A Cardinality Estimation Approach Based on Two Level Histograms

XUDONG LIN¹, XIAONING ZENG², XIAOWEI PU¹ AND YANYAN SUN¹

¹Department of Information Engineering
Environmental Management College of China
Qinhuangdao, Hebei, 066004 P.R. China
E-mail: 06112076@bjtu.edu.cn, fydong_xl@126.com; fyy_sun@126.com

²College of Mathematics and Information Technology
Hebei Normal University of Science and Technology
Qinhuangdao, Hebei, 066004 P.R. China
E-mail: ning_xz@126.com

For the mainstream relational database management systems, histograms play important roles in cardinality estimation. The main histogram-based cardinality estimation approaches can be classified into two categories: proactive approaches and reactive approaches. For the former, histograms are constructed and updated by periodical data scan which is also the essential reason affecting the accuracy and performance of this kind of approaches. Data scan is avoided in the latter, as an alternative, query feedback records (QFRs) are collected to construct and update histograms. But some time-consuming algorithms such as the effective QFR set calculation, the hole drilling algorithm and the iterative scaling algorithm are used by reactive approaches, which makes it inefficient.

In this paper, we propose a novel cardinality estimation approach by combining proactive approach with QFRs. In our approach, data scan will be executed only once to construct the initial first-level histogram. And then, corresponding to all buckets of the first-level histogram, second-level histograms will be constructed and updated based on QFRs. The existence of second-level histograms and the elaborated mechanism dealing with the data update problem can improve the accuracy of cardinality estimation remarkably. Different from the traditional histogram-based approaches, we do not construct only one big histogram covering the whole value range of an attribute, but construct a serials of small second-level histograms covering different parts of the whole value range. These second-level histograms can be constructed and updated independently over time to ensure the performance of the approach. Extensive comparison experiments fully demonstrate the advantages of our approach in accuracy and performance.

Keywords: cardinality estimation, proactive approach, reactive approach, histogram, query feedback record

1. INTRODUCTION

Cardinality estimation is an important problem in query optimizations. The choices of query plans rely heavily on the accuracy of cardinality estimation. For the mainstream relational database management systems, histograms play important roles in cardinality estimation. The first kind of histogram used to approximate data distributions within a database system is called the equi-width histogram [1]. And then, [2] proposes the equi-depth histogram and its multi-dimensional form is presented in [3]. [4] proposes the his-
togram with frequency as the sort parameter. Several years later, it is improved with the forms of $\nu$-optimal histogram [5], maxdiff histogram, compressed histogram [6] and entropy-based histogram [7]. All above approaches can be uniformly called proactive approaches because data scans must be executed periodically to make histograms consistent with underlying data, which lead to the following drawbacks:

- Periodical data scan aggravates the database overhead and affect the performance of routine queries.
- Data updates cannot be incorporated into a histogram in real time. Between twice reconstructions of a histogram, data updates may lead to large errors of cardinality estimation.
- To execute data scan in short time and avoid affecting database performance remarkably, the number of buckets in a histogram cannot be too many, which also brings negative impact on estimation accuracy.

To avoid periodical data scan and incorporate data updates into histograms in real time, people begin to use query feedback records (QFRs) to construct histograms [8]. QFRs can be gathered with relatively little overhead from query execution engines and used to refine histograms progressively. Hereafter, the similar approaches are proposed continuously [9, 10]. In this paper, we call this kind of approaches as reactive approaches. Although data scans are avoided, some new deficiencies prevent reactive approaches being practical. We can summarize the deficiencies of the representative reactive approach ISOMER [10] as follows:

- Besides the poor performance, iterative scaling algorithm is unstable and vulnerable. The experiments show that for the same under data, if currently used QFR sets are different, the histograms calculated by iterative scaling algorithm may be very different. And it is very difficult to obtain a valid histogram with a small number of QFRs.
- The criteria of ISOMER deciding whether a QFR is outdated or invalid are unreasonable. These criteria cannot ensure the accuracy of the retained QFRs, thereby leading to the errors of histogram calculation. The corresponding details can be found in Section 6.
- Because of containing time-consuming algorithms, the number of buckets in a histogram cannot be too many, which affects cardinality estimation accuracy inevitably.

The serious deficiencies in proactive approaches and reactive approaches prompt us to find new notions to improve the accuracy and the performance of the histogram-based cardinality estimation. In this paper, we propose a specific combination form of proactive and reactive approaches, and expect the effect of cardinality estimation can be improved by combining the advantages of the two kinds of approaches. It needs to emphasize that the application of the automatic column group detection technology in the modern relational DBMS [12] makes the huge multi-dimensional histogram unnecessary and the statistics about multi-attributes can be gathered as a one-dimensional histogram.
Therefore, we only research the problem using one-dimensional histograms in this paper.

In our approach, data scan will be executed only once to construct the initial first-level histogram. And then, within the value range of an attribute covered by each bucket in the first-level histogram, a second-level histogram will be constructed based on QFRs. Therefore, we call our approach the Cardinality Estimation approach based on Two Level Histograms (CETLH).

Hereinafter, the analysis of CETLH is based on the relation and the corresponding histograms described in Example 1.

Example 1: Consider a relation orderInfo with attributes order_id, customer_id, product_id, order_date and order_quantity. 200 thousand orders with 82 thousand distinct order quantities are stored in the relation. The minimum and the maximum of the order quantities are 1 and 99999 respectively. Parts of the data in the relation orderInfo are shown in Table 1. The first-level histogram and the second-level histograms constructed over the attribute order_quantity are shown in Fig. 1.

<table>
<thead>
<tr>
<th>order_id</th>
<th>customer_id</th>
<th>product_id</th>
<th>order_date</th>
<th>order_quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c1</td>
<td>p1</td>
<td>20120115</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>c1</td>
<td>p200</td>
<td>20120130</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>c2</td>
<td>p4</td>
<td>20120306</td>
<td>15,016</td>
</tr>
<tr>
<td>4</td>
<td>c2</td>
<td>p80</td>
<td>20120321</td>
<td>99,999</td>
</tr>
<tr>
<td>……</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200,000</td>
<td>c2132</td>
<td>p35</td>
<td>20130128</td>
<td>23,000</td>
</tr>
</tbody>
</table>

In Example 1, the first-level histogram over the attribute order_quantity contains 100 buckets, so there are also 100 second-level histograms totally. Each second-level histogram contains 30 buckets. Compared with proactive approaches and reactive approaches, the main improvements of CETLH can be summarized as:

- The framework of cardinality estimation which is composed of two level histograms can improve the accuracy of cardinality estimation remarkably.
- A novel mechanism is proposed to locate the ranges of remarkable data updates accurately and incorporate these updates into histograms in real time. Simultaneously, the histogram adjustment mechanism can ensure that each second-level histogram always locates in a range with the relatively uniform data distribution.
- Data scan is executed only once and does not affect database performance seriously. And due to the existing of second-level histograms, the number of the buckets in a first-level histogram is limited, and the performance of constructing a first-level histogram can be very high.
- The performance of the construction of second-level histograms can also be very high because: (i) each second-level histogram is limited at a relatively small range covered by a bucket of a first-level histogram; (ii) In each moment, only one second-level histogram is possible to be constructed or updated.
- The bucket number in each second-level histogram is fixed and the hole drilling operation is not needed. Simultaneously, iterative scaling algorithm is replaced by the Min-
imum-norm Least-squares Algorithm (MLA), which improves the stability and the robustness of the construction of second-level histograms.

The rest of the paper is organized as follows: Section 2 discusses the related work. Section 3 gives the notations and preliminary. Section 4 and Section 5 describe the details of the construction of two level histograms. The mechanism dealing with data update problem is analyzed in Section 6. The results of extensive experiments are demonstrated in Section 7. Section 8 summarizes the paper and discusses future directions.

2. RELATED WORK

One of the first attempts at cardinality estimation was done by the designers of System R [13]. And the first approach to use histograms to approximate data distributions within a database system was in Kooi’s PhD thesis [1]. The histogram used in [1] is called the equi-width histogram as the value range of an attribute over which a histogram is constructed will be partitioned into several smaller sub ranges with equal widths. [2] proposes the equi-depth histogram and its multi-dimensional form is presented in [3]. [4] proposes histograms with frequency as the sort parameter, which represented the first departure from value-based grouping of buckets. Several years later, it is improved with the forms of v-optimal histogram [5], maxdiff histogram [6], compressed histogram[6] and entropy-based Histogram [7].

[8] and [9] begin to use query feedback mechanisms which take into account actual sizes of query results to dynamically modify histograms so that their estimates are closer to reality. [10] introduces ME principle into feedback-driven histogram, and [14] proposes an alternative formulation for consistency to improve the performance of the ME-based approach. [15] leverages all available feedback information and is scalable to multiple dimensions and large number of query feedback records.

Sampling technology [16-18] can be considered as a complementary to histograms because static histograms are usually constructed based on a sample of the original data.

Wavelet technology is also integrated into histograms [19, 20]. [19] proposes a method to build efficient and effective histograms using wavelet decomposition. [20] gives a novel approach for the dynamic maintenance of wavelet-based histogram with very little online time and space costs.

[10] and [21] apply ME principle to the space of frequencies of observed statistics,
and [22] begins to apply the ME principle to the probability distribution on the underlying (discrete) relations. [23] and [24] improves the work in [22] continuously. [23] solves the model computation problem: given a statistical program and a set of full inclusion constraints, find a solution to the ME-based model. And the full-fledged solution is established in [24] which can use the ME-based approach to predict the cardinality of conjunctive queries.

3. NOTATIONS AND PRELIMINARY

All notations used in the paper are shown in Table 2.

<table>
<thead>
<tr>
<th>Table 2. Notations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic notations</td>
</tr>
<tr>
<td>$r$ and its subscripted forms</td>
</tr>
<tr>
<td>$a$ and its subscripted forms</td>
</tr>
<tr>
<td>$p$ and its subscripted forms</td>
</tr>
<tr>
<td>$h$, its subscripted and superscripted forms</td>
</tr>
<tr>
<td>$b$ and its subscripted forms such as $b_i$</td>
</tr>
<tr>
<td>$v$ and its variations</td>
</tr>
<tr>
<td>$\min(x), \max(x)$</td>
</tr>
<tr>
<td>notations related to the attribute $a$</td>
</tr>
<tr>
<td>$\rg(a)$</td>
</tr>
<tr>
<td>$</td>
</tr>
<tr>
<td>notations related to the bucket $b$</td>
</tr>
<tr>
<td>$\rg(b)$</td>
</tr>
<tr>
<td>$v(b)$</td>
</tr>
<tr>
<td>$b = (\min(\rg(b)), \max(\rg(b)), v(b))$</td>
</tr>
<tr>
<td>notations related to the histogram $h$</td>
</tr>
<tr>
<td>$\rg(h)$</td>
</tr>
<tr>
<td>$</td>
</tr>
<tr>
<td>$d(h)$</td>
</tr>
<tr>
<td>$w(h)$</td>
</tr>
<tr>
<td>$h^1$</td>
</tr>
<tr>
<td>$h^1_a$</td>
</tr>
<tr>
<td>$h^2$</td>
</tr>
<tr>
<td>$h^2_{a,b}$</td>
</tr>
<tr>
<td>notations related to the predicate $p$</td>
</tr>
<tr>
<td>$\rg(p)$</td>
</tr>
<tr>
<td>$\qfr(p)$</td>
</tr>
<tr>
<td>$QFR$</td>
</tr>
</tbody>
</table>
Table 2. (Cont’d) Notations.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(v(qfr(p)))</td>
<td>the actual number of tuples satisfying (p) which is obtained from a query execution engine</td>
</tr>
<tr>
<td>(qfr(p))</td>
<td>the inner QFR of the bucket (b), i.e., the QFR (qfr(p)) satisfying (rg(p) \subseteq rg(b))</td>
</tr>
<tr>
<td>(p = (\text{min}(rg(p)), \text{max}(rg(p))))</td>
<td>(p) can be extended to a two-tuple</td>
</tr>
<tr>
<td>(qfr(p) = (\text{min}(rg(p)), \text{max}(rg(p)))), (v(qfr(p)))</td>
<td>(p) can be extended to a triple</td>
</tr>
</tbody>
</table>

Based on the definition of inner QFR given in Table 2, all of the inner QFRs which are being used to calculate a second-level histogram \(h_{a,b}^{s}\) compose the currently used inner QFR set of \(b\) or \(h_{a,b}^{s}\). Non-inner QFRs can be transformed into inner QFRs as follows: given a predicate \(p\) and \(t\) adjacent buckets \(b_m, \ldots, b_{m+t-1} \in (h_{a,b}^{s})\) which satisfy \(rg(p) \not\subseteq rg(b_i)\) and \(rg(p) \cap rg(b_i) \neq \emptyset\) for \(i = m, \ldots, m+t-1\). The predicate \(p\) can be partitioned into \(t\) mutually disjoint sub-predicates \(p_m, \ldots, p_{m+t-1}\) which satisfy \(\bigcup_{i=m}^{m+t-1} rg(p_i) = rg(p)\) and \(rg(p_i) \subseteq rg(b_i)\) for \(j = m, \ldots, m+t-1\). And then, we can get \(v(qfr^t(p)) = \sum v(qfr^t_i(p_i))\). In this formula, only two end values \(v(qfr^t_m(p_m))\) and \(v(qfr^t_{m+t-1}(p_{m+t-1}))\) are unknown. If the second-level histogram \(h_{a,b}^{s}\) is known, we can calculate \(v(qfr^t_m(p_m))\) and the non-inner QFR \(qfr(p)\) can be transformed into the inner QFR \(qfr_{m+t-1}(p_{m+t-1})\). Sometimes, both \(h_{a,b}^{s}\) and \(h_{a,b}^{s+t}\) may be unknown. If we want to calculate \(h_{a,b}^{s+t}\), we use \(v(b_a)\) and the traditional uniformity assumption to calculate the approximate \(v(qfr^t_m(p_m))\) and vice versa. Although the approximation will deteriorate the accuracy of the calculation of second-level histograms, it can improve the utilization of QFRs and the accuracy of the calculation of second-level histograms can be ameliorated rapidly with the increase of the calculated second-level histograms.

4. CONSTRUCTION OF FIRST-LEVEL HISTOGRAM

In CETLH, the main purpose of constructing a first-level histogram is to play the following two auxiliary roles:

Firstly, before the construction of second-level histograms, a first-level histogram can be used to coarsely estimate the cardinality of a predicate. For a predicate \(p\) and a bucket \(b\) satisfying \(rg(p) \cap rg(b) \neq \emptyset\), the number of tuples falling in \(b\) and satisfying \(p\) can be estimated based on the uniformity assumption:

\[
n_s(p) = v(b) \times \frac{\text{vol}(rg(p) \cap rg(b))}{\text{vol}(rg(b))},
\]

for discrete data, \(\text{vol}(R)\) denotes the number of discrete points that lie in \(R\).

Secondly, the buckets of a first-level equi-depth histogram can partition the whole value range of an attribute into several smaller sub ranges which are used as the borders of second-level histograms. Due to the property of equi-depth histogram, inside each sub-range, data distribution is relatively uniform, which is helpful to improve the construction accuracy of second-level equi-width histograms.
To construct a first-level histogram, the off-the-shelf procedures included in the commercial database systems can be used. For example, in Oracle 12c [12], we can use the following SQL script to construct the first-level equi-depth histogram over the attribute `order_quantity` with the under data described in Example 1:

```
Begin
    DBMS_Stats.Gather_Table_Stats (
        Ownname => 'lin',
        Tabname => 'orderInfo',
        CASCADE => True,
        Estimate_Percent => 100,
        Method_OPT => 'For Collums Order_Quantity Size00'
    );
End;
```

The first-level equi-depth histogram constructed over attribute `order_quantity` by the above SQL script is shown in Fig. 1. After the construction of a first-level histogram, the maximum and the minimum of the actual number of tuples falling in each bucket will be recorded. The difference of the maximum and the minimum will decide whether a first-level histogram should be updated. The details can be found in Section 6.2.

5. CONSTRUCTION OF SECOND-LEVEL HISTOGRAMS

The construction of second-level histograms can reduce the value range applying the uniformity assumption and remarkably improve the application effect of the uniformity assumption. We will not construct a large second-level histogram covering the whole value range of an attribute; alternatively, a series of smaller equi-width second-level histograms will be constructed independently according to different inner QFR sets. When a new inner QFR arrives, only one but not all second-level histograms may be reconstructed. That means all second-level histograms can be constructed and updated in staggered time, which can fully ensure the performance of the construction of second-level histograms.

Before describing the details of the construction of second-level histograms, the core algorithm of the construction, MLA will be introduced firstly.

5.1 Minimum-norm Least-Squares Algorithm

The main function of MLA is to calculate a second-level histogram based on the currently used inner QFR set of it. For ISOMER approach, to make a calculated histogram consistent with all currently used QFRs, Maximization Entropy (ME) principle [25, 26] is introduced and an approximate ME-based solution is calculated by the iterative scaling algorithm. But we do not continue to adopt the iterative scaling algorithm in our approach because of its deficiencies in stability and robustness. The stability deficiency means for a value range of an attribute and two inner QFR sets with little differences, the calculated histograms may be very different. The robustness deficiency means it is very difficult to obtain a valid histogram when the number of QFRs participating in the calcu-
lation is much less than the number of buckets of the calculated histogram.

In this paper, we use a new algorithm MLA to replace iterative scaling algorithm to finish the construction of second-level histograms. Although ME principle is not used in MLA, the consistency of second-level histograms with inner QFRs can still be held.

For a second-level histogram $h'$ with $k$ buckets $b_1, \ldots, b_k$, the inner QFR set of $h'$ with $t$ inner QFRs $qfr(p_1), \ldots, qfr(p_t)$ can compose the following linear system:

$$
n_h(p_1) + n_h(p_2) + \ldots + n_h(p_t) = v(qfr(p_1)) \\
n_h(p_1) + n_h(p_2) + \ldots + n_h(p_t) = v(qfr(p_2)) \\
\vdots \\
n_h(p_1) + n_h(p_2) + \ldots + n_h(p_t) = v(qfr(p_t))$$

(2)

Applying Formulas 1 and 2 can be transformed as a linear system about $v(b_1), \ldots, v(b_k)$, i.e., the numbers of tuples which fall in the buckets $b_1, \ldots, b_k$.

MLA [27] is capable of solving underdetermined, determined and overdetermined linear system. Therefore, we can use MLA to solve the linear system about $v(b_1), \ldots, v(b_k)$ and the solution is a second-level histogram consistent with all currently used inner QFRs. The detail of MLA is omitted and can be found in [27]. Experiments show that MLA is more stable and robust than the iterative scaling algorithm.

5.2 Steps of Construction

For a relation $r(a_1, \ldots, a_n)$ and a first-level histogram $h_{i,a_i}^0$, $1 \leq i \leq n$, $B(h_{i,a_i}^0) = \{b_1, \ldots, b_m\}$. At the value range of $a_i$ covered by a bucket $b_j \in B(h_{i,a_i}^0)$, $1 \leq j \leq m$, the second-level histogram $h_{i,a_i,b_j}^1$ can be constructed following the steps in Fig. 2.

![Fig. 2. Steps of the construction of second-level histograms.](Image)
We use Example 1 to further analyze each step in Fig. 2 as follows:

**Step 1:** for Example 1, \(|B_{h_{\text{order_quantity}}, b_j}| = 30\) for \(j = 1, \ldots, 100\).

**Step 2:** the initial \(h_{a_i, b_j}\) is an equi-width and equi-depth histogram with \(w(h_{a_i, b_j}) = \frac{v(b_j)}{|B_{h_{a_i, b_j}}|}\) and \(d(h_{a_i, b_j}) = \frac{\nu(b_j)}{|B_{h_{a_i, b_j}}|}\). In Example 1, \(w(h_{\text{order_quantity}, b_j}) = 798 - 1 = 26.6\) and \(d(h_{\text{order_quantity}, b_j}) = \frac{2000}{30} = 66.7\). The initial \(h_{\text{order_quantity}, b_j}\) is shown in Fig. 3 (a). Intuitively, the initial second-level histograms will be replaced by the second-level histograms calculated by MLA later. To distinguish with the initial second-level histograms, the second-level histograms calculated by MLA are called the *normal* second-level histograms hereinafter.

**Step 3:** no more explanation.

**Step 4:** Based on the robustness and the stability of MLA, we can set a moderate threshold number \(n_q_0\) to ensure that MLA can be called in an appropriate time. For Example 1, we set \(n_q_0 = 10\).

**Step 5-7:** when \(|arr_{b_j}| < n_q_0\), the newly arrived inner QFRs will be stored into \(arr_{b_j}\) but MLA will not be called. Once \(|arr_{b_j}| = n_q_0\), MLA will be called for the first time to calculate \(h_{a_i, b_j}\). The calculation may be not successful because of the insufficient inner QFRs. But henceforth, once adding a new inner QFR into \(arr_{b_j}\), the MLA will be called until a normal second-level histogram is obtained. For Example 1, as the inner QFRs arrive over time, the changes of \(arr_{b_j}\) are shown in the tables of Figs. 3 (a)-(d).

### 5.3 Cardinality Estimation

For a relation \(r(a_1, \ldots, a_n)\) and a first-level histogram \(h_{a_i}^r, 1 \leq i \leq n, B(h_{a_i}^r) = \{b_1, \ldots, b_m\}\). Given that for a predicate \(p\), there exist \(t\) adjacent buckets \(b_o, \ldots, b_{o+t-1} \in B(h_{a_i}^r)\) which satisfy \(\text{rg}(b_j) \cap \text{rg}(p) \neq \emptyset\) for \(1 \leq o \leq m, j = 0, \ldots, t - 1\). For any bucket \(b_j \in \{b_o, \ldots, b_{o+t-1}\}\), if \(h_{a_i, b_j}\) is an initial second-level histogram, \(n_{b_j}(p)\) can be calculated using...
Formula 1. Otherwise, a more accurate result can be calculated using Formula 3 based on the bucket set $B(h_{ai,j}) = \{b_{j}, ..., b_{k}\}$:

$$n_{j}(p) = \sum_{q=1}^{m} v(b_{m}) \times \frac{vol(rg(p) \cap rg(b_{j}))}{vol(rg(b_{j}))}$$

(3)

And the cardinality of $p$ can be estimated as:

$$n(p) = \sum_{j=1}^{m} n_{j}(p)$$

(4)

For Example 1, the predicate $p = (733, 1923)$ over the attribute order_quantity corresponds to $v(qfr(p)) = 3052$. The buckets $b_{j}$ for $j = 1, 2, 3$ of the first-level histogram $h_{order\_quantity}$ shown in Fig. 1 satisfy $rg(b_{j}) \cap rg(p) \neq \emptyset$. Based on the normal second-level histograms $h_{order\_quantity,b_{j}}$, we can get $n_{1}(p) = 497.6$, $n_{2}(p) = 1984.1$ and $n_{3}(p) = 563.5$ by use of Formula 3. Thereby, $n(p) = 3045.2$. Compared with $v(qfr(p))$, the estimation accuracy is very high.

6. AUTOMATICALLY INCORPORATION OF DATA UPDATES

For reactive approaches, data updates in a database are incorporated into histograms by replacing invalid QFRs with newly arrived QFRs and the key problem is to judge which QFRs are invalid. In ISOMER, the QFRs with older ages or smaller Lagrange multipliers will be discarded. But a QFR $qfr(p)$ with older age may be always accurate as long as the values of an attribute at the range of $rg(p)$ are not updated. And given the Lagrange multiplier of a QFR $qfr(p_1)$ is smaller than another QFR $qfr(p_2)$, although $qfr(p_1)$ can be discarded, the accuracy of the retained $qfr(p_2)$ cannot be ensured. In a word, the judging methods used in ISOMER cannot ensure retaining the accurate QFRs and discarding the error ones. In fact, without the information about the details of data updates, it is impossible to judge whether a currently used QFR is accurate or not.

In CETLH, we adopt a novel mechanism to deal with the data update problem: we no longer attempt to judge the accuracy of a currently used inner QFR, but use the newly arrived inner QFRs to locate the remarkable data updates. The core of our thought is trying to find the value ranges of an attribute with remarkable data updates and reconstruct the buckets located in these value ranges. The details of the mechanism will be described in Section 6. In the mechanism, the update of a first-level histogram is based on the update of the corresponding second-level histograms, so we will describe the update of second-level histograms firstly.

6.1 Update of Second-level Histograms

For a relation $r(a_{1}, ..., a_{n})$ and a first-level histogram $h_{ai,j}$, $1 \leq i \leq n$, $B(h_{ai,j}) = \{b_{j}, ..., b_{k}\}$. At the value range of $a_{i}$ covered by a bucket $b_{j} \in B(h_{ai,j}), 1 \leq j \leq m$, the second-level histogram $h_{ai,b_{j}}$ contains $k$ buckets which compose the bucket set $B(h_{ai,b_{j}}) = \{b_{j}, ..., b_{k}\}$. When the data at the value range of $a_{i}$ covered by the bucket $b_{j}$ change remarkably, the update of $h_{ai,b_{j}}$ can be completed following the steps in Fig. 4.
We use Example 1 to further analyze each step in Fig. 4 as follows:

**Step 1**: no more explanation.

**Step 2**: for Example 1, \( n_{q_1} = 5 \).

**Step 3**: for Example 1, \( t_{h_1} = 20 \).

**Step 4-8**: For each new inner QFR \( qfr^w(p_w), 1 \leq w \leq n_{q_1} \) of \( b_j \), find the bucket set \( \{b_{j'_1}, ..., b_{j'_w}\} \subseteq B(h'_j, s) \) which satisfy \( rg(b_{j'_w}) \cap rg(p_w) \neq \emptyset, s_1 \leq x \leq e_s \) firstly. And then, calculate \( n_e(p_w) \) based on the bucket set \( \{b_{j'_1}, ..., b_{j'_w}\} \).

**Step 5**: find the bucket set \( \{b_{j'_1}, ..., b_{j'_w}\} \subseteq B(h'_j, s) \) satisfying \( rg(b_{j'_w}) \cap rg(p_w) \neq \emptyset, s_1 \leq x \leq e_s \) firstly. And then, calculate \( n_e(p_w) \) based on the bucket set \( \{b_{j'_1}, ..., b_{j'_w}\} \).

**Step 6**: calculate \( n_e(p_w) \) based on the bucket set \( \{b_{j'_1}, ..., b_{j'_w}\} \).

**Step 7**: mark each bucket in the bucket set \( \{b_{j'_1}, ..., b_{j'_w}\} \) as the `changed`, `unchanged` or `unmarked` state.

**Step 8**: estimate \( |arr_{\infty}\| < n_{q_1} \) if \( \text{Yes} \), or \( \text{No} \).

**Step 9**: clear \( arr^w \).

**Step 10**: Reconstruct the currently used inner QFR set of \( b_j \) in \( arr^w \).

**Step 11**: call MLA to generate normal second-level histogram \( h'_{obj} \).

---

**Fig. 4. Steps of the update of second-level histograms.**
er $b_i$ which have been marked as unchanged should maintain their marks because some other accurately estimated inner QFRs have shown that no remarkable data updates happen in these ranges. After this step, all $b_{ij} \in B(h_{ij})$ will be in the states of changed, unchanged or unmarked, and the remarkable data updates can be located inside ranges covered by the ones with change state. For Example 1, the complete second-level histogram $h_{order\_quantity,b}$ is shown at the top of Fig. 5 (a). An SQL update statement, update orderInfo set order_quantity = order_quantity -700 where order_quantity>850 and order_quantity < 950, is executed to update data. After the data update, 5 new inner QFRs of $b_1$ arrive one by one. With the arriving of each new inner QFR, the state changes of all buckets of $h_{order\_quantity,b}$ are shown in the table at the bottom of Fig. 5 (a).

**Step 9:** all currently used inner QFRs stored in $arr^b$ will be discarded and the whole currently used inner QFR set will be rebuilt in a more reliable way.

**Step 10:** the new currently used inner QFR set of $b$ will be composed of two kinds of statistical information: firstly, because all of the $nq$ new QFRs stored in $arr_{new}$ are just collected from the query execution engine, they are considered accurate and will be copied into $arr^b$ directly; secondly, for the buckets $b_{ij} \in B(h_{ij})$ with the states unchanged or unmarked, it is considered that no remarkable data updates happen in $rg(b_{ij})$, so these buckets will be transferred into new inner QFRs of $b_j$ and stored into $arr^b$. That is, a bucket $b_{ij} = (\min(rg(b_{ij})), \max(rg(b_{ij})), v(b_{ij}))$ can be transformed as an inner QFR $qfr_{ij} = (\min(rg(b_{ij})), \max(rg(b_{ij})), v(b_{ij}))$. Compared with the method discarding QFRs with older ages or smaller Lagrange multipliers, above two kinds of statistical information can provide more reliable basis to reconstruct second-level histograms. For Example 1, after the data updates, the new inner QFRs of $b_1$ which are stored in $arr^b$ to reconstruct the second-level histogram $h_{order\_quantity,b}$ are shown in Fig. 5 (b).

**Step 11:** no more explanation.

| new inner QFR | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| null          |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| (553,798,1628)|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| (140,779,2291)|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| (185,275,490) |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| (275,794,1609)|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| (421,695,716) |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

As each new inner QFR arrives, the state changes of the 1st bucket to the 30th bucket of $h_{order\_quantity,b}$ are shown in the following table:

| new inner QFR | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| null          |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| (553,798,1628)|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| (140,779,2291)|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| (185,275,490) |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| (275,794,1609)|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| (421,695,716) |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

(a)
6.2 Update of First-level Histogram

Every time updating a second-level histogram, the number of tuples which fall in each bucket of the first-level histogram will be checked. Once the number in one bucket of a first-level histogram is much different from the others, the borders of all buckets in the first-level histogram will be adjusted to make it return to an equi-depth state.

For a relation \( r(a_1, \ldots, a_n) \) and a first-level histogram \( h'_i, 1 \leq i \leq n \), \( B(h'_i) = \{b_1, \ldots, b_p \} \). At the value range of \( a_i \) covered by a bucket \( b_j \in B(h'_i), 1 \leq j \leq m \), the second-level histogram \( h'_{i,j} \) contains \( k \) buckets which can be denoted by \( B(h'_{i,j}) = \{b_j, \ldots, b_p \} \). When the data change remarkably at the value range of \( a_i \) covered by the bucket \( b_j \), the first-level histogram \( h'_i \) and its corresponding second-level histograms will be updates can be completed following the steps shown in Fig. 6.

We use Example 1 to further analyze each step in Fig. 6 as follows:

Fig. 6. Steps of the update of first-level histograms.
Step 1-5: no more explanation.

Step 6: for Example 1, the final result of the update of the first-level histogram $h_{\text{order quantity}}$ is shown in Fig. 7. From the bottom of Fig. 7, we can see that the borders of the second-level histogram $h'_{\text{order quantity}, b_1}$ are adjusted from $(1, 799)$ to $(1, 692)$.

Step 7: during the reconstruction, $|B(h'_{\text{order quantity}})|$ maintains unchanged and equals the value which is set in the first step of Section 5.2. For Example 1, the new width of $h'_{\text{order quantity}, b_1}$ can be calculated as $w(h'_{\text{order quantity}, b_1}) = \frac{692 - 1}{30} = 23$.

Step 8: for Example 1, the border adjustment of the first-level histogram $h_{\text{order quantity}}$ can be seen from the top of Fig. 7.

Step 9: the calculation will be based on the buckets of the currently used second-level histograms. For Example 1, the recalculated second-level histogram $h'_{\text{order quantity}, A}$ can be seen from the bottom of Fig. 7.

7. EXPERIMENTS

The experiments are performed on a 3.2GHz Intel CPU machine running Windows XP sp3, with 4GB memory and 1TB hard disk. Before analyzing the experimental results, we describe the experimental settings firstly.

7.1 Experimental Settings

7.1.1 Data sets

We use both real and synthetic data sets for the experiments.

Real data set (denoted by c): This data set is transformed from USCensus1990raw in the UCI Machine Learning Repository [28]. It consists of 2,458,285 tuples. An initial equi-depth histogram with 100 buckets is constructed over the attribute $\text{income1}$ with $|\text{rg}(\text{income1})| = 55,088$, $\text{min}(\text{rg}(\text{income1})) = 0$ and $\text{max}(\text{rg}(\text{income1})) = 197,927$.

Synthetic data set I (denoted by g2): This data set consists of 500,000 tuples which are sampled from 30 Gaussians [29] with the same standard deviation 25 of different peak values selected uniformly at random from 0 to 5000. The total number of tuples contained in each Gaussian bell follows the Zipf distribution [29] with the skew parameter $z$.
1. In real databases, frequency distributions often follow the Zipf distribution [30, 31].

**Synthetic data set II (denoted by gu):** This data set also consists of 500,000 tuples which are sampled from 30 Gaussians with the same standard deviation 25 of different peak values selected uniformly at random from 0 to 5000. The total number of tuples contained in each Gaussian bell is selected uniformly at random.

### 7.1.2 Workloads

We use two predicate generation models proposed by [32] to generate the training predicates and the validation predicates. For each model, the predicate centers are generated based on a certain *center distribution* firstly, and then, each predicate is expanded from a predicate center to its neighborhood with a certain range.

**Data-dependent predicate model (denoted by d):** In this model, predicate centers are sampled from the underlying data distribution. And then, each predicate is expanded from a predicate center to its neighborhood with the range at most 20 percent of the whole value range of an experimental attribute. The total number of predicates around each predicate center follows the Zipf distribution with the skew parameter $z=1$.

**Uniform predicate model (denoted by u):** In this model, predicate centers are sampled uniformly at random from the whole value set of an experimental attribute. And then, each predicate is expanded from a predicate center to its neighborhood with the range at most 20 percent of the whole value range of the experimental attribute. The total number of predicates around each predicate center also follows the random distribution.

For each different approach, a training workload containing 100 to 1,000 predicates is generated to tune the histograms. And then, a validation workload with the same distribution and quantity as the training workload is generated to calculate the relative accurate rates defined in the next subsection.

### 7.1.3 Metrics

Firstly, we can define the *relative error*, $re(p)$, of a predicate $p$ using Formula 5:

$$
re(p) = \frac{abs(n(p) - v(qfr(p)))}{v(qfr(p))},
$$

But relative errors are not suitable to be used directly to measure the accuracy of a cardinality estimation approach. For example, for a predicate $p$, the relative errors obtained by using two different cardinality estimation approaches are 0.8 and 80 respectively. But from a practical point of view, we cannot consider that the first approach is better than the second one because for a realistic query optimizer, both of the two approaches can not estimate the cardinality of $p$ accurately. Therefore, we define the *relative accuracy rate*, $rar(ce)$, of a cardinality estimation approach $ce$ as the criterion to measure the accuracy of a cardinality estimation approach:
where \( ce \) denotes a cardinality estimation approach, \( cn(ce) \) denotes the number of validation predicates whose relative errors are less than \( s \), and \( tn(ce) \) denotes the total number of validation predicates. In our experiments, we set \( s = 0.2 \) and consider a predicate whose relative error is less than 0.2 as a correctly estimated predicate.

### 7.1.4 Programs

In the experiments, the comparison approaches of cardinality estimations include the approach proposed in this paper, the representative reactive approach ISOMER and the one adopted by the query optimizer of Oracle, which are denoted by CETLH, ISOMER and Optimizer respectively.

For Optimizer approach, the equi-depth histograms with 100 buckets over the attribute \( income1 \) of the data set \( c \), the attribute \( data \) of the data set \( d \) and the attribute \( data \) of the data set \( u \) are constructed.

For CETLH, the number of buckets in a first-level histogram is also set as 100. And the number of buckets in each second-level histogram will be set as 30.

All of the experimental algorithms are realized under JDK 1.6.0_10. And we record each experimental result averaged over 10 runs.

### 7.2 Accuracy Experiments

#### 7.2.1 Static Data

*Static data* means there are no data updates during the constructions of histograms and the executions of predicates. The relative accuracy rates of cardinality estimations of different approaches are shown in Figs. 8 (a)-(f) respectively.

It is easy to see that CETLH approach has the highest accuracy and stability for

![Fig. 8. Relative accuracy rates of different approaches.](image-url)
static data. For each combination of data set and workload, the relative accuracy rate of CETLH shows a stable improvement tendency with the increase of the number of training predicates. For the workload \( d \), the improvement tendency always remains smooth. But for the workload \( u \), the relative accuracy rates of CETLH have rapid improvement phases which can be observed from Figs. (b), (d) and (f). The rapid improvement phases correspond to the range of 200 to 400 training predicates. We know that for the workload \( d \), most training predicates locate around few predicate centers. Therefore, even for 200 training predicates, 2 to 4 second-level histograms can be calculated accurately, which ensure the accuracy of the validation predicates locate in these ranges. But the training predicates in the workload \( u \) are relatively dispersed, and no second-level histograms can be calculated successfully as the total number of training predicates is less than 200. As the increase of the number of training predicates, second-level histograms begin to be calculated little by little, and the relative accuracy rate is improved rapidly. From here, we can see the important role of second-level histograms in the process of cardinality estimation.

The tendency of the relative accuracy rate of ISOMER is very different from the ones of CETLH and Optimizer, and smooth changes are replaced by the frequent and significant fluctuations, which testify the mentioned instability and vulnerability of ISOMER. In some cases, ISOMER can estimate cardinality even more accurately than CETLH, but it can not be continuously maintained. In Fig. 9, we expand parts of Fig. 8(a) and 8(b) to show the original results of 10 runs of accuracy experiments using CETLH and ISOMER approaches at the range of 200 to 400 training predicates. In almost each sub graph of Fig. 9, we can find the points at which the relative accuracy rates of ISOMER are higher than the corresponding ones of CETLH such as the third and the fourth runs in Fig. 9(a), the first and the third runs in Fig. 9 (c), etc. But after synthesizing the results of 10 runs, CETLH is excellent than ISOMER which has been shown in Fig. 8. As the number of training predicates increases, the superiority of CETLH is more evi-
dent which can be testified by the changes from Figs. 9 (a) to (c) and the changes from Figs. 9 (d) to (f).

For CETLH, an important parameter must be explained further, i.e. the number of buckets in a second-level histogram. To observe the influence of the parameter, we test the relative accuracy rates of CETLH with different numbers of buckets from 10 to 80 in a second-level histogram using 500 and 700 training predicates respectively. The experimental results are shown in Fig. 10. We can observe that the relative accuracy rate does not always go up with the increase of the number of buckets in a second-level histogram. When the number of buckets in a second-level histogram is between 20 and 40, the relative accuracy rate arrives at the peak. When the number of buckets in a second-level histogram is more than 40, this second-level histogram may be not calculated successfully because there are not enough training predicates. Especially for the relatively dispersed character of the workload $u$, as the number of buckets in a second-level histogram increases, the second-level histograms which can be calculated successfully will decrease rapidly, which lead to the speed of the decrease of relative accuracy rate using workload $u$ being much faster than workload $d$. Therefore, we set the number of buckets in a second-level histogram as 30 in our experiments, which is also a balance between the accuracy and the performance of cardinality estimation.

7.2.2 Dynamic data

Dynamic data means there are data updates during the constructions of histograms and the executions of predicates. Because the Optimizer approach cannot deal with data update in real time, we only compare CETLH with ISOMER in the accuracy experiments on dynamic data and the whole experimental results are shown in Fig. 11.

The process of the accuracy experiments on dynamic data can be described as follows: (i) for the data set $c$, and the workloads $d$ or $u$ with 200 or 500 training predicates,
an initial relative accuracy rate is calculated and is shown as a value of ordinate in Fig. 11 with the value of abscissa as 0. For 200 training predicates, the initially calculated histograms are not stable and the relative accuracy rates of the subsequent validation predicates have further improvement space, and for 500 training predicates, the initially calculated histograms can reach a stable state and the relative accuracy rate of the subsequent validation predicates will be at the same level as the initial relative accuracy rate.

(ii) 50 source sub ranges and 50 target sub ranges are selected uniformly at random from the whole value ranges of the attribute income. Each sub range has a uniform width of 400. The data in each source sub range will be updated into a corresponding target sub range by executing a SQL update statement. (iii) Execute 800 new validation predicates totally and record a relative accuracy rate every 100 validation predicates which is shown as a value of ordinate in Fig. 11 with the values of abscissa as 1 to 8. (iv) Repeat steps (ii) and (iii) twice again to make the experimental results more objective.

In the actual experiments, the total number of the tuples which are updated in each step (ii) is between 98,170 and 236,829. From the Figure, we can see that after each data update, the relative accuracy rates of the first 100 validation predicates decrease a little in most cases. And they will restore to the normal level little by little. But it is obvious that CETLH can better adapt to data updates than ISOMER, which can be reflected from the following three aspects. Firstly, the speed of the amelioration of CETLH is fasted than ISOMER. For the second 100 validation predicates, the relative accuracy rate of CETLH will rebound in each case. But the downward trend of ISOMER will last longer in some cases such as the first and the second updates in Fig. 11 (a), the second update in Fig. 11 (b), the first and the third updates in Fig. 11 (c) and the first and the second updates in

![Fig. 11. Relative accuracy rates of dynamic data.](image-url)
Fig. 11 (d). Secondly, in most cases, the relative accuracy rates of CETLH are higher than the corresponding ones of ISOMER under data updates. Especially in Fig. 11 (a) and 11 (b), the improvement space of relative accuracy rates of CETLH is filled little by little by the executions of the subsequent validation predicates. But the space budget of histograms of ISOMER always limits the relative accuracy rate at a low level. Thirdly, CETLH shows higher stability than ISOMER and the overall fluctuations of the relative accuracy rates of CETLH are smaller than the ones of ISOMER, which is relevant to the iterative scaling algorithm adopted by ISOMER.

From Figs. 11 (c) and (d), we can see that as the increase of space budget, the overall relative accuracy rates of ISOMER are improved to a level which is very close to CETLH, but the amelioration has only the theoretic meaning because the cost of the amelioration is very high, which will be shown in detail in Section 7.3.

7.3 Performance Experiments

We compare the performances of CETLH and ISOMER in the following experiments. The space budget of ISOMER in these experiments is 200.

The reason that Optimizer is not compared is that it has not performance comparability with CETLH and ISOMER. When we consider the performances of CETLH and ISOMER, besides the cost of the cardinality calculation, the cost of the construction and the update of histograms cannot be ignored. In fact, the latter composes the main part of the whole cost of a QFR-based cardinality estimation approach which is shown as the ordinate in each subfigure of Fig. 12. For Optimizer, the histograms retain unchanged except periodical data scans and the cost of the construction and the update of histograms should not be considered as a part of the whole estimation time. But periodical data scans affect the performance of an actual database seriously, so it is meaningless to only compare the cost of cardinality calculation of Optimizer with the whole cost of QFR-based cardinality estimation approaches.

In Fig. 12, logarithmic ordinates are adopted because of the huge differences of performance between CETLH and ISOMER. And the details of the whole cost of each approach are shown in a table below the corresponding subfigure. In the tables, $C$ and $I$ denote the CETLH approach and the ISOMER approach respectively. Each whole cost is averaged over 10 runs. And for CETLH, the whole execution time is composed of the cost of the update of second-level histograms, the cost of the update of first-level histograms and the cost of cardinality calculation which are denoted by $c_{ssh}$, $c_{sfh}$ and $c_{ccc}$ in the tables. For ISOMER, the whole execution time is composed of the cost of the construction of histograms and the cost of cardinality calculation which are denoted by $c_{ch}$ and $c_{ccc}$ respectively.

For CETLH, the arrival of a predicate can only lead to the call of MLA once to reconstruct one second-level histogram with only 30 buckets. And the follow-up adjustment of two level histograms is very efficient. Therefore, the whole execution time of CETLH is no more than one second in most cases. Compared with CETLH, ISOMER has a very poor performance. Each time constructing a histogram with 200 buckets, it spends more than 14 seconds and the longest time is even up to 109 seconds. Therefore, we do not compare CETLH with the ISOMER approach with more space budgets than 200. And it is impossible to use a cardinality estimation approach with such a performance in an actual commercial database.
8. CONCLUSION

In this paper, we attempt to ameliorate the effect of cardinality estimation by combining traditional proactive approaches with QFRs. To divide the whole value range of an attribute into the smaller sub ranges with relatively uniform data distribution, we construct a first-level histogram by data scan. But it is executed only once to minimize the negative impact on database performance. In order to improve the accuracy of cardinality estimation, second-level histograms are constructed based on inner QFRs. Each second-level histogram can be constructed and updated independently. And in each moment, only one second-level histogram is possible to be constructed or updated, which fully ensure the performance of the whole approach. Simultaneously, the elaborated update mechanism of histograms makes our approach adapt to the data changes well. Extensive comparison experiments have shown that our approach is satisfactory in the accuracy and the performance of cardinality estimation. In the future, we will try to introduce multi-dimensional histograms into the proposed framework.

REFERENCES

of Data, 1984, pp. 256-276.
19. Y. Matias, J. Vitter, and M. Wang, “Wavelet-based histograms for selectivity esti-
A CARDINALITY ESTIMATION APPROACH BASED ON TWO LEVEL HISTOGRAMS

1755


Xudong Lin (蔺旭东) received his Ph.D. degree in School of Computer and Information Technology from Beijing Jiaotong University of China in 2010. He is currently an Associate Professor of Information Engineering at Environmental Management College of China. His main research interests include query optimization, keyword search, semi-structural data, etc.
Xiaoning Zeng (曾晓宁) received her MS degree in School of Information Science and Engineering from Yanshan University of China in 2006. She is currently an Associate Professor of Information Engineering at Hebei Normal University of Science and Technology. Her main research interests include relational database, image processing, etc.

Xiaowei Pu (浦晓威) received her MS degree in School of Electrical Engineering and Information from Northeast Petroleum University of China in 2008. She is currently an Associate Professor of Information Engineering at Environmental Management College of China. Her main research interests include query optimization, information retrieval, etc.

Yanyan Sun (孙岩岩) received her MS degree in School of Information Science and Engineering from Yanshan University of China in 2006. She is currently an Associate Professor of Information Engineering at Environmental Management College of China. Her main research interests include information retrieval, algorithm analysis, etc.