A Pareto Model for OLAP View Size Estimation

Thomas P. Nadeau* **and Toby J. Teorey**
**Computer Science and Engineering Division (CSE), Department
of Electrical Engineering and Computer Science (EECS),
The University of Michigan, 1301 Beal Avenue, Ann Arbor,
MI 48109-2122, USA
E-mail: nadeau@engin.umich.edu
E-mail: teorey@eecs.umich.edu**

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 usu-
ally 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 ag-
gregates in question. We present an original algorithm for esti-
mating the number of rows in an aggregate based on the Pareto
distribution model. We test the Pareto Model Algorithm empiri-
cally against four published algorithms, and conclude the Pareto
Model Algorithm is consistently the best of these algorithms for
estimating view size.*

Key Words. *Pareto distribution, OLAP, view size estimation,
materialized view selection*

1. Motivation

Accumulation of data in industry and organizations has
led to large archives of data in recent years. Quick ac-
cess 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 en-
gineered 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 aggre-
gated. Each attribute of the fact table key is typically a
foreign key matching the primary key of a dimension
table. Fig. 1 illustrates an example of a star schema. The
CustID, *DateID* and *BindID* together make up the pri-
mary key of the *Fact Table*. Thus there are three dimen-
sions. Notice that a dimension can also have a hierarchy.
For example, time can be grouped by *DateID*, *Month*,
Quarter or *Year*.

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 strat-
egy. Pre-aggregation requires the result to be saved to
disk. The number of possible aggregates is exponential
in the number of dimensions. Faced with combinato-
rial explosion and limited disk space, we must decide
which aggregates to calculate in anticipation to queries.
The cost/benefit analysis involves estimating the stor-
age 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 four published
algorithms, and conclude the Pareto Model Algorithm
is the best of these algorithms for estimating view
size.

---

*To whom correspondence should be addressed.

Customer

Fact Table

Calendar

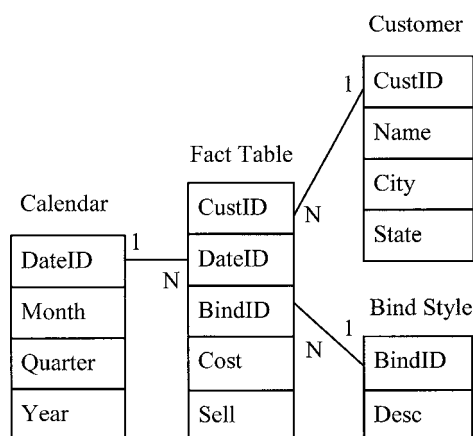| Calendar		Fact Table		Customer	

Fig. 1. A simple star schema.

The remainder of the paper is organized as follows. Section 2 gives a brief overview of related work. Section 3 discusses our preliminary work of testing four published algorithms. 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. The Pareto Model Algorithm is presented in Section 5. Section 6 outlines the synthetic data sets utilized for testing and 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 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 (Cardenas, 1975):

Let n be the number of rows.
Let v be the number of possible values.

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

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 Flajolet and Martin (1985). 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.

Distinct sampling (Gibbons, 2001) augments probabilistic counting with sample data, allowing approximate answers to distinct value range queries. Greater accuracy is reported for distinct sampling when compared to standard sampling techniques, but requires processing the full dataset for each view of interest. The assumption is made that the views of interest are few in number, and are known in advance.

A sampling approach based on the binomial multifractal distribution model is presented in Faloutsos, Matias, and Silberschatz (1996). 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 Shukla et al. (1996). 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 Runapongsa, Nadeau, and Teorey (1999). 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 four published algorithms 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 (Cardenas, 1975), the Sample Frequency Algorithm (Runapongsa, Nadeau, and Teorey, 1999), and the algorithm developed in Faloutsos, Matias, and Silberschatz (1996) 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 (Nadeau, Runapongsa, and Teorey, 2001). 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.

We did not implement the probabilistic counting method (Flajolet and Martin, 1985) 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.

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 5.7×10^{-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.

Table 1. *Attributes of the real world database*

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.)

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. Four algorithms estimated the number of rows in each possible view of the book database. Cardenas' Formula utilizes Eq. (1), and is actually not a sampling approach, hence the measures do not vary with sample size. Sample Frequency attempts to adjust from Cardenas' Formula based on skew detected in a sample (Runapongsa, Nadeau, and Teorey, 1999). FMS takes a sample, finds a good fit to the parameters for a binomial multifractal model based on the number of distinct values in the sample, and the frequency of the most common value. FMS then utilizes Eqs. (2) and (3) which are described in Section 4. The Curve Fitting approach also uses the binomial multifractal model along with Eqs. (2) and (3), but finds a good fit to the parameters based on the number of distinct values in two samples of differing sizes (Nadeau, Runapongsa, and Teorey, 2001). Table 2 indicates the larger of the two sample sizes in the case of Curve Fitting.

We tested at three different sample sizes: 1%, 3% and 10% of the fact table. This is a small database. Larger databases would tend to 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

Table 2. *Preliminary test results on real world data*

	Sample size								
	Mean of estimate/actual			Standard deviation			Coefficient of variation		
Algorithm	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

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/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 mean of estimate/actual. 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. Improving the Distribution Model

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 multifractal models. Fig. 2 illustrates a binomial multifractal distribution tree with a small example. The decision tree depth in Fig. 2 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 Fig. 2 is the probability that a random tuple belongs in that bin. Note the bins group as sets. There
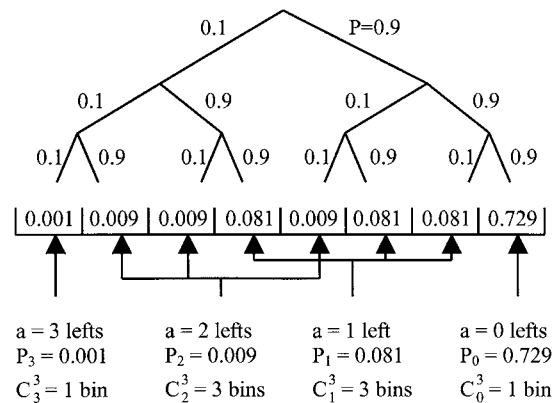
Fig. 2. *Example of a binomial multifractal distribution tree.*

is a relationship between the number of bins in each set, and the elements in Pascal's triangle. The superscript 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.

The theory behind the approach presented in Faloutsos, Matias, and Silberschatz (1996) is that by calculating the parameters of a multifractal 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 Faloutsos, Matias, and Silberschatz (1996) for this purpose.

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

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

Fig. 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 Fig. 2, given 5 rows calculate the expected distinct values:

Expected distinct values

$$= 1 \cdot (1 - (1 - 0.729)^5) + 3 \cdot (1 - (1 - 0.081)^5)$$
$$+ 3 \cdot (1 - (1 - 0.009)^5) + 1 \cdot (1 - (1 - 0.001)^5)$$
$$\approx 1.965 \tag{4}$$

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 Fig. 3.

The FMS algorithm overestimates for larger aggregates and underestimates the size of smaller aggregates. The huge underestimates in the left half of Fig. 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.
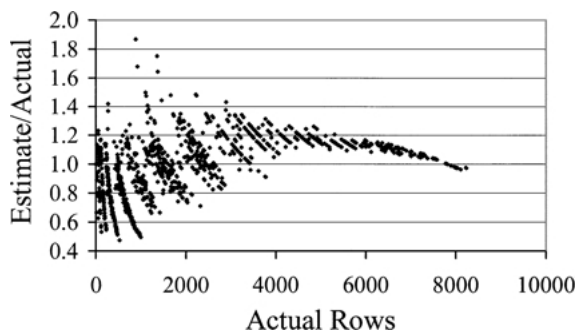
The underlying assumption in the binomial multifractal distribution model is that large-scale structure resembles small-scale structure, and that the skew detected in a sample will continue throughout the unseen items in the domain. Perhaps there is a fundamental flaw in this assumption. Next, we turn 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 multifractal 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

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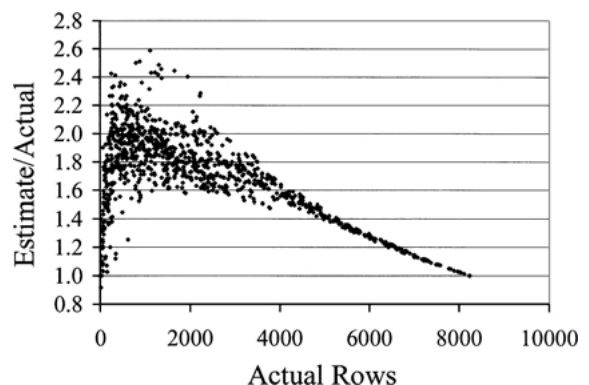

Fig. 4. Scatter plot of the simple algorithm illustrating trends in estimate/actual versus actual rows.

4.3.3. Conclusions on the simple algorithm. Compare Fig. 4 with Fig. 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.

5. 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 (Zipf, 1949). If the items in a Zipf distribution are ordered by descending frequency, the relative frequencies are described by the following equation:

$$f(X_i) = 1/(X_i^\alpha) \qquad (5)$$

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 (DeGroot, 1970, p. 41) is:

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

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 αX_0^α normalizes the curve to give a probability density 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:

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

$[\alpha X_r^\alpha / (X_r^\alpha - 1)]$ is a normalization factor. The resulting area under the curve is 1, as required for a probability density 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 multifractal 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 multifractal 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.

5.1. Pareto model algorithm
The Pareto Model Algorithm is summarized in Fig. 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:

$$\begin{aligned} &\text{select group-by-list, count(*)} \\ &\text{from sample} \\ &\text{group by group-by-list} \\ &\text{order by count(*) desc;} \end{aligned} \qquad (8)$$

If the database is hiererchical, 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
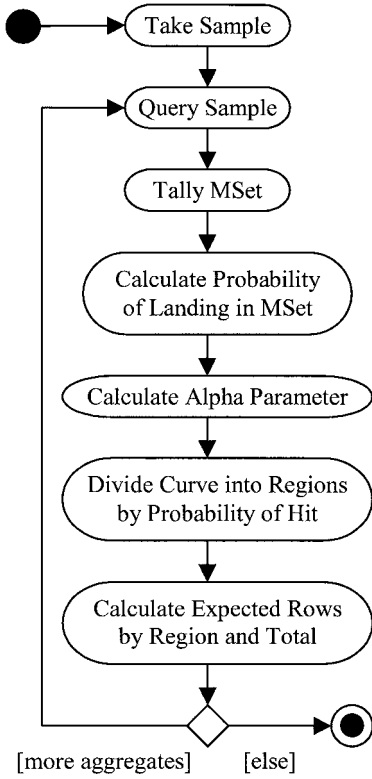
Fig. 5. *Activity diagram of Pareto model algorithm.*

distinct aggregate rows the MSet. Let MRows be the number of rows in the MSet. Let MCount be the number of rows in the sample table that aggregate into the MSet.

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

$$P = \text{MCount}/\text{SampleRows} \qquad (9)$$

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. $j = \text{MRows} + 1$). 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).

$$P = X_r^\alpha (1 - j^\alpha) / j^\alpha (1 - X_r^\alpha) \qquad (10)$$

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.

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 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 $D = |\text{Rows in full data set}|$. From Eq. (7), we derive Eq. (11) for calculating the probability of a hit at X.

$$T = 1 - \left(1 - \left[\alpha X_r^\alpha / (X_r^\alpha - 1)\right]/X^{\alpha-1}\right)^D \qquad (11)$$

Solving for X we have

$$X = \left[\alpha X_r^\alpha / (X_r^\alpha - 1)\right]/\left[(1 - (1 - T)^{1/D})^{1/(\alpha-1)}\right] \qquad (12)$$

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.
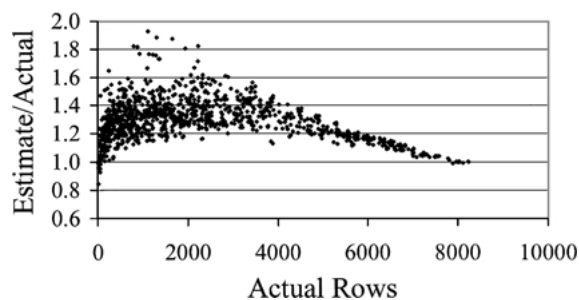
5.2. Results of Pareto model algorithm

Table 3 summarizes the statistical results over the real world data set, comparing our Pareto model algorithm versus FMS (Fig. 6). The organization of Table 3 is similar to that of Table 2, described in Section 3.2.

5.3. 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

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

	Sample size								
	Mean of estimate/actual			Standard deviation			Coefficient of variation		
Algorithm	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

Fig. 6. Scatter plot of the Pareto model algorithm illustrating trends in estimate/actual versus actual rows.
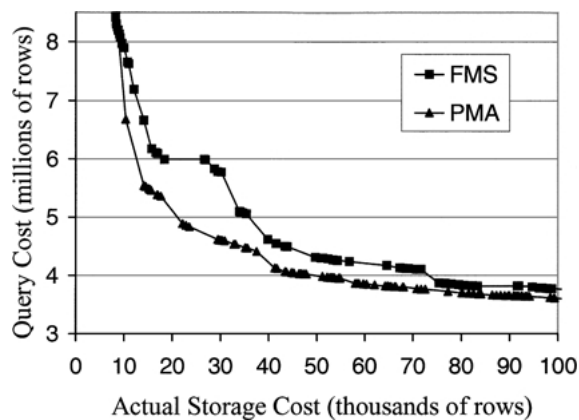
Fig. 7. Improved view selection. Our PMA view size estimator improves the resulting view selection when compared with the FMS view size estimator. Query cost is the total number of rows read if a query is posed against every possible view. This test was implemented with the space limited greedy view selection algorithm from Harinarayan, Rajaraman, and Ullman (1996).

space limited greedy view selection algorithm from Harinarayan, Rajaraman, and Ullman (1996). 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 Fig. 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.

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.

6.   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.

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

Table 4. The schema for the synthetic datasets

	Cardinalities of hierarchy levels		
Dimension number	Base data	Level 1	Level 2
1	500	50	5
2	1000	200	50
3	2000	1000	200

Table 5. Accuracy measurements over scale-up testing

	Sample size											
	Mean of estimate/actual				Standard deviation				Coefficient of variation			
Fact table rows algorithm	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

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 α 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 1/2 the width of the domains.

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. 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 under-estimate 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

Table 6. Elapsed time 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 7. Accuracy measurements over data with normal and uniform distributions

Data distribution algorithm	Sample size											
	Mean of estimate/actual				Standard deviation				Coefficient of variation			
	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

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 (actual − estimate)/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 Chaudhuri, Motwani, and Narasayya (1998), and in the context of estimating the number of distinct values in a single column in Charikar et al. (2000). One approach examined in Chaudhuri, Motwani, and Narasayya (1998) 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 have examined existing approaches to selecting views for materialization. The number of possible views is exponential in the number of dimensions in the database. Existing static view selection algorithms examine every possible view, and are therefore exponential in complexity. We have developed an algorithm for view selection which is polynomial time relative to the number of dimensions. This algorithm improves the scalability of view selection as the number of dimensions increases.

We are beginning to examine alternative OLAP data structures, and their impact on maintaining materialized views. Once we have contributed to the literature on materialized view maintenance, the next problem we will work on is query optimization with materialized views. Our final goal is an integrated approach for improving OLAP optimization, based on materialized views.

References

Cardenas A. Analysis and performance of inverted database structures. *Communications of the ACM* 1975;18(5):253–263.

Charikar M, Chaudhuri S, Motwani R, Narasayya V. Towards estimation error guarantees for distinct values. In: *Proceedings of the Nineteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'00)*, Dallas, 2000;268–279.

Chaudhuri S, Motwani R, Narasayya V. Random sampling for histogram construction: How much is enough? In: *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, Seattle, 1998;436–447.

DeGroot M. *Optimal Statistical Decisions*. McGraw-Hill Book Company, 1970.

Faloutsos C, Matias Y, Silberschatz A. Modeling skewed distributions using multifractals and the '80–20' law. In: *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96)*, Mumbai, 1996;307–317.

Flajolet P, Martin G. Probabilistic counting algorithms for database applications. *Journal of Computer and System Sciences* 1985;31:182–209.

Gibbons P. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In: *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, Roma, 2001;541–550.

Harinarayan V, Rajaraman A, Ullman J. Implementing data cubes efficiently. In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD'96)*, Montreal, 1996;205–216.

Kimball R. *The Data Warehouse Toolkit*. John Wiley, 1996.

Nadeau T, Runapongsa K, Teorey T. Binomial multifractal curve fitting for view size estimation in OLAP. In: *SCI 2001 Proceedings, Vol. II, Information Systems*, Orlando, 2001;194–199.

Nadeau T, Teorey T. A Pareto Model for OLAP view size estimation. In: *Proceedings of CASCON 2001*, Toronto, 2001;1–13.

Runapongsa K, Nadeau T, Teorey T. Storage estimation for multidimensional aggregates in OLAP. In: *Proceedings of CASCON 1999*, Toronto, 1999;40–54.

Shukla A, Deshpande P, Naughton J, Ramasamy K. Storage estimation for multidimensional aggregates in the presence of hierarchies. In: *Proceedings of the 22nd Very Large Data Bases (VLDB'96)*, Mumbai, 1996;522–531.

Zipf G. *Human Behavior and Principle of Least Effort: An Introduction to Human Ecology*. Cambridge: Addison Wesley, 1949.

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, data warehousing, and OLAP. His email address is teorey@eecs.umich.edu, and his web home page is http://www.eecs.umich.edu/~teorey/