Obtaining Quick Results for Approximate Answers*

Qaizar Ali BAMBOAT
Institute of Technical Information Systems, University of Magdeburg
P.O.Box 4120, D-39016 Magdeburg, Germany

and

Oliver DUNEMANN
Institute of Technical Information Systems, University of Magdeburg
P.O.Box 4120, D-39016 Magdeburg, Germany

{qaizar|dunemann}@iti.cs.uni-magdeburg.de

Abstract

In recent times the size of the databases has grown many folds and many of the decision support systems need a very fast and interactive response from these DBMS, but the queries to retrieve the results are at times very complex. In this work we have forwarded a new approach to this problem, by pre-computing the summary statistics in the form of histograms on the data and evaluating the random samples from them in order to answer the given queries quickly, but approximately.
We believe that most of the decision support systems or OLAP applications can tolerate small errors in terms of getting the answers quickly.
This work constitute a part of our larger work which is the development of an ef£cient Incremental data analysis engine.

Keywords: Approximation, Histograms, Sampling, OLAP, Data analysis.

1 Introduction

The amount of data managed by databases like data warehouses for decision support increases rapidly. In modern times we have seen the rise and increase in not only the size but also the need to retrieve the piece of information for the data analysis for scienti£c and business applications. To get the relevant piece of information there is some trade off between accuracy, space and time.

The applications developed for the data analysis during last few years are generally termed as *on-line analytical processing (OLAP)* applications or *decision support systems*. These applications support the analysis of the data and the deduction of interesting trend information. In order to return us the trend information they analyze a large amount of data and also involve complex aggregate queries. So they are expected to return the response result at the earliest but despite the appreciable support from commercial databases for OLAP applications during the last some years, they still require to improve the capability to reduce the response times. In these data analysis applications we often want to get the '£rst' results quickly, e.g., in order to £nd interesting regions of the information space.

In this paper, we are presenting a novel approach that will appreciably reduce the response times to the queries placed to the system and will quickly return approximate answers to the queries. The basic idea is to pre-compute a statistical summary of the actual data and then return the answers based on this summary. As this statistical summary can often £t in the main memory the response time can be reduced signi£cantly. This work forms the part of our main work to develop an ef£cient data analysis engine. The basis of this approach lies on that many OLAP applications can tolerate small errors in query responses in return.

The remainder of this paper is organized as follows. After a short survey of related work, the subsequent section gives a brief introduction to the techniques of histograms and sampling. These are two possible paradigms to get a summary that describes the distribution of values. After this introduction the application of these methodologies to the generation of approximate results is presented. The paper £nishes with the directions of future work and the conclusions.

2 Related Work

In [4] Hellerstein et al. proposed a method for on-line aggregation in which the results of aggregate based queries are re£ned and updated incrementally until the exact answer is obtained. In this approach the answers are provided from the original data which resides on the disk.

Where as to trade off accuracy for the response time a technique was developed by Poosla and Ganti which is based on creating approximate histograms, which summarizes the contents of the data cube and provide approximate answers to any aggregate query from these summaries [15].

Histograms do play an important role in getting the approxi-

*This research is supported by the grant FOR 345/1 from the DFG (German Research Foundation).

mate answers as they are able to gather the statistical summary of the data. The computation and maintenance of histograms was addressed by [3, 6]. The class of v-optimal histograms can be obtained as proposed by Poosla [11] by using a randomized algorithm. This algorithm is applicable independent of the sort and source parameter choice. It is based on the iterative improvement technique, which has been proposed earlier in the context of search studies for query optimization [13, 5].

The authors of [10] and [9] describe different kinds of uniform random sampling techniques in a DBMS. They discuss several techniques for uniform random sampling on relations or on output of relational operators.

One problem concerning sampling in relational DBMS is the placement of the sampling operation in the access plan. The *naive sampling* strategy applies sampling after all other operations have been executed. This in most cases is ineffcient, because the time consuming operations like joins work on all the data even if the desired sample size is relatively small. In [2] further join sample strategies were proposed. These also need statistical information from histograms.

3 Reducing the Data for Fast Query Approximation

In this section two methodologies of reducing the data are introduced. Afterwards the idea of combining them to a compound technique is discussed. The first approach to minimize the amount of data to be considered for retrieving approximate answers is to gather statistics which describe the data distributions. In spite of examining the complete data set these statistics, usually stored in histograms, can be used to estimate the result values. Another technique is to calculate samples of the original data in order to decrease the number of tuples.

3.1 Estimation of Aggregates Based on Histograms

A histogram should reduce the data by describing the data distributions. Assume the attribute X of the relation R is defined upon the domain $\mathcal{D}$ and $\mathcal{V} \subseteq \mathcal{D}$ is the value set of X with all values that are present in R. Furthermore let $\mathcal{V} = \{v_i | 1 \leq i \leq D\}$ with $v_i < v_j$ for all $i < j$. The frequency $f(v_i)$ or f_i of value v_i is the number of tuples $t \in R$ with $t.X = v_i$. The cumulative frequency c_i is the number of tuples $t \in R$ with $t.X \leq v_i$ and accordingly $c_i = \sum_{j=1}^{i} f_i$. A data distribution is then a set of pairs $\mathcal{T} = \{(v_1, f_1), (v_2, f_2), ..., (v_D, f_D)\}$. The cumulative data distribution is $\mathcal{T}^C = \{(v_1, c_1), ..., (v_D, c_D)\}$.

In classic mathematical literature a histogram is often described as a graphical representation of values in rectangles. The value of the variable is the height of rectangle. In this way a mathematical function of the according distribution can be approximated. According to [11] a histogram is constructed by partitioning the data distribution in disjoint subsets called buckets.

In literature are several classes of histograms discussed, which are listed below:

- **Equi-sum(V,S)** histograms are also known as **equi-width**. They group contagious ranges of attributes into buckets. The sum of the spreads of the values in one bucket is approximately equal to $1/\beta$-times the sum of the spreads of all values.
- The class of **equi-sum(V,F)** histograms is the same like **equi-height** or **equi-depth** histograms. The sum of the frequencies of the values in all buckets should be equal. If the frequency F_i of a value v_i is higher than the buckets size, the value v_i will appear in different buckets.
- The **v-optimal** histogram is the histogram with the least variance of all histograms with the same number of buckets. The variance is defined as $V = \sum_{i=1}^{\beta} n_i V_i$ where n_i is the number of values in the i-th bucket ($1 \leq i \leq \beta$) and V_i is the variance of the source values in the i-th bucket.
- Another partition constraint is **MaxDiff**. A bucket boundary is placed between two adjacent source parameter values (in sort parameter order), if the difference between them is one the $\beta - 1$ largest.
- In a **compressed** histogram the n most frequent source values are stored separately in n singleton buckets. The remaining values are divided into $\beta - n$ buckets like proposed the equi-height constraint. There are different methods to compute n.
- **Spline-based** histograms minimize the maximum difference between a source value and the average of the assigned bucket.

Unlike histograms on single attribute, multidimensional histograms are built over multiple attributes in order to capture the joint frequency distributions accurately. Muralikrishna and DeWitt [8] have presented techniques to construct multidimensional histograms effciently. For the scope of this paper we present an example of two dimensional histogram, for the detail study the suggested reading is by [11].

The histogram is built on two attributes of a relation. Let A_0 and A_1 be the attributes of some relation R, the set $D_0 \times D_1$ is partitioned into buckets, which might look like as shown in Fig. 1.

Figure 1: Multidimensional histogram [11]

If histograms are available, they can be used to accelerate aggregate queries like demonstrated in the following simplified example by estimating the aggregate values. Here we use an equi-height histogram to demonstrate the principle. The formulas except the last one are directly applicable to other histogram types.

Suppose we have a relation R with just one attribute a and 20 data sets like shown in Tbl. 1. Let the number of buckets be four. Then Tbl. 2 shows the respective equi-height histogram, where the first value is the smallest attribute value of the base relation:

2	99	23	65	34	93	7	16	31	20
27	10	55	34	87	24	27	41	87	98

Table 1: Example values of attribute a

v_0	v_1	v_2	v_3	v_4
2	23	34	65	99

Table 2: Histogram with four buckets

Each bucket holds £ve values. The upper bounds of the four buckets are 23, 34, 65 and 99. Suppose that the user requests the following information:

```
select avg(a) from R where a <= 50
```

As the average can be deducted from the sum and the count of attributes, we at £rst take a look at theses aggregation functions. The exact count of data sets with an attribute value less or equal than 50 computes to 13. In the above paragraph there is an example of an estimation of the number of attribute values in a given range. The general form of the estimation formula is given in Equation 1 for different types of intervals, where $k_{\min}$ and $k_{\max}$ denote the relative part of the whole bucket size according to $a_{\min}$ and $a_{\max}$ respectively.

$$\begin{aligned} count\,(a) &= \sum_{i=1}^{D} f(v_i) \\ count\,(a \le a_{\max}) &= \sum_{v_i \le a_{\max}} f(v_i) + k_{\max} \\ count\,(a_{\min} \le a) &= k_{\min} + \sum_{v_{i-1} > a_{\min}} f(v_i) \\ count\,(a_{\min} \le a \le a_{\max}) & \\ &= k_{\min} \\ &+ \sum_{\substack{v_{i-1} > a_{\min} \\ v_i \le a_{\max}}} f(v_i) \\ &+ k_{\max} \end{aligned} \tag{1}$$

The de£nition of the relative parts k is shown in Eq. 2:

$$\begin{aligned} k_{\min} &= f(v_{i_{\min}}) \cdot \frac{v_{i_{\min}} - a_{\min}}{v_{i_{\min}} - v_{i_{\min}-1}} \\ &\text{,where } v_{i_{\min}} = \min\{v | v > a_{\min}\} \\ k_{\max} &= f(v_{i_{\max}}) \cdot \frac{a_{\max} - v_{i_{\max}-1}}{v_{i_{\max}} - v_{i_{\max}-1}} \\ &\text{,where } v_{i_{\max}} = \min\{v | v > a_{\max}\} \end{aligned} \tag{2}$$

So, for our example the estimated answer to the query

```
select count(a) from R where a <= 50
```

is calculated as

$$\begin{aligned} count(a \le 50) &\approx \sum_{v_i \le 50} f(v_i) + k_{\max} \\ &= 5 + 5 + 5 \cdot \frac{50 - 34}{65 - 34} \approx 12.6, \end{aligned}$$

while the exact answer is 13.

The next aggregation function to be examined is the sum. The exact sum of all attribute values which are less than or equal to 50 is 326. The estimation formula is provided in Eq. 3 for the different types of intervals:

$$\begin{aligned} sum\,(a) &= \sum_{i=1}^{D} f(v_i) \cdot (v_i - \frac{v_i - v_{i-1}}{2}) \\ &= \sum_{i=1}^{D} f(v_i) \frac{v_i + v_{i-1}}{2} \\ sum\,(a \le a_{\max}) &= \sum_{v_i \le a_{\max}} f(v_i) \frac{v_i + v_{i-1}}{2} \\ &+ k_{\max} \frac{v_{i_{\max}} + v_{i_{\max}-1}}{2} \\ sum\,(a_{\min} \le a) &= k_{\min} \frac{v_{i_{\min}} + a_{\min}}{2} \\ &+ \sum_{v_{i-1} > a_{\min}} f(v_i) \frac{v_i + v_{i-1}}{2} \\ sum\,(a_{\min} \le a \le a_{\max}) & \\ &= k_{\min} \frac{v_{i_{\min}} + a_{\min}}{2} \\ &+ \sum_{\substack{v_{i-1} > a_{\min} \\ v_i \le a_{\max}}} f(v_i) \frac{v_i + v_{i-1}}{2} \\ &+ k_{\max} \frac{v_{i_{\max}} + v_{i_{\max}-1}}{2} \end{aligned} \tag{3}$$

Using this formula for the example a result of 332.7 is estimated as follows:

$$\begin{aligned} &sum(a \le 50) \\ &\approx 5\frac{2+23}{2} + 5\frac{23+34}{2} + 5\frac{50-34}{65-34} \cdot \frac{34+65}{2} \approx 332.7 \end{aligned}$$

As mentioned above the average value now can be estimated by combining the estimated sum of the attribute values and the count of data sets:

$$avg = \frac{sum}{count} \tag{4}$$

For the initial example query this value computes to 26.4 while the exact value is approximately 25.1. The formulas out of Eq. 1 through 3 can be simpli£ed, if the buckets of the histogram are equi-height like in the example discussed before. Here just the estimation formula for the sum is given. Let f be the frequency of all buckets. Then the sum of the values of a given range can be estimated by:

$$\begin{aligned} &sum\,(a_{\min} \le a \le a_{\max}) \\ &= \frac{f}{2} \left(\frac{v_{i_{\min}}^2 - a_{\min}^2}{v_{i_{\min}} - v_{i_{\min}-1}} \right. \\ &+ \sum_{v_{i-1} > a_{\min} \wedge v_i \le a_{\max}} (v_i + v_{i-1}) \\ &\left. + \frac{(a_{\max} - v_{\max-1})\,(v_{\max} + v_{\max-1})}{v_{\max} - v_{\max-1}} \right) \end{aligned} \tag{5}$$

In some cases only retrieving a single aggregate value may not be enough for the desired analysis purposes. Theses are typical applications of sampling methods.

3.2 Estimation of Samples Based on Histograms

Sampling techniques reduce the data by picking out a given number of data sets from the total volume. Because in database systems there usually is no random access to the data, sequential sampling algorithms have to be deployed. The algorithms differ widely whether the number of initial data sets has to be known or not. If no cardinality information is available sampling with reservoir [7] is necessary. These algorithms do not need the initial size, but provide the £rst tuples only after the complete scan of the complete data set. In contrast non-blocking algorithms like described in [14] can be used, if the number of data sets is known and direct access to them is possible.

In this work we propose a novel approach of estimating samples. As samples are usually used to gain a cloudy picture of the real data, in many cases an estimated sample should be suf£cient. This is also at £rst discussed on the simple one-dimensional example introduced in Section 3.1. Let the query be:[1]

```
select a from R
where a <= 50
limit sample 5
```

If the user only wants to know how the selected rows are distributed in their domain, it is not that important, that he gets £ve in fact existing rows out of R. The desired information is in which areas the attribute values are.

In the example we have three histogram's buckets to consider. The £rst two intervals $[2; 23]$ and $[23; 34]$ contain 5 values each, while the third interval $[34; 50]$ contains approximately $k_{\max}$ values. So, the probability P, that a value less than 50 resides in the according buckets is:

$$P(2 \leq a \leq 23) \approx \frac{5}{5+5+2.58} \approx 39.75\%$$
$$P(23 \leq a \leq 34) \approx \frac{5}{12.58} \approx 39.75\%$$
$$P(34 \leq a \leq 50) \approx \frac{2.58}{12.58} \approx 20.5\%$$

The probability that a value of an interval is part of the base relation is computed by the number of elements divided by the width of the interval. So the probabilities, that a single value is a result of the initial query compute as:

$$P(a = 20) \approx \frac{5}{12.58} \cdot \frac{1}{23-2+1} \approx 1.8\%$$
$$P(a = 30) \approx \frac{5}{12.58} \cdot \frac{1}{34-23+1} \approx 3.3\%$$
$$P(a = 40) \approx \frac{2.58}{12.58} \cdot \frac{1}{50-34+1} \approx 1.2\%$$

[1]The 'limit sample' clause is one example of a SQL extension for sampling and is part of the FRAQL query language [12].

Having these equations the cumulative probability $P(a \leq x)$ can derived. Now it is just necessary to generate £ve uniform distributed random numbers and choose the according values from the cumulative probability function as results. For example let the £rst generated number be 0.47. The according attribute value is 25, because of $P(\hat{a} = 25) \approx 22 \cdot 0.018 + 2 \cdot 0.033 = 0.462$. So this can be returned as the approximate result to the user.

The generalized formula of the above exemplarily outlined computation is shown in Eq. 6:

$$\begin{aligned}
&P(a = \hat{a}) \\
&\quad \approx \frac{k_{\min}}{count\,(a_{\min} \leq a \leq a_{\max})\,(v_{i_{\min}} - a_{\min} + 1)}, \\
&\quad \text{if} \quad a_{\min} \leq \hat{a} \leq v_{i_{\min}}. \\
&P(a = \hat{a}) \\
&\quad \approx \frac{f(v_i)}{count\,(a_{\min} \leq a \leq a_{\max})\,(v_{i_{\min}} - v_{i_{\min}-1} + 1)}, \\
&\quad \text{if} \quad v_{i_{\min}-1} \leq \hat{a} \leq v_{i_{\min}}. \\
&P(a = \hat{a}) \\
&\quad \approx \frac{k_{\max}}{count\,(a_{\min} \leq a \leq a_{\max})\,(a_{\max} - v_{i_{\max}} + 1)}, \\
&\quad \text{if} \quad v_{i_{\max}} \leq \hat{a} \leq a_{\max}.
\end{aligned} \tag{6}$$

The cumulative form is obtained by adding the probabilities for all values less or equal the searched parameter. With this function sample values can quickly be estimated. These values are not necessarily part of the underlying base relations. They only serve to sketch the real data distribution.

If more than one attribute is selected for the result set, one-dimensional histograms are not longer suf£cient. This is shown by a small example. Let the rows in the base table be like shown in Tbl. 3: Relying on one-dimensional histograms would lead to

a	b
1	0
1	0
0	1
0	1

Table 3: Example values of two attributes a and b

the estimation, that for each attribute a and b there are the same number of 1 and 0. For that reason sample rows like $\langle 1, 0\rangle$ would have the same probability to be generated as $\langle 1, 1\rangle$, which is not part of the base relation. Here the use of the multi-dimensional histograms mentioned above is necessary.

4 Future Directions

For the future work we suggest that work can be carried in the following directions. The £rst point is to check the quality of the estimated results with different data base sizes and histogram types. If necessary the error in estimations should be brought down and the con£dence level of the results returned should be increased. If possible the histogram type and the number of

buckets should be derivable from the desired con£dence level and the amount of data in the base relations.

Next the area of multi-dimensional histograms should be examined in detail.

This work is the part of our larger work on developing an ef£cient incremental data analysis engine so we believe that there should be an implementation such that it satis£es the working on the varied type of the real data. Here the aspect of not restricting the analysis only to numerical data has to be considered.

5 Conclusions

In this paper a method was discussed, which provides fast approximate answers to the aggregate and sample queries. To achieve this aim the demand for exactness was relaxed. Nevertheless, the results should feature a certain quality level. The approach is to construct histograms over the attributes of the base relations. The size of these histograms can be chosen, so that they can £t in the main memory. There is a set of equations given, which calculate results to aggregate and sample queries just by using these histograms without accessing the base relations.

References

[1] Haran Boral and Per-Åke Larson, editors. *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, June 1-3, 1988*. ACM Press, 1988.

[2] S. Chaudhuri, R. Motwani, and V.R. Narasayya. On Random Sampling over Joins. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania,USA*, pages 263–274. ACM Press, 1999.

[3] P.B. Gibbons, Y. Matias, and V.Poosla. Fast Incremental Maintenance of Approximate Histograms. *In Proc. of the 23rd. Intl. Conf. on Very Large Databases*, August 1997.

[4] Joseph M. Hellerstein, Peter J. Haas, and Helen Wang. Online Aggregation. In Joan Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 171–182. ACM Press, 1997.

[5] Yannis E. Ioannidis and Eugene Wong. Query Optimization by Simulated Annealing. In Umeshwar Dayal and Irving L. Traiger, editors, *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference, San Francisco, California, May 27-29, 1987*, pages 9–22. ACM Press, 1987.

[6] H.V. Jagadish, N. Koudas, S. Muthukrishnan, V.Poosla, K. Sevcik, and T. Suel. Optimal Histograms With Quality Guarantees. *Proc. of ACM SIGMOD Conf*, 1998.

[7] Kim-Hung Li. Reservoir-sampling algorithms of time complexity O(n(1 + log(N/n))). *ACM Transactions on Mathematical Software*, 20(4):481–493, December 1994.

[8] M. Muralikrishna and David J. DeWitt. Equi-Depth Histograms For Estimating Selectivity Factors For Multi-Dimensional Queries. In Boral and Larson [1], pages 28–36.

[9] Frank Olken. *Random Sampling from Databases*. PhD thesis, UC Berkeley, 1993.

[10] Frank Olken and Doron Rotem. Simple Random Sampling from Relational Databases. In Wesley W. Chu, Georges Gardarin, Setsuo Ohsuga, and Yahiko Kambayashi, editors, *VLDB'86 Twelfth International Conference on Very Large Data Bases,August 25-28, 1986, Kyoto, Japan, Proceedings*, pages 160–169. Morgan Kaufmann, 1986.

[11] Viswanath Poosala. *Histogram-based Estimation Techniques in Database Systems*. PhD thesis, University of Wisconsin-Madison, 1997.

[12] K. Sattler, S. Conrad, and G. Saake. Adding Con¤ict Resolution Features to a Query Language for Database Federations. In M. Roantree, W. Hasselbring, and S. Conrad, editors, *Proc. 3nd Int. Workshop on Engineering Federated Information Systems, EFIS'00, Dublin, Ireland, June*, pages 41–52, Berlin, 2000. Akadem. Verlagsgesellschaft.

[13] Arun N. Swami and Anoop Gupta. Optimization of Large Join Queries. In Boral and Larson [1], pages 8–17.

[14] Jeffrey Scott Vitter. An Ef£cient Algorithm for Sequential Random Sampling. *ACM Transactions on Mathematical Software*, 13(1):58–67, March 1987.

[15] V.Poosla and V. Ganti. Fast Approximate Answers to Aggregate Queries on a Data Cube. *In Proc. of the Intl. Conf. on Scienti£c and Statistical Database Management*, 1999.