

## Integrating Query-Feedback Based Statistics into Informix Dynamic Server

Alexander Behm <sup>1</sup>, Volker Markl <sup>2</sup>, Peter Haas <sup>3</sup>, Keshava Murthy <sup>4</sup>

<sup>1</sup> Berufssakademie Stuttgart / IBM Germany  
alexbehm@gmx.de

<sup>2</sup> IBM Almaden Research Center  
marklv@us.ibm.com

<sup>3</sup> IBM Almaden Research Center  
peterh@almaden.ibm.com

<sup>4</sup> IBM Menlo Park  
rkeshav@us.ibm.com

**Abstract:** Statistics that accurately describe the distribution of data values in the columns of relational tables are essential for effective query optimization in a database management system. Manually maintaining such statistics in the face of changing data is difficult and can lead to suboptimal query performance and high administration costs. In this paper, we describe a method and prototype implementation for automatically maintaining high quality single-column statistics, as used by the optimizer in IBM Informix Dynamic Server (IDS). Our method both refines and extends the ISOMER algorithm of Srivastava et al. for maintaining a multidimensional histogram based on query feedback (QF). Like ISOMER, our new method is based on the maximum entropy (ME) principle, and therefore incorporates information about the data distribution in a principled and consistent manner. However, because IDS only needs to maintain one-dimensional histograms, we can simplify the ISOMER algorithm in several ways, significantly speeding up performance. First, we replace the expensive STHoles data structure used by ISOMER with a simple binning scheme, using a sweep-line algorithm to determine bin boundaries. Next, we use an efficient method for incorporating new QF into the histogram; the idea is to aggregate, prior to the ME computation, those bins that do not overlap with the new feedback records. Finally, we introduce a fast pruning method to ensure that the number of bins in the frequency distribution stays below a specified upper bound. Besides refining ISOMER to deal efficiently with one-dimensional histograms, we extend previous work by combining the reactive QF approach with a proactive sampling approach. Sampling is triggered whenever (as determined from QF records) actual and estimated selectivities diverge to an unacceptably large degree. Our combined proactive/reactive approach greatly improves the robustness of the estimation mechanism, ensuring very high quality selectivity estimates for queries falling inside the range of available feedback while guaranteeing reasonably good estimates for queries outside of the range. By automatically updating statistics, query execution is improved due to better selectivity estimates, and the total cost of ownership (TCO) is reduced since the database administrator need not update statistics manually for monitored columns.

## 1 Introduction

In today's database market performance and the total cost of ownership (TCO) are two important factors that provide a competitive advantage. Performance is dependant on the query optimizer's choices of physical execution plans, which heavily rely on the accuracy of available statistics [IC91, Lyn88]. The total cost of ownership is influenced by the administration cost, among others. A superior performance and TCO will provide a significant competitive advantage. Traditionally, statistics are updated manually by the database administrator (DBA) at intervals according to vendor-recommendations, based on experience or in response to bad performance. On the one hand, this approach requires expertise and time of the DBA which may increase the total cost of administration. On the other hand, manually updating statistics at certain intervals may result in either too frequent gathering of statistics (which is expensive and may impact the processing of user queries) or too infrequent gathering of statistics, resulting in suboptimal query plans due to inaccurate selectivity estimates. So far, there exists no optimal solution to determine when to update the statistics for which tables and columns. For tackling this problem we propose a method for automatically and autonomously gathering statistics. It ensures accurate statistics because they automatically evolve with changes in the database and it minimizes the need for manual intervention by the DBA which reduces costs. The general idea of feedback based statistics as described in [SLMK01, CR94] suggests exploiting results of actual queries that run on the database, creating a feedback-loop to improve future selectivity estimates by "learning" from previous queries. Embodiments of this principle may include suggesting tables and columns for which to update statistics (e.g. based on a significant discrepancy between estimated and actual cardinalities), suggesting statistical parameters such as the number of frequent values and quantiles to gather [Beh05], and creating multi-dimensional histograms (see chapter 2), and others.

In this work we present a query-feedback based method for creating and maintaining single-column statistics in IBM Informix Dynamic Server. Our work is a specialisation of ISOMER [SHM<sup>+</sup>06] to the one dimensional-case which allows for performance improvements (see 2). We gather a sampling-based distribution for a given column and continuously refine it using QF. The process of refinement consists of eliminating inconsistencies based on a timestamp (recent information is preferred) and creating a maximum entropy (ME) cumulative frequency distribution. This distribution is consistent with all available information (QF, previous sampling information) and uniform in the absence of information (see chapter 6). In contrast to the work described in [Beh05], we do not use the ME distribution for recommending statistical parameters but make this distribution available to the optimizer directly for selectivity estimation. Moreover, this approach offers flexibility when thinking of cases in which feedback only delivers information on small regions in the column domain. During the course of time the accuracy of statistics will degrade as the data in the column changes. Hence, there is a good chance that queries not covered by feedback will yield a large discrepancy between estimated and actual cardinalities. If a significant discrepancy were to be detected one might schedule an updating of statistics via sampling. So, in between sampling procedures the QF is used to continuously refine the distribution until a said discrepancy is detected. In case feedback continuously delivers information on the whole column domain it may not be necessary to sample the column

ever again. Apart from this flexibility, our hybrid approach is reasonable in terms of computational costs. Sampling a column is very Input/Output (I/O) intensive since the actual data is tapped. Refining an existing distribution with the said method is very CPU intensive, but very low on I/O costs because only a very low number of records have to be read (the feedback records). As a rule of thumb, in terms of total costs I/O is nine times more expensive than CPU cycles. Therefore, the refinement process is less expensive by orders of magnitude and at the same time a lot is to be gained due to better execution plans.

## 2 Related Work

Single and multi-dimensional statistics are traditionally created proactively using techniques involving data-scans as described in [PI97, GKVD00, DGR01, MD88, TGIK02] or sampling techniques as in [CMN98]. On the one hand these approaches are costly and scale poorly with growing table sizes or require large sample sizes, respectively. On the other hand these approaches disregard the fact that a user's workload may not require high accuracy in all parts of the domain. So, a reactive method may prove to be more feasible because learning from previous queries ensures high accuracy in often queried regions of the domain and scanning the actual data becomes obsolete or less frequently required. The idea of using QF for creating statistics needed for selectivity estimation is known since [CR94]. Various methods have been proposed to reactively improve selectivity estimates [CR94, SLMK01] but we will focus on the construction of histograms. This work specializes the method described in ISOMER [SHM<sup>+</sup>06], which consistently creates multi-dimensional histograms based on QF, to the case of one-dimensional histograms. Previous work [BCG01, PI97, AC99, LWV03] in the area of feedback based multi-dimensional histograms lacked the feature of consistently incorporating new feedback, thus requiring heuristics to determine which feedback to keep. This reduces the overall accuracy of the histogram. ISOMER is a full-blown solution for feedback based statistics that uses the STHoles datastructure [BCG01] which is superior to the one in MHist [PI97] but even so, imposes an unnecessary overhead for the case of one-dimensional histograms. The one-dimensional case allows for optimization. We use a sweep-line algorithm to determine the atoms (bins) for the ME histogram instead of iterating through all levels in the STHoles structure. In order to meet storage requirements we use a one-pass pruning algorithm which does not need to incrementally recompute the ME solution as is the case in ISOMER. Furthermore, for adding new feedback to the histogram we use an efficient refinement algorithm based on the principle of maximum entropy. In essence, we propose a QF based method for consistently creating and refining one-dimensional histograms based on the ME principle. This method is similar to ISOMER but focuses on the one dimensional case which allows for improvements (compared to ISOMER). This effort is novel in the sense that we are not aware of any previous work specializing in QF based one-dimensional histograms.

### 3 Background

#### 3.1 Informix Dynamic Server (IDS)

IBM Informix Dynamic Server (IDS) is a light-weight relational database management system that focuses on online transaction processing. It uses a cost-based optimizer for choosing an efficient physical execution plan. Currently, only single-column statistics are used for selectivity estimation. These statistics are gathered proactively by sampling. So, we wish to enhance selectivity estimates by the use of QF. Since only single-column statistics are currently used we expect a big impact by improving their accuracy.

#### 3.2 Distributions in Informix Dynamic Server

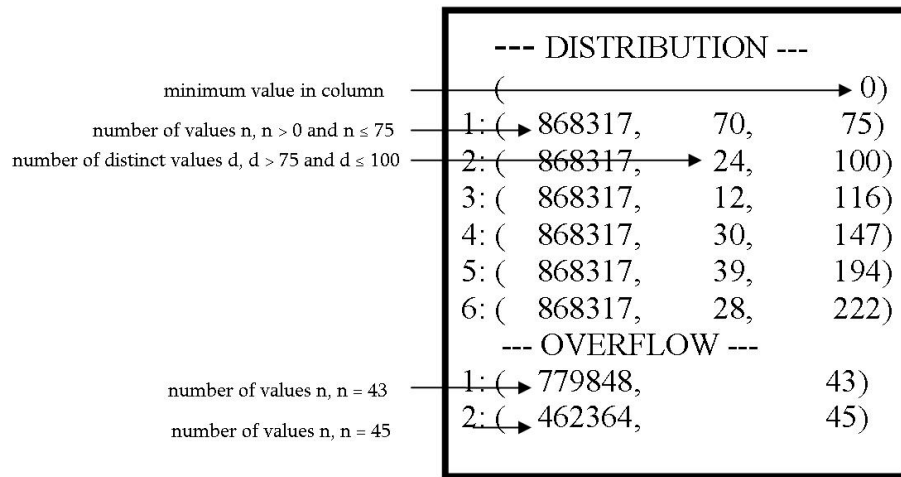


Figure 1: Single-column, sampling-based distribution in IDS

#### 3.3 Updating Statistics in Informix Dynamic Server

Apart from data distributions IDS gathers other statistical information on tables, columns and indexes which are useful to the IDS optimizer when choosing an execution plan. The “UPDATE STATISTICS” command is used to update statistics. This command offers three basic modes of operation: UPDATE STATISTICS [LOW—MEDIUM—HIGH] We add a new mode of UPDATE STATISTICS for enabling the feedback based updating of statistics described in this work:

FBS (feedback-based statistics)

1. monitor queries on specified column gathering estimated and actual cardinalities
2. create/refine maximum-entropy distribution at given condition (e.g. update-delete-insert counter or the number of feedback records reach a certain threshold)
3. use maximum-entropy distribution for selectivity estimation in optimizer

### 3.4 Representing Feedback/Distribution Information

In order to ultimately achieve a ME frequency distribution we need to define a common way of representing available information on the respective column. The information retrieved from the Query-Feedback Warehouse (QFW), from the sampling-based distribution and from an existing ME distribution is represented as a set of triples:  $I = \{(l_1, u_1, f_1), (l_2, u_2, f_2), \dots, (l_n, u_n, f_n)\}$  where for  $n$  pieces of information  $i \in \{1, 2, 3, \dots, n\}$   $l_i$  is the lower boundary of the interval,  $u_i$  is the upper boundary of the interval and  $f_i$  is the relative frequency of occurrence of values in  $(l_i, u_i]$ .

For example consider the following SQL-Query:

`SELECT COUNT(*) FROM TAB WHERE COL > A and COL ≤ B`

With  $|TAB|$  as the table cardinality,  $C$  as the returned count and  $F = \frac{C}{|TAB|}$  we represent the result of this query with the triple  $(A, B, F)$ .

Now that all information is in a standard format we can start preparing constraints for the ME distribution. Since some intervals may overlap or we may lack information on some parts of the column domain we need to find a partitioning of the column domain such that no bins in the partitioning overlap. We refer to “bins” as mutually disjoint intervals that partition the whole column domain.

Consider the following example:

Let there be a set of triples  $I = \{(l_1, u_1, f_1), (l_2, u_2, f_2), (l_3, u_3, f_3)\}$  which we can represent graphically by the following diagram:

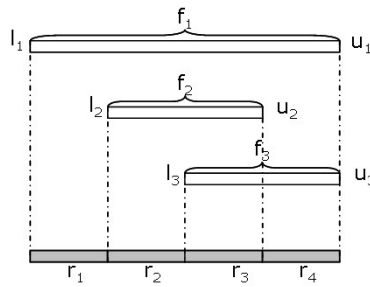


Figure 2: Graphical representation of a set of triples and resulting bins

The set of non-overlapping triples  $\{r_1, r_2, r_3, r_4\}$  is the refinement of the set of overlapping triples. It is the smallest set of mutually disjoint intervals, which we call “bins”, such that each interval in  $I$  can be expressed as a finite union of bins. We distinguish non-overlapping triples from overlapping triples by representing each non-overlapping one as triple of the form  $(a, b, m)$  equivalent to  $(l, u, f)$  for intervals. We denote the set of bins by  $R = \{r_1 = (a_1, b_1, m_1), r_2 = (a_2, b_2, m_2), \dots, r_k = (a_k, b_k, m_k)\}$ , where for  $k$  number of bins  $i \in \{1, 2, 3, \dots, k\}$   $a_i$  is the lower boundary of the bin,  $b_i$  is the upper boundary of the bin and  $m_i$  is the relative frequency of occurrence of column-values in  $(a_i, b_i]$ . Looking at Figure 2 one can easily derive a system of linear equations describing the frequencies of our intervals as a sum of frequencies of bins. In this example the system of linear equations looks like:

$$f_1 = m_1 + m_2 + m_3 + m_4$$

$$f_2 = m_2 + m_3$$

$$f_3 = m_3 + m_4$$

We call the set of linear equations “constraints” and denote the set of constraints by  $T = \{t_1, t_2, t_3, \dots, t_n\}$ , each element in  $T$  corresponding to a linear equation of the form  $f_n = \delta_1 m_1 + \delta_2 m_2 + \delta_3 m_3 + \dots + \delta_k m_k$  with  $\delta_1, \dots, \delta_k \in \{0, 1\}$ . Note that for a relative frequency distribution all frequencies must add up to one. This is why we always add the interval  $(l_1, u_1, 1.0)$  to  $I$ , where  $l_1$  is the minimum value in the column domain and  $u_1$  is the maximum value in the column domain. The resulting constraint we add to  $T$  looks like this:  $f_1 = m_1 + m_2 + m_3 + \dots + m_k = 1.0$ .

### 3.5 The Maximum Entropy Principle

Our knowledge on a given column as presented in the previous sections may, on the one hand, be incomplete due to lack of information in some parts of the column domain and, on the other hand, some intervals may overlap. We seek to find a set of  $k$  bins with frequencies  $\{m_0, m_1, \dots, m_k\}$  that are consistent with all available information. There exist many sets of bins satisfying all constraints because frequencies of bins on which no information is present can be arbitrarily chosen (as long as the sum of all frequencies equals one). However, we seek exactly the set of bins which is uniform in the absence of information. The principle of maximum entropy delivers a solution to this problem:

“Model everything that is known and assume nothing about the unknown.”  
[GS85]

Previous work using this principle includes [MMK<sup>+</sup>05] in which consistent estimates for selectivities of conjuncts of predicates are created in the presence of incomplete, overlapping knowledge. ISOMER [SHM<sup>+</sup>06] uses the principle to consolidate information retrieved from QF. The latter applies to this work except that we focus on the one-dimensional case. Hence, the entropy we need to maximize is similar to the one in ISOMER.

ISOMER defines the following entropy:

$$H(R) = - \sum_{i=1}^k m_i \ln \left( \frac{m_i}{h_i} \right) \quad (1)$$

where  $h_i$  denotes the volume of bin (bucket)  $i$

Since we focus on the one-dimensional case  $h_i$  is simply the length of bin  $i$  ( $u_i - l_i$  for integers).

### 3.6 Inconsistency and Implied Zero Removal

**Inconsistency Removal:** Query feedback is gathered over time and hence is a time series of snapshots giving insight into the actual distribution of data within a given column at a given time. Changes to the respective column occur all the time due to normal database usage. This is the reason why inconsistencies between feedback records or between feedback records and an existing distribution occur. The ME principle cannot be applied to an inconsistent system because an optimal solution does not exist. The problem of inconsistency elimination has been addressed in [KHM<sup>+</sup>06, MHK<sup>+</sup>06] and we follow the linear programming solution approach described therein.

**Implied Zero Removal:** As recognized in [KHM<sup>+</sup>06, MHK<sup>+</sup>06] bins having an implied frequency of zero pose a threat to solving the ME problem efficiently. The iterative scaling algorithm will try to push the frequency of said bins to zero requiring a high number of expensive iterations to converge, assigning a frequency unequal to zero to these bins (the logarithm of zero is undefined). The latter introduces an estimation error which may exceed acceptable limits if many implied zeros exist. In order to ensure that the iterative scaling algorithm converges fast and to eliminate the said error we need to detect and remove implied zero bins. For removing implied zero bins we pursue the approximate zero detection approach described in [KHM<sup>+</sup>06, MHK<sup>+</sup>06].

### 3.7 The Iterative Scaling Algorithm

The constrained optimization problem induced by the ME principle can be numerically approximated efficiently using the iterative scaling algorithm [DR72] or variations thereof [MMK<sup>+</sup>05]. The algorithm determines the bin frequencies without violating the constraints in  $T$  while adhering to the ME principle.

## 4 Feedback-Based Statistics Maintenance in IDS

### 4.1 Overview: Maintaining Statistics

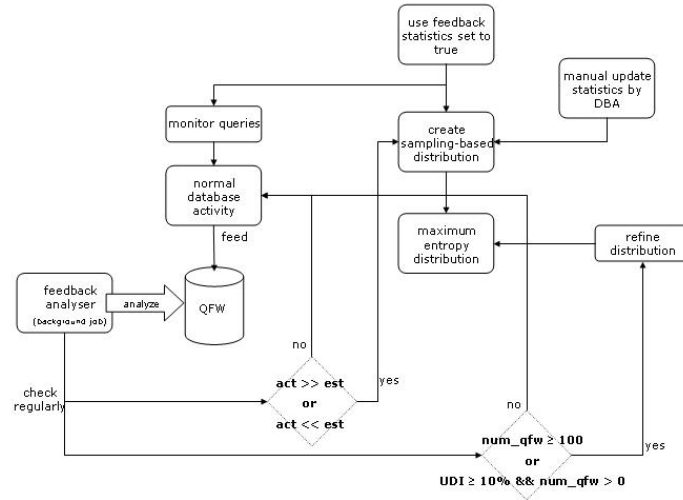


Figure 3: Informix Dynamic Server architecture for automatically maintaining statistics

#### Glossary:

act	actual cardinality
est	estimated cardinality
num_qf	number of query feedback records for a given column
UDI	Update, Delete, Insert counter for a given column
DBA	database administrator
QFW	query feedback warehouse, stores estimated and actual cardinalities of intermediate results of queries

### 4.2 Determining the Bin Boundaries

A sweep-line algorithm as described in [JP82] is an efficient method for intersecting geometric figures. We use our own, specific implementation for determining the boundaries and lengths of each bin constructing the constraint matrix on-the-fly. The constraint matrix stores the system of linear equations column-wise and is an internal representation for the constraints. A detailed description of the sweep-line algorithm used in this work can be found in [Beh05]. In summary this is how the sweep-line algorithm works: First we need



the set of intervals sorted by lower boundary. The sweep-line traverses from lower boundary to lower boundary. In every iteration (or “sweep-event”) the current interval is added to the “sweep-event structure”. In our case the sweep-event structure is a heap having the interval with the lowest upper boundary as head. Now we distinguish several cases for finding the bins. Each time a bin is found the constraint matrix is updated accordingly. An interval is deleted from the sweep-event structure only if its upper boundary has been used to add a bin.

Keep in mind that we are ultimately interested in the frequencies of all bins. The sweep-line algorithm performs the first step to achieving this, namely finding the boundaries and length of each bin. The frequencies are still to be determined up to this point.

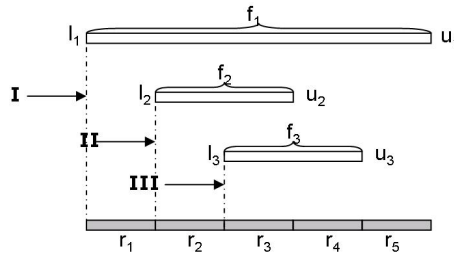


Figure 4: Example of sweep line algorithm showing iterations

Figure 4 shows a set of intervals  $\{(l_1, u_1, f_1), (l_2, u_2, f_2), (l_2, u_2, f_2)\}$  and a set of bins  $\{r_1, r_2, r_3, r_4, r_5\}$  whose boundaries and lengths are to be determined by the sweep-line algorithm. I, II and III denote the iterations (or sweep-events) of the algorithm. During these iterations bins  $\{r_1, r_2, r_3\}$  are recognized. Afterwards, all intervals are on the sweep-event structure (heap). The heap is “cleaned” and in this process bins  $\{r_4, r_5\}$  are found. The difference between the “cleaning” and the normal iterations is that the “cleaning” recognizes bins that have an upper boundary of an interval as lower boundary. Normal iterations recognize bins having a lower boundary that is a lower boundary of an interval. Additionally, the sweep-line algorithm constructs the following matrix that represents our constraints (the system of linear equations):

	$f_1$	$f_2$	$f_3$
$m_1$	1	0	0
$m_2$	1	1	0
$m_3$	1	1	1
$m_4$	1	0	1
$m_5$	1	0	0

Figure 5: Constraint matrix resulting from intervals in figure 4

### 4.3 Determining the Frequencies with Maximum Entropy

**Resolving Conflicts:** Consider the example given in figure 4. Assume that the length of interval 1 and 2 are the same. Further, let  $f_2 = 0.4$  and  $f_3 = 0.2$ . We can try to determine the frequencies  $\{m_2, m_3, m_4\}$  algebraically:

$$m_2 + m_3 = 0.4 \quad (2)$$

$$m_3 + m_4 = 0.2 \quad (3)$$

replacing  $m_3$  in equation 2 with  $0.2 - m_4$  yields:

$$m_2 - m_4 = 0.2 \quad (4)$$

We see there are many possibilities to assign frequencies to the bins without violating any known information. However, we do not want to imply any knowledge and hence we seek the set of frequencies which does not violate any information, but is as uniform as possible. In other words, we seek the frequencies which maximize the entropy, namely  $m_2 = 0.3, m_3 = 0.1, m_4 = 0.1$ .

**Filling Gaps:** On some parts of the column domain we may not have any information because there is no QF covering them. As seen in figure 4 we have no QF giving insight into the frequencies  $\{m_1, m_5\}$ . Again, we use the principle of maximum entropy to assign frequencies to these “gaps”. Assuming that the lengths of bins  $\{r_1, r_5\}$  are identical, we can easily compute the frequencies having the maximum entropy:

$$m_1 + 0.3 + 0.1 + 0.1 + m_5 = 1 \quad (5)$$

using the principle of maximum entropy we conclude  $m_1 = m_5 = 0.25$ .

The iterative scaling algorithm will deliver exactly the desired solution.

### 4.4 Meeting a Space Budget - Pruning

The number of bins in a distribution grows as the distribution continuously is refined with new feedback information. Since we cannot spend an infinite amount of memory for a distribution and the costs for refining a distribution grow exponentially with an increasing number of intervals (see section 5.1) we seek a method for reducing the number of bins. In general, the pruning mechanism could be based either on a tolerable error or based on a predefined maximum number of bins. We choose to pursue the latter approach since it requires less implementation effort and allows for precise control over storage requirements.

Let  $max$  be the maximum number of bins for a column and  $err(r_x, r_{x+1})$  be the resulting error when merging bins  $x$  and  $x + 1$ . Consider the following example:

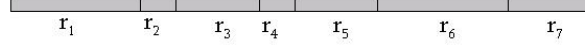


Figure 6: Set of bins before pruning

$$\begin{aligned}
 h_1 &= 4 & m_1 &= 0.2 \\
 h_2 &= 2 & m_2 &= 0.1 \\
 h_3 &= 3 & m_3 &= 0.1 \\
 h_4 &= 2 & m_4 &= 0.2 \\
 h_5 &= 3 & m_5 &= 0.3 \\
 h_6 &= 4 & m_6 &= 0.05 \\
 h_7 &= 3 & m_7 &= 0.05 \\
 max &= 5 & k &= 7
 \end{aligned}$$

We must choose  $k - max = 2$  bin pairs to merge. Of course, we wish to take the pairs which minimize the resulting error which we estimate by:

$$err(r_x, r_{x+1}) = h_x \left| \frac{m_x}{h_x} - \left( \frac{m_x + m_{x+1}}{h_x + h_{x+1}} \right) \right| + h_{x+1} \left| \frac{m_{x+1}}{h_{x+1}} - \left( \frac{m_x + m_{x+1}}{h_x + h_{x+1}} \right) \right| \quad (6)$$

The error equals zero if and only if two bins imply uniformity.

In this example merging bins 1,2 and 4,5 minimizes the error because:

$$\begin{aligned}
 err(r_1, r_2) &= 4 \left( \left| \frac{0.2}{4} - \frac{0.2+0.1}{4+2} \right| \right) + 2 \left( \left| \frac{0.1}{2} - \frac{0.2+0.1}{4+2} \right| \right) = 0 \\
 err(r_4, r_5) &= 2 \left( \left| \frac{0.2}{2} - \frac{0.2+0.3}{2+3} \right| \right) + 3 \left( \left| \frac{0.3}{3} - \frac{0.2+0.3}{2+3} \right| \right) = 0
 \end{aligned}$$

#### 4.5 Adding Feedback to the ME Distribution Efficiently

Assume there is an existing ME distribution with  $k$  bins that we wish to refine with  $n$  QF records. The straightforward way to achieve this is to start the process “from scratch” with  $n + k$  intervals (see figure 7) . However, since the cost of the process grows exponentially with an increasing number of intervals (see 5.1) a more efficient way for refinement is desirable. An important observation is that in real-life scenarios feedback covers only some parts of the column domain. Thus, some parts of the existing maximum entropy distribution are “independent” of the new feedback (they are not really independent but we can apply the principle of maximum entropy in a simpler fashion, see below). We exploit this to reduce the total number of intervals, saving costs.

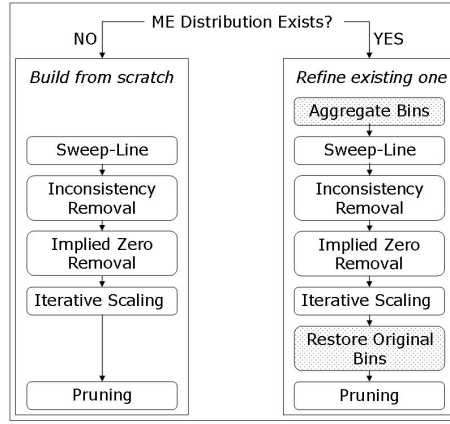


Figure 7: Differences between creating a ME distribution from scratch and refining an existing one



 bin of ME distribution  
  QF interval  
  aggregated bin

Figure 8: Legend for efficient refinement illustrations

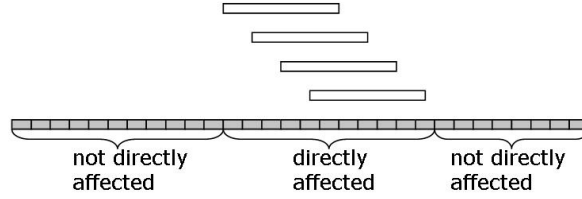


Figure 9: Affected parts of ME distribution during refinement

**Aggregating Bins:** In figure 9 we see that some parts of the existing ME distribution do not overlap with any new QF interval. We say these bins are “not directly affected” by the new QF. To reduce the total number of intervals we can aggregate not directly affected bins as follows:

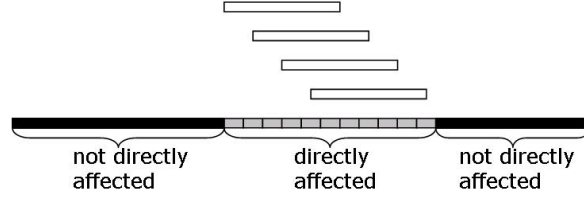


Figure 10: Aggregated bins for reducing the number of intervals during refinement

Instead of the original bins we use the aggregated ones in the refinement process (sweep-line, inconsistency+zero removal, iterative scaling). So, in total we have the QF intervals, the directly affected bins and the aggregated bins as intervals in the refinement process.

**Restoring the Original Bins:** After having run the refinement process we need to replace the aggregated bins with the original ones. Note that this step takes place before the pruning. The new maximum entropy distribution looks as follows:

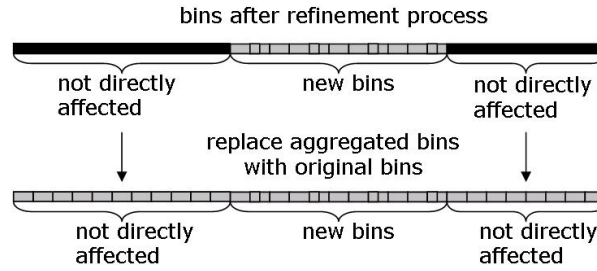


Figure 11: Restoring the original bins from the aggregated bins

We now replace the aggregated bins with the corresponding original bins. If the frequencies of the aggregated bins are not changed during the inconsistency removal we are done at this point. Otherwise, we must compensate for the changes made to the aggregated bins by modifying the frequencies of the original bins, respectively. Note that the linear program (section 3.6) will “randomly” adjust the frequencies to achieve consistency. Since we have no further information from QF we follow the principle of maximum entropy and distribute the change in frequency (equation 7) proportionately among the original, unaffected bins.

Let  $G^o$  denote the set of all aggregated bins with unmodified frequencies (after the step “aggregate bins”),  $G^n$  denote the set of all aggregated bins with modified frequencies (after the step “iterative scaling”),  $O(Y)$  denote the set of all original bins that aggregated bin  $Y$  consists of,  $F^o(X)$  denote the frequency of aggregated bin  $X \in G^o$ ,  $F^n(X)$  denote the frequency of aggregated bin  $X \in G^n$  and  $F(X)$  denote the frequency of original bin (not aggregated)  $X \in O(Y)$ . Then the total change in frequency we need to distribute

proportionately among the original, unaffected bins is given by:

$$\Delta_F = \sum_{i \in G^n} F^n(i) - \sum_{j \in G^o} F^o(j) \quad (7)$$

Now, for each aggregated bin  $Y \in G^o$ , we must adjust the frequencies of each original bin  $X$  that aggregated bin  $Y$  consists of, according to the following equation:

$$F_{new}(X) = F(X) + \frac{F(X)}{\sum_{j \in G^o} F^o(j)} \Delta_F \quad (8)$$

for all  $X \in O(Y)$  and all  $Y \in G^o$ .

## 5 Measurements

### 5.1 Performance

In this experiment we want to examine the performance of creating a maximum entropy distribution using the already introduced method with an increasing number of intervals. In order to identify bottlenecks the performance of each step is measured separately. Since the size of the two linear programs and the complexity of the iterative scaling algorithm are directly related to the number of intervals we expect an exponential increase in time needed.

Test settings:	maximum number of bins	200
	column domain	20000 rows, [0, 10000], skewed
	feedback range	[0, 10000] randomly generated intervals

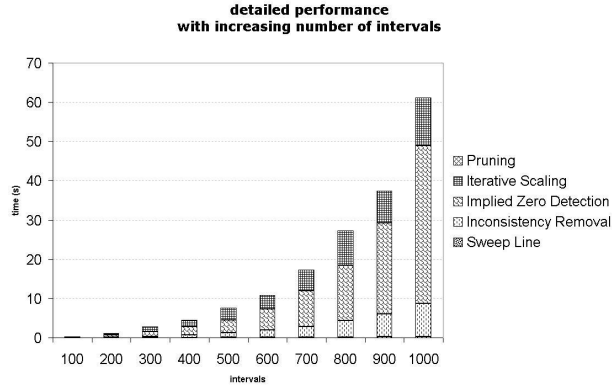


Figure 12: Stacked bar chart, performance with an increase in intervals I

We see the costs for the sweep-line and pruning are negligible. As expected the increase

in needed time is exponential. Having a more detailed look reveals that the bottleneck is the linear program for detecting implied zeros being outperformed even by the iterative scaling algorithm. This leaves room for future improvement because we use the generic method of solving a linear program, namely the simplex algorithm.

## 5.2 Quality

In this chapter we wish to analyse how the maximum entropy distribution compares to the actual data distribution with an increasing number of feedback records.

We use the Kolmogorov Distance<sup>1</sup> and the average distance<sup>2</sup> over the cumulated distributions to evaluate the accuracy of a given ME distribution. With an increasing number of intervals we expect both distances to converge but not actually reach zero because of fixing the maximum number of bins.

	maximum number of bins	200
Test settings:	column domain	20000 rows, [0, 10000], skewed
	feedback range	[0, 10000] randomly generated intervals

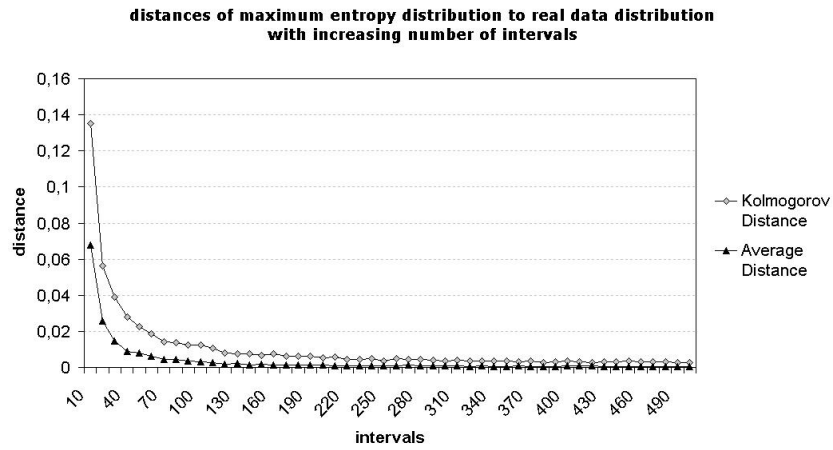


Figure 13: Accuracy of maximum entropy distribution with increasing intervals

Both curves converge towards zero quickly with an increasing number of intervals. Considering that the performance is acceptable for up to 500 intervals and the maximum error is below 2% at approx. 100 intervals we see that the feasible region in terms of cost/benefit lies within 100 and 500 intervals. However, keep in mind that the generated feedback delivers information on the whole column domain. This is the best-case scenario for feedback based statistics and unlikely to happen in a real-life environment. The next experiment shall give insight into how the Kolmogorov-Distance and the average distance

<sup>1</sup>Maximum absolute distance of two cumulative frequency distributions

<sup>2</sup>Average distance of two cumulative frequency distributions

behave when feedback allows for only limited insight into the column domain.

	maximum number of bins	200
	number of intervals	100
Test settings:	column domain	20000 rows, [0, 10000], skewed
	feedback range	varying, randomly generated intervals covering percentage of column domain

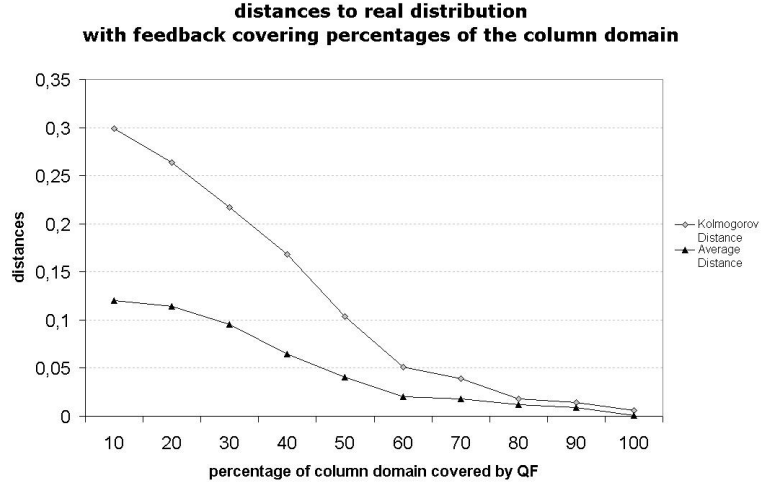


Figure 14: Distances with QF covering increasing percentages of column domain

Here we can observe a clear trend. The distances improve continuously with feedback covering more and more of the column domain. The improvement is almost proportional to the percentage covered. The accuracy of the maximum entropy distribution is regulated by the QF, especially how much of the column domain is covered by QF. Do keep in mind, though, that this distribution is completely feedback based. The bad accuracy with low percentages of the column domain covered by QF is the main reason why we combine a sampling-based distribution with the feedback.

## 6 Conclusion

### 6.1 Simplifying ISOMER

ISOMER [SHM<sup>+</sup>06] is a full-blown solution for creating and maintaining multi-dimensional histograms with the use of QF. In principle, our work is a simplification of ISOMER to the one-dimensional case. This allows for optimization. Here is an overview of the differences of this work to the solution in ISOMER:



**Combining proactive and reactive methods:** In contrast to ISOMER our method does not solely rely on feedback for histogram construction. We start with a sampling based distribution which we continuously refine using feedback. If at any given time the histogram delivers poor estimates (e.g. when queries tend not to query the same portion of the data) we can update statistics the conventional way - by sampling. So, we combine the proactive and the reactive approach in a consistent manner.

**Datastructure:** ISOMER uses the STHoles datastructure [BCG01]. For a one-dimensional histogram this structure imposes an unnecessary overhead when estimating selectivities. For a given predicate the STHoles structure requires summing up partial selectivities in possibly all levels of the tree. In contrast to STHoles we store the histogram as an array with cumulated selectivities, so selectivity estimation requires less computation.

**Determining the atoms:** In order to cope with overlapping knowledge the STHoles datastructure “drills” holes into the histogram creating a tree-structure. In the one-dimensional case this “drilling” becomes unnecessarily expensive especially when several feedback records are to be added at the same time. This is because the STHoles datastructure would require each feedback record to be added to the histogram consecutively, requiring the splitting of bins and the drilling of new holes each time. Since we assume that several records are added to the histogram at the same time we need to overcome this drawback (the refinement of the histogram occurs periodically). We use an efficient sweep-line algorithm to determine the bins (atoms) in one pass (see section 4.2).

**Efficient refinement with feedback:** For efficiently incorporating new feedback into a histogram ISOMER incrementally computes the new ME distribution. However, the method in ISOMER only speeds up the iterative scaling. In this work we introduce an efficient algorithm that speeds up all parts of the process, including the inconsistency removal, zero elimination and iterative scaling. This is specific to the one-dimensional case.

**Meeting a space budget:** ISOMER exploits the fact that the ME computation delivers an “importance measure” for each bucket. So, when the number of buckets exceeds an acceptable limit ISOMER removes the bucket with the least importance, then incrementally computes then new ME solution and repeats this process until the number of buckets is acceptable. In this work we propose a fast pruning algorithm that merges bins in order to fulfill storage constraints and minimizes the impact on accuracy in one pass. Here, no recomputation of the ME solution is required.

## 6.2 Summary

Feedback based statistics are a low-cost alternative to statistics created proactively using either data scans or sampling. We focus the ideas in ISOMER to single-column statistics combining the feedback-based approach with sampling-based statistics. This is specific to the one-dimensional case because creating multidimensional statistics via sampling is very

expensive and hence not desirable in ISOMER. Further, we have shown, for the single-column case, that purely feedback based histograms can yield acceptable accuracies at relatively low costs even if the whole column domain is not covered by QF. This is mainly due to the features of consistently adding new feedback information while making no assumptions about the unknown. We employ well founded concepts such as an efficient sweep-line algorithm for information preparation, linear programming for inconsistency removal and the principle of maximum entropy for ensuring uniformity in the absence of information. Furthermore, in order to be compliant with storage constraints a fast pruning algorithm is used that minimizes the impact on accuracy. New feedback is added quickly with an efficient refinement method that reduces the total number of intervals. Further improvements can be made on the implied zero detection because it is the performance bottleneck, especially with a large number of intervals. Having a predefined rule set (when exactly to trigger refinement) the database management system can autonomously refine feedback based histograms when new feedback is generated by normal database activity. The optimal parameters for these rules are still to be determined, e.g. referring to figure 3 when do we consider “act  $\ll$  est” or “act  $\gg$  est”. In essence, we have specialized the ideas in ISOMER to the one-dimensional case and introducing an industrial-strength, automatic and autonomous method for feedback based single-column statistics that is founded on well known concepts and delivers accurate selectivity estimates for either previously seen or unseen queries at reasonable costs.

## References

- [AC99] A. Aboulnaga and S. Chaudhuri. Self-Tuning Histograms: Building Histograms Without Looking at Data. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, 1999.
- [BCG01] N. Bruno, S. Chaudhuri, and L. Gravano. STHoles: A Multi-Dimensional Workload-Aware Histogram. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, 2001.
- [Beh05] Alexander Behm. DB2 Learning Optimizer, Query Feedback, Frequent Values and Quantiles. Thesis Assistentenarbeit, 2005.
- [CMN98] S. Chaudhuri, R. Motwani, and V. Narasayya. Random Sampling For Histograms Construction: How Much Is Enough? In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, 1998.
- [CR94] C. M. Chen and N. Roussopoulos. Adaptive Selectivity Estimation Using Query Feedback. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, 1994.
- [DGR01] A. Deshpande, M. Garofalakis, and R. Rastogi. Independence Is Good: Dependency-Based Histogram Synopses for High-Dimensional Data. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, 2001.
- [DR72] J. N. Darroch and D. Ratcliff. Generalized Iterative Scaling for Log-Linear Models. *The annuals of Mathematical Statistics*, 43, 1972.

- [GKVD00] D. Gunopulos, G. Kollios, V. Tsortras, and C. Domeniconi. Approximating Multi-Dimensional Aggregate Range Queries Over Real Attributes. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 2000.
- [GS85] S. Guiasu and A. Shenitzer. The Principle of Maximum Entropy. *The Mathematical Intelligencer*, 7(1), 1985.
- [IC91] Yannis E. Ioannidis and Stavros Christodoulakis. On the Propagation of Errors in the Size of Join Results. In *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*, 1991.
- [JP82] Nievergelt J. and Preparata. Plane-Sweep Algorithms for Intersecting Geometric Figures. 1982.
- [KHM<sup>+</sup>06] M. Kutsch, P. J. Haas, V. Markl, N. Megiddo, and T. M. Tran. Integrating a maximum-entropy cardinality estimator into DB2 UDB. In *Proceedings of the 10th International Conference on Extending Database Technology (EDBT)*, 2006.
- [LWV03] L. Lim, M. Wang, and J. Vitter. SASH: A Self-Adaptive Histogram Set For Dynamically Changing Workloads. In *Proceedings of the 29th International Conference on Very Large Data Bases*, 2003.
- [Lyn88] C. A. Lynch. Selectivity Estimation and Query Optimization in Large Databases with Highly Skewed Distribution of Column Values. In *Proceedings of the 14th International Conference on Very Large Data Bases*, 1988.
- [MD88] M. Muralikrishna and D. DeWitt. Equi-Depth Histograms for Estimating Selectivity Factors For Multi-Dimensional Queries. In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, 1988.
- [MHK<sup>+</sup>06] V. Markl, P. J. Haas, M. Kutsch, N. Megiddo, U. Srivastava, and T. M. Tran. Consistent Selectivity Estimation via Maximum Entropy. In *Proceedings of VLDB (electronic edition)*, 2006.
- [MMK<sup>+</sup>05] V. Markl, N. Megiddo, M. Kutsch, T. M. Tran, P. Haas, and U. Srivastava. Consistently Estimating the Selectivity of Conjunctions of Predicates. In *Proceedings of the 31st International Conference on Very Large Data Bases*, 2005.
- [PI97] M. Poosala and Y. Ioannidis. Selectivity Estimation Without the Attribute Value Independence Assumption. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, 1997.
- [SHM<sup>+</sup>06] U. Srivastava, P. J. Haas, V. Markl, N. Megiddo, M. Kutsch, and T. M. Tran. ISOMER: Consistent Histogram Construction Using Query Feedback. 2006. ICDE.
- [SLMK01] Michael Stillinger, Guy Lohman, Volker Markl, and Mokhtar Kandil. LEO, DB2's Learning Optimizer. In *Proceedings of the 27th International Conference on Very Large Data Bases*, 2001.
- [TGIK02] N. Thaper, S. Ghua, P. Indyk, and N. Koudas. Dynamic Multi-Dimensional Histograms. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, 2002.