pages 250–262, Poznan, Poland, April 1999.

[23] M. Thiyagarajah and B. J. Oommen. Prototype Validation of the Trapezoidal Attribute Cardinality Map for Query Optimization in Database Systems. In *International Conference on Enterprise Information Systems - ICEIS'99*, pages 156–162, Setubal, Portugal, March 1999.

[24] Murali Thiyagarajah. *Attribute Cardinality Maps : New Query Result Size Estimation Techniques for Database Systems.* PhD thesis, Carleton University, Ottawa, Canada, 1999.

[25] Transaction Processing Council (TPC). TPC-D Benchmark Specification. Feb 1998.

[26] G. K. Zipf. *Human Behavior and the Principle of Least Effort.* Addision-Wesley, Reading, MA, 1949.