International Journal of Innovative
Computing, Information and Control
Volume 11, Number 1, February 2015

 ISSN 1349-4198
pp. 123–135

CETLH: A NEW HISTOGRAM-BASED CARDINALITY ESTIMATE APPROACH

Xudong Lin[1], Xiaoning Zeng[2] and Xiaowei Pu[1]

[1]Department of Information Engineering
Environmental Management College of China
No. 73, West Hebei Avenue, Qinhuangdao 066004, P. R. China
{fydong_xl; fyy_xun}@126.com

[2]College of Mathematics and Information Technology
Hebei Normal University of Science and Technology
No. 360, West Hebei Avenue, Qinhuangdao 066004, P. R. China
qhdzxn@126.com

Received March 2014; revised July 2014

Abstract. *For the mainstream relational database management systems, histograms play important roles in cardinality estimate. The main histogram-based cardinality estimate approaches can be classified into two categories: the proactive approach and the reactive approach. For the former, histograms are constructed and updated by periodical data scans. Data updates can not be incorporated into a histogram in real time, so between two data scans, large errors of cardinality estimate will occur. For the latter, data scans are avoided, as an alternative, query feedback records (QFRs) are collected to construct and update histograms. Although data updates can be incorporated into a histogram by replacing stale QFRs in real time, the cost of time is very expensive. For each histogram reconstruction, all buckets in the histogram will be recalculated and the large amount of computation leads to the inefficiency of the reactive approach. In this paper, we propose a novel QFR-based cardinality estimate approach which can balance the efficiency issue and the data update issue: on the one hand, it can improve the efficiency of QFR-based cardinality estimate to a practical level; on the other hand, it can incorporate data updates into a histogram in real time to fully ensure the accuracy of cardinality estimate. In our approach, a serial of small second-level histograms covering different parts of the whole value range of an attribute will be constructed. These second-level histograms can be updated independently over time to ensure the performance of the approach. As the update of each second-level histogram, QFRs still play their important roles in incorporating data updates into the histogram in real time. Extensive comparison experiments fully demonstrate the advantages of our approach in performance and accuracy.*
Keywords: Cardinality estimate, Proactive approach, Reactive approach, Histogram, Query feedback record

1. **Introduction.** Cardinality estimate is an important problem in query optimizations. The choices of query plans rely heavily on the accuracy of cardinality estimate. For the mainstream relational database management systems, histograms play important roles in cardinality estimate. The first approach using histograms to approximate data distributions within a database system is proposed 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 is partitioned into smaller sub ranges with equal widths. [2] proposes the equi-depth histogram and its multi-dimensional form is presented in [3]. [4] proposes the histogram with frequency as the sort parameter, which represents the first departure from the value-based grouping of buckets. Several years later, it is improved

with the forms of v-optimal histogram [5], maxdiff histogram, compressed histogram [6] and entropy-based histogram [7]. All above approaches construct histograms through data scans, so they can be uniformly called proactive approaches. For this kind of approaches, data scans must be executed periodically to make histograms consistent with underlying data, which lead to the following drawbacks:

- Data updates cannot be incorporated into a histogram in real time. Between two reconstructions of a histogram, data updates may lead to large errors of cardinality estimate.
- Periodical data scans aggravate the database overhead and affect the performance of routine queries.

To avoid periodical data scans but 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. Hereafter, the similar approaches are proposed continuously [9,10]. In this paper, we call this kind of approaches the reactive approach. For this kind of approaches, data scans can be avoided, and data updates can be incorporated into histograms by replacing outdated or invalid QFRs. However, some new deficiencies, especially the efficiency issue, prevent the reactive approach being practical. We can summarize the deficiencies of the representative reactive approach ISOMER [10] as follows:

- Although data scans are avoided, the whole histogram covering the complete domain of an attribute will be recalculated in each histogram reconstruction, and the large amount of computation becomes the most important reason of its inefficiency.
- Some time-consuming operations of ISOMER including replacing stale QFRs, drilling holes or merging buckets, and executing the iterative scaling algorithm (ISA) [11] also make the whole approach inefficient.
- Besides the poor performance, the ISA adopted by ISOMER is unstable and vulnerable.
- The criteria of ISOMER deciding whether a QFR is outdated or invalid are unreasonable. These criteria cannot ensure the accuracy of the retained QFRs and lead to the errors of histogram recalculation.

The serious deficiencies in the proactive approach and the reactive approach prompt us to find new notions to improve the accuracy and the performance of the histogram-based cardinality estimate approach. In [10], the combination of ISOMER with proactive approach is mentioned. And it is possible to ameliorate the existing approaches by combining their respective advantages. However, unfortunately, no specific form of the combination is proposed in [10].

In this paper, we propose a specific combination form of the proactive approach and the reactive approach for the first time. 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. The second-level histograms corresponding to all buckets of the first-level histogram compose the second-level histogram set which will play main roles in cardinality estimate. Therefore, we call our approach the Cardinality Estimate approach based on Two Level Histograms (CETLH).

Example 1.1. *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 Figure 1. The first-level histogram and the second-level histograms constructed over the attribute order_quantity are shown in Figure 2.

order_id	customer_id	product_id	order_date	order_quantity
1	c1	p1	20120115	100
2	c1	p200	20120130	1
3	c2	p4	20120306	15,016
4	c2	p80	20120321	99,999
...				
200,000	c2132	p35	20130128	23,000

FIGURE 1. Relation orderInfo

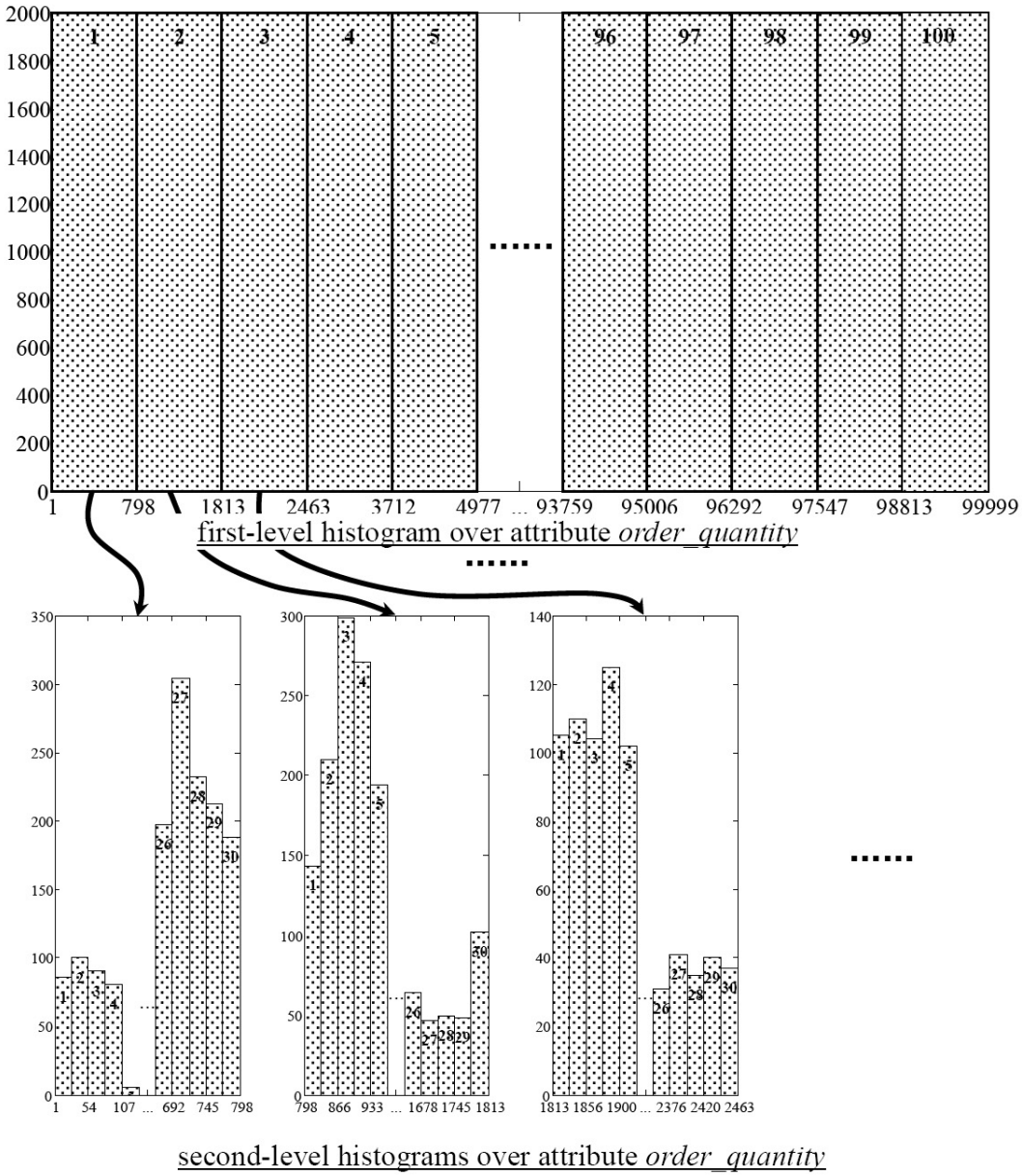


FIGURE 2. Two level histograms

In Example 1.1, the first-level histogram over the attribute *order_quantity* contains 100 buckets, so there exist also 100 second-level histograms totally. Each second-level histogram contains 30 buckets. Compared with the proactive approach and the reactive approach, the main improvements of CETLH can be summarized as:

- For each histogram reconstruction, only the buckets in one second-level histogram will be recalculated, the computation will be decreased remarkably and the efficiency of cardinality estimate can be fully ensured;
- The bucket number in each second-level histogram is fixed and no time-consuming operations such as replacing stale QFRs, drilling holes or merging buckets are needed, which also improve the efficiency of CETLH;
- A novel mechanism is proposed to locate the ranges of remarkable data updates accurately and incorporate these updates into histograms in real time.

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

2. **Notations and Preliminary.** In this paper, without special explanations, the letters r, a, p, h and their subscripted forms such as r_x, a_x, p_x, h_x are used to denote the common concepts *relation*, *attribute*, *predicate* and *histogram* in a relational database. And the letter b and its subscripted form denote the *bucket* of a histogram. The letter v and its variations are usually used to denote values.

The value range of the attribute a is denoted by $rg(a)$ and the number of values in $rg(a)$ is denoted by $|rg(a)|$. The value range of an attribute at which a predicate p is true can be denoted by $rg(p)$. The number of tuples which fall in a bucket b is denoted by $v(b)$, and $rg(b)$ denotes the value range of an attribute covered by a bucket b. Based on $rg(b)$ and $v(b)$, a bucket b can be extended to a triple as necessary, i.e., $b = (\min(rg(b)), \max(rg(b)), v(b))$. In the paper, $\min(x)$ and $\max(x)$ are used to denote the minimum and the maximum of the data set x.

For a histogram h, all of its buckets compose a bucket set which can be denoted by $B(h)$. The number of buckets in $B(h)$ is denoted by $|B(h)|$. The value range of an attribute covered by a histogram h is denoted by $rg(h)$. $rg(h) = \bigcup_{b \in B(h)} rg(b)$. For an equi-depth histogram h, the number of tuples which fall in each bucket $b_i \in B(h)$ is equal and we call this number the *depth* of h and denote it by $d(h)$. For any $b \in B(h)$, $d(h) = v(b)$. For an equi-width histogram h, the value range of an attribute covered by each bucket $b \in B(h)$ is equal and we call this value range the *width* of h and denote it by $w(h)$. For any $b \in B(h)$, $w(h) = \max(rg(b)) - \min(rg(b))$.

In this paper, a histogram may be a first-level histogram, or a second-level histogram, which can be marked as h^f and h^s respectively. A h^f constructed over the attribute a can be marked as h^f_a. For a h^s constructed at the value range of the attribute a covered by the bucket $b \in B(h^f_a)$, h^s can be marked as $h^s_{a,b}$.

For a predicate p, the statistical information obtained from a query execution engine which records the number of tuples satisfying p is called the *Query Feedback Record* (*QFR*) of the predicate p and can be denoted by $qfr(p)$. Hereinafter, a predicate p can be extended to a two-tuple as necessary, i.e., $p = (\min(rg(p)), \max(rg(p)))$, and a QFR $qfr(p)$ can be extended to a triple $qfr(p) = (\min(rg(p)), \max(rg(p)), v(qfr(p)))$ where $v(qfr(p))$ denotes the actual number of tuples satisfying p which is obtained from a query execution engine.

If a predicate p and a bucket $b \in B(h_a^f)$ satisfy $rg(p) \subseteq rg(b)$, the QFR $qfr(p)$ is called an *inner QFR* of b and can be marked as $qfr^b(p)$. $qfr^b(p)$ can be used to calculate the second-level histogram $h_{a,b}^s$. One or more inner QFRs can compose an *inner QFR set*. 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 $h_{a,b}^s$.

It is necessary to emphasize that non-inner QFRs can be transformed into inner QFRs as follows. Given a predicate p and t adjacent buckets $b_m, \ldots, b_{m+t-1} \in B(h_{a_i}^f)$ which satisfy $rg(p) \not\subset rg(b_i)$ and $rg(p) \cap rg(b_i) \neq \varnothing$ 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 $rg(p_j) \subseteq rg(b_j)$ for $j = m, \ldots, m+t-1$. And then, we can get $v(qfr(p)) = \sum_{i=m}^{m+t-1} v(qfr^{b_i}(p_i))$. In this formula, only two end values $v(qfr^{b_m}(p_m))$ and $v(qfr^{b_{m+t-1}}(p_{m+t-1}))$ are unknown, and all medial values can be get as $v(qfr^{b_i}(p_i)) = v(b_i)$ for $i = m+1, \ldots, m+t-2$. Therefore, if the second-level histogram h_{a,b_m}^s is known, we can calculate $v(qfr^{b_m}(p_m))$. And the non-inner QFR $qfr(p)$ can be transformed into the inner QFR $qfr^{b_{m+t-1}}(p_{m+t-1})$ to calculate $h_{a,b_{m+t-1}}^s$ and vice versa. Sometimes, both h_{a,b_m}^s and $h_{a,b_{m+t-1}}^s$ may be unknown. In this case, if we want to calculate $h_{a,b_{m+t-1}}^s$, we use $v(b_m)$ and the traditional *uniformity assumption* to calculate the approximate $v(qfr^{b_m}(p_m))$ 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 calculation accuracy of second-level histograms can be ameliorated rapidly with the increase of the number of the calculated second-level histograms.

3. **Construction of First-Level Histogram.** In our approach, data scans need to be executed only once to construct the initial first-level equi-depth histogram. The main purpose of constructing a first-level histogram is not to estimate the cardinalities of predicates, but 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. The uniformity assumption is applied within each bucket of a first-level histogram. Therefore, for a predicate p and a bucket b satisfying $rg(p) \cap rg(b) \neq \varnothing$, the number of tuples fall in the bucket b for which p is true can be estimated as:

$$n_b(p) = v(b) * \frac{vol(rg(p) \cap rg(b))}{vol(rg(b))} \tag{1}$$

where $vol(R)$ denotes the usual Euclidean volume of the region R when the data is real-valued; for discrete data, $vol(R)$ 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.

4. **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. Before describing the details of the construction of second-level histograms, the core algorithm of the construction, *Minimum-norm Least-squares Algorithm* (*MLA*), will be introduced firstly.

4.1. **Minimum-norm least-squares algorithm.** The main function of MLA is to calculate a second-level histogram based on a currently used inner QFR set of it. We conceive that a second-level histogram h^s with k buckets $b_1, \ldots, b_k$, and the inner QFR set of h^s with t inner QFRs $qfr(p_1), \ldots, qfr(p_t)$ can compose the following linear system:

$$\begin{aligned} &n_{b_1}(p_1) + n_{b_2}(p_1) + \ldots + n_{b_k}(p_1) = v(qfr(p_1)) \\ &n_{b_1}(p_2) + n_{b_2}(p_2) + \ldots + n_{b_k}(p_2) = v(qfr(p_2)) \\ &\ldots \\ &n_{b_1}(p_t) + n_{b_2}(p_t) + \ldots + n_{b_k}(p_t) = v(qfr(p_t)) \end{aligned} \tag{2}$$

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

MLA [12] 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 [12].

4.2. **Steps of construction.** For a relation $r(a_1, \ldots, a_n)$ and a first-level histogram $h^f_{a_i}$, $1 \leq i \leq n$, $B(h^f_{a_i}) = \{b_1, \ldots, b_m\}$. At the value range of a_i covered by a bucket $b_j \in B(h^f_{a_i})$, $1 \leq j \leq m$, the second-level histogram $h^s_{a_i,b_j}$ can be constructed as follows:

(1) Set $|B(h^s_{a_i,b_j})|$, the number of buckets in $B(h^s_{a_i,b_j})$.
(2) Apply the uniformity assumption on first-level histogram and create the *initial* second-level histogram $h^s_{a_i,b_j}$. To distinguish with the initial second-level histograms, the second-level histograms calculated by MLA are called the *normal* second-level histograms hereinafter.
(3) Allocate an array arr^{b_j} to store the inner QFRs of b_j.
(4) Collect inner QFRs from the query execution engine and store them into arr^{b_j}.
(5) Set $|arr^{b_j}|_{\min}$, the minimum number of inner QFRs stored in arr^{b_j} which can trigger the call of MLA. Once $|arr^{b_j}| = |arr^{b_j}|_{\min}$, the newly arrived inner QFRs are still stored into arr^{b_j} and MLA will be called to calculate $h^s_{a_i,b_j}$.

4.3. **Cardinality estimate.** For a relation $r(a_1, \ldots, a_n)$ and a first-level histogram $h^f_{a_i}$, $1 \leq i \leq n$, $B(h^f_{a_i}) = \{b_1, \ldots, b_m\}$. Given that for a predicate p, there exist t adjacent buckets $b_o, \ldots, b_{o+t-1} \in B(h^f_{a_i})$ which satisfy $rg(b_{o+j}) \cap rg(p) \neq \varnothing$ 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 $h^s_{a_i,b_j}$ 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 $B(h^s_{a_i,b_j}) = \{b_{j1}, \ldots, b_{jk}\}$:

$$n_{b_j}(p) = \sum_{q=1}^{k} v(b_{jq}) * \frac{vol(rg(p) \cap rg(b_{jq}))}{vol(rg(b_{jq}))} \tag{3}$$

And the cardinality of p can be estimated as:

$$n(p) = \sum_{j=1}^{m} n_{b_j}(p) \tag{4}$$

5. **Automatically Incorporation of Data Updates.** In CETLH, we 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. 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.

5.1. **Update of second-level histograms.** For a relation $r(a_1, \ldots, a_n)$ and a first-level histogram $h^f_{a_i}$, $1 \le i \le n$, $B(h^f_{a_i}) = \{b_1, \ldots, b_m\}$. At the value range of a_i covered by a bucket $b_j \in B(h^f_{a_i})$, $1 \le j \le m$, the second-level histogram $h^s_{a_i,b_j}$ contains k buckets which compose the bucket set $B(h^s_{a_i,b_j}) = \{b_{j1}, \ldots, b_{jk}\}$. When the data at the value range of a_i covered by the bucket b_j change remarkably, the update of $h^s_{a_i,b_j}$ can be completed as follows:

(1) Set nq, the number of progressively arrived new inner QFRs of b_j which can be used to update $h^s_{a_i,b_j}$ together.
(2) Allocate an array $arr^{b_j}_{new}$ with nq storage spaces to temporarily store the accumulated new inner QFRs of b_j.
(3) Collect nq new inner QFRs of b_j from the query execution engine and store them into $arr^{b_j}_{new}$.
(4) For each new inner QFR $qfr^{b_j}(p_w)$, $1 \le w \le nq$ of b_j stored in $arr^{b_j}_{new}$, find the bucket set $\{b_{js_w}, \ldots, b_{je_w}\} \subseteq B(h^s_{a_i,b_j})$ which satisfy $rg(b_{jx}) \cap rg(p_w) \neq \varnothing$, $s_w \le x \le e_w$.
(5) Set an error threshold th_1 and calculate $n_{b_j}(p_w)$ using Formula (3).
(6) If $abs(n_{b_j}(p_w) - v(qfr^{b_j}(p_w))) \le th_1$, all $b_{jx}, s_w \le x \le e_w$ will be marked as *unchanged*. If $abs(n_{b_j}(p_w) - v(qfr^{b_j}(p_w))) > th_1$, the *unmarked* $b_{jx}, s_w \le x \le e_w$ will be marked as *changed*. The remarkable data updates can be located inside the value ranges of a_i covered by the $b_{jx} \in B(h^s_{a_i,b_j})$ with *change* state.
(7) Clear arr^{b_j}.
(8) Reconstruct the currently used inner QFR set of b_j.
(9) Call MLA to reconstruct the second-level histogram $h^s_{a_i,b_j}$ based on the reconstructed currently used inner QFR set of b_j.

5.2. **Update of first-level histogram.** For a relation $r(a_1, \ldots, a_n)$ and a first-level histogram $h^f_{a_i}$, $1 \le i \le n$, $B(h^f_{a_i}) = \{b_1, \ldots, b_m\}$. At the value range of a_i covered by a bucket $b_j \in B(h^f_{a_i})$, $1 \le j \le m$, the second-level histogram $h^s_{a_i,b_j}$ contains k buckets which can be denoted by $B(h^s_{a_i,b_j}) = \{b_{j1}, \ldots, b_{jk}\}$. When the data change remarkably at the value range of a_i covered by the bucket b_j, the first-level histogram $h^f_{a_i}$ and its corresponding second-level histograms will be updated as follows:

(1) Recalculate the number of tuples which fall in $b_j \in B(h^f_{a_i})$, $1 \le j \le m$ as $v(b_j) = \bigcup_{b_{jx} \in B(h^s_{a_i,b_j})} v(b_{jx})$, where each $v(b_{jx})$, $1 \le x \le k$ has been recalculated with the steps described in Section 5.1.
(2) Set an error threshold th_2 and update $v(b_j)_{\max}$ or $v(b_j)_{\min}$, the maximum and the minimum of all $v(b_j)$ for $j = 1, \ldots, m$ if necessary.
(3) If $v(b_j)_{\max} - v(b_j)_{\min} \ge th_2$, recalculate a new depth of $h^f_{a_i}$ as $d(h^f_{a_i}) = \frac{1}{m} * \sum_{b_j \in B(h^f_{a_i})} v(b_j)$.
(4) Adjust the borders of all second-level histograms $h^s_{a_i,b_1}, \ldots, h^s_{a_i,b_m}$ to ensure each new second-level histogram $h^s_{a_i,b_j}, 1 \le j \le m$ satisfying $\bigcup_{b_{jx} \in B(h^s_{a_i,b_j})} v(b_{jx}) = d(h^f_{a_i})$.
(5) The buckets in each second-level histogram will be reconstructed as $w(h^s_{a_i,b_j}) = \frac{\max(rg(h^s_{a_i,b_j})) - \min(rg(h^s_{a_i,b_j}))}{|B(h^s_{a_i,b_j})|}$, where $|B(h^s_{a_i,b_j})|$ remains unchanged and equals the value which is set in the first step of Section 4.2.

(6) Corresponding to the new borders of each second-level histogram, adjust the borders of the buckets of the corresponding first-level histogram.
(7) Allocate an array arr_{trf} to temporarily store the newly *transformed* inner QFRs which can be used to calculate the number of tuples which fall in each bucket of the new $h^s_{a_i,b_j}$ obtained in Step 5.
(8) Transform the buckets of the old second-level histograms $h^s_{a_i,b_1}, \ldots, h^s_{a_i,b_m}$ into inner QFRs and stored them into arr_{trf}.
(9) Call MLA to calculate the number of tuples which fall in each bucket of the new second-level histograms $h^s_{a_i,b_j}$ for $j = 1, \ldots, m$ based on the inner QFRs stored in arr_{trf}.

6. Experiments.

The experiments are performed on a 3.2GHz Intel CPU machine running Windows XP sp3, with 4GB memory and 1TB hard disk.

6.1. Experimental settings.

6.1.1. *Data sets.*

Real data set (denoted by *c*): A relation *census_1990_raw* is created in the commercial relation database Oracle [13] to store the USCensus1990raw [14] data which consists of 2,458,285 tuples. A 1-dimensional equi-depth histogram with 100 buckets is constructed over the attribute *income1* with $|rg(income1)| = 55,088$, $\min(rg(income1)) = 0$ and $\max(rg(income1)) = 197,927$. All experiments are carried out over the attribute *income1*.

Synthetic data set I (denoted by *gz*): A relation *gauss_zip* with a single attribute *data* is created in the commercial relation database Oracle to store this data set which consists of 500,000 tuples sampled from 30 Gaussians [15] 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 Zipfian distribution [15] with the skew parameter $z = 1$.

Synthetic data set II (denoted by *gu*): A relation *gauss_uniform* with a single attribute *data* is created in the commercial relation database Oracle to store this data set which 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.

6.1.2. *Workloads.*

For each workload 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 Zipfian 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.

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

$$re(p) = \frac{abs(n(p) - v(qfr(p)))}{v(qfr(p))} \tag{5}$$

And then, we define the *relative accuracy rate*, $rar(ce)$, of a cardinality estimate approach ce as the criterion to measure the accuracy of ce:

$$rar(ce) = \frac{cn_s(ce)}{tn(ce)} \tag{6}$$

where $cn_s(ce)$ 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.

6.1.4. *Programs.* In the experiments, the comparison approaches of cardinality estimates 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. All cardinality approaches are realized under JDK 1.6.0_10.

6.2. **Accuracy experiments.**

6.2.1. *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 estimates of different approaches obtained by running the workloads d and u over the datasets c, gz and gu are shown in Figure 3(a) to Figure 3(f) respectively.

It is easy to see that the CETLH approach has the highest accuracy and stability in all cases. 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. Without second-level histograms, the relative accuracy rate of the Optimizer always wanders at a relatively low level. The tendency of the relative accuracy rate of

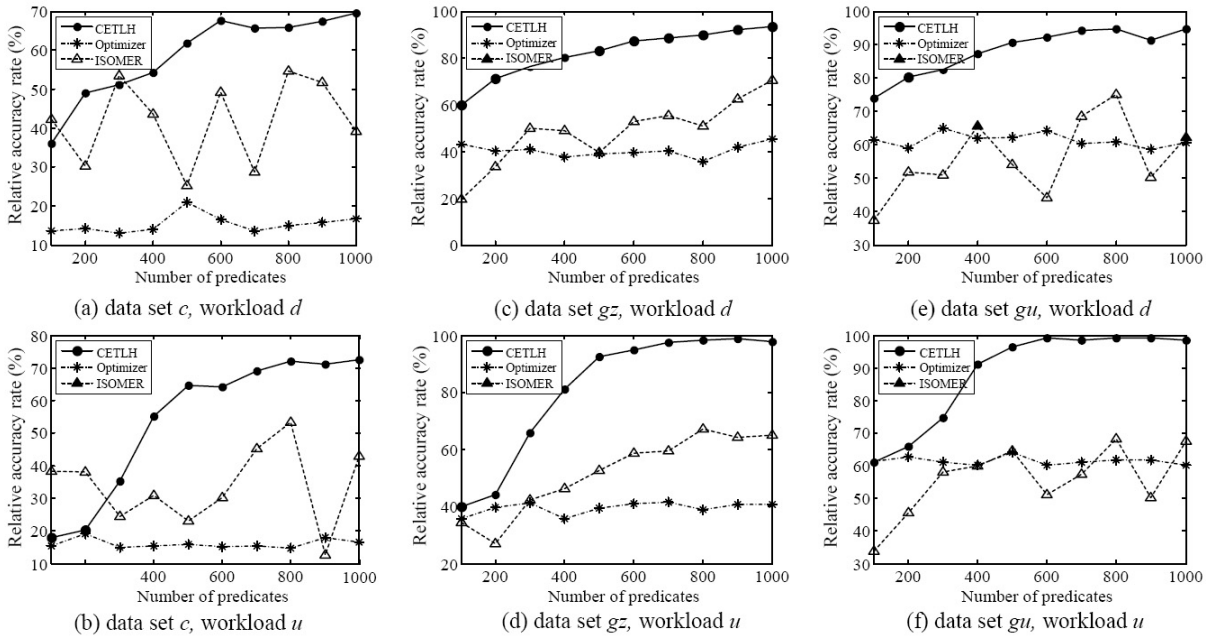
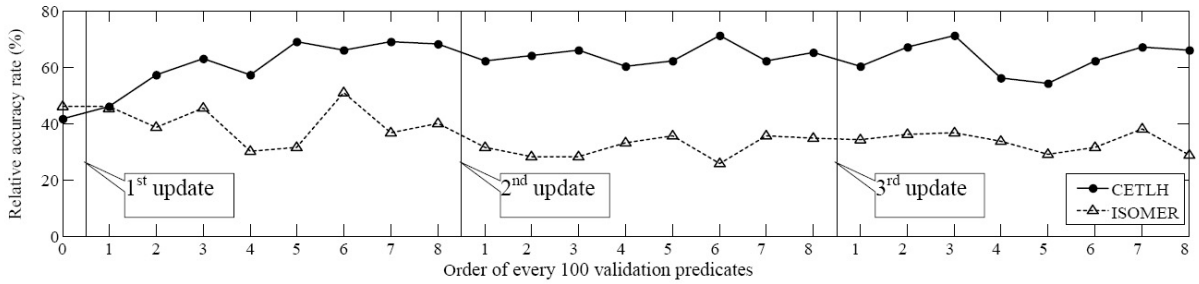


FIGURE 3. Relative accuracy rates of different approaches

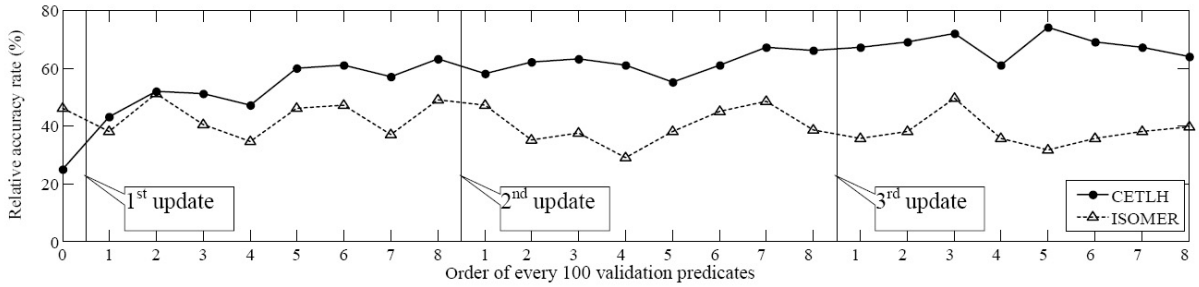
ISOMER is very different from the ones of CETLH and Optimizer, and smooth changes are replaced by the frequent and significant fluctuations, which testify the mentioned instability and vulnerability of ISOMER.

6.2.2. *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 Figure 4.

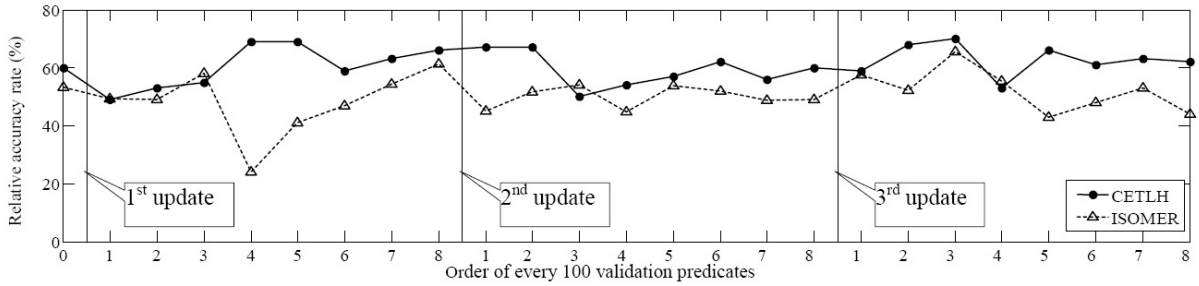
The process of the accuracy experiments on dynamic data can be described as follows: (i) for the data set c, and the workloads d or u with 200 or 500 training predicates, an initial relative accuracy rate is calculated and is shown as a value of ordinate in Figure



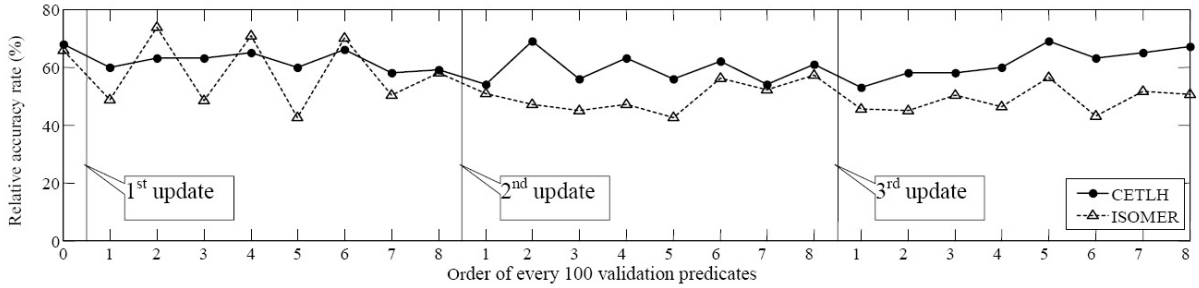
(a) data set c, workload d (200 initial predicates, 100 update predicates)



(b) data set c, workload u (200 initial predicates, 100 update predicates)



(c) data set c, workload d (500 initial predicates, 100 update predicates)



(d) data set c, workload u (500 initial predicates, 100 update predicates)

FIGURE 4. Relative accuracy rates of dynamic data

4 with the value of abscissa as 0; (ii) 50 *source* sub ranges and 50 *target* sub ranges are selected uniformly at random from the whole value ranges of the attribute *income1*. 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 an 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 Figure 4 with the values of abscissa as 1 to 8; (iv) Repeat steps (ii) and (iii) twice again to make the experimental results more objective.

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. However, 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 faster than ISOMER; secondly, in most cases, the relative accuracy rates of CETLH are higher than the corresponding ones of ISOMER under data updates; 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.

6.3. **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 construction and the update of the histograms used in CETLH and ISOMER rely on QFRs. As the new QFRs arriving continuously, histograms will be reconstructed frequently. Therefore, when we consider the performances of CETLH and ISOMER, except the cost of the cardinality calculation of a predicate, the time 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 estimate approach and is shown as the ordinate in Figure 5. In the figure, logarithmic ordinates are adopted because of the huge differences of performance between CETLH and ISOMER. For CETLH, the arrival of a predicate can only lead to the call of MLA once to reconstruct one second-level histogram

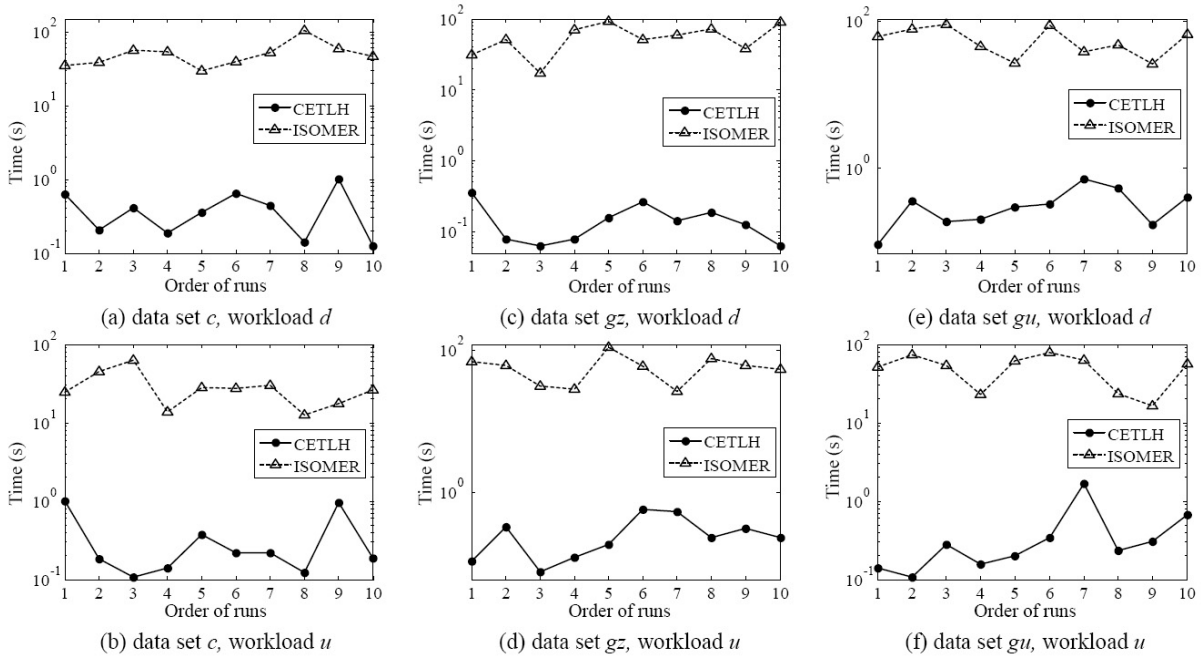


FIGURE 5. Performances of different approaches

with only 30 buckets. And the follow-up adjustment of two level histograms is very efficient and can be finished within 200 milliseconds in general for the data sets *c*, *gz* and *gu*. Compared with CETLH, ISOMER has a very poor performance. Each time constructing a histogram with 200 buckets using ISOMER approach, it spends more than 14 seconds and the longest time is even up to 109 seconds. And it is impossible to use a cardinality estimate approach with such a performance in an actual commercial database.

7. **Conclusion.** In this paper, we attempt to ameliorate the effect of cardinality estimate by combining the proactive approach with QFRs. To improve the performance of histogram update, the whole value range of an attribute is divided into smaller sub ranges with relatively uniform data distribution according to a first-level histogram. Within each sub range, a second-level histogram is constructed according to inner QFRs. And in each moment, only one second-level histogram is possible to be constructed or updated, which can bring two great improvements: (i) the computation of each histogram update will be decreased remarkably and the efficiency of cardinality estimate can be fully ensured; (ii) data updates can be incorporated into histograms by inner QFRs in real time to make histograms always consistent with underlying data. Extensive comparison experiments have shown that our approach is satisfactory in the accuracy and the performance of cardinality estimate. In the future, we will introduce the v-optimal histogram, the maxdiff histogram and the compressed histogram into the proposed framework to find an optimal solution.

REFERENCES

[1] R. P. Kooi, *The Optimization of Queries in Relational Databases*, PhD Thesis, Case Western Reserve University, 1980.

[2] G. Piatetsky-Shapiro and C. Connell, Accurate estimate of the number of tuples satisfying a condition, *Proc. of ACM SIGMOD International Conference on Management of Data*, Boston, MA, USA, pp.256-276, 1984.

[3] M. Muralikrishna and D. J. DeWitt, Equi-depth histograms for estimating selectivity factors for multi-dimensional queries, *Proc. of ACM SIGMOD International Conference on Management of Data*, Chicago, IL, USA, pp.28-36, 1988.

[4] Y. E. Ioannidis and S. Christodoulakis, Optimal histograms for limiting worst-case error propagation in the size of join results, *Transactions on Database Systems (ACM)*, vol.18, no.4, pp.709-748, 1993.

[5] Y. E. Ioannidis and V. Poosala, Balancing histogram optimality and practicality for query result size estimate, *Proc. of ACM SIGMOD International Conference on Management of Data*, San Jose, CA, USA, pp.233-244, 1995.

[6] V. Poosala, Y. E. Ioannidis, P. J. Haas and E. J. Shekita, Improved histograms for selectivity estimate of range predicates, *Proc. of ACM SIGMOD International Conference on Management of Data*, Montreal, Quebec, Canada, pp.294-305, 1996.

[7] H. To, K. Chiang and C. Shahabi, Entropy-based histograms for selectivity estimate, *Proc. of the 22nd ACM International Conference on Information and Knowledge Management*, San Francisco, CA, USA, pp.1939-1948, 2013.

[8] A. Aboulnaga and S. Chaudhuri, Self-tuning histograms: Building histograms without looking at data, *Proc. of ACM SIGMOD International Conference on Management of Data*, Philadelphia, PA, USA, pp.181-192, 1999.

[9] N. Bruno, S. Chaudhuri and L. Gravano, STHoles: A multidimensional workload-aware histogram, *Proc. of ACM SIGMOD International Conference on Management of Data*, Santa Barbara, CA, USA, pp.294-305, 2001.

[10] U. Srivastava, P. J. Haas, V. Markl, M. Kutsch and T. M. Tran, Isomer: Consistent histogram construction using query feedback, *Proc. of the 22nd International Conference on Data Engineering*, Atlanta, Georgia, USA, p.39, 2006.

[11] J. N. Darroch and D. Ratcliff, Generalized iterative scaling for log-linear models, *The Annals of Mathematical Statistics*, vol.43, no.5, pp.1470-1480, 1972.

[12] Å. Björck, *Numerical Methods for Least Squares Problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.
[13] *Oracle Database SQL Tuning Guide 12c Release 1 (12.1), E15858-15*, http://docs.oracle.com/cd/E16655_01/server.121/e15858.pdf, 2013.
[14] D. Aha, P. Murphy et al., *UCI Repository of Machine Learning Databases*, http://archive.ics.uci.edu/ml/index.html, 2013.
[15] S. A. William, H. Press, B. P. Flannery and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, UK, 1993.