

A Pareto Model for OLAP View Size Estimation

Thomas P. Nadeau (nadeau@engin.umich.edu)
Toby J. Teorey (teorey@eecs.umich.edu)
University of Michigan
Ann Arbor, Michigan

Abstract

On Line Analytical Processing (OLAP) aims at gaining useful information quickly from large amounts of data residing in a data warehouse. To improve the quickness of response to queries, pre-aggregation is a useful strategy. However, it is usually impossible to pre-aggregate along all combinations of the dimensions. The multi-dimensional aspects of the data lead to combinatorial explosion in the number and potential storage size of the aggregates. We must selectively pre-aggregate. Cost/benefit analysis involves estimating the storage requirements of the aggregates in question. We present an original algorithm for estimating the number of rows in an aggregate based on the Pareto distribution model. We test the Pareto Model Algorithm empirically against three published algorithms, and conclude the Pareto Model Algorithm is consistently the best of these algorithms for estimating view size.

1 Motivation

Accumulation of data in industry and organizations has led to large archives of data in recent years. Quick access to the information in these archives has become critical for decision-making. The need to excel has given rise to new data models and decision support systems. Typically the queries posed involve operations of aggregation such as sum or count. The queries also typically include “group by” expressions. For example, the CEO of a book manufacturing company may want to examine trends in profitability of different types of books over time. The answer could be found by doing a sum of the cost and sell values of jobs, grouped by bind style

and quarter. Data warehouses have been engineered to answer queries of aggregation with “group by” expressions efficiently.

Data warehouses are commonly organized with one large central fact table, and many smaller dimension tables. The fact table is keyed by the attributes to be used in “group by” expressions. The fact table also contains measure attributes, the values to be aggregated. Each attribute of the fact table key is typically a foreign key matching the primary key of a dimension table. Figure 1 illustrates an example of a star schema. The *CustID*, *DateID* and *BindID* together make up the primary key of the *Fact Table*. Thus there are three dimensions. Notice that a dimension can also have a hierarchy. For example, time can be grouped by *DateID*, *Month*, *Quarter* or *Year*.

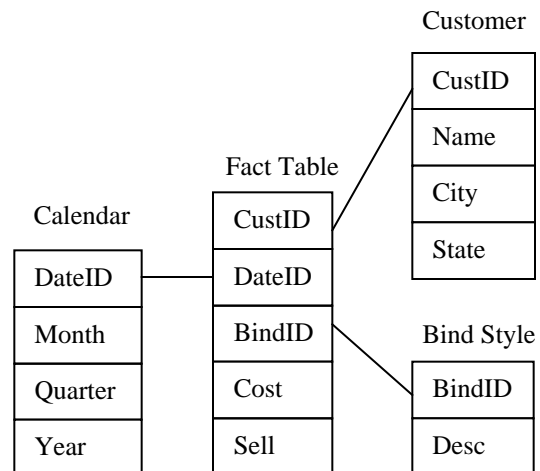


Figure 1: A simple star schema.

A fact table in a data warehouse may contain many millions of rows, and processing a single aggregate can require significant resources. To

improve the quickness of response to queries, pre-aggregation is a useful strategy. Pre-aggregation requires the result to be saved to disk. The number of possible aggregates is exponential in the number of dimensions. Faced with combinatorial explosion and limited disk space, we must decide which aggregates to calculate in anticipation to queries. The cost/benefit analysis involves estimating the storage requirements of the aggregates in question.

This paper focuses on estimating the space required for an aggregate. We present an original algorithm for estimating the number of rows in an aggregate based on the Pareto distribution model. We test the Pareto Model Algorithm (PMA) empirically against three published algorithms, and conclude the Pareto Model Algorithm is the best of these algorithms for estimating view size.

The remainder of the paper is organized as follows. Section 2 gives a brief overview of related works. Section 3 discusses our preliminary work of testing three published algorithms, and one original algorithm. Section 4 covers the progression of thought and experimentation that led us to the Pareto Model Algorithm. Test results over the real world data set are included in section 4, since these results were instrumental in the development process. Section 5 outlines the synthetic data sets utilized for testing. Section 6 presents the experimental results over the synthetic data sets. Analysis and conclusions are made in section 7. Future work is covered in section 8.

2 Related Work

This section first outlines the sources of ideas used in this paper for estimating the number of rows in an aggregate.

There is a simple equation for estimating the number of rows in an aggregate. The approach is known as Cardenas' formula [1]:

Let n be the number of rows.

Let v be the number of possible values.

$$\text{Expected distinct values} = v - v(1 - 1/v)^n \quad (1)$$

Cardenas' formula assumes uniform distribution. However, the data distribution affects the number of rows in an aggregate. In order to capture the

effect of data distribution, other methods have been developed.

Probabilistic counting was introduced as a new approach in [5]. A hashing function is applied to the values, and meta-data is gathered on the output. Probabilistic analysis is applied to the meta-data, determining an estimate of the number of distinct values. The approach uses very little memory, but requires a full scan of the data.

A sampling approach based on the binomial multifractal distribution model is presented in [6]. Parameters of the distribution are estimated from a sample. The number of rows in the aggregate for the full data set can then be estimated using the parameter values determined from the sample. Further details of this approach can be found in section 4.1.

Three approaches are tested and compared in [11]. They examine Cardenas' formula, a sampling approach we call linear projection, and the probabilistic counting method. The probabilistic counting method is the most accurate of the three algorithms tested, for the given data sets.

Two algorithms, which are hybrids of Cardenas' formula and sampling approaches, are presented and tested in [10]. The proportional skew effect algorithm, and the sample frequency algorithm tested favorably compared to Cardenas' formula and linear projection.

3 Preliminary Work

Our preliminary work involved testing three published algorithms and one original algorithm over a real world data set. The details of the real world data set are covered in section 3.1. The algorithms tested in the early development stage included Cardenas' formula [1], the Sample Frequency Algorithm [10], and the algorithm developed in [6] which we will call FMS for short. FMS is based on the binomial multifractal distribution model (discussed in detail in section 4.1). We developed another algorithm based on this distribution model, which we named the Curve Fitting algorithm [9]. We will not go into the details of these algorithms in this section. Our purpose in this section is to demonstrate why we set the goal of bettering the FMS algorithm. The FMS approach and our recent work is the focus of section 4.

Attribute	Cardinality	Explanation/Examples
Bind Style	14	Paper back, hard cover, comb bound etc.
Trim Width	13	The width of the book (e.g. 6")
Trim Length	14	The length of the book from top to bottom (e.g. 9")
Pages	31	The number of pages in the book
Quantity	28	The number of books ordered
Stock Color	18	The color of the paper used in the book (e.g. white)
Stock Weight	5	An industry standard measurement of paper weight for a fixed amount of paper (e.g. 50#, 60# etc.)
Stock Width	12	The width of the paper run on the press (e.g. 29")
Stock Length	12	The length of the paper run on the press (e.g. 42")
Press	5	The press the book was run on (e.g. Miehle, Planeta etc.)

Table 1: Attributes of the real world database.

We did not implement the probabilistic counting method [5] because that approach requires a full scan of the fact table. We are looking for approaches efficient over huge fact tables.

3.1 The real world data set

The motivations for testing aggregate storage size estimation algorithms on real world data are many. The usefulness of any algorithm ultimately depends on the effectiveness of its application to real world problems. Real world data presents a challenge in that the distribution of the data is usually unknown. Should the data be modeled using Zipf distributions, normal distributions, binomial multifractal distributions, or some other distribution model?

We obtained a real world data set in cooperation with a book manufacturing company, McNaughton & Gunn, Inc. McNaughton & Gunn specializes in short to medium run titles. Their customers include thousands of publishers across the country. The data set they supplied is a history of job specifications for books they have produced.

McNaughton & Gunn periodically analyses the job mix in their plant. Analyzing a fact table containing job specifications is a realistic application. Specifications for 14,438 jobs were gathered. A fact table was built using ten job attributes as the key. An eleventh field was used for tracking the number of jobs with the given job

specifications. Some jobs have the same key values. The job count field is not part of the key, rather it is a measurement field. Job count is the information to be aggregated. Some jobs have duplicate specifications, so the resulting fact table has fewer rows than the original set of jobs. The fact table has 8,238 rows. The dimensions are shown in Table 1 on the following page.

A quick calculation shows the total number of possible tuples in the base data is 143,315,827,200. The density of the base data is $5.7 \cdot 10^{-8}$. Even with such sparsity, 43% of the jobs have the same specifications as other jobs. This already gives a clue that the data is very skewed.

There are 10 dimensions, and therefore $2^{10}-1$ different aggregates, not counting the original fact table. The aggregates will dominate the fact table in the number of rows after cubing.

3.2 Preliminary test results

Table 2 shows the results of our preliminary testing. We tested at three different sample sizes: 1%, 3% and 10% of the fact table. This is a small database. Larger databases would tend not require as large a sample percentage. Larger databases are explored in section 6 during our scale-up testing. For this round of testing, we ran three independent runs for each algorithm at each sample size. Each run estimates the size of 1023 aggregates. Thus each number in Table 2 represents a statistic of 3069 estimates. The first measurement is the mean of the estimated rows /

Algorithm	Mean of Estimate/Actual			Standard Deviation			Coefficient of Variation		
	Sample Size			Sample Size			Sample Size		
	1%	3%	10%	1%	3%	10%	1%	3%	10%
Cardenas' Formula	5.329	5.329	5.329	5.052	5.052	5.052	0.948	0.948	0.948
Sample Frequency	2.892	1.787	0.987	1.826	1.097	0.572	0.631	0.614	0.579
FMS	1.342	1.105	1.012	0.660	0.370	0.200	0.492	0.335	0.198
Curve Fitting	0.886	0.921	0.893	0.687	0.427	0.233	0.775	0.463	0.261

Table 2: Preliminary rest results on real world data.

actual rows. Ideally this should be 1. Values above 1 indicate over-estimates. Values below 1 indicate under-estimates. The next measurement is the standard deviation of the estimate/actual. Ideally this should be as small as possible. The last measurement is the coefficient of variation. This is the ratio of the standard deviation divided by the estimate/actual value. Ideally the coefficient of variation would be as small as possible.

Cardenas' formula drastically over-estimates. This clearly indicates the need to account for skew in the data distribution. The FMS algorithm is the best estimator at this point. We investigate several new approaches to see if we can improve on the FMS algorithm.

4 Current work

We will now examine in detail the binomial multifractal distribution model, which is the basis of FMS. We discover empirically a flaw in the assumptions made by that model. Through an experiment with a simple algorithm we gain further insight. Then we develop a successful approach based on the Pareto distribution model.

4.1 Binomial multifractal distribution model

Large-scale structure resembles small-scale structure in multi-fractal models. Figure 2 illustrates a binomial multi-fractal distribution tree with a small example. The decision tree depth is $k = 3$. The probability of a right edge is the bias parameter $P = 0.9$. The probability of a right branch remains the same regardless of the depth

in the tree. Each bin at the bottom of the tree represents a distinct value in the data set. The number in each bin of Figure 2 is the probability that a random tuple belongs in that bin. Note the bins group as sets. There is a relationship between the number of bins in each set, and the elements in Pascal's triangle. The super-script of C is the depth into Pascal's triangle, starting with row 0 at the top of Pascal's triangle. The sub-script of C is the position into the row of Pascal's triangle, beginning with the left-most item as element 0.

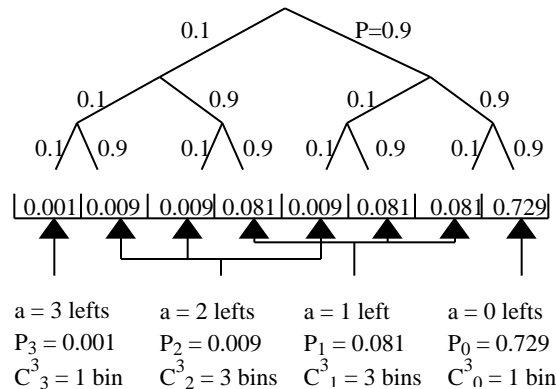


Figure 2: Example of a binomial multi-fractal distribution tree.

The theory behind the approach presented in [6] is that by calculating the parameters of a multi-fractal distribution based on a small sample, the number of distinct members can be predicted for a larger set of data. Equations (2) and (3) are presented in [6] for this purpose.

$$\text{Expected distinct values} = \sum_{a=0}^k C_a^k (1 - (1 - P_a)^N). \quad (2)$$

$$P_a = P^{k-a} (1 - P)^a. \quad (3)$$

Figure 2 illustrates with an example. Order k is the decision tree depth. C_a^k is the number of bins in the set reachable by taking some combination of a left hand edges and $k - a$ right hand edges in the decision tree. P_a is the probability of reaching a given bin whose path contains a left hand edges. N is the number of rows in the data set. Bias P is the probability of selecting the right hand edge at a choice point in the tree.

We illustrate the calculations of Eq. (2) with a small example. An actual database would yield much larger numbers, but the concepts and the equations are the same. These calculations can be done with logarithms, resulting in very good scalability. Based on figure (2), given 5 rows calculate the expected distinct values:

$$\begin{aligned} \text{Expected distinct values} = & \\ & 1 (1 - (1 - 0.729)^5) + 3 (1 - (1 - 0.081)^5) \\ & + 3 (1 - (1 - 0.009)^5) + 1 (1 - (1 - 0.001)^5) \quad 1.965 \quad (4) \end{aligned}$$

4.2 The discovery of opportunity

Our attempts to develop a better algorithm based on the binomial multifractal distribution model resulted in only small improvements. We decided to produce some scatter plots to better visualize the problem. There are 1023 aggregates in our test database. Each point in the scatter plots represents a test result for a single aggregate. We plotted the estimate/actual ratio versus the actual number of rows. All algorithms based on the binomial multifractal distribution produced a common pattern. The plot for the FMS algorithm is illustrated in figure 3.

The FMS algorithm overestimates for larger aggregates and underestimates the size of smaller aggregates. The huge underestimates in the left half of figure 3 pose a problem. When estimating required disk space, it is preferable to overestimate rather than underestimate. Furthermore, the systematic trends will unduly bias a view selection algorithm to materialize the very aggregates whose sizes have been underestimated.

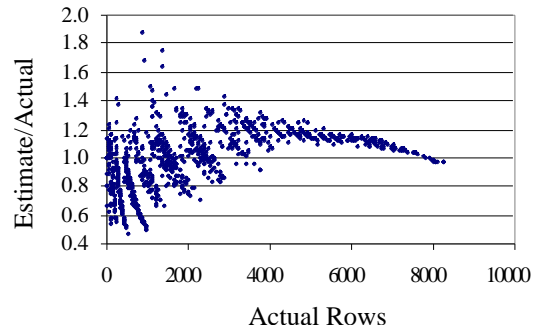


Figure 3: Scatter plot for FMS algorithm, illustrating trends in estimate/actual versus actual rows.

The underlying assumption in the binomial multifractal distribution model is that large-scale structure resembles small-scale structure, and that the skew detected in a sample will continue throughout the unseen items in the domain. Perhaps there is a fundamental flaw in this assumption. Next, we turn to an original approach that does not assume skew in the unseen items.

4.3 Estimating with uniform distribution over unseen items

Our next approach is also based on sampling. However, in contrast to the binomial multi-fractal model, we will not assume the skew in the sample continues throughout the unseen items in the domain. We will make the radically different assumption that there is no skew in the unseen items, and compare the scatter plot with that of FMS for enlightenment.

4.3.1 The simple algorithm

The approach is very straightforward. Examine a sample. Count the number of items in the sample occurring more than once. Call this set of items the MSet, and the number of distinct rows in the MSet MRows. Tally the frequencies in the sample for the items in the MSet. Divide this tally by the sample size. This will estimate the probability a row at random from the data will duplicate an MSet item. Figure out how many rows are expected to be outside the MSet. Assume they are uniformly distributed over the items in the domain

that lie outside of the MSet. Use Cardenas' formula to calculate the expected number of distinct rows outside of the MSet for the full data set. Add this to the MRows for the final estimate.

4.3.2 Results of the simple algorithm

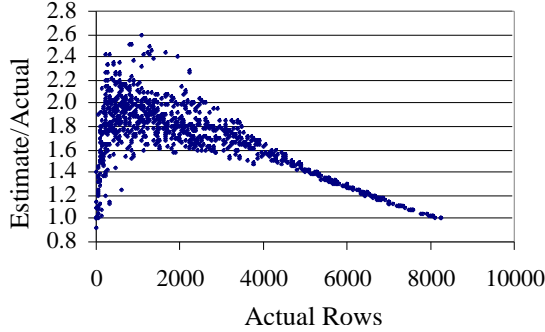


Figure 4: Scatter plot of the simple algorithm illustrating trends in estimate/actual versus actual rows.

4.3.3 Conclusions on the Simple Algorithm

Compare figure 4 with figure 3. Observe the simple algorithm overestimates the most on those aggregates that the FMS approach underestimates the most. The FMS approach assumes the skew observed in the sample will extend into the unseen items of the domain. The simple algorithm assumes there is no skew unless it is observed directly in the sample. These two approaches are diametrically opposed. The test results suggest the truth lies somewhere in between.

4.4 Pareto model approach

The last sampling approach we designed, implemented and tested is the Pareto Model Algorithm. The Pareto distribution is similar to the Zipf distribution model. Zipf observed a pattern common in real world distributions [12]. If the items in a Zipf distribution are ordered by descending frequency, the relative frequencies are described by the following equation:

$$f(X_i) = 1 / (X_i) \quad (5)$$

X_i is the position of the item. The α parameter is a measure of skew. If $\alpha = 0$ the distribution is uniform. α can be any non-negative number. The higher the α value, the more skewed the distribution. Here is an example of a Zipf distribution: Let $\alpha = 2$, then the progression is: 1, 1/4, 1/9, 1/16 ...

The equation for a Pareto distribution [4] is:

$$\begin{aligned} f(X) &= X_0 / X^{\alpha+1} \text{ for } X > X_0 \\ f(X) &= 0 \text{ otherwise.} \end{aligned} \quad (6)$$

The Pareto distribution is bound on the left by X_0 . Note: If $X_0 = 1$, the Pareto distribution becomes the continuous version of the Zipf distribution. The numerator X_0 normalizes the curve to give a probability distribution function. Notice the α value in the Pareto distribution is offset by 1 from the analogous α value in the Zipf distribution equation. This is for ease of integration. The domain of X values in the Pareto distribution as defined above is not bound on the right. For our purposes we need to set a bound on the right, since our domains are finite. We will set $X_0 = 1$, and let X_r be the right hand bound of the domain. These modifications to the Pareto distribution result in the continuous version of a Zipf distribution, bound over a finite domain. Our distribution is as follows:

$$\begin{aligned} f(X) &= [X_r / (X_r - 1)] / X^{\alpha+1} \text{ for } 1 < X < X_r \\ f(X) &= 0 \text{ otherwise.} \end{aligned} \quad (7)$$

$[X_r / (X_r - 1)]$ is a normalization factor. The resulting area under the curve is 1, as required for a probability distribution function.

The reason we have chosen the Pareto distribution model for our next algorithm is because the inherent assumptions lie between those of the binomial multi-fractal distribution model and the uniform distribution model. Recall in section 4.3.3 we observed reality appears to lay between the assumptions of the binomial multi-fractal distribution model and the uniform distribution model. The skew of the Pareto distribution is most heavily evident at the left end of the curve where the most frequent items lie. The distribution flattens as the curve progresses. The distribution of the remaining items approaches a uniform distribution as we move along the series. The model thus assumes skew affects the most common items most heavily and

less common items are more uniformly distributed. These properties make the Pareto distribution a good candidate for modelling real world data.

4.4.1 Pareto model algorithm

The Pareto Model Algorithm is summarized in figure 5. We will describe the details of each step in turn. The first step is to take a sample of the fact table. This only needs to be done once. We selected rows at random from the fact table. An efficient implementation is to generate a set of random numbers, sort the random numbers, fetch the rows from the fact table in sort order, and materialize the sample.

The next step begins a loop, which is processed once for each aggregate we wish to estimate. The loop begins by posing an SQL group-by query corresponding to a given aggregate. The query has this general form:

```
select group-by-list, count(*) (8)
from sample
group by group-by-list
order by count(*) desc;
```

If the database is hierarchical, the query must also include joins between the sample table and the pertinent dimension tables.

The query is used to identify aggregate rows with multiple occurrences in the sample. We call this set of distinct aggregate rows the MSet. Let MRRows be the number of rows in the MSet. Let MCount be the number of rows in the sample table that aggregate into the MSet.

Eq. 9 estimates the probability P that a row at random from the full data set will land in the MSet.

$$P = MCount / SampleRows \quad (9)$$

We treat the result of the SQL query in Eq. 8 as a histogram, and find a Pareto distribution that approximates the histogram. We visualize the aggregate row with the highest count as a column ranging from $X = 1$ to 2. The aggregate row with the second highest count becomes a column ranging from 2 to 3, and so on. Let j be the X value corresponding to where the MSet ends (i.e. $j = MRRows + 1$). Let $X_r = |\text{Possible rows in aggregate space} + 1|$. The possible rows in the

aggregate space can be calculated as the product of the number of distinct values in the dimension tables for the attributes of the group-by-list. Integrating Eq. 7 over the domain of the MSet (i.e. 1 to j), we arrive at Eq. 10.

$$P = X_r (1 - j) / j (1 - X_r) \quad (10)$$

We know the value of P , X_r and j . We calculate the value of α based on Eq. 10. We now have an instance of a Pareto distribution approximating the histogram from the sample.

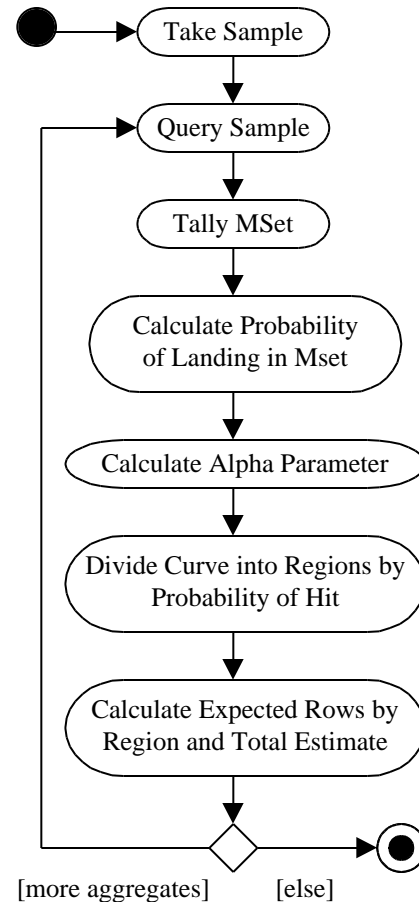


Figure 5: Activity diagram of Pareto Model Algorithm.

We divide the curve of the Pareto distribution from j to X_r into regions by probability of a hit from the full data set. The Mset will be treated separately. Our implementation creates 30 regions, and we find this to give satisfactory accuracy. The first region contains the domain of

Algorithm	Mean of Estimate/Actual			Standard Deviation			Coefficient of Variation		
	Sample Size			Sample Size			Sample Size		
	1%	3%	10%	1%	3%	10%	1%	3%	10%
Pareto Model Algorithm	2.051	1.754	1.305	0.776	0.440	0.151	0.379	0.251	0.116
FMS	1.342	1.105	1.012	0.660	0.370	0.200	0.492	0.335	0.198

Table 3: Test results on real world data, Pareto model algorithm versus FMS.

items where the probability of a hit is 0.9 or higher. The second region contains the domain of items where the probability ranges from 0.81 to 0.9. In general, the divider between region_i and region_{i+1} is at the point where the probability of a hit from the remaining data set is (0.9)ⁱ. The divider points can be determined with Eq. 12. Let T be the target probability for the divider. Let D = |Rows in full data set|. From Eq. 7, we derive Eq. 11 for calculating the probability of a hit at X.

$$T = 1 - (1 - [X_r / (X_r - 1)] / X^{+1})^D \quad (11)$$

Solving for X we have

$$X = [X_r / (X_r - 1)] / [(1 - (1 - T)^{1/D})^{1/(+1)}] \quad (12)$$

For each region, we multiply the width of the domain by the probability of a hit for that region, and then total up the expected rows for all regions. We add MRows to account for the MSet. The last region does not have its own divider, and must be handled differently. We estimate the expected rows for the last region by integrating Eq. 7 for the region and multiplying by D. This approach works since there are very few multiple occurrences of values within the last region. The resulting total is the expected number of distinct aggregate rows over the full data set. If there are more aggregates to process, we continue with the loop.

4.4.2 Results of Pareto model algorithm

Table 3 on summarizes the statistical results over the real world data set.

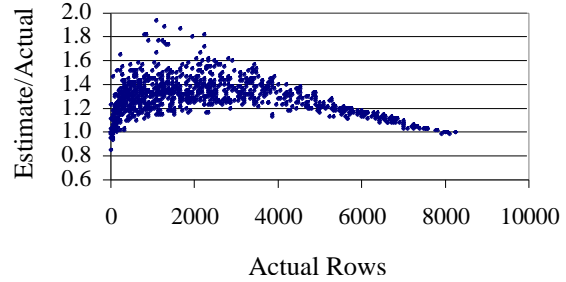


Figure 6: Scatter plot of the Pareto model algorithm illustrating trends in estimate/actual versus actual rows.

4.4.3 Conclusions on Pareto model algorithm

The Pareto model algorithm estimates with a smaller coefficient of variation than the other algorithms. This leads to better decisions when materialized views are selected. We have verified this in conjunction with the space limited greedy view selection algorithm from [7]. We ran the view selection algorithm based on the view size estimates obtained from the FMS algorithm, and our Pareto model algorithm. The results of a typical run are shown in figure 7.

We verified experimentally that PMA typically results in a better selection of views than FMS. The improvement is most pronounced at small sample sizes, with a small portion of the possible views being materialized. This is valuable because view size estimation runs faster at small sample sizes, and typically OLAP systems materialize only a small portion of the possible views.

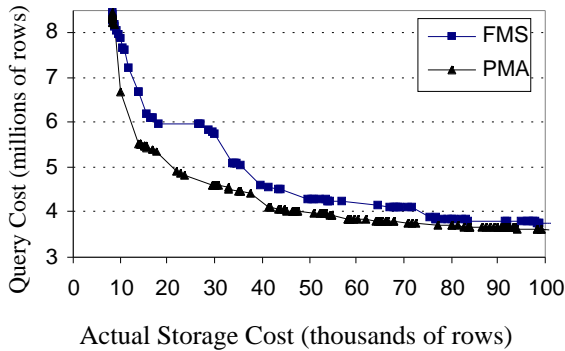


Figure 7: Improved view selection. Our PMA view size estimator improves the resulting view selection when compared with the FMS view size estimator. Query cost is the total number of rows read if a query is posed against every possible view. This test was implemented with the space limited greedy view selection algorithm from [7]

The Pareto model algorithm is also superior to the FMS algorithm in another important aspect. The Pareto model algorithm rarely underestimates the number of rows in the aggregate for the full dataset. FMS often underestimates, which is an undesirable quality when estimating required disk space.

5 The synthetic data sets

Our next group of experiments involved testing over synthetically generated data. All the synthetic databases utilize the same schema. There are three dimensions, each with its own hierarchy.

Dimension Number	Cardinalities of Hierarchy Levels		
	Base Data	Level 1	Level 2
1	500	50	5
2	1000	200	50
3	2000	1000	200

Table 4: The schema for the synthetic datasets.

The cardinalities in table 4 represent the number of possible values along each dimension

and hierarchy level. There are four possible “group by” configurations for each dimension: Exclusion, base level, level 1 or level 2. Thus there are $4^3 - 1 = 63$ possible aggregates in this schema (not counting the fact table itself).

We generated data with three distribution models: Normal, uniform and Pareto. The uniform and Pareto distributions were generated with software we wrote. The normal distribution was generated with software available from Jim Gray’s web site (<http://research.microsoft.com/~Gray/DBGen/DBGen.Zip>).

We generated three databases of varying sizes to measure scalability. These three databases were generated with Pareto distributions. The parameter was set at 0.000001 while generating these datasets. This setting generates a close approximation of a Zipf distribution with $\alpha = 1$. We varied the number of rows in the fact table. The smallest database contained 100,000 rows in the fact table. The second contained 500,000. The largest database contained 1,000,000 rows in the fact table.

The databases with the normal distribution and the uniform distribution contain 1,000,000 rows each. The normal distribution was generated with the mean along each dimension at the midpoint of the domain. The standard deviations were $\frac{1}{2}$ the width of the domains.

6 Test results from synthetic data sets

We ran five independent runs on each of the five synthetic datasets, at four different sample sizes. Table 5 shows the results of the scale-up testing. At every setting, each of the 5 runs has an estimate/actual ratio for each of the 63 aggregates. Thus each number in table 5 is a statistic over 315 estimates at the given settings.

The FMS algorithm appears more accurate if you only examine the mean of estimate/actual measurement. However, this measurement can be misleading. The FMS algorithm often underestimates by large percentages, bringing the mean down. The most meaningful measurement here is the coefficient of variation. When a view selection algorithm compares the benefits of materializing different views, the relative benefits are the driving factor in the decision process.

Fact Table Rows Algorithm	Mean of Estimate/Actual				Standard Deviation				Coefficient of Variation			
	Sample Size				Sample Size				Sample Size			
	0.03%	0.10%	0.30%	1.00%	0.03%	0.10%	0.30%	1.00%	0.03%	0.10%	0.30%	1.00%
100,000 Rows												
Pareto Model	1.431	1.238	1.186	1.151	0.664	0.312	0.180	0.137	0.464	0.252	0.151	0.119
FMS	0.948	1.035	1.030	0.990	0.432	0.364	0.221	0.151	0.456	0.351	0.214	0.152
500,000 Rows												
Pareto Model	1.331	1.292	1.243	1.184	0.442	0.310	0.255	0.191	0.332	0.240	0.205	0.161
FMS	1.088	1.037	0.933	0.918	0.407	0.299	0.223	0.176	0.374	0.288	0.239	0.192
1,000,000 Rows												
Pareto Model	1.335	1.285	1.255	1.186	0.425	0.333	0.292	0.210	0.319	0.259	0.233	0.177
FMS	1.089	0.991	0.948	0.893	0.396	0.288	0.230	0.195	0.363	0.290	0.242	0.218

Table 5: Accuracy measurements over scale-up testing.

Fact Table Rows	Sample Size	Elapsed Time (seconds)			Pareto Speedup Ratio
		FMS	Pareto Model	Compute Exact	
100,000	0.03%	1.1	1.1	275	250
	0.10%	1.8	2.2	275	125
	0.30%	3.5	3.5	275	79
	1.00%	10.1	10.3	275	27
500,000	0.03%	2.7	2.0	1611	826
	0.10%	5.8	6.5	1611	248
	0.30%	14.8	15.3	1611	105
	1.00%	43.0	45.2	1611	36
1,000,000	0.03%	7.2	6.9	3588	523
	0.10%	14.4	14.4	3588	249
	0.30%	30.5	31.2	3588	115
	1.00%	87.8	92.8	3588	39

Table 6: Elapsed time measurements over scale-up testing.

Data Distribution Algorithm	Mean of Estimate/Actual				Standard Deviation				Coefficient of Variation			
	Sample Size				Sample Size				Sample Size			
	0.03%	0.10%	0.30%	1.00%	0.03%	0.10%	0.30%	1.00%	0.03%	0.10%	0.30%	1.00%
Normal												
Pareto Model	0.967	0.941	0.920	0.915	0.056	0.071	0.073	0.066	0.058	0.075	0.079	0.073
FMS	0.869	0.939	0.983	1.019	0.211	0.140	0.110	0.114	0.242	0.149	0.112	0.112
Uniform												
Pareto Model	0.958	0.929	0.914	0.913	0.054	0.077	0.079	0.073	0.057	0.083	0.086	0.081
FMS	0.862	0.931	0.974	1.014	0.210	0.140	0.102	0.105	0.243	0.151	0.104	0.104

Table 7: Accuracy measurements over data with normal and uniform distributions.

Reducing the coefficient of variation will lead to better decisions in the selection of materialized views.

The elapsed times of the scale-up testing are shown in table 6. The third and fourth columns show the elapsed times for each algorithm. These times include the sampling, and the estimating for 63 aggregates. The fifth column shows the elapsed time to calculate the exact number of rows for the aggregates from the fact table. The last column shows the speedup obtained by sampling and estimating, versus computing the exact answer.

Table 7 shows the accuracy performance over normal and uniform distributions. The organization of table 7 is similar to table 5. Each number in table 7 is a statistic over 315 estimates at the given settings. Note: There are conditions where the Pareto model does underestimate by small percentages. The reason for this can best be understood in the case of a uniform distribution. Each item in the domain of a uniform distribution has an equal likelihood of occurring. This does not mean the number of occurrences will be equal. If we flip a coin 1000 times, we would be surprised if it came up heads exactly 500 times. When a database is sampled, the randomness of the data produces some item frequencies higher than others. The Pareto algorithm is seeing this as a sign of skew. It should be possible to improve our algorithm further by compensating for this distortion.

7 Conclusions

The Pareto model algorithm has the most accurate performance of the five algorithms we tested. The FMS algorithm was the closest competitor. The coefficient of variation indicates the Pareto model algorithm is on average 30% more accurate than FMS when tested against the real world data, and 38% more accurate when tested across the five synthetic data sets. This increased proportional consistency leads to better decisions when selecting views to materialize, as verified empirically.

The FMS approach frequently underestimates by a large percentage the disk space requirements of aggregates. Underestimating disk space requirements is not a desirable quality. Overestimates are preferable, since they represent a more conservative approach to disk space management. The FMS algorithm underestimated on 37.1% of the aggregates. The average error of these underestimates was 25.1%, measured as $(\text{actual} - \text{estimate})/\text{actual}$. The Pareto model algorithm by contrast only underestimated on 19.8% of the aggregates, by an average of 8.5%. Thus the Pareto model algorithm underestimates about half as often as FMS, and with about 1/3 the amount of error when it does underestimate.

The results indicate the Pareto model algorithm is robust. The coefficient of variation of the Pareto model algorithm was superior to that of FMS in 22 of the 23 database configurations tested. The algorithm performs well under a variety of different data distributions, including

normal, uniform, Pareto and real world distributions.

Lastly, the Pareto model scales well with the size of the fact table. The sampling is done once, and has time complexity $O(n \log(n) + d)$ where n is the number of rows in the sample, and d is the number of rows in the data set. The time complexity of estimating the size of an aggregate is $O(n \log(n))$. We have verified the scalability empirically. Determining the exact number of rows in an aggregate can be extremely time consuming. The approach of sampling the fact table and estimating the number of rows can speed up the process by orders of magnitude.

8 Future work

The results show our Pareto Model Algorithm is robust over different data distributions, and varying database sizes. Thus the value of the algorithm should be applicable in general to other real world data sets. This should be verified empirically by testing over other real world data sets.

One problem that arises when implementing a commercial product is determining how much data constitutes a sufficient sample. This problem is examined in the context of equi-height histograms in [3], and in the context of estimating the number of distinct values in a single column in [2]. One approach examined in [3] is cross-validation. Basically, they begin with a predetermined level of desired accuracy. A small sample is taken, and an initial approximation is made. The process is repeated with a second sample of the same size. The resulting approximations are compared. If the results are within the desired error tolerance, the sampling process is terminated. Otherwise the two existing samples are combined, and another sample equal in size to the existing sample is taken. Thus the sample size is repeatedly doubled until the desired accuracy is reached. This approach should also be applicable in the context of view size estimation for real world data sets, where the distribution is typically unknown. Automatic determination of sample size sufficient for view size estimation is another area for future research.

We are currently examining existing approaches to selecting views for materialization. The number of possible aggregates is exponential

in the number of dimensions. We view this complexity as an opportunity for improvement. One of our goals is to improve the scalability of view selection as the number of dimensions increases. We will then integrate our Pareto model algorithm for view size estimation into a scalable algorithm for view selection.

Once we have integrated our work on view size estimation with view selection, we will address the problems of maintaining materialized views, and query optimization with materialized views. Our final goal is an integrated approach for improving OLAP, based on materialized views.

About the Authors

Thomas P. Nadeau is currently a Ph.D. candidate in the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. He received his B.S. degree in Computer Science (1981) and his M.S. degree in Electrical Engineering and Computer Science (1999) from the University of Michigan, Ann Arbor. His interests include data warehousing, OLAP, data mining, and machine learning. His email address is nadeau@engin.umich.edu, and his home page is <http://www-personal.engin.umich.edu/~nadeau/>

Toby J. Teorey is currently a Professor of Electrical Engineering and Computer Science at the University of Michigan at Ann Arbor. He received the B.S. (1964) and M.S. (1965) degrees in Electrical Engineering from the University of Arizona, Tuscon, and a Ph.D. in Computer Science (1972) from the University of Wisconsin, Madison. He is the author of *Database Modeling and Design* (3rd edition, Morgan Kaufmann, 1999). Professor Teorey's current research interests include database design and data warehousing, OLAP, data mining, and advanced database systems. His email address is teorey@eecs.umich.edu, and his web home page is <http://www.eecs.umich.edu/~teorey/>

References

- [1] A. F. Cardenas. Analysis and Performance of Inverted Database Structures. *Comm. ACM* 13, (May 5, 1975), pp.253 - 264.

- [2] M. Charikar, S. Chaudhuri, R. Motwani, V. Narasayya. Towards Estimation Error Guarantees for Distinct Values. In Proceedings of the 19th ACM SIGACT-SIGMOD-SIGART Symposium on PODS, Dallas, 2000.
- [3] S. Chaudhuri, R. Motwani, V. Narasayya. Random Sampling for Histogram Construction: How much is enough? In *Proc. of 1998 ACM SIGMOD Conference*, pp. 436 – 447, Seattle, 1998.
- [4] M. H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill Book Company, 1970.
- [5] P. Flajolet, G. N. Martin. Probabilistic Counting Algorithms for Database Applications. *Journal of Computer and System Sciences* 31, 1985, pp. 182 - 209.
- [6] C. Faloutsos, Y. Matias, A. Silberschatz. Modeling skewed distributions using multifractal and the ‘80-20 law’. In *Proc. 22nd VLDB Conference*, pp. 307 – 317, Mumbai, 1996.
- [7] V. Harinarayan, A. Rajaraman, J. D. Ullman. Implementing Data Cubes Efficiently. In *Proceedings of 1996 ACM-SIGMOD Conf.*, pp. 205 - 216, Montreal, Canada.
- [8] R. Kimball. *The Data Warehouse Toolkit*. John Wiley & Sons, 1996.
- [9] T. P. Nadeau, K. Runapongsa, T. J. Teorey. Binomial Multifractal Curve Fitting for View Size Estimation in OLAP. In *SCI 2001 Proceedings, Vol . II, Information Systems*, pp. 194 – 199, Orlando, 2001.
- [10] K. Runapongsa, T. P. Nadeau, T. J. Teorey. Storage Estimation for Multidimensional Aggregates in OLAP. In *Proc. 10th CASCON Conference*, pp. 40 – 54, Toronto, 1999.
- [11] A. Shukla, P. M. Deshpande, J. F. Naughton, K. Ramasamy. Storage estimation for multidimensional aggregates in the presence of hierarchies. In *Proc. 22nd VLDB Conference*, pp. 522 – 531, Mumbai, 1996.
- [12] G. K. Zipf. *Human Behavior and Principle of Least Effort: an Introduction to Human Ecology*. Addison Wesley, Cambridge, 1949.