JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 35, 201-222 (2019)
DOI: 10.6688/JISE.201901_35(1).0011

Cardinality Estimation Based on Cluster Analysis

XIAONING ZENG[1], XUDONG LIN[2], CAIYAN PEI[1] AND JING CAO[1]
[1]*School of Mathematics and Information Science and Technology*
Hebei Normal University of Science and Technology
Hebei, 066004 P.R. China
E-mail: {qhdzxn; peicaiyan78; caojxxf}@163.com
[2]*Department of Information Engineering*
Hebei University of Environmental Engineering
Hebei, 066004 P.R. China
E-mail: fydong_xl@126.com

In the cardinality estimation solutions based on multi-dimensional self-tuning histograms, periodical data scans are avoided and self-tuning histograms are constructed according to query feedback records (QFRs). We call this kind of cardinality estimation solutions the *reactive solutions*. In the existing reactive solutions, self-tuning histograms are constructed over the entire value ranges of the queried attributes and have large scales. The bucket number of a multi-dimensional self-tuning histogram increases exponentially with the dimension. That means the existing reactive solutions are stuck with the issue of "curse of dimension". Simultaneously, to construct and maintain a multi-dimensional self-tuning histogram over the entire value ranges of the queried attributes, a large number of QFRs must be accumulated at a long time span, and the cumbersome operations have to be executed repeatedly to meet a space budget, which makes the existing reactive solutions unpredictable and time-consuming.

To address above issues, a new reactive solution is proposed in the paper. In the solution, the global self-tuning histogram covering the entire value range of the queried attributes is abandoned. When a new predicate p is executed, the Ward's minimum variance method is used to find k nearest QFRs with p from the QFR warehouse. Based on the found k QFRs, a micro self-tuning histogram only covering the neighborhood of p is constructed to help estimate the cardinality of p. The solution can be considered as a beneficial attempt to improve the cardinality estimation efficiency under high dimensions, and notably alleviate the issue of "curse of dimension". Furthermore, old QFRs can be replaced rapidly by the new QFRs and the data and workload changes can be timely reflected by micro self-tuning histograms. The process of meeting a space budget is eliminated completely, which makes the whole solution reliable and dexterous.

Keywords: cardinality estimation, cluster analysis, ward's minimum variance method, self-tuning, query feedback record

1. INTRODUCTION

In a query optimizer, cardinality estimation plays an important role in choosing optimal query plans. For a predicate referring to multiple attributes, if the cardinality is estimated only based on 1-dimensional data summarization techniques, the correlations among multiple attributes are ignored and the accuracy of cardinality estimation cannot be guaranteed. Therefore, multi-dimensional data summarization techniques have important practical significance for cardinality estimations.

Received June 14, 2017; revised September 26, 2017; accepted November 22, 2017.
Communicated by Meng Chang Chen.

Traditional multi-dimensional cardinality estimation solutions rely on data scans. Therefore, these solutions are called the *proactive solutions* in the paper. The first proactive solution is based on the multi-dimensional equi-depth histogram [1]. And then, the improved proactive solutions are proposed continuously [2-7]. The proactive solutions proposed in [8, 9] are based on the multi-dimensional wavelet transforms [10, 11]. But none of the existing proactive solutions is actually adopted in the mainstream databases because of two serious deficiencies:

(1) Lots of system resources are occupied by periodical data scans and the performance of routine queries are influenced seriously.
(2) The solutions are stuck with the "curse of dimension".

The cardinality estimation solution in [12] is different from the proactive solutions. It uses the multi-dimensional self-tuning histogram to replace the proactive data summarization technologies. A multi-dimensional self-tuning histogram is constructed and maintained based on query feedback records (QFRs). We call this kind of multi-dimensional cardinality estimation solutions the *reactive solutions* in the paper. The succeeding reactive solution in [13] shows how to build low-dimensional self-tuning histograms from high-dimensional queries using the delta rule. The reactive solution in [14] improves the accuracy of a self-tuning histogram by subtilizing the granularity of QFRs. The information-theoretic principle of maximum entropy is introduced in [15]. [16] uses the equi-width approach and the sparse-vector recovery based approach to maintain self-tuning histograms. [17] initializes a multi-dimensional self-tuning histogram based on subspace clustering. [18] tries to improve the efficiency of maintaining self-tuning histograms by constructing two level histograms.

Summing up the existing reactive solutions, periodical data scans are avoid, but they are not yet practical due to the following common issues:

(1) Reactive solutions are still stuck with the "curse of dimension". In the existing reactive solutions, self-tuning histograms are constructed and maintained over the entire value range of the queried attributes (henceforth called *global* self-tuning histograms). As dimension increases, the bucket number of a global self-tuning histogram increases exponentially just as a proactive histogram.
(2) To construct and maintain a global self-tuning histogram, a large number of QFRs must be accumulated at a long time span. As the changes of data and workload, the accumulated QFRs may become inaccurate and contradictory for each other.
(3) To limit the bucket number of a global self-tuning histogram, the space budgets is widely adopted, which leads to accuracy deterioration of cardinality estimation. Furthermore, the process of reducing bucket number may be called more than once, which makes reactive solutions unpredictable and time-consuming.

To address above issues, a new reactive solution – the *Cardinality Estimation solution applying Ward's minimum variance Method* (*CEWM*) is proposed in the paper. When a new predicate p is executed, the *Ward's minimum variance method* (*Ward method* for short) [19] is used to find k nearest QFRs with p from the QFR warehouse. And a *micro* self-tuning histogram only covering the neighborhood of p is constructed to help estimate the cardinality of p. The main contributions of CEWM can be summarized as:

(1) The Ward method is introduced firstly to find k nearest QFRs for a new predicate p. The executed predicates which locate in the neighborhood of p have the relatively similar cardinalities with p. The Ward method is used to find these predicates according to its function of clustering similar classes.
(2) Instead of using a global self-tuning histogram, a micro self-tuning histogram is constructed swiftly based on a small number of QFRs of the similar executed predicates. It only covers the neighborhood of a new predicate and can be considered as a beneficial attempt to alleviate the issue of "curse of dimension".
(3) After the execution of each new predicate, QFR warehouse is updated by the corresponding new QFR, and the data and workload changes can be timely reflected by the micro self-tuning histogram.
(4) Due to the small scale of a micro self-tuning histogram, space budget is unnecessary and the process of reducing bucket number can be eliminated completely, which make the whole solution reliable and dexterous.

The rest of the paper is organized as follows. Section 2 gives the notations. Section 3 describes the details of finding k nearest QFRs using the Ward method. The micro histogram and its construction process are analyzed in Section 4. Based on the micro histogram, the processes of cardinality estimation for different cases are given in Section 5. The new QFR update mechanism is explained in Section 6. The results of extensive experiments are demonstrated in Section 7. Section 8 discusses the related work, and Section 9 summarizes the paper and discusses future directions.

2. NOTATIONS

All notations used in the paper are shown in Table 1.

Table 1. Notations.

basic notations	meanings
$min(x)$, $max(x)$	the minimum and the maximum of the data set x
$\|x\|$	the usual Euclidean volume of the area x when the data are real-valued; for discrete data, $\|x\|$ denotes the number of discrete points that lie in x.
r and its subscripted forms	relation
t and its subscripted forms	tuple
a and its subscripted forms	attribute
q and its subscripted forms	query
v and its subscripted forms	value
p and its subscripted forms	predicate: a predicate p has the form $(v_{11} \leq a_1 \leq v_{12}) \wedge (v_{21} \leq a_2 \leq v_{22}) \wedge \ldots \wedge (v_{m1} \leq a_m \leq v_{m2})$ where $a_1, a_2, \ldots, a_m$ are the different attributes in one relation. v_{i1} and v_{i2} are two values within the value range of a_i which satisfy $v_{i1} \leq v_{i2}$ for $i = 1, 2, \ldots, m$. Given a predicate p, if it is just submitted and will be executed soon, p is called a new predicate; if p has been executed and the QFR of p has been collected, p is called an executed predicate.
h and its subscripted forms	histogram
b and its subscripted forms	the bucket of a histogram

notations related to an attribute	meanings
$d(a)$	the value range of a
$d(b, a)$	the value range of a which is covered by b
$d(p, a)$	the value range of a where p is true
notations related to a bucket	meanings
$d(b)$	the value range of b: for a bucket b of h over the attributes $a_{s+1}, \ldots, a_{s+k}$ in the relation r, $d(b) = [min(d(b, a_{s+1})), max(d(b, a_{s+1}))] * \ldots * [min(d(b, a_{s+k})), max(d(b, a_{s+k}))]$
$f(b)$	the frequency of b, *i.e.*, the number of tuples which fall in b
notations related to a histogram	meanings
$d(h)$	the value range of h: a k-dimensional histogram h over the attributes $a_{s+1}, \ldots, a_{s+k}$ in the relation r is obtained by partitioning the value space $[min(d(a_{s+1})), max(d(a_{s+1}))] * \ldots * [min(d(a_{s+k})), max(d(a_{s+k}))]$ into one or more buckets and records the number of tuples falling in each buckets. $d(h) = [min(d(a_{s+1})), max(d(a_{s+1}))] * \ldots * [min(d(a_{s+k})), max(d(a_{s+k}))]$
$B(h)$	the bucket set composed of all buckets of h
notations related to a predicate	meanings
$n(p)$	the real number of tuples which satisfy the predicate p
$esp(p)$	the estimated number of tuples which satisfy the predicate p
$esp_b(p)$	the estimated number of tuples which fall in the bucket b and satisfy the predicate p
$d(p)$	the value range of p: for a $p = (v_{11} \leq a_1 \leq v_{12}) \wedge (v_{21} \leq a_2 \leq v_{22}) \wedge \ldots \wedge (v_{m1} \leq a_m \leq v_{m2})$, $d(p) = [min(d(p, a_1)), max(d(p, a_1))] * \ldots * [min(d(p, a_m)), max(d(p, a_m))]$
$s(p)$	The dimension of p: for a $p = (v_{11} \leq a_1 \leq v_{12}) \wedge (v_{21} \leq a_2 \leq v_{22}) \wedge \ldots \wedge (v_{m1} \leq a_m \leq v_{m2})$, $s(p)$ denotes the number of attributes $a_1, a_2, \ldots, a_m$.
$qfr(p)$	the QFR of p: $qfr(p) = (p, n(p), m_p)$ where m_p is the executing moment of p

3. FIND *K* NEAREST QFRS USING WARD METHOD

3.1 Ward Method

Cluster analysis [20, 21] is the process of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar than those in other groups (clusters).

As a hierarchical cluster analysis method, the Ward method measures the distance between two classes based on the increment of the Sum of Squares of Deviations (SSD). Assuming n samples are categorized into k classes $G_1, G_2, \ldots, G_k$, X_{it} is the vector composed of the relevant variables of the i sample in G_t, and n_t denotes the number of samples in G_t. $\overline{X_t}$ is the center of gravity of G_t. The SSD of G_t can be expressed as:

$$S_t = \sum_{i=1}^{n_t} (x_{it} - \overline{X_t})^T (X_{it} - \overline{X_t}). \tag{1}$$

It reflects the degree of dispersion of the internal samples of G_t.

Assuming two classes G_p and G_q are merged into a new class G_r, SSD will increase. Based on the increment of SSD, the distance between G_p and G_q can be calculated as:

$$D_{pq} = \sqrt{S_r - S_p - S_q}. \tag{2}$$

This distance is called the *Ward distance* in the paper.

3.2 Ward Distance between Predicates

In CEWM, each predicate is considered as a sample in the Ward method. We define the *Ward distance between predicates* firstly, and then, the k nearest QFRs with a new predicate can be found using the Ward method.

For a predicate $p = (v_{11} \leq a_1 \leq v_{12}) \wedge (v_{21} \leq a_2 \leq v_{22}) \wedge \ldots \wedge (v_{m1} \leq a_m \leq v_{m2})$, the vector composed of its location variables can be expressed as:

$$X_p = (v_{11}, v_{12}, v_{21}, v_{22}, \ldots, v_{m1}, v_{m2})^T. \tag{3}$$

Supposing the class G_{p_1} and G_{p_2} contain the predicate p_1 and p_2 respectively. Simultaneously, the class $G_{p_1p_2}$ contains both p_1 and p_2. Based on Eqs. (1), (2) and (3), the Ward distance between p_1 and p_2 can be defined as:

$$D_{p_1p_2} = \sqrt{S_{p_1p_2} - S_{p_1} - S_{p_2}} = \sqrt{\sum_{i=1}^{2} (X_{p_i} - \overline{X_{p_1p_2}})^T (X_{p_i} - \overline{X_{p_1p_2}})}. \tag{4}$$

3.3 Algorithm of Finding *k* Nearest QFRs

Given a new predicate p_{n1} and the QFRs $qfr(p_1)$, …, $qfr(p_n)$ corresponding to the executed predicates $p_1, \ldots, p_n$, finding k $(1 \leq k \leq n)$ nearest QFRs with p_{n1} from $qfr(p_1)$, …, $qfr(p_n)$ is to find k executed predicates from $p_1, \ldots, p_n$ which have the 1st shortest Ward distance to the kth shortest Ward distance with p_{n1}. The algorithm of finding k nearest QFRs is shown as Algorithm 1:

Algorithm 1: finding k nearest QFRs with the new predicate p_{n1}

```
fkqfr($p_{n1}$, $p[1\ldots n]$)
1  $G_{pn1} \leftarrow p_{n1}$
2  for (each $i \in [1\ldots n]$) do
3      $G_{p[i]} \leftarrow p[i]$
4      $wd[i] \leftarrow calWardDis(G_{pn1}, G_{p[i]})$        //calculating Ward distances
5  $loc \leftarrow sort(wd[1], \ldots, wd[n])$              //sorting Ward distances
6  $k \leftarrow configK(p_{n1}, p[1\ldots n], loc[1\ldots n])$   //configuring $k$, elaborated in Algorithm 2
7  return $loc[1\ldots k]$
```

For Algorithm 1, the pseudo-codes in Line 2 to Line 4 calculate all Ward distances with the time complexity of $O(n*2s(p_{n1}))$ because one predicate corresponds to two coordinates in each dimension. The pseudo-code in Line 5 sorts the Ward distances with the time complexity of $O(n*\log(n))$. From Section 3.4, we can know the time complexity of the pseudo-code in Line 6 is $O(UL\text{^}2*4s(p_{n1}))$ where UL is the upper limit of k value and will be explained in Section 3.4. In general, $s(p_{n1})$ and UL are less than 10. Therefore, the time complexity of the whole Algorithm 1 is about $O(m*n)$.

For example, a 2-dimensional new predicate p_{n1} and ten executed predicates $p_1, \ldots, p_{10}$ are shown in Fig. 1. The new predicate is shown as a grey rectangle and the rectangles with dashed borders denote the executed predicates. Configure $k = 3$, the executed predicates corresponding to the k nearest QFRs with p_{n1} are shown as the rectangles with bold solid borders. They can be efficiently obtained by Algorithm 1.

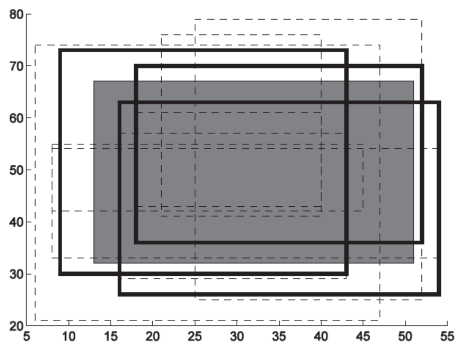


Fig. 1. A new predicate and ten executed predicates.

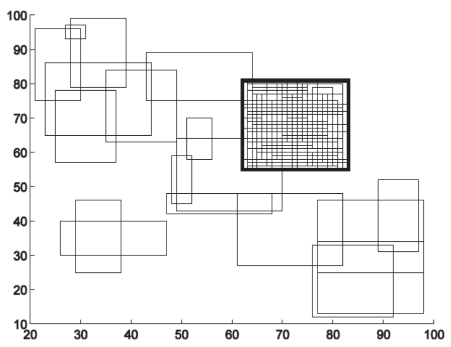


Fig. 2. A skewed query workload example.

3.4 The Choice of *k* Value

Using Algorithm 1, the *k* nearest QFRs with a new predicate can be found. But *k* value must be chosen carefully. To choose an appropriate *k* value, the distribution feature of query workload should be analyzed firstly. For an actual database, especially an OLTP system, the query workload often shows a certain skewed distribution feature where some tuples are frequently queried but many tuples are not [22, 23]. Fig. 2 shows a skewed query workload example in a 2-dimensional space. Each predicate is denoted by a rectangle. From the figure, we can observe that the area surrounded by a rectangle with bold solid borders is more frequently queried than the other areas.

For a new predicate p, the k value can be chosen as follow:

$$k = \min(C_p, UL). \tag{5}$$

When $k = \min(C_p, UL) = C_p$, the top C_p QFRs in the ascending QFR sequence obtained in Algorithm 1 fulfill: (1) the value range union of the corresponding C_p executed predicates can cover $d(p)$; (2) the value range union of the corresponding $C_p - 1$ executed predicates cannot cover $d(p)$.

When $k = \min(C_p, UL)=UL$, the value range union of the $UL - 1$ executed predicates corresponding to the top $UL - 1$ QFRs in the ascending QFR sequence obtained in Algorithm 1 cannot cover $d(p)$.

For a new predicate p_{n1}, the drilling hole operation [15] will be executed to calculate k value. The pseudo-codes of configuring k value are shown as Algorithm 2.

Algorithm 2: Configuring k value for the new predicate p_{n1}

```
configK($p_{n1}$, $p[1…n]$, $loc[1…n]$)
1 $k \leftarrow UL$        //$UL$ is the upper limit of $k$
2 $ucas \leftarrow ucas \cup uca_{n1}$  //the array $ucas$ stores all uncovered areas inside $d(p_{n1})$ according to the $k$
                     executed predicates. It is initialized with the area $uca_{n1}$ covering the
                     whole $d(p_{n1})$.
3 for (each $i \in [1…UL-1]$) do
4     if($|ucas|$ == 0)
```

```
5              $k \leftarrow i$
6              break
7          else
8              $tempUcas \leftarrow \varnothing$
9              for (each $uca \in ucas$) do
10                 if ($d_{uca} \subseteq d_{p[\mathrm{loc}[i]]}$)
11                     $ucas \leftarrow ucas - uca$
12                 else if ($d_{uca} \cup d_{p[\mathrm{loc}[i]]} \neq \varnothing$)
13                     $tempUcas \leftarrow tempUcas \cup holeDrilling(p[\mathrm{loc}[i]], uca)$
14                     $ucas \leftarrow ucas - uca$
15             $ucas \leftarrow ucas \cup tempUcas$
16     return $k$
```

For Algorithm 2, the time complexity is $O(UL*|ucas|_{max}*2s(p_{n1}))$. $|ucas|$ is different in each outer loop beginning from Line 3, and $|ucas|_{max}$ denotes the maximum of all $|ucas|$ in the UL-1 outer loops. It is difficult to deduce the precise $|ucas|_{sum}$ because the number of uncovered areas generated by each drilling hole operation can be different. From experiments, we know the drilling hole operations based on x predicates can generate 1.5x to 2.0x uncovered areas. Therefore, the time complexity of Algorithm 2 can be estimated as $O(UL\hat{}2*4s(p_{n1}))$.

For the new predicate shown in Fig. 1, the process of configuring k value is shown in Fig. 3. The uncovered areas inside the value range of the new predicate are filled with grey. After the drilling hole operations according to 3 executed predicates, no any uncovered areas can be found. Therefore, the k value can be configured as 3.

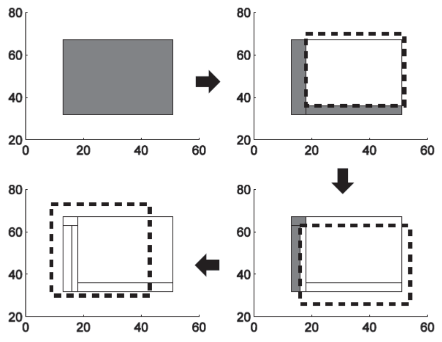


Fig. 3. Process of configuring k value.

4. CONSTRUCT MICRO HISTOGRAM

4.1 Micro Histogram and Global Histogram

A micro histogram is a histogram only covering a local of the value ranges of the queried attributes. In contrast, a histogram covering the entire value ranges of the queried attributes is called a global histogram. The reason that we adopt the micro histogram but not the global histogram in CEWM is: (1) the cardinality of a predicate is only decided by the data distribution of its neighborhood but not the entire value ranges of the queried attributes; (2) it is more efficient to construct and maintain a micro histogram because a micro histogram only contains very few buckets and the number of QFRs participating in the construction of a micro histogram is very limited; (3) due to the efficient performance of construction, an micro histogram can be constructed for each new predicate, and the changes of the underlying data can be embedded into a micro histogram in a more timely manner.

4.2 Construct Micro Histogram

For a multi-dimensional micro histogram h, each bucket $b_i \in B(h)$ covers a hyper rectangle with two constant boundaries in each dimension. Inside each hyper rectangle, there may be mutually disjoint sub hyper rectangles which are covered by the other buckets $b_{i1}, \ldots, b_{ij}$. We say $b_{i1}, \ldots, b_{ij}$ are the children of b_i and all buckets in the histogram h compose a tree structure. The value range of b_i can be calculated as follow:

$$d(b_i) = d(R_{b_i}) - \bigcup_{x=1}^{j} d(b_{ix}). \quad (6)$$

Here $d(R_{b_i})$ denotes the entire value range of the hyper rectangle covered by b_i. An example of the buckets in a multi-dimensional micro histogram and the corresponding bucket tree are shown in Figs. 4 (a) and (b) respectively.

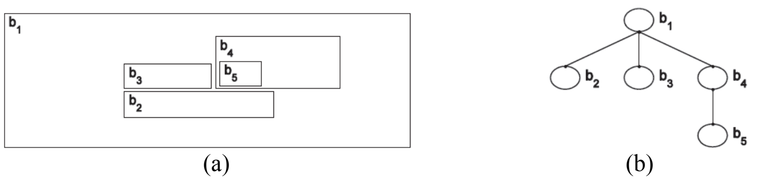


Fig. 4. A micro histogram and the corresponding bucket tree.

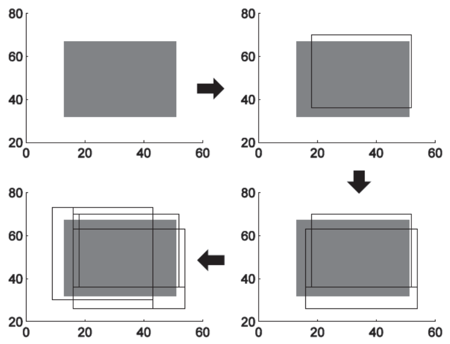


Fig. 5. Process of generating all buckets in a micro histogram.

The construction of a multi-dimensional micro histogram in CEWM includes two steps: firstly, the drilling hole operation is executed to get the value range of each bucket;

secondly, the frequency of each bucket is calculated based on the information-theoretic principle of maximum entropy [24, 25]. For the new predicate shown in Fig. 1, the process of generating all buckets in the micro histogram is shown in Fig. 5, where the new predicate is shown as a grey rectangle.

The pseudo-codes of constructing a micro histogram are shown as Algorithm 3:

Algorithm 3: constructing the micro histogram h for the new predicate p_{n1}

```
constLh($p_{n1}$, $p[1\ldots n]$, $loc[1\ldots k]$)
1 $T_{Bh} \leftarrow T_{Bh} \cup b_0$   //The bucket tree $T_{Bh}$ is initialized with the bucket $b_0$ covering the whole $d(p_{loc[1]})$.
2 for (each $i \in loc[2\ldots k]$) do
3     if (!$srhEqu(T_{Bh}, p[i])$)   //searching the bucket $b'$ satisfying $d(b') = d(p_i)$
4         $T_{temp} \leftarrow \varnothing$
5         for (each $(b \in T_{Bh})$ ^ ( $d_b \cap d_{p[i]} \neq \varnothing$)) do
6             $T_{temp} \leftarrow T_{temp} \cup holeDrilling(p[i], b)$
7             $T_{Bh} \leftarrow T_{Bh} - b$
8         $T_{Bh} \leftarrow T_{Bh} \cup T_{temp}$
9 $h \leftarrow is(T_{Bh}, n(p[loc[1]])\ldots n(p[loc[k]]))$   //executing the IS algorithm
```

For Algorithm 3, the pseudo-codes in Lines 1 to 8 generates all buckets in the micro histogram with the time complexity of $O(k*|T_{Bh}|_{max}*2s(p_{n1}))$. And Line 9 executes the IS algorithm with the time complexity $O(k\hat{}2*|T_{Bh}|_{max}*2s(p_{n1}))$ [26]. $|T_{Bh}|$ is different in each outer loop beginning from Line 2, and $|T_{Bh}|_{max}$ denotes the maximum of all $|T_{Bh}|$ in the $k-1$ outer loops. From experiments, we know the drilling hole operations based on x predicates can generate $1.5x$ to $2.0x$ buckets. Therefore, the time complexity of Line 1 to Line 8 can be estimated as $O(k\hat{}2*4s(p_{n1}))$. According to the actual efficiency experiments, it only spends several 10ms to construct a micro histogram in general. Therefore, a micro histogram can be constructed for each new predicate.

5. CARDINALITY ESTIMATION

Definition 1: Cardinality Estimation Before executing a predicate p, the estimation about the tuples satisfying p is called the cardinality estimation of p.

Assuming the micro histogram h is constructed to estimate the cardinality of the new predicate p. For a bucket $b_i \in B(h)$ satisfying $d(p) \cap d(b_i) \neq \varnothing$, the number of tuples which fall in b_i and satisfy p can be estimated according to the uniformity assumption:

$$est_{b_i}(p) = f(b_i)*|d(b_i) \cap d(p)|/|d(b_i)|. \tag{7}$$

Based on Eq. (7), the cardinality of p can be estimated in three different cases:

Case 1: If there exist the bucket set $B'(h) \subseteq B(h)$ satisfying $d(p) \subseteq \bigcup_{b_i \in R'(h)} d(b_i)$, and for each $b_i \in B'(h)$, $d(p) \cap d(b_i) \neq \varnothing$, the cardinality of p is:

$$est(p) = \sum_{b_i \in B'(h)} est_{b_i}(p). \tag{8}$$

Case 2: if no bucket set $B'(h) \subseteq B(h)$ satisfying $d(p) \subseteq \bigcup_{b_i \in R'(h)} d(b_i)$ can be found, but there exist a bucket set $B''(h) \subseteq B(h)$ satisfying $d(p) \cap d(b_i) \neq \varnothing$ for each $b_i \in B''(h)$, the cardinality of p is:

$$est(p) = \sum_{b_i \in B''(h)} est_{b_i}(p) * |d(p)| \Big/ \sum_{b_i \in B''(h)} |d(b_i) \cap d(p)|. \tag{9}$$

Case 3: if no any bucket $b_i \in B(h)$ satisfies $d(p) \cap d(b_i) \neq \varnothing$, the cardinality of p is:

$$est(p) = \sum_{b_i \in B(h)} f(b_i) * |d(p)| \Big/ \sum_{b_i \in B(h)} |d(b_i)|. \tag{10}$$

For example, the cardinalities of the new predicates p_{n1}, p_{n2} and p_{n3} in Fig. 6 can be calculated using Eqs. (8), (9) and (10) respectively.

For a new predicate p in Case 1, the error of cardinality estimation only originates from the uniformity assumption. For a new predicate p in Case 2, the extra error of cardinality estimation may be introduced by estimating the entire cardinality in the value range $d(p)$ with the local cardinality in the value range $\bigcup_{b_i \in R'(h)} d(b_i) \cap d(p)$.

For the new predicate p in Case 3, no information about the data distribution in its value range can be provided by h, and the remarkable error of cardinality estimation may be caused by the possible different data distributions between $d(p)$ and $d(h)$. Due to the high degree of overlap of the predicates in the frequently queried areas, the cardinalities of such predicates can be estimated using Eq. (8) or (9). Therefore, the holistic accuracy of cardinality estimation can be fully guaranteed in CEWM.

6. QFR UPDATE MECHANISM

In CEWM, we adopt the following QFR update mechanism: *for a new predicate p and its k nearest QFRs*, *qfr*(*p*) *will replace the oldest one of the k nearest QFRs after p is executed.*

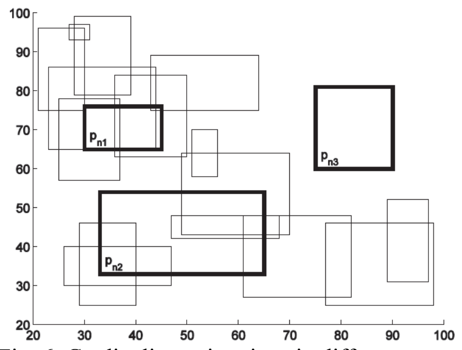


Fig. 6. Cardinality estimations in different cases.

Assuming the value ranges of ten successively executed predicates p_1 to p_{10} are shown in Fig. 7 (a). We can observe the skewed distribution feature of p_1 to p_{10} and most of them locate in A_1 area. The QFRs $qfr(p_1)$ to $qfr(p_{10})$ are stored into a QFR warehouse after the executions of p_1 to p_{10}.

Subsequently, ten new predicates p_{n1} to p_{n10} which have the same distribution feature with p_1 to p_{10} are executed successively. Based on the QFR update mechanism, the changes of the QFR warehouse are shown in Figs. 7 (b)-(d) respectively (Some medium statuses are omitted).

For p_{n1} shown as a grey rectangle in Fig. 7 (b), $k = 3$ and the k nearest QFRs are shown as rectangles with bold borders. We assume that the oldest one is shown as a rectangle with dashed bold borders, and it will be replaced by $qfr(p_{n1})$ after p_{n1} is executed.

For p_{n2} to p_{n10}, the corresponding QFRs can be updated in the same manner as p_{n1}. $qfr(p_1)$ to $qfr(p_{10})$ can be replaced by $qfr(p_{n1})$ to $qfr(p_{n10})$ gradually, and the final status of the QFR warehouse is shown in Fig. 7 (d). The possible underlying data changes can be timely reflected by the new QFRs.

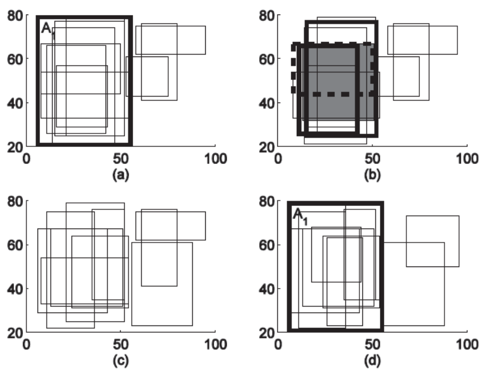


Fig. 7. QFR updates without query workload changes.

We also assume the other ten successively executed new predicates p_{n11} to p_{n20} have the different distribution feature with p_1 to p_{10}. The changes of the QFR warehouse are shown in Figs. 8 (b)-(d) respectively (Some medium statuses are omitted). The frequently query area moved from A_1 to A_2 which is shown in Fig. 8 (d). Therefore, we say the QFR update mechanism in CEWM can make a micro histogram matches not only the changes of underlying data but also the changes of query workload distribution.

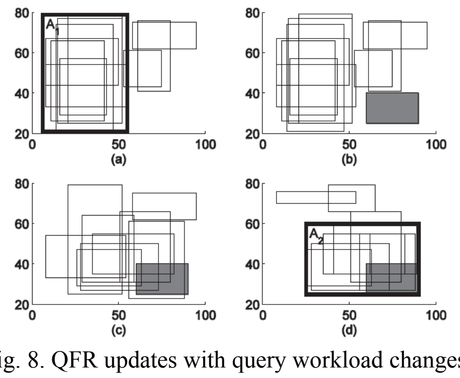


Fig. 8. QFR updates with query workload changes.

7. EXPERIMENTS

The experiments are performed on a 3.2GHz Intel CPU machine running Windows 7 sp1, with 4GB memory and 1TB hard disk.

7.1 Experimental Settings

7.7.1 Data sets

To test CEWM comprehensively, a real data set [27] and the TPC-H benchmark [28] with scale factor of 1 are used for the experiments. The distribution of the former shows skewed feature and the latter is a uniform data set.

Real data set (denoted by ds_1): It contains the census data of U.S. with 2,458,285 tuples. The experiments are carried out over the attributes *income*1 and *income*2.

TPC-H benchmark (denoted by ds_2): It contains 1,500,000 tuples which are generated by the DBGEN program. The experiments are carried out over the attributes *o_orderdate* and *o_orderId*.

7.1.2 Query workloads

In our experiments, two query workload models qw_1 and qw_2 are adopted.

qw_1: It follows the 2-dimensional Zipfian distribution [29] with the skew parameter $z = 1$. The domains of the queried attributes are divided into 5 mutually disjoint parts. The sizes of the 5 parts follow the 2-dimensional Zipfian distribution; the numbers of predicates within the 5 parts follow the reverse 2-dimensional Zipfian distribution.

qw_2: It follows the superposition of three 2-dimensional Gaussian distributions [30], and 3 median pairs are selected at random from the domain of the queried attributes. Around each median pair, the predicates are generated following a 2-dimensional Gaussian distribution with the same standard deviation *d*. The parameter *d* is configured as 5 percent of the width of the domains of the queried attributes.

7.1.3 Metrics

Firstly, we can define *re*(*p*), the relative error of a predicate *p* using Eq. (11):

$$re(p) = \frac{abs(n(p) - est(p))}{n(p)}. \tag{11}$$

Based on relative errors, we define the relative accuracy rate, *rar*(*ce*), of a cardinality estimation solution *ce* as the criterion to measure the accuracy of a cardinality estimation solution:

$$rar(ce) = \frac{cn_s(ce)}{tn(ce)}, \tag{12}$$

where *ce* denotes a cardinality estimation solution, $cn_s(ce)$ denotes the number of predicates whose relative errors are lower than s, and $tn(ce)$ denotes the total number of predicates. In our experiments, we configure $s = 0.2$.

7.1.4 Programs

In our experiments, the comparison solutions include CEWM proposed in the paper and the representative reactive solution ISOMER.

CEWM and ISOMER are realized under JDK 1.6.0_10. For CEWM, the initial capacity of QFR warehouse and the upper limit of k value are configured as 300 and 10 respectively. For ISOMER, the space budget of histogram affects the experimental results remarkably. Therefore, we compare two kinds of ISOMER solutions − the ISO2 solution and the ISO3 solution with 200 and 300 space budgets of histogram respectively.

Static and dynamic experiments are carried out based on each combination of data set and query workload. In static experiments, the underlying data and the distribution features of query workloads remain unchanged. In dynamic experiments, the underlying data and the distribution features of query workloads are always changing.

7.2 Static Experiments

At the preparation stage of each static experiment, 300 *training predicates* are executed and the corresponding QFRs are stored into the QFR warehouse for CEWM. And for ISO2 and ISO3, the initial histograms with about 200 and 300 buckets are constructed using 150 and 200 training predicates.

And then, in the formal experimental stage, 1,000 *validation predicates* with the same distribution feature as the training predicates are executed using different solutions. During the execution of the 1,000 validation predicates, for each 100 ones, and the relative accuracy rate and the overall execution time are recorded for each solution.

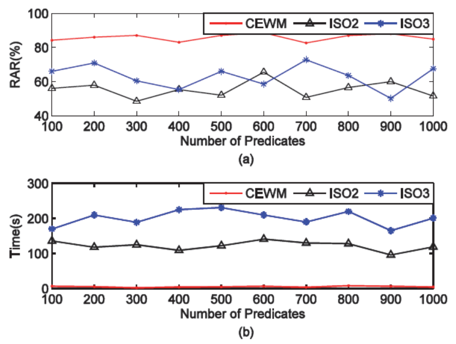
The results of the static experiments based on ds_1 and qw_1 are shown in Fig. 9 (I). From Fig. 9 (I.a), we can see that CEWM shows excellent accuracy of cardinality estimation and the relative accuracy rates of CEWM are always higher than 80 percent.

For ISOMER, the accuracy of ISO3 is better than the one of ISO2 due to the improved space budget of histogram, but the overall accuracy level of ISOMER is about 20 to 30 percent lower than CEWM. Furthermore, in Fig. 9 (I.a), we can observe obvious fluctuations from the relative accuracy rate curves of ISO2 and ISO3. In ISOMER, several hundreds of QFRs are needed to calculate a global histogram together. In this case, the IS algorithm adopted in ISOMER shows some instability. For CEWM, the adopted micro histograms only contain few buckets and the IS algorithm can always provide the relatively accurate results.

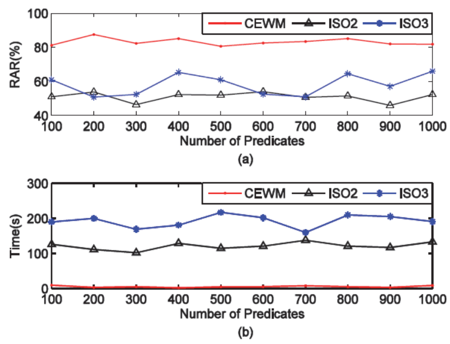
From Fig. 9 (I.b), we can also see the superiority of CEWM in efficiency. CEWM can finish the cardinality estimations of each 100 predicates within 10 seconds in general.

But for ISOMER, the time costs are 10 to 50 multiples of CEWM due to the periodical reconstructions of global histograms. Furthermore, as the space budgets of histograms increase, the efficiency of ISOMER deteriorates rapidly.

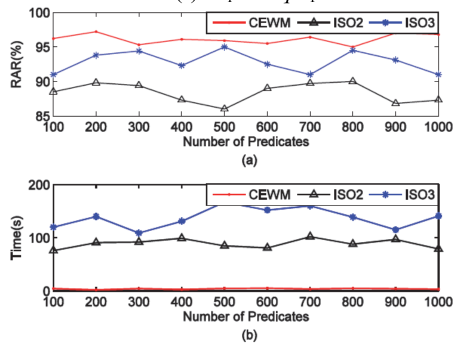
Fig. 9 (II) shows the results of the static experiments based on ds_1 and qw_2. CEWM is still accurate, stable and efficient. And the fluctuation of ISOMER is still obvious.



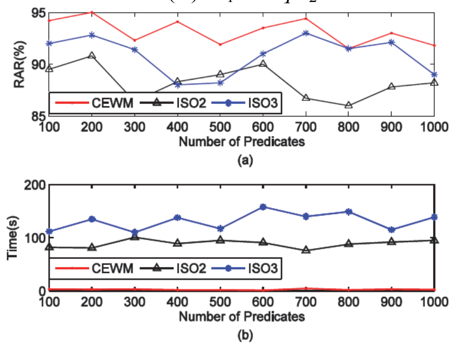
(I) ds_1 and qw_1



(II) ds_1 and qw_2



(III) ds_2 and qw_1



(IV) ds_2 and qw_2

Fig. 9. Static experiments.

Fig. 9 (III) shows the results of the static experiments based on ds_2 and qw_1. The relative accuracy rates of all solutions improve to different degrees. But the overall accuracy level of CEWM is still higher than ISOMER.

The results of the static experiments based on ds_2 and qw_2 are shown in Fig. 9 (IV). The tiny difference of skewness between ds_1 and ds_2 is the main reason leading to the slight deterioration of CEWM in accuracy and stability over ds_2.

The averages in different static experiments are summarized in Table 2. CEWM shows excellent performance in static experiments. Compared with ISOMER, CEWM can finish cardinality estimations more accurately in much shorter time.

Table 2. Averages in different static experiments.

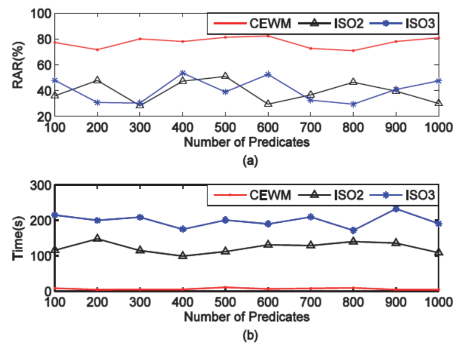
Solutions	averages of RAR(%)/Time(s)			
	ds_1 and qw_1	ds_1 and qw_2	ds_2 and qw_1	ds_2 and qw_2
CEWM	85.8/5.2	83.2/5.5	96.1/4.1	93.2/2.9
ISO2	55.4/122.4	51.0/121.3	88.4/89.1	88.3/89.0
ISO3	63.1/201.1	58.2/192.5	92.9/137.4	90.9/131.3

7.3 Dynamic Experiments

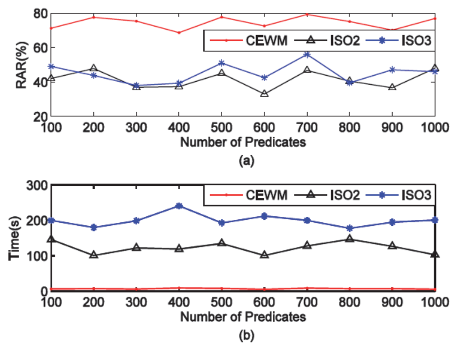
The preparation stage and the formal stage of each dynamic experiment are similar with the static experiment.

In each dynamic experiment, as the main process is being executed, another data updating process is running simultaneously. For the data set ds_1, the data updating process contains two refresh functions *INS* and *DEL*, which can insert 10 percent new tuples and delete 10 percent old tuples respectively. Both *INS* and *DEL* will be executed once before each 100 of the 1,000 validation predicates are executed, which can ensure at least 20 percent of the data in ds_1 can be updated. For the data set ds_2, the refresh functions $RF1$ and $RF2$ which are defined in the TPC-H benchmark will be used to finish the update of the underlying data. Before each 100 of the 1,000 validation predicates are executed, $RF1$ and $RF2$ will be executed 100 times continuously to ensure at least 20 percent of the data in ds_2 can be updated.

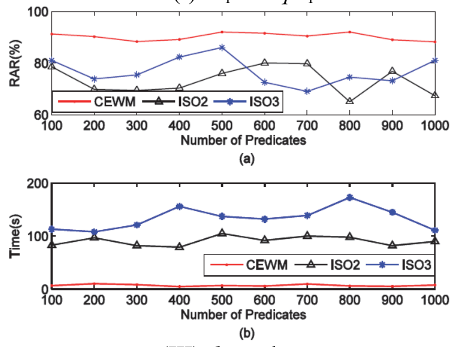
As the underlying data are updated by the refresh functions, the query workloads are also changed. For the data set ds_1, each 100 of the 1,000 validation predicates will satisfy the 2-dimensional Zipfian distribution with the new centers for the 5 mutually disjoint parts. And for the data set ds_2, each 100 of the 1,000 validation predicates will satisfy the superposition of 2-dimensional Gaussian distributions with 3 new median pairs.



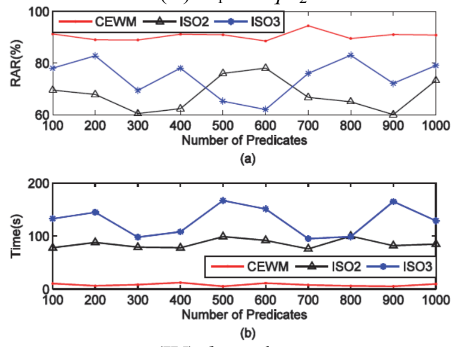
(I) ds_1 and qw_1



(II) ds_1 and qw_2



(III) ds_2 and qw_1



(IV) ds_2 and qw_2

Fig. 10. Dynamic experiments.

The results of the dynamic experiments based on ds_1 and qw_1 are shown in Fig. 10 (I). Compared with the results of the corresponding static experiments in Fig. 9, we can observe the relative accuracy rates of all solutions decline. The difference between the maximum and the minimum of the relative accuracy rates of CEWM increase, which shows larger fluctuations. But compared with ISOMER, CEWM is still a much more stable solution. It can always provide accurate cardinality estimations for more than 70 percent validation predicates with only 5 percent to 10 percent time costs of the ISO2 solution. Furthermore, CEWM is more adaptive to the changes than ISOMER, which owe to the micro histogram and the QFR update mechanism adopted in CEWM.

Fig. 10 (II) shows the results of the dynamic experiments based on ds_1 and qw_2. The changing tendencies of the relative accuracy rate and the execution time are similar with the ones in Fig. 10 (I).

Fig. 10 (III) shows the results of the dynamic experiments based on ds_2 and qw_1. Apparently, the instability is always a serious deficiency of ISOMER. For the 600th to the 700th predicates, the relative accuracy rate of the ISO3 solution is only 69 percent, but for the 400th to the 500th predicates, the value is 86 percent. ISOMER cannot stably provide accurate cardinality estimation even for the uniform underlying data.

The results of the dynamic experiments based on ds_2 and qw_2 shown in Fig. 10 (IV) show more fluctuations than the ones shown in Fig. 10 (III).

The averages in different dynamic experiments are summarized in Table 3. CEWM can adapt to the changes of the underlying data and the query workloads much better than ISOMER.

Table 3. Averages in different dynamic experiments.

Solutions	averages of RAR(%)/Time(s)			
	ds_1 and qw_1	ds_1 and qw_2	ds_2 and qw_1	ds_2 and qw_2
CEWM	77.3/6.6	74.4/7.1	90.2/7.1	90.5/8.3
ISO2	39.3/123.5	41.4/122.9	73.3/90.8	67.9/85.7
ISO3	40.5/199.6	45.2/199.9	76.9/133.5	74.6/129.0

7.4 Parameter Influence Experiments

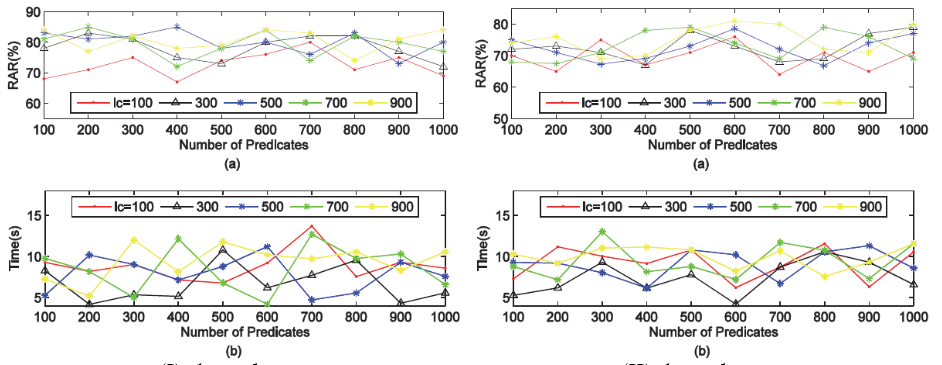
During the execution of CEWM, two parameters can be configured freely: the initial capacity of the QFR warehouse and the upper limit of k value. The influences of the two parameters influence are tested by the corresponding experiments.

To test the influence of the initial capacity of the QFR warehouse, we configure it as 100, 300, 500, 700 and 900 (QFRs) respectively. And then the results of the dynamic experiments based on ds_1 and qw_1 are shown in Fig. 11 (I), where *ic* denotes the initial capacity of the QFR warehouse. When the initial capacity of the QFR warehouse equals 100, the relative accuracy rate of CEWM is a little lower. When the initial capacity of the QFR warehouse increases to 300 or more, we cannot observe the apparent differences for the relative accuracy rates. Based on the experimental results, we conclude that 100 is not an enough initial capacity of the QFR warehouse for CEWM, and the cardinalities of some predicates cannot be estimated accurately because of inadequate similar QFRs. As the initial capacity of the QFR warehouse increases to 300, the frequently queried areas can be fully covered by the QFRs of the executed predicates.

From Fig. 11 (I.b), we know the efficiency of CEWM cannot be influenced by the initial capacity of the QFR warehouse remarkably.

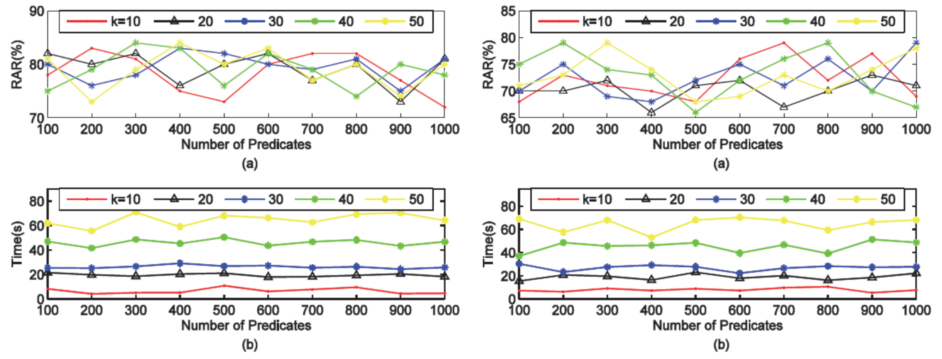
Fig. 11 (II) shows the experimental results of the initial capacity of the QFR warehouse based on ds_1 and qw_2. No remarkable difference can be observed.

We also carry out the experiments about the influence of the upper limit of k value. The results based on ds_1 and qw_1, ds_1 and qw_2 are shown in Figs. 12 (I) and (II) respectively. The upper limit of k value (denoted by k) is configured as 10, 20, 30, 40 and 50 respectively. No obvious difference can be found from the relative accuracy rates under different upper limits of k value.



(I) ds_2 and qw_1 (II) ds_2 and qw_2

Fig. 11. Experiments for initial capacity.



(I) ds_2 and qw_1 (II) ds_2 and qw_2

Fig. 12. Experiments for different k based on ds_2 and qw_1.

But from Figs. 12 (I.b) and (II.b), we can see different upper limits of k value mainly influence CEWM on efficiency. As the increase of the upper limit of k value, the average time of cardinality estimation of each predicate increases from 90ms to 650ms. Especially when the upper limit of k value is 40 or more, the deterioration of efficiency becomes faster. Therefore, it is unnecessary to adopt an upper limit of k value more than 10.

In summary, the performance of CEWM is not seriously influenced by the initial capacity of the QFR warehouse and the upper limit of k value. In general, CEWM can work well under the initial capacity of the QFR warehouse 300, and the upper limit of k value 10.

8. RELATED WORKS

The first solution which applies the 1-dimensional histogram in cardinality estimation is proposed in [31]. And then, the improved solutions [1, 22, 32-34] are continuously proposed. In the mainstream relational databases, 1-dimensional histograms have been widely used to help estimate cardinality. 1-dimensional equi-depth histograms, compressed histograms and maxdiff histograms are adopted in Oracle [35], DB2 [36] and SQL Server [37] respectively. To keep a 1-dimensional histogram consistent with changing data and query workload distribution, the underlying data is scanned periodically in an actual database.

But for a predicate referring to multiple attributes, the cardinality estimation based on a 1-dimensional histogram ignores the correlations among multiple attributes and leads to serious cardinality estimation errors inevitably. Multi-dimensional histograms have more practical values for the cardinality estimation because few actual data satisfy the attribute value independence assumption. The experimental multi-dimensional histograms have been researched for many years [1-5]. But in the existing solutions, the bucket number in a multi-dimensional histogram will increase exponentially with dimensionality and it is very time-consuming to maintain a multi-dimensional histogram according to the data changes. This is the main reason that no multi-dimensional histogram is adopted in the mainstream databases until now.

[38] begins to use QFRs to help estimate cardinality, but no histogram is constructed in it. The first self-histogram based on QFRs appears in [12]. The solutions in [12] and [13] adopt heuristics mechanism to update self-tuning histograms, which leads to inconsistency between a histogram and the corresponding QFRs. The solution in [14] improves the accuracy of a self-tuning histogram by subtilizing the granularity of QFRs, but the extremely detailed feedback information requirements in the solution leads to expensive time cost. The self-tuning histogram structure in [14] is also used in [15], but the histogram in [15] can be consistent with all currently valid QFRs by introducing the information-theoretic principle of maximum entropy. [39] proposes an alternative formulation for consistency to improve the performance of the maximum entropy based solution. [40] leverages all available query feedback information based on the information-theoretic principle of maximum entropy and is scalable to multiple dimensions and large number of QFRs. [16] proposes a simple learning-theoretic formalization of self-tuning histograms. A self-tuning histogram is learned based on QFRs to minimize the expected cardinality estimation error on future queries. [17, 41] improve the accuracy of a self-tuning histogram by starting with a carefully chosen initial configuration. [18] tries to improve the efficiency of maintaining self-tuning histograms by constructing two level self-tuning histograms. But the first-level histogram must be constructed by scanning data.

9. CONCLUSION

In the paper, a new cardinality estimation solution using micro self-tuning histograms is proposed. The Ward method is introduced to find *k* nearest QFRs for a new predicate, and the micro self-tuning histogram is constructed based on the *k* nearest QFRs to alleviate the issue of "curse of dimension" and improve the efficiency of the solution.

Simultaneously, after the execution of each new predicate, QFR warehouse is updated by the corresponding new QFR, and the data and workload changes can be timely reflected by the micro self-tuning histogram. The complex and cumbersome operations in the process of meeting a space budget are eliminated completely, which make the whole solution reliable and dexterous. Extensive comparison experiments have shown that our solution is satisfactory in the accuracy and the efficiency of cardinality estimation.

In the future, we will try to use the application framework in the paper to improve the cardinality estimation of join predicates which are related to multiple attributes in different relations.

ACKNOWLEDGMENT

This work is supported in part by the research funds from the science and technology plan of Hebei Province under Grant No. 16210348, and the humanities and social sciences major research project of Hebei Provincial Department of Education under Grant No. ZD201710. We would like to thank the anonymous reviewers for their constructive suggestions.

REFERENCES

1. M. Muralikrishna and D. J. DeWitt, "Equi-depth histograms for estimating selectivity factors for multi-dimensional queries," in *Proceedings of ACM International Conference on Management of Data*, 1988, pp. 8-36.
2. V. Poosala and Y. Ioannidis, "Selectivity estimation without the attribute value independence assumption," in *Proceedings of ACM International Conference on Very Large Data Bases*, 1997, pp. 486-495.
3. D. Gunopulos, G. Kollios, V. Tsotras, and C. Domeniconi. "Approximating multidimensional aggregate range queries over real attributes," in *Proceedings of ACM International Conference on Management of Data*, 2000, pp. 463-474.
4. A. Deshpande, M. Garofalakis, and R. Rastogi, "Independence is good: Dependency-based histogram synopses for high-dimensional data," in *Proceedings of ACM International Conference on Management of Data*, 2001, pp. 199-210.
5. N. Thaper, S. Guha, P. Indyk, and N. Koudas, "Dynamic multi-dimensional histograms," in *Proceedings of ACM International Conference on Management of Data*, 2002, pp. 428-439.
6. H. Wang and K. C. Sevcik, "A multi-dimensional histogram for selectivity estimation and fast approximate query answering," in *Proceedings of Conference of the Centre for Advanced Studies on Collaborative Research*, 2003, pp. 328-342.
7. L. Baltrunas, A. Mazeika, and M. H. Böhlen, "Multi-dimensional histograms with tight bounds for the error," in *Proceedings of International Database Engineering and Applications Symposium*, 2006, pp. 105-112.
8. J. S. Vitter and M. Wang, "Approximate computation of multidimensional aggregates of sparse data using wavelets," in *Proceedings of ACM International Conference on Management of Data*, 1999, pp. 193-204.
9. K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim, "Approximate query pro-

cessing using wavelets," in *Proceedings of International Conference on Very Large Data Bases*, 2000, pp. 111-122.

10. E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, *Wavelets for Computer Graphics – Theory and Applications*, Morgan Kaufmann Publishers, CA, 1996.
11. F. Dubeau, S. Elmejdani, and R. Ksantini "Non-uniform Haar wavelets," *Applied Mathematics and Computation*, Vol. 159, 2004, pp. 675-693.
12. A. Aboulnaga and S. Chaudhuri, "Self-tuning histograms: Building histograms without looking at data," in *Proceedings of ACM International Conference on Management of Data*, 1999, pp. 181-192.
13. L. Lim, M. Wang, and J. Vitter, "SASH: A self-adaptive histogram set for dynamically changing workloads," in *Proceedings of International Conference on Very Large Data Bases*, 2003, pp. 369-380.
14. N. Bruno, S. Chaudhuri, and L. Gravano, "STHoles: A multi-dimensional workload-aware histogram," in *Proceedings of ACM International Conference on Management of Data*, 2001, pp. 294-305.
15. U. Srivastava, P. J. Haas, V. Markl, M. Kutsch, and T. M. Tran, "ISOMER: Consistent histogram construction using query feedback," in *Proceedings of International Conference on Data Engineering*, 2006, pp. 39-51.
16. R. Viswanathan, *et al.*, "A learning framework for self-tuning histograms," *CoRR*, abs/1111.7295, 2011.
17. A. Khachatryan, *et al.*, "Improving accuracy and robustness of self-tuning histograms by subspace clustering," *IEEE Transactions on Knowledge & Data Engineering*, Vol. 27, 2015, p. 1.
18. X. Lin, *et al.*, "A cardinality estimation approach based on two level histograms," *Journal of Information Science and Engineering*, Vol. 31, 2015, pp. 1733-1756.
19. F. Murtagh and P. Legendre, "Ward's hierarchical clustering method: Clustering criterion and agglomerative algorithm," *CoRR*, abs/1111.6285, 2011.
20. G. A. Mauser, R. A. Peterson, and R. A. Kerin, "Clustering algorithms," *Journal of Marketing Research*, Vol. 14, 1977, p. 124.
21. A. K. Jain, "Data clustering: 50 years beyond *k*-means," *Pattern Recognition Letter*, Vol. 31, 2010, pp. 651-666.
22. Y. E. Ioannidis and V. Poosala, "Balancing histogram optimality and practicality for query result size estimate," in *Proceedings of ACM International Conference on Management of Data*, 1995, pp. 233-244.
23. J. J. Levandoski, P. A. Larson, and R. Stoica, "Identifying hot and cold data in main-memory databases," in *Proceedings of International Conference on Data Engineering*, 2013, pp. 26-37.
24. E. T. Jaynes, *Probability Theory: the Logic of Science*, Cambridge University Press, Cambridge, UK, 2003.
25. V. Markl, P. Haas, M. Kutsch, N. Megiddo, U. Srivastava, and T. Tran, "Consistent selectivity estimation via maximum entropy," *The VLDB Journal*, Vol. 16, 2007, pp. 55-76.
26. J. N. Darroch and D. Ratcliff, "Generalized iterative scaling for log-linear models," *The Annals of Mathematical Statistics*, Vol. 43, 1972, pp. 1470-1480.
27. D. Aha and P. Murphy, "UCI repository of machine learning databases," http://archive.ics.uci.edu/ml/index.html, 2013.

28. Transaction Processing Performance Council, "TPC benchmark H standard specification revision 2.17.0," http://www.tpc.org/tpch/spec/tpch2.17.0.pdf, 2014.
29. G. K. Zipf, *Human Behavior and the Principle of Least Effort*, Addison-Wesley, Reading, MA, 1949.
30. S. A. William, H. Press, B. P. Flannery, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, UK, 1993.
31. R. P. Kooi, "The optimization of queries in relational databases," PhD Thesis, Department of Electrical Engineering and Computer Science, Case Western Reserve University, 1980.
32. G. Piatetsky-Shapiro and C. Connell, "Accurate estimate of the number of tuples satisfying a condition," in *Proceedings of ACM International Conference on Management of Data*, 1984, pp. 256-276.
33. V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita, "Improved histograms for selectivity estimate of range predicates," in *Proceedings of ACM International Conference on Management of Data*, 1996, pp. 294-305.
34. H. To, K. Chiang, and C. Shahabi, "Entropy-based histograms for selectivity estimate," in *Proceedings of ACM International Conference on Information and Knowledge Management*, 2013, pp. 1939-1948.
35. D. R. Augustyn, "Applying advanced methods of query selectivity estimation in oracle DBMS," *Advances in Intelligent and Soft Computing*, Vol. 59, 2009, pp. 585-593.
36. M. Stillger, G. Lohman, V. Markl, and M. Kandil, "LEO-DB2's learning optimizer," in *Proceedings of International Conference on Very Large Data Bases*, 2001, pp. 19-28.
37. S. Agrawal, S. Chaudhuri, L. Kollár, A. P. Marathe, V. R. Narasayya, and M. Syamala, "Database tuning advisor for Microsoft SQL server 2005," in *Proceedings of International Conference on Very Large Data Bases*, 2004, pp. 1110-1121.
38. C. M. Chen and N. Roussopoulos, "Adaptive selectivity estimation using query feedback," in *Proceedings of ACM International Conference on Management of Data*, 1994, pp. 161-172.
39. R. Kaushik, C. Ré, and D. Suciu, "General database statistics using entropy maximization," in *Proceedings of the 12th International Symposium on Database Programming Languages*, 2009, pp. 84-99.
40. C. Ré and D. Suciu, "Understanding cardinality estimation using entropy maximization," in *Proceedings of ACM SIGMOD/PODS Conference*, Vol. 37, 2012, p. 6.
41. A. Khachatryan, *et al.*, "Efficient selectivity estimation by histogram construction based on subspace clustering," *Scientific and Statistical Database Management*, Springer Berlin Heidelberg, 2011, pp. 351-368.

Xiaoning Zeng (曾晓宁) received her M.S. degree in School of Information Science and Engineering from Yanshan University of China in 2006. She is currently an Associate Professor at Hebei Normal University of Science and Technology. Her main research interests include relational database, query optimization, *etc.*

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 Hebei University of Environmental Engineering. His main research interests include query optimization, keyword search, semi-structural data, *etc.*

Caiyan Pei (裴彩燕) received her M.S. degree in School of Information Science and Engineering from Yanshan University of China in 2009. She is currently a Lecturer at Hebei Normal University of Science and Technology. Her main research interests include query optimization, information retrieval, *etc.*

Jing Cao (曹靖) received her M.S. degree in School of Information Science and Engineering from Yanshan University of China in 2009. She is currently an Associate Professor at Hebei Normal University of Science and Technology. Her main research interests include information retrieval, algorithm analysis, *etc.*