Approximate Query Processing Based on Approximate Materialized View

Yuhan Wu[1], Haifeng Guo[1], Donghua Yang[1(✉)], Mengmeng Li[1], Bo Zheng[2], and Hongzhi Wang[1]

[1] Harbin Institute of Technology, Harbin, China
yang.dh@hit.edu.cn
[2] ConDB, Beijing, China

Abstract. In the context of big data, the interactive analysis database system needs to answer aggregate queries within a reasonable response time. The proposed AQP++ framework can integrate data preprocessing and AQP. It connects existing AQP engine with data preprocessing method to complete the connection between them in the process of interaction analysis.

After the research on the application of materialized views in AQP++ framework, it is found that the materialized views used in the two parts of the framework both come from the accurate results of precomputation, so there's still a time bottleneck under large scale data. Based on such limitations, we proposed to use approximate materialized views for subsequent results reuse. We take the method of identifying approximate interval as an example, compared the improvement of AQP++ by using approximate materialized view, and trying different sampling methods to find better time and accurate performance results.

By constructed larger samples, we compared the differences of time, space and accuracy between approximate and general materialized views in AQP++, and analyzed the reasons for the poor performance in some cases of our methods.

Based on the experimental results, it proved that the use of approximate materialized view can improve the AQP++ framework, it effectively save time and storage space in the preprocessing stage, and obtain the accuracy similar to or better than the general AQP results as well.

Keywords: Approximate materialized view · Materialized views reuse · AQP++ optimization · Approximate query processing

1 Introduction

With the increasing amount of data, practical applications have higher requirement in query. The query results within the precise or error threshold should

This paper was supported by The National Key Research and Development Program of China (2020YFB1006104) and NSFC grant (62232005).

Z. Tari et al. (Eds.): ICA3PP 2023, LNCS 14488, pp. 168–185, 2024.
https://doi.org/10.1007/978-981-97-0801-7_10

be returned in the acceptable time. Sampling based approximate query processing and aggregate precomputation are the two methods proposed in the past to try to solve this problem. During interactive queries, the database will produce a large number of materialized views, and the queries on certain data sets are usually concentrated in practice, which makes the query results reusable.

AQP++ framework integrates AQP with aggregate precomputation. AQP usually transforms query conditions in a reasonable way to get approximate results, data preprocessing focuses on the reasonable organization of data set for efficient query. However, the reuse results of the two parts of AQP++ all come from the precise answer of precomputation, so there is still a time bottleneck in the case of huge data scale.

Considering the reusability and reduction of the materialized views, we proposed a new perspective of approximate materialized view. That is to say, data preprocessing is based on sampling set, using general methods to generate approximate materialized views for reuse in the subsequent query process.

Using an approximate materialized view will approximate the results twice, so we are faced with the challenge of how to choose a way of view generation and AQP method. We use the precomputing method based on data aggregation on the results of simple random sampling to generate the interval results as the approximate materialized view, and identify the approximate intervals on them, so as to realize the approximate query.

We make the following contributions in this paper:

- Sampling and aggregation precomputing of data sets, and generate approximate materialized view on the intervals.
- Identifying the approximate interval and get the approximate result.
- Comparison and improvement of sampling methods: select different sampling methods, or combine multiple sampling methods to find the relatively optimal method according to their respective time and accuracy performance.

The remaining sections of this paper are organized as follows. In Sect. 2, we make overview for this paper. In Sect. 3, we introduce the framework and steps of using approximate materialized view. In Sect. 4, the experimental results indicate the performance of our method and we carry out comparative experiments and error analysis. In Sect. 5, we study the influence of parameters. In Sect. 6, we survey related work for this paper. In Sect. 7, we provide the conclusions and give a brief overview of our future work.

2 Overview

For the range query of attributes with continuous values on the dataset, the data cube based method for data preaggregation is a way of AQP. We will apply this method to sample sets to generate approximate materialized views. In this section, we will give the definition of the problem and review the interval partition method.

2.1 Problem Definition

We consider aggregate queries on continuous attributes. Let the value range of the query attribute be $[x, y]$, aggregate query is defined as the number of samples whose values belong to the query interval $[a, b]$. Suppose through data preaggregation, we have obtained the approximate aggregate value of the attribute on k different intervals: $[a, x_1], [x_1, x_2, ..., [x_{k-1}, b]$, these results will be reused as our materialized views. Take the age query on census data as an example, the new range query is:

$$\text{SELECT age, COUNT(*) FROM census WHERE } a \leq age \leq b$$

We aim to use one or more views as approximate solutions of actual query intervals. For example, use the sum of aggregate values of interval $[x_{t-1}, x_t]$ and $[x_t, x_{t+1}]$ to replace $[a, b]$ if $\|a - x_{t-1}\| \leq \epsilon$ and $\|b - x_{t+1}\| \leq \epsilon$.

2.2 Data Aggregation

This paper uses the preaggregation and approximate interval recognition methods based on interval partition provided by previous strudies. It finds some partition points and approximate point of query according to interval evaluation.

Partition Evaluation. The current partition points on the ordered dataset is $x_1, x_2, ..., x_n$. For any query point x, I_x and H_x are represented the first partition point less than x and the first partition point greater than x respectively. The interval $[I_x, H_x]$ is also divided into two parts $\overline{L}_x$ and L_x as shown in Fig. 1.

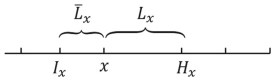


Fig. 1. Interval Partition

Approximate method will select I_x or H_x to replace the query point x, so the error comes from the smaller value between $\overline{L}_x$ and L_x. Define error as:

$$error_x = \min\{\frac{\lambda N}{\sqrt{n}}\sqrt{var(A_{L_x})}, \frac{\lambda N}{\sqrt{n}}\sqrt{var(A_{\overline{L}_x})}\} \quad (1)$$

If i and j are the two points with the largest error in the whole dataset, then the upper bound of the interval error will be $error_i + error_j$.

Data Aggregation and Precomputation. Interval Partition use the adaptive climbing method, starting from the initial state, trying to move the partition point to a better position according to the evaluation (1). The specific steps of the algorithm will be given in Sect. 3.

Aggregation Recognition. After interval pre partition, for query in interval $[a, b]$, there are four most relevant intervals: $[I_a, I_b], [I_a, H_b], [H_a, I_b], [H_a, H_b]$. Calculate confidence on these candidate intervals:

$$\lambda N \sqrt{\frac{var(cond(A = 0))}{n}} \tag{2}$$

where $cond(A = 0)$ means $\{x | x \in data, x \notin A\}$. Calculate the confidence of candidate interval on subsampling, and choose one of them with the minimum confidence as the approximate result.

3 Reuse of Approximate Materialized View

In this section, we divide the reuse of approximate materialized view into two parts: view generation and query approximate interval recognition. In the part of view generation, we find the partition points of the sampling data according to the method described in Sect. 2 and store the aggregation results of the partition intervals. Note that these aggregation results are based on the sampling set, so they are approximate. In the part of interval recognition, based on the method described in Sect. 2, we find four candidate values corresponding to the query interval, and choose the best one according to their confidence. That is to achieve the approximate treatment of approximate materialized view.

3.1 Aggregation and Precomputing

In this part, we need to sample the dataset, and find a resonable partition of the sampled data, use the aggregation results of each partition as a materialized view for subsequent reuse.

Firstly we use simple random sampling, assume that the sampled data has been sorted, and a parameter k is given to represent the number of partition points. Starting with an initial partition, each iteration calculates the upper bound of the current partition error, only when the new iteration reduces the upper bound can the partition points move. When it need to move the partition point, we move the point with the minimum moving error to the position with the maximum error in the dataset. The details will be introduced with the description of the pseudo code in Algorithm 1.

The pseudo code of partition points generation is shown in Algorithm 1. We first sample and sort the dataset (Line 1). Then we initialize the partition points, here we find k points evenly on the sample set (Line 2). We define three conditions to stop the algorithm: (a)iterations exceeds the threshold, (b)new iteration cannot reduce the upper bound of error, (c)no better partition points can be found. We initialize the above three conditions (Line 3–5).

Algorithm 1: Interval Partition

Input: dataset D, number of partition points k, maximum iteration I_{max}
Result: partition points set $P = \{p_{1,2}, ..., p_k\}$

```
1  S ← Sample_Sort(D);
2  P ← EqualPartition(S, k);
3  iterator ← 0;
4  upper ← +∞;
5  stop ← False;
6  while not stop and ErrorBound(P) ≤ upper and iterator ≤ I_max do
7  |   i_1 ← arg max_i error(S);
8  |   i_2 ← arg max_{i≠i_1} error(S);
9  |   if i_1, i_2 ∈ P then
10 |   |   stop ← True;
11 |   else
12 |   |   t ← arg min_{t∈[1,k]} {max_{r∈S∧r∈[p_{t-1},p_{t+1}]} error(S_r)};
13 |   |   if i_1 ∉ P then
14 |   |   |   p_t ← i_1;
15 |   |   else
16 |   |   |   p_t ← i_2;
17 |   |   end
18 |   end
19 end
```

In an iteration, we find two points i_1, i_2 with the largest error in the sampling set (Line 7–8). We should move the partition points to i_1 or i_2 to reduce the upper bound of error. If i_1, i_2 are already partition points, means algorithm has found the partition result (Line 9–10). Otherwise, we need to select a partition point to move to the location of i_1 or i_2. Suppose three consecutive partition points are p_{t-1}, p_t, p_{t+1}, after moving p_t, only the query on $[p_{t-1}, p_{t+1}]$ will be affected by p_t and may choose other approximate points. So we define the movement error of p_t as the maximum error of interval $[p_{t-1}, p_{t+1}]$, find the partition point with the minimum moving error to move (Line 12). Note that error of i_1 is greater than that of i_2, so we move to the location of i_1 preferentially (Line 13–17).

At the end of the algorithm, we get the set of partition points on the sampling set, the aggregation values on the intervals according to these points will be used as approximate materialized view.

3.2 Approximate Interval Recognition

In Sect. 3.1, we store approximate aggregation values on several intervals. We aim to identify the available partition intervals and find the approximate results when a new query comes. We have reviewed the principle of approximate identification in Sect. 2.2, details and the pseudo code will be explained in Algorithm 2.

The pseudo code of finding approxiamte result is shown in Algorithm 2. We first find the four closest partition points to the upper and lower bounds a and b

Algorithm 2: Interval Recognition

Input: sample set S, partition points set P, approximate aggregate values $V = \{V_i | sum(p_{i-1} \leq x \leq p_i) \wedge x \in S\}$, query interval $[a, b]$
Result: approximate aggregation value $\hat{Q}$

```
1  I_a, I_b, H_a, H_b ← LowAndHighPoint(P, a, b);
2  C ← {[I_a, I_b], [I_a, H_b], [H_a, I_b], [H_a, H_b]};
3  c ← +∞;
4  interval ← ∅;
5  S' ← Subsample(S);
6  for t in C do
7  |   tmp ← confidence(t, S');
8  |   if tmp < c then
9  |   |   c ← tmp;
10 |   |   interval ← t;
11 |   end
12 end
13 Q̂ ← AggregationValue(interval, V);
```

of the query (Line 1), according to these four points constructed four candidate intervals (Line 2). Initialize to find the candidate interval (Line 3–4). In order to improve the computational efficiency of confidence, the set is subsampled once (Line 5). Calculate the confidence of candidate interval in subsample set S' and find the minimum (Line 6–12). Finally, the approximate result can be obtained by simple calculation according to the stored aggregated values V and the found interval (Line 13).

Example 3.2. We want to find out the age distribution in the census data, and there comes a query:

$$\text{SELECT age, COUNT(*) FROM census WHERE } 23 \leq age \leq 67.$$

Assume that the age distribution on the census is between $[10, 100]$, and after Algorithm 1 we got a set of 6 partition points as $P = \{22, 35, 47, 61, 85, 100\}$. So we know $I_a = 22, I_b = 61, H_a = 35, H_b = 85$ and get the four candidate intervals as $[22, 61], [22, 85], [35, 61], [35, 85]$. After confidence calculation, we get that the interval $[22, 61]$ is the best approximate result, then we can get the aggregation result by adding the precomputed values of $[10, 22], [22, 35], [35, 47], [47, 61]$.

3.3 Analysis for Twice Approximations

We use the following theorem to prove that the error of twice approximation methods using approximate materialized views is controllable.

Result Estimation. Given dataset D and the sample set S on D, for some aggregate queries q, the answer can be estimated from the sample:

$$q(D) \approx \hat{q}(S) \tag{3}$$

AQP++ uses samples to estimate the difference between query results and pre calculated aggregate values. Let q denote user's query, pre denote precomputed aggregate quary on complete dataset, i.e.

q: SELECT f(A) FROM D WHERE Condition
q: SELECT f(A) FROM D WHERE Condition

AQP++ estimates the differences in query results as follows:

$$q(D) - pre(D) \approx \hat{q}(S) - \hat{pre}(S) q(D) \approx pre(D) + \hat{q}(S) - \hat{pre}(S) \tag{4}$$

In the general case, after data pre aggregate, the $pre(D)$ obtained by AQP++ is a known constant. Moreover, AQP++ can estimate the results of user's query $\hat{q}(S)$ and the precomputed results $\hat{pre}(S)$ by formula (3). Bring these two estimates into formula (4) to get the final approximate results. When data pre aggregation is based on samples, we use Δ to represent the difference between sample set S and dataset D, i.e. $pre(D) = pre(S) + \Delta$. Substitute it into formula (??), there will be:

$$q(D) \approx pre(S) + \Delta + \hat{q}(S) - \hat{pre}(S) \tag{5}$$

where $pre(S)$ has been precomputed. Similarly, we can get $\hat{q}(S)$ and $\hat{pre}(S)$ by formula (3).

Error Estimation of Twice Approximation. We use the approximate materialized views for estimation, formula (5) is further approximated by rounding off Δ:

$$q(D) \approx pre(S) + \hat{q}(S) - \hat{pre}(S) \tag{6}$$

Compare formula (4) and (6), the difference is only the constant value $pre(D)$ and $pre(S)$ in the precomputation part on the right side of the formula, so Lemma 1 still holds.

Lemma 1. *For any aggregation function f, if AQP can anwer queries like:* SELECT *f(A)* FROM *D* WHERE *Condition, then AQP++ can also answer the query.*

Because the constant value from $pre(D)$ to $pre(S)$, when AQP gets unbiased estimation, the answer returned by twice approximation is likely to be biased. If AQP get the unbiased answer, i.e. $q(D) = E[\hat{q}(S)]$ and $pre(D) = E[\hat{pre}(S)]$. Based on formula (6) we get the result $pre(S) + \hat{q}(S) - \hat{pre}(S)$, calculate its expectation:

$$\begin{aligned} & E[pre(S) + \hat{q}(S) - \hat{pre}(S)] \\ & = E[pre(S)] + (E[\hat{q}(S)] - E[\hat{pre}(S)]) \\ & = pre(S) + (q(D) - pre(D)) \\ & = q(D) + (pre(S) - pre(D)) \\ & = q(D) - \Delta \end{aligned} \tag{7}$$

When precomputation is based on the complete datasetm, i.e. $pre(S)$ equals $pre(D)$, AQP++ get unbiased estimation. But when the precomputation is based

on the sample set, the expectation difference after twice approximation only comes from the difference Δ. Then we consider the variance of twice approximation:

$$\begin{aligned}
&D[pre(S) + \hat{q}(S) - \hat{pre}(S)] \\
&= E\{[pre(S) + \hat{q}(S) - \hat{pre}(S) - (pre(S) + \hat{q}(S) - \hat{pre}(S))]^2\} \\
&= E\{[pre(S) + \hat{q}(S) - \hat{pre}(S) - (q(D) + (pre(S) - pre(D)))]^2\} \\
&= E\{[pre(D) - \hat{pre}(S)]^2\} \\
&= E\{pre^2(D) - 2pre(D)\hat{pre}(S) + \hat{pre}^2(S)\} \\
&= pre^2(D) - 2pre(D)\hat{pre}(S) + D(\hat{pre}(S)) + E^2[\hat{pre}(S)] \\
&= D[\hat{pre}(S)]
\end{aligned} \tag{8}$$

Formula (8) explain that the variance of the result estimation is completely from the error of sample estimation in the pre aggregation stage. Therefore, it can be inferred from the results of formula (7) and (8) that a reasonable sampling method can make the preprocessing result on the sample dataset approximate to that on the complete dataset. It means the error of twice approximation is controllable.

4 Experimental Results

In this seciton, we exprimentally study the proposed algorithms.

4.1 Experimental Setup

We will introduce the hardware, datasets and some important parameters before we describe and analyze the experimental results.

Hardware All the experiments were conducted on a laptop with an Intel Core i7 CPU with 2.20 GHz clock frequency and 16 GB of GAM.

Datasets We use UCI's public dataset: *Adult Data Set*. This dataset is extracted from census data, with 15 attributes and 32561 instances. Attributes include numerical (age, income, etc.) and discrete (gender, nationality, etc.) The attribute $fnlwgt$ is the weight control value, because it has no important meaning in the queries, we ignored it in the experiment. Each row of the dataset has an instance, the attributes are arranged in order and separated by commas.

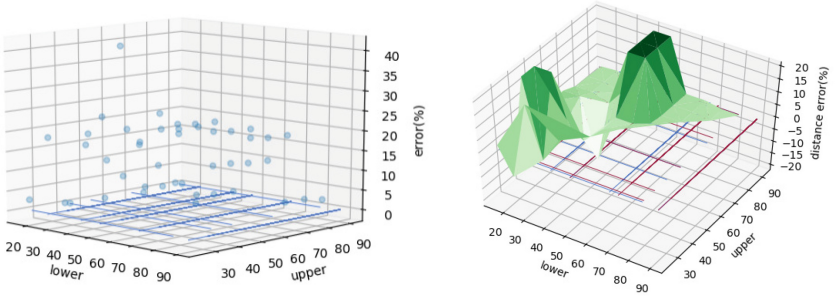
Parameters We execute experiments on attribute *age*, which is a integer attribute and has the value range $[17, 75]$. The default number of interval partition points $k = 5$.

4.2 Accuracy

The sampling methods are simple random sampling and stratified sampling based on attribute *age*, about 10% of the data are extracted from both sampling methods. We analyze the accuracy of approximate materialized views reuse from two perspectives:

– The difference between the approximate query results on the approximate materialized view and the exact query results on the complete dataset: compare absolute error, X and Y axis are the upper and lower bounds of recognition interval, the error is taken as Z axis. The mark lines for X and Y axis represent the results of interval partition in preprocessing.
– The difference between the approximate query results on the approximate materialized view and the approximate query results on the complete dataset: the interval partitions are obtained on the sampling sample and the complete dataset respectively. Then get the approximate results according to the partition points. We compare the difference between them and show it in the form of triangular surface graph. The difference is represented by $Q_C - Q_{AMV}$, where Q_C is the results on complete dataset, Q_{AMV} is the result based on approximate materialized views. The blue and red mark line of X and Y aixs represent the repective results of interval partitions.

Simple Random Sampling. Using simple random sampling, the results are shown in Fig. 2. The processing time to generate the partition is about 627 s.



(a) Compared with the accurate results (b) Compared with the approximate results on complete dataset

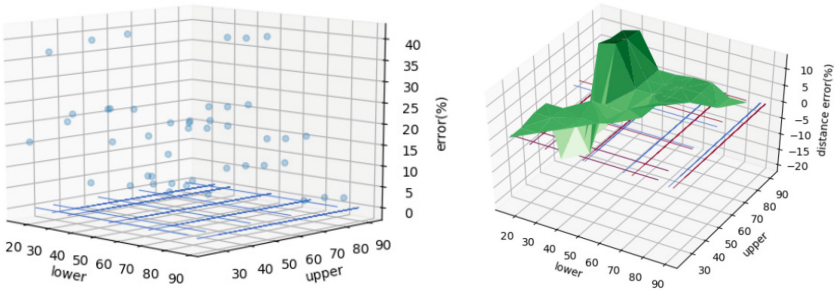
Fig. 2. Accuracy of simple random sampling

Figure 2(a) indicates the errors are concentrated in the region of 0% and 15%, few points (only one in Figure) is more than 35%. It shows that the error can be stablized within 15% by using our method.

Analyze the results in Fig. 2(b), it can be roughly divided into three different areas: high-rise area, smooth area with error difference about 0% and concave area. The smooth area represents the approximate query result based on the approximate materialized view is similar to that using the exact materialized view. The high-rise area indicates that our method performs better, and concave area means it is not good as using materialized view from the complete dataset. Since most of the regions in the graph are smooth and high-rise, it can be shown

that the approximate query based on approximate materialized views has better performance in accuracy on simple random sampling.

Attribute Stratified Sampling. We set every 10 years as a layer for stratified sampling and the results shown in Fig. 3. The processing time to generate the partition is about 624 s.



(a) Compared with the accurate results

(b) Compared with the approximate results on complete dataset

Fig. 3. Accuracy of stratified sampling

Errors in Fig. 3(a) are also concentrated in the region of 0% and 15%. But compared with Fig. 2(a), there are more points with large error (more than 35%), indicating that stratified sampling is not as accurate as simple random sampling in reusing approximate materialized views. Similarly divide Fig. 3(b) into three areas. Compared with Fig. 2(b), there is a significant difference in the scale of Y axis, which indicates that stratified sampling for approximate materialized view reuse can get a closer result than directly reusing materialized view. And most of the areas in Fig. 3(b) are above 0%, shows that stratified sampling can get better accuracy results.

Error Analysis. Based on the precomputation of Fig. 2(a) and 3(b), we take multiple perspectives of the result graph to analyze the error. We use red arrows and red dotted line box to identify the parts with lower error in the result of the diagram, and use black dashed line to mark the line $lower = upper$ on the $lower - upper$ plane. The results of two sampling methods are shown in Fig. 4 and 5.

Notice the red marked parts in Fig. 4 and 5, the lower errors are concentrated near the blue border on the $lower - upper$ plane. It shows that the error of approximate query is lower when query interval approaches the partition interval. This is corresponding to with the theory, i.e. when query interval is replaced by the partition intervals, there has lower error. When query interval moves to the inside of the blue line, the error trends to rise, and reach the peak in the most

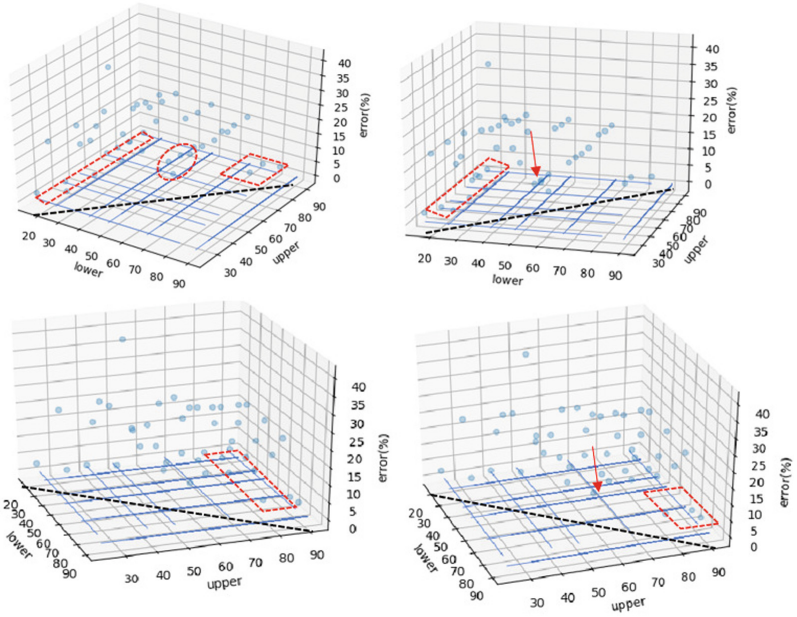


Fig. 4. Accuracy on simple random sampling. (Color figure online)

central areas, which also indicates that the query accuracy is lower when query interval distance from partition intervals.

By observing the scattered points projected near the black dotted line, the query error varies from high to low, showing an obvious fluctuation trend. The possible reason is: when the upper and lower bounds of query are close, the approximate query interval is easily changed to $[p_i, p_i]$ (p_i is a partition point), so the returned result will be 0. Once the actual data is distributed on $[p_i - \varepsilon, p_i + \varepsilon]$, the approximate result is far from the actual result.

Analyze the Fig. 2(b) and 3(b) more specifically. Use red arrows and red dotted line box the mark the high-rise part, and use the black dotted line to mark the line $lower = upper$ on the $lower - upper$ plane. The results are shown in Fig. 6.

The red mark in Fig. 6 indicates that the error of using approximate materialized view is lower than that of using general views. Moreover, these intervals show a trend of a lower bound and higher upper bound, that is, when the interval span is larger, using approximate materialized views can improve the effect obviously. The projection area near the black dotted line is basically at 0%, which indicates that when intervals have similar lower and upper bounds, the approximate materialized view has almost the same accuracy as the general method. It can be seen that this type of query interval is prone to high and low query error, which is the original limitation of using data aggregation method, and it can not be improved by using approximate materialized view.

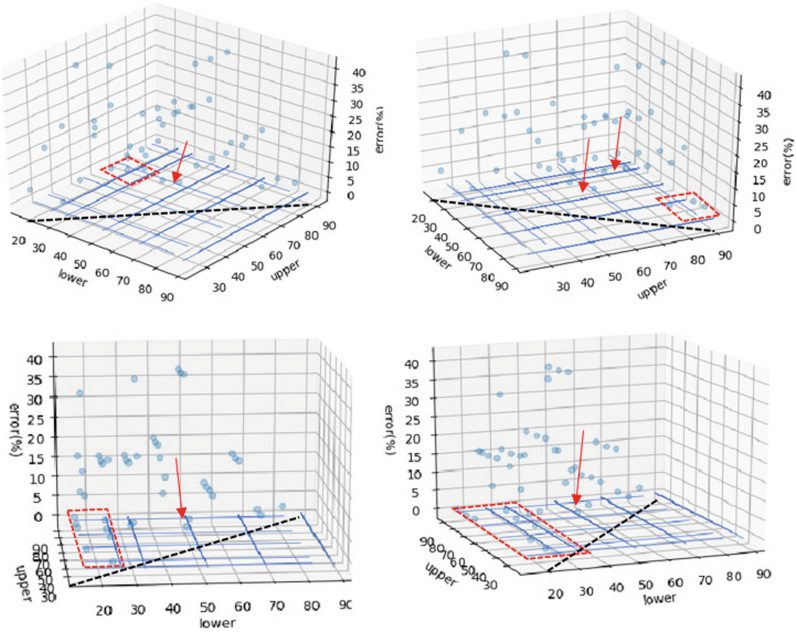


Fig. 5. Accuracy on stratified sampling. (Color figure online)

4.3 The Summary of Experimental Results

We summarize the experimental results as follows:

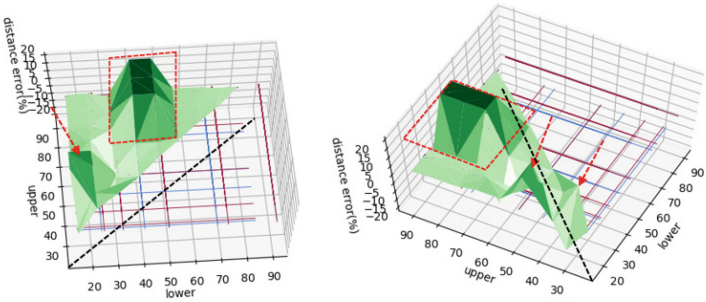
- In most cases, the performance of reusing approximate materialized view is better than that of reusing general materialized view. It can not only save preprocessing time and storage space, but also have better accuracy.
- The error of reusing approximate materialized view is mainly limited by the method of data pre aggregation.

5 Influence of Parameters

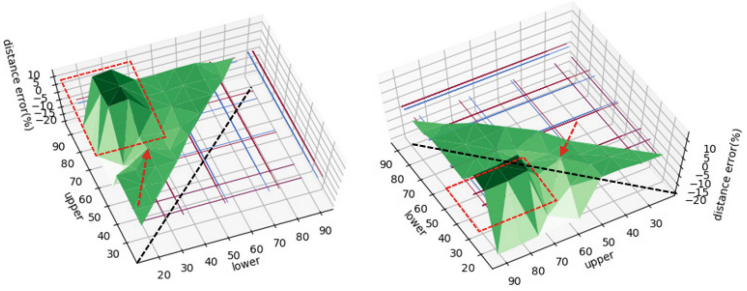
In this section, we modified the experimental parameters to study their influence on the results. These parameters include sample size and the number of partition points k.

5.1 Sample Size

We stay the other experimental conditions the same, using simple random sampling, selected 10%, 20%, 30% and 50% from the complete dataset to repeat the above experiments. The results are shown in Fig. 7. Figure 7 indicates that under different sample size, the error of using approximate materialized view is about



(a) Simple random sampling



(b) Stratified sampling

Fig. 6. Accuracy of stratified sampling

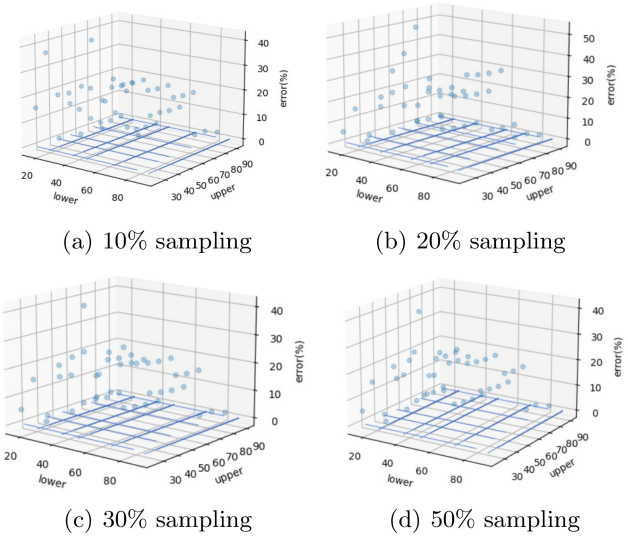
15%, and the increase of sample size has little effect on the accuracy. We think the possible reason is: the query processing use the data aggregation recognition method, query error of this method mainly comes from the difference between the pre partition points and the real query interval. The change of sample size will not lead to a huge difference of interval partition, so it has no obvious impact on the accuracy.

We compare the interval partitions under different sample size as shown in Fig. 8. Figure 8(a) indicates there is no significant difference in the time of interval partition within the increase of sample size, and Fig. 8(b) shows the results of interval partition are similar. So we can infer that the sample size has little effect on improving the pretreatment time and accuracy of reusing approximate materialized view. But considering the space occupation, lower sample size has better space performance.

5.2 Number of Partition Points

We stay the other experimental conditions the same, using simple random sampling to take about 10% samples, set the number of partition points as 3, 5, 8, 10 to repeat these experiments. The accuracy are shown in Fig. 9.

Figure 9 shows the error concentration area decreases significantly with the increase of k, so the parameter k can improve the approximate materialized



(a) 10% sampling (b) 20% sampling

(c) 30% sampling (d) 50% sampling

Fig. 7. Accuracy under different sampling scale

view. Then we compare the time-consuming of interval partition and the reduced preprocessing time by using sampling under different k, results are shown in Fig. 10.

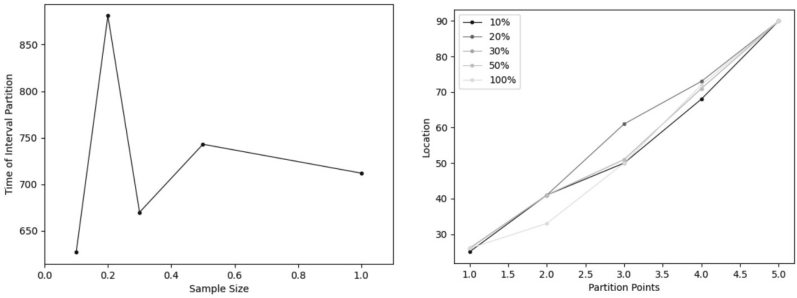
From the results of Fig. 10, the preprocessing time is positively correlated with number of partition points k, and the use of sampling can effectively reduce the time of interval partition. Moreover, this performance improvement increases with the increase of k. It shows that our method can improve the performance greatly when k is large.

Finally we compared the difference between using approximate materialized view and general view under different parameter k, as shown in Fig. 11.

According to previous description, the area above 0% in Fig. 11 indicates that our approximate materialized view have better performance. Figure 11(a) shows when k is too small, there is almost no high-rise area. As the number of partition points from $k = 5$ to $k = 8$, high-rise area increases obviously, and the depression area of $k = 8$ is smaller than that of $k = 5$. It shows that the increase of k makes the use of approximate materialized view more efficient than general views. When k becomes large ($k = 10$), the increase of high-rise area is no longer obvious, but the peak value decrease, which indicates that the difference between our method and general method is reduced. However, most of the areas are still above the horizontal plane, so using approximate materialized view still has the advantage of accuracy.

5.3 The Summary of Parameter Setting

We summarize the influence of parameters as follows:



(a) Time of Interval Partition (b) Partition Points

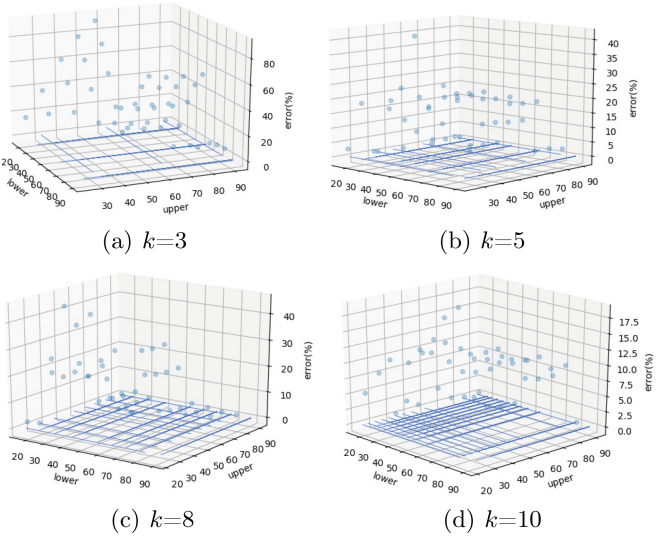
Fig. 8. Different Sample Size

- From the perspective of sampling method, single attribute stratified sampling and simple random sampling have little influence on the results.
- From the perspective of sample size, increasing the sampling scale can effectively improve the accuracy of AQP. However, the too large sample size is no longer obviously improve the performance, and the space occupation increases. The 30% sampling ratio in the experiment can get the best results.
- From the selection of parameter k, it directly affect the accuracy of AQP. When the number of partition is increased, the accuracy will be improved but the preprocessing time will be increased too. In our experiment, choosing the number of partition points with $k = 8$ can get the best comprehensive result.

6 Related Work

Tranditional database management systems execute query paradigm based on blocking. In order to deal with the challenge of interactive analysis database system answering aggregate query in a reasonable time under large scale of data, the technology of pre aggregation (materialized view, data cube) can significantly reduce query latency. But they need a lot of preprocessing, there are dimension bottlenecks and the cost of storing a complete data cube is usually very expensive. The methods that try to overcome these problems (such as imMens and NanoCubes) usually limit the number of attributes that can be filtered at the same time, and limit the possible exploration paths.

In order to achieve low latency query, the system for interactive data exploration must rely on AQP, which provide query result estimation with bounded error. Most AQP systems conduct research on sampling, including using some form of biased sampling (such as AQUA, BlinkDB [2], DICE), and study how to generate better hierarchical samples [2–6], or trying to supplement samples with auxiliary index [3,7]. However, these methods usually need a lot of preprocessing time to obtain prior knowledge, which is still insufficient in the face of unknown queries. In addition to sampling based AQP, some non sampling techniques are



(a) k=3 (b) k=5

(c) k=8 (d) k=10

Fig. 9. Accuracy under different k

also proposed in the study [8,9]. They provide certainty by using indexes rather than samples. But for interactive query types, their effectiveness is not as good as sampling based AQP.

Based on the combination of preprocessing and approximate query, the proposed AQP++ framework [10,11] is used to connect any existing AQP engine with aggpre for the connection of AQP and aggregate precomputation of interaction analysis.

In many AQP systems, the ueusability of materialized views is brought into full play [12]. And there are many sample collection methods to help us study the generation of approximate materialized view [13,14].

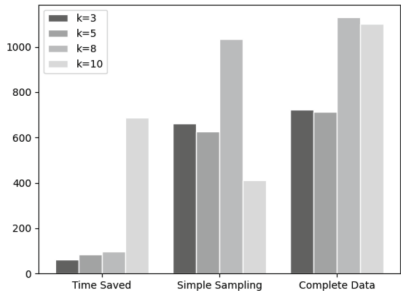
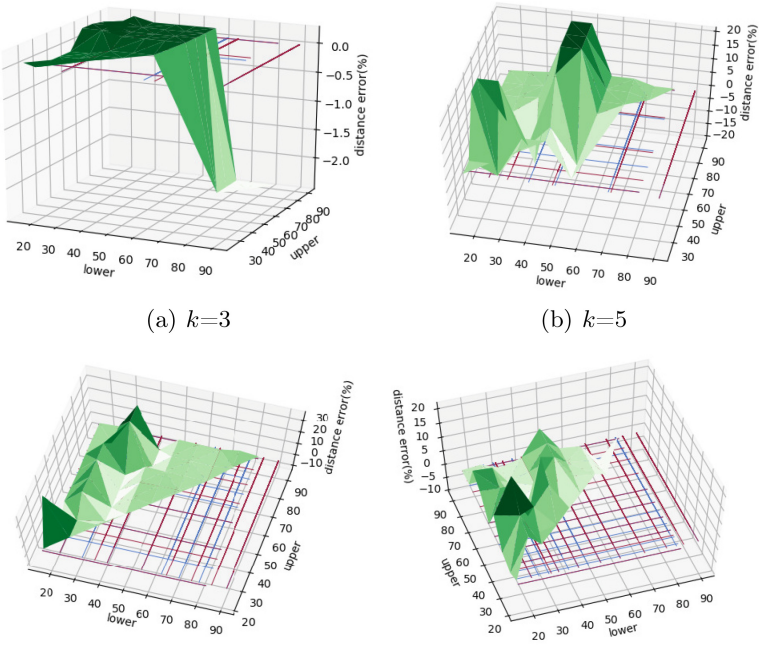


Fig. 10. Time Performance under different k



(a) k=3 (b) k=5

(c) k=8 (d) k=10

Fig. 11. Compared with General Views

7 Conclusion

In this paper, we proposed to generate approximate materialized views on sample datasets and combine it with AQP++ framework to improve its performance in the data preprocessing stage. The experimental results show that sampling can reduce the preprocessing time based on data aggregation, thus greatly improving the overall time performance and reduces the occupation of storage space. And when the sample size is not too small, using approximate materialized views can get approximate query results with smaller error than using general view in more cases. Generate a better approximate materialized view is still a problem to be studied. Our research is limited to the method of data pre aggregation. We plan to horizontally select more approximate query methods using materialized views for comparison, and consider more complex sample extraction methods to test our views.

References

1. Gray, J., et al.: Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub totals. In: Data Mining and Knowledge Discovery, pp. 29–53 (1997)

2. Agarwal, S., Mozafari, B., Panda, A., Milner, H., Madden, S., Stoica, I.: BlinkDB: queries with bounded errors and bounded response times on very large data. In: EuroSys (2013)
3. Chaudhuri, S., Das, G., Datar, M., Motwani, R., Narasayya, V.R.: Overcoming limitations of sampling for aggregation queries. In: ICDE (2001)
4. Acharya, S., Gibbons, P.B., Poosala, V.: Congressional samples for approximate answering of group-by queries. ACM SIGMOD Rec. **29**(2), 487–498 (2000)
5. Chaudhuri, S., Das, G., Narasayya, V.R.: A robust, optimization-based approach for approximate answering of aggregate queries. In: SIGMOD (2001)
6. Ganti, V., Lee, M., Ramakrishnan, R.: ICICLES: self-tuning samples for approximate query answering. In: VLDB (2000)
7. Moritz, D., Fisher, D., Ding, B., Wang, C.: Trust, but verify: optimistic visualizations of approximate queries for exploring big data. In: CHI (2017)
8. Cao, Y., Fan, W.: Data driven approximation with bounded resources. PVLDB **10**(9), 973–984 (2017)
9. Potti, N., Patel, J.M.: DAQ: a new paradigm for approximate query processing. PVLDB **8**(9), 898–909 (2015)
10. Peng, J., Zhang, D., Wang, J., et al.: AQP++: connecting approximate query processing with aggregate precomputation for interactive analytics. In: The 2018 International Conference. ACM (2018)
11. Wang, Y., Xia, Y., Fang, Q., et al.: AQP++: a hybrid approximate query processing framework for generalized aggregation queries. J. Comput. Sci. **26**, 419–431 (2017)
12. Galakatos, A., Crotty, A., Zgraggen, E., et al.: Revisiting reuse for approximate query processing. Proc. VLDB Endow. **10**(10), 1142–1153 (2017)
13. Gibbons, P.B., Matias, Y.: New sampling-based summary statistics for improving approximate query answers. ACM SIGMOD Rec. **27**(2), 331–342 (1998)
14. Babcock, B., Chaudhuri, S., Das, G.: Dynamic sample selection for approximate query processing. In: The 2003 ACM SIGMOD International Conference on Management of Data. ACM (2003)