Approximate Range-Sum Queries over Data Cubes Using Cosine Transform

Wen-Chi Hou, Cheng Luo, Zhewei Jiang, and Feng Yan

Abstract—In this research, we propose to use the discrete cosine transform to approximate the cumulative distributions of data cube cells' values. The cosine transform is known to have a good energy compaction property and thus can approximate data distribution functions easily with small number of coefficients. The derived estimator is accurate and easy to update. We perform experiments to compare its performance with a well-known technique - the (Haar) wavelet. The experimental results show that the cosine transform performs much better than the wavelet in estimation accuracy, speed, space efficiency, and update easiness.

Keywords—DCT, Data Cube

I. Introduction

DATA warehouse is a large collection of integrated data, built to assist knowledge workers, such as executives, managers, analysts, etc., to make better and faster decisions. It is often required that the data be summarized at various levels of detail and on various combinations of attributes for on-line analytical processing (OLAP), which allows analysts to gain insight into the data through a variety of views. Typical OLAP applications include product performance and profitability, effectiveness of sales programs or marketing campaigns, sales forecasting, capacity planning, etc. Data warehousing and OLAP have increasingly become a focus of the database industry. OLAP systems generally support a multidimensional data model, known as a data cube [6]. A range-sum query, a very common and useful type of query over data cubes, is to compute the sum of measure attribute values of data cube cells that fall in the ranges specified by the query. It is very useful in finding trends and discovering relationships between attributes in OLAP. To facilitate range-sum query processing, prefix-sum cubes are proposed [3], [4], [5], [7]. Although these methods generally can answer range-sum queries quickly, updates to data cubes can propagate to large portions of the prefix-sum cubes, incurring tremendous overheads. In addition, these approaches all require at least as much space as the original data cubes to store the prefix-sum cubes.

Approximate query answering present an appealing alternative to conventional query processing when exact answers are too slow or costly to derive. Fast approximate answers are very useful in exploratory data analyses, such as OLAP, decision support, and data mining. They provide quick summary information to users to help refine search in a potentially tedious mining process or in an ad-hoc drill-down and roll-up in OLAP [1]. In this research, we attempt to find a method that not only can provide fast approximate answers to range-sum queries, but also uses very little space. In addition, the approach is dynamically updatable in the presence of updates to data cubes.

In this research, we propose to use discrete cosine transform (DCT) to approximate prefix-sum cubes. The discrete cosine transform is known to have a good energy compaction property and thus can approximate the distribution of data cube cells' values easily using only a few low frequency terms. The derived estimator is accurate and easy to update. We perform experiments to compare its performance with a well-known technique, the (Haar) wavelet [11], [15]. Experimental results show that DCT performs better than the wavelet in estimation accuracy, speed, space efficiency, and update easiness.

The paper is organized as follows. Section II reviews previous studies on data cube compression and prefix-sum cubes. Section 3 introduces the notations. Section 4 discusses approximation using cosine transform. Section 5 presents the range-sum query estimation. Section 6 discusses the updatability of the cosine estimator. Section 7 presents the experimental results. Section 8 concludes the paper.

II. Related Work

The condensed cube [16] condenses tuples from different cuboids that are aggregated from the same set of tuples into one tuple. The quotient cube method [8] partitions a cube into classes of cells with identical aggregate values to save storage space. Dwarf [14] accomplishes size reduction of data cubes by factoring redundant prefixes and suffixes out of the data warehouse. Unfortunately, the constructions of such cubes are generally complex and the effectiveness of these reduction methods heavily depend upon the properties of the data themselves. For range-sum queries, substantial portions of the size-reduced cubes may have to be accessed. Li et al. [10] indicated that poor query performance was observed in condensed and quotient cubes.

Quasi-cubes [2] slice the data cubes and approximate the distributions of subcubes by linear functions. It is not clear how range-sum queries can be answered when subcubes partially intersect with the ranges of queries. In addition, updates are difficult to handle on the fly and periodical reconstructions of the linear functions may be required.

To facilitate range-sum query processing, Ho et al. [7] computed the prefix sums of data cubes. Although this method can answer queries fast, an update in the worst case can propagate

W. Hou, Z. Jiang, and F. Yan are with the Department of Computer Science, Southern Illinois University, Carbondale, IL, 62901 Email: hou@cs.siu.edu, zjiang@cs.siu.edu, and fyan@cs.siu.edu.

C. Luo is with the Department of the Mathematics and Computer Science, Coppin State University, 2500 West North Avenue, Baltimore, MD, 21216 Email: cluo@coppin.edu

to the entire prefix-sum cube, which is as large as the original data cube. To control the cascading of updates, Geffner et al. [4], [5] decomposed the prefix-sum cubes recursively and Chan et al. [3] organized the prefix-sum cubes hierarchically; but the complexity of update still increases exponentially with the number of dimensions. In general, update propagation is a common problem for all these prefix-sum data cube approaches. Note that all these approaches require tremendous amounts of space, at least as large as the sizes of the original data cubes, to store the prefix-sum cubes.

The wavelet transforms [13] decompose the original signal by applying high-pass and low-pass filters repeatedly until a predefined decomposition level is reached. The Haar transform is conceptually simple and fast. It is proved that the largest coefficients in absolute value carry the most important information of the original signal in the Haar transform. Thus, the original signal can be compressed using a small number of coefficients that have largest absolute values. The wavelet transform has been used to compress histograms for selectivity estimation [11]. Vitter et al. [15] compressed the data cubes using the Haar wavelet and showed that estimates of aggregation queries can be derived quickly and accurately. Matias et al [12] have enhanced the method with a dynamic update scheme for its coefficients.

In this study, we use the discrete cosine transform (DCT) to approximate the cumulative distribution of the measure attribute values in the data cubes. DCT is known to have a good energy compaction property and thus can approximate a distribution easily using a few (low-frequency) coefficients. DCT also has a simple and efficient update method. Since the domains of dimension attributes are all discrete or already discretized, we shall use ”discrete cosine transform” and ”cosine transform” interchangeably in the paper for simplicity.

The wavelet method [15] is most relevant to our approach as both attempt to apply mathematical techniques to compress the data cubes. We shall have more in depth comparisons of the two approaches in subsequent sections.

III. Notations

A. Attribute Value Normalization

Consider a data cube with d dimension attributes X_1, ..., X_d. By converting a categorical domain to numerical, all dimension attributes can be viewed as numerical. Let M be a measure attribute whose values are of interest. We assume M has the set of real numbers R as its domain.

To simplify notations and algorithm implementation, dimension attribute values are normalized to a predetermined domain $[0, 1]$. Let $maxX_i$ and $minX_i$ be the maximal and minimal values of attribute X_i, respectively. Then, each value x_i of X_i can be normalized as follows:

$$x_i^z = \frac{x_i - \min X_i}{\max X_i - \min X_i}, \qquad \min X_i \leq x_i \leq \max X_i \tag{1}$$

From now on, we shall assume all attribute values are so normalized and shall not distinguish x$_i$from x$_i^z$, unless otherwise stated.

B. Range-Sum Queries

Let $X = (X_1, \ldots, X_d)$ be the set of dimension attributes. For each *1*≤ *i*≤ *d*, denoted by x_i is a value of the attribute X_i. Given $x = (x_1, \ldots, x_d) \in [0,\ 1]^d$, and $y = (y_1, \ldots, y_d) \in [0,\ 1]^d$, we say $x \leq y$ if $x_i \leq y_i$ for all *i=1, ..., d*.

We assume all range constraints in a range-sum query have the form $a_i < X_i \leq b_i$. Other forms of range constraints, such as $a_i \leq X_i \leq b_i$, $a_i \leq X_i < b_i$, $a_i < X_i < b_i$ can all be converted to this form. For example, $a_i \leq X_i \leq b_i$ can be rewritten as $a_i^- < X_i \leq b_i$ where a_i^- is largest X_i value that is smaller than a_i. Let $\{X_{i1}, ..., X_{ik}\}$ be the set of attributes on which range constraints, such as $a_i < X_i \leq b_i$, are posted in the queries. By denoting $a_i = 0$ and $b_i = 1$ for $i \notin \{i_1, ..., i_k\}$, a range-sum query $Q(a, b)$, where $a = (a_1, \ldots, a_d) \in [0, 1]^d$, $b = (b_1, \ldots, b_d) \in [0, 1]^d$, and $a_i \leq b_i$ for all $i = 1, \ldots, d$, is to compute the sum of measure attribute values for those cells whose dimension attribute values satisfy $a_i < X_i \leq b_i$ for all $1 \leq i \leq d$.

C. Empiric Distribution

Consider a random variable M that has a cumulative distribution function F. If F is known, the probability of M falling in the range $(a, b]$ is

$$P\{a < X \leq b\} = F(b) - F(a) \tag{2}$$

For simplicity, we shall use the term “distribution” for “cumulative distribution” from now on. The empiric (cumulative) distribution represents exactly the data distribution of the data cube without losing any information. Consider a data cube as a sample from the measure attribute domain R. Each cell of the data cube can be viewed as an observation. Given a set of n non-zero (measure attribute) observations $\{v_1, ..., v_n\}$, whose coordinates in the data cube are $\{y_1, y_2, \ldots, y_n\}$, the empiric distribution function $\hat{F}(x)$ is defined by

$$\hat{F}(x) = \sum_k v_k, \quad whose\, y_k \leq x, \quad x \in [0,\ 1]^d \tag{3}$$

Then given a sample of a random variable M, we can use the empirical distribution constructed from the sample to estimate the distribution of M.

Consider a one-dimensional example here. Given a sample of 6 measure attribute values {3, 1, 4, 1, 2, 2} at coordinates {0.2, 0.3, 0.4, 0.5, 0.6, 0.8}, we can estimate the probability $P\{X \leq 0.4\}$ as the ratio of the sum of the values whose coordinates are less than or equal to 0.4 (i.e., 8) to the sum of all the values (i.e., 13), that is 8/13.

In this paper, our goal is to find an approximation, denoted by $\hat{F}_m(X)$, to the empiric distribution $\hat{F}(X)$. The approximation shall use much less space and yet with little information loss.

IV. Empirical Distribution Estimation via Cosine Series

In this section, we discuss how to derive an estimator for the empiric distribution.

Let the cosine series be denoted as $\phi_i(x),\ \ i \geq 0$,

$$\phi_i(x) = \begin{cases} 1, & i = 0; \\ \sqrt{2}\ \cos i\pi x, & i > 0; \end{cases}$$

That is, $\{1,\ \sqrt{2}cos\pi x,\ \sqrt{2}cos2\pi x, \ldots, \sqrt{2}cosi\pi x, \ldots\}$. Let $\Phi_i(x) = \int_x^1 \phi_i(u)du$, That is,

$$\Phi_i(x) = \begin{cases} 1 - x, & i = 0; \\ -\sqrt{2}\sin i\pi x/i\pi, & i > 0; \end{cases}$$

We assume for simplicity all dimension attributes have the same domain size D. Consider a d-dimensional data cube with n non-zero measure attribute values (or cell values) $v_1, v_2, \ldots, v_n$, whose coordinates are $y_1,\ y_2,\ \ldots,\ y_n$, respectively. Then, the empirical distribution $\hat{F}(x_1, ..., x_d)$ of the cells' values can be represented by a cosine series with D^d coefficients, $\hat{\beta}_{i_1,...,i_d}$ $0 \leq i_1, ..., i_d \leq D-1$, as

$$\hat{F}(x_1, ..., x_d) = \sum_{i_1=0}^{D-1} .... \sum_{i_d=0}^{D-1} \overset{\wedge}{\beta}_{i_1,...,i_d} \prod_{j=1}^{d} \phi_{i_j}(x_j) \qquad (4)$$

where $\hat{\beta}_{i_1,...,i_d}$'s are

$$\hat{\beta}_{i_1,...,i_d} = \sum_{k=1}^{n} v_k (\prod_{j=1}^{d} \Phi_{\ i_j}(y_{kj})) \qquad (5)$$

where y_{kj} is the jth-dimension coordinate of y_k,$1 \leq k \leq n$.

While the empirical function $\hat{F}(x_1, ..., x_d)$ describes the exact distribution of tuples, the storage of such a function could be large, especially when the number of domain attributes d and the domain sizes thereof are large. To save storage space, we opt to approximate the function by a smaller number of cosine coefficients.

The cosine transform has a good energy compaction property; most energy is preserved in the first few low frequency terms. Thus, one can approximate, without much information loss, the empirical distribution $\hat{F}(x_1, ..., x_d)$ with its first m^d (low frequency) terms, denoted $\hat{F}_m$, as

$$\hat{F}(x_1, ..., x_d) \approx \hat{F}_m(x_1, ..., x_d) = \sum_{i_1=0}^{m-1} .... \sum_{i_d=0}^{m-1} \overset{\wedge}{\beta}_{i_1,...,i_d} \prod_{j=1}^{d} \phi_{i_j}(x_j) \qquad (6)$$

In general, the larger the m value, the better the approximation. It is noted that only the m^d coefficients (real numbers) need to be stored for the approximate frequency function. In contrast, besides the coefficients, the wavelet needs to store the indexes of coefficients too (to indicate which terms are selected). Thus, the cosine method is more space-efficient than the wavelet method.

Example 4.1: Consider a one-dimensional data cube with 6 non-zero measure attribute values v's: { 2, 1, 5, 4, 2,1} and their respective coordinates y's :{0.12, 0.32, 0.33, 0.66, 0.80, 0.90}. The cosine transform of this distribution is derived as follows.

$$\hat{\beta}_0 = \sum_{j=1}^{6} v_j \Phi_0(y_j) = \sum_{j=1}^{6} v_j(1 - y_j) = 7.65$$

$$\hat{\beta}_1 = \sum_{j=1}^{6} v_j \Phi_1(y_j) = \sum_{j=1}^{6} v_j[-\frac{\sqrt{2}}{\pi} \sin \pi y_j)] = -4.8951$$

The estimator of the empiric distribution with 2 coefficients is $\hat{F}_2(x) = 7.65 - 4.8951(\sqrt{2}\cos \pi x)$.

V. Estimation of Range-Sum Queries

In this section, we elaborate on the estimation of range-sum queries using the approximate empiric distributions derived in the previous section.

Let us illustrate our idea through a 1-dimensional case. Suppose X is a random variable such that $X \in [0,\ 1]$ with a cumulative distribution function $F(x)$. The probability that $a < X \leq b$ is

$$P\{a < X \leq b\} = F(b) - F(a) \approx \hat{F}_m(b) - \hat{F}_m(a) \qquad (7)$$

Let us extend X to a d-variate random vector. Let $a = (a_1, ..., a_d) \in [0,\ 1]^d$, $b = (b_1, ..., b_d) \in [0, \cdots 1]^d$, and $a < b$ A vertex of the hyperinterval (a, b]

$$\begin{aligned} (a,\ b] &= \{x = (x_1, ..., x_d) \in [0,\ 1]^d : a_1 < x_1 \\ &\leq b_1, ..., a_d < x_d \leq b_d\} \end{aligned} \qquad (8)$$

is denoted as $u = (u_1, ..., u_d) \in [0,\ 1]^d$ with $u_i \in \{a_i,\ b_i\}$ for $i = 1, \ldots, d$. Let $\Delta_k(a,\ b)$ be the set of all vertices u with $u_i = a_i$ for exactly k coordinates and $u_j = b_j$ for the remaining coordinates. Then,

$$\begin{aligned} &P\{a_i < X_i \leq b_i, \quad \forall\, 1 \leq i \leq d\} \\ = \ &\sum\nolimits_{k=0}^{d} (-1)^k \sum\nolimits_{u \in \Delta_k(a,b)} F(u) \end{aligned} \qquad (9)$$

Thus, the range-sum query $Q(a, b)$ is estimated as

$$Q(a, b) \approx \sum\nolimits_{k=0}^{d} (-1)^k \sum\nolimits_{u \in \Delta_k(a,b)} F(u). \qquad (10)$$

Example 5.1. Suppose $d = 2$, $a = (a_1, a_2) \in [0, 1]^d$, $b = (b_1, b_2) \in [0, 1]^d$, $a_1 \leq b_1$, $a_2 \leq b_2$. Then, by Eq. (9)

$$\begin{aligned} &P\{a_i < X_i \leq b_i, \quad \forall\, 1 \leq i \leq 2\} \\ = \ &F(b_1,\ b_2) + F(a_1,\ a_2) - F(a_1,\ b_2) - F(b_1,\ a_2). \end{aligned}$$

Algorithm 1 summarizes the computation of a range-sum query $Q(a, b)$.

The above algorithm calls Empiric-Distribution, depicted in Algorithm 2, to compute the empirical distribution of the measure attribute values $\hat{F}_m(p)$ (i.e., Eq.(6) at a specific point p.

A. Storage of the Estimator

By utilizing the good energy compaction property, one can further filter out high frequency terms without much information loss. A technique, called Triangle Sampling [9] can be applied. It stores only those coefficients whose indexes satisfy $i_1 + ... + i_d \ \leq\ m - 1$. Thus, the number of the coefficients finally stored is $C(m + d - 1, \quad d) \approx m^d/d!$, which is much smaller than m^d. Note that the indexes (i_1, $\ldots$, i_d) of the coefficients need not be stored because they are unique and can be derived on the fly.

For example, consider a 2-dimensional case ($d = 2$) and m has been set to 3. Then, there will be m^d (=3^2) coefficients in the approximation function, denoted as $C_{i,j}$, $0 \leq i, j \leq m-1$.

```
Algorithm 1: Range-sum(m,a,b,$\beta$[],T)
Input: an integer $m > 0$, two vectors $a = (a_1, \ldots, a_d)$,
       $b = (b_1, \ldots, b_d)$, $l \leq a_i \leq b_i \leq r$, and $m^d$
       coefficients $\{\beta[0, \ldots, 0], \ldots, \beta[m-1, \ldots, m-1]\}$
       of the orthogonal series estimators $\hat{F}_m(x)$.
Output: an estimation of $Q(a, b)$.
begin
    prob $\leftarrow$ 0;
    $k \leftarrow 1$;
    for $i \leftarrow 0$ to $2^d - 1$ do
        for $j \leftarrow 0$ to $d - 1$ do
            if $i$ mod $2^j = 0$ then
                $p[j] = a[j]$;
                $k \leftarrow (-k)$;
            end
            else
                $p[j] = b[j]$;
            end
        end
        prob $\leftarrow$ prob $+k\times$
        Empiric-Distribution$(m, p, \beta[])$;
    end
    if prob$> 0$ then
        return prob;
    end
    else
        return 0;
    end
end
```

```
Algorithm 3: Empiric-Distribution($m, p, \beta[]$)
Input: an integer $m > 0$, a vector $p = (b_1, \ldots, b_d)$,
       $l \leq b_i \leq r$, and $m^d$ coefficients
       $\{\beta[0, \ldots, 0], \ldots, \beta[m-1, \ldots, m-1]\}$ of the
       orthogonal series estimators $\hat{F}_m(p)$.
Output: the empiric distribution $\hat{F}_m(p)$ valued at $p$.
begin
    $s \leftarrow 0$;
    $k \leftarrow 1$;
    for $i_1 \leftarrow 0$ to $m - 1$ do
        $\ldots\ldots$;
        for $i_d \leftarrow 0$ to $m - 1$ do
            $s \leftarrow s + \beta[i_1, \ldots, i_d]\Pi_{j=1}^d \phi_{i_j}(p_j)$;
        end
    end
    return $s$;
end
```

Fig. 1. Compute the Empiric Distribution $\hat{F}_m(p)$ at p

```
Algorithm 2: Empiric-Distribution($m, p, \beta[]$)
Input: an integer $m > 0$, a vector $p = (b_1, \ldots, b_d)$,
       $l \leq b_i \leq r$, and $m^d$ coefficients
       $\{\beta[0, \ldots, 0], \ldots, \beta[m-1, \ldots, m-1]\}$ of the
       orthogonal series estimators $\hat{F}_m(p)$.
Output: the empiric distribution $\hat{F}_m(p)$ valued at $p$.
begin
    $s \leftarrow 0$;
    $k \leftarrow 1$;
    for $i_1 \leftarrow 0$ to $m - 1$ do
        $\ldots\ldots$;
        for $i_d \leftarrow 0$ to $m - 1$ do
            $s \leftarrow s + \beta[i_1, \ldots, i_d]\Pi_{j=1}^d \phi_{i_j}(p_j)$;
        end
    end
    return $s$;
end
```

By triangle sampling, only 6 (=$C(m + d - 1, \quad d)$) of them that satisfy the condition: $i_1 + i_2 \leq m - 1 = 2$, are kept. They are: $C_{0,0}$, $C_{0,1}$, $C_{0,2}$, $C_{1,0}$, $C_{2,0}$, $C_{1,1}$. We will incorporate this technique in our implementation.

VI. Dynamic Maintenance of the Estimator

As observed from Eq. (5), each coefficient $\hat{\beta}_{i_1,\ldots,i_d}$ of the transform is basically the sum of the product of the measure attribute values and the products of basis functions on the measure attribute value's coordinates. Therefore, for insertion or deletion of a measure attribute value, we can just compute the "contribution" of the value to the transform and then combine them with the old coefficients. That is, for insertion of a new value t at $y = (y_1, y_2, \ldots, y_d)$, $\hat{\beta}_{i_1,\ldots,i_d}$ is updated as

$$\hat{\beta}_{i_1,\ldots,i_d} = \hat{\beta}_{i_1,\ldots,i_{d1}} + t\prod_{j=1}^{d} \Phi_{i_j}(y_j) \tag{11}$$

Similarly, for deletion of a value t at $y = (y_1, y_2, \ldots, y_d)$, it is updated as

$$\hat{\beta}_{i_1,\ldots,i_d} = \hat{\beta}_{i_1,\ldots,i_{d1}} - t\prod_{j=1}^{d} \Phi_{i_j}(y_j) \tag{12}$$

An update to the measure attribute value can be accomplished by a deletion followed by an insertion. Let m be the number of coefficients of the estimator. The complexities of an insertion, a deletion, and an update are all $O(m)$.

Coefficients can be updated easily and dynamically. This property makes the cosine transform well suited in data stream environments, where tuples continuously flow in. The updates of the coefficients can be performed on the fly as well as in batch. In addition, the computation workload can be easily distributed among processors as the "contributions" of tuples can be computed separately.

VII. Experimental Results

In this section, we report experimental results of estimating range-sum queries using the cosine and wavelet methods.

A. Experiment Setup

We have implemented both methods in C++ and compiled them with GNU C/C++ Compiler V3.2.3. The test platform is Redhat Linux Enterprise 4 running on a Dell Precision 360 workstation with 3.3 GHZ CPU and 1GB RAM.

Experiments are run on both synthetic and real-life datasets. The purpose of using synthetic data is to study the methods in relation to different characteristics of data in a controlled environment. The synthetic relations are generated following the TPC-D benchmark [18] with attribute values distributed Zipfianly [17]. We generate distributions with two different z values, 0.5, and 1.0, which represent, roughly speaking, a slightly skewed and skewed distribution, respectively. The domain size of each dimension attribute is $1,024$ and the sum of measure attribute values is 10^6.

We have also used a real-life dataset from the Bureau of Census [19]. We select the data for a period of three-months, from January to March 2004. The dataset has around $140,000$ tuples for each month. The dimension attributes are Age, Education and State, whose ranges are $[1,99]$, $[1,46]$ and $[1,99]$, respectively, and the measure attribute is count (or the number of tuples).

B. Estimation Accuracy

We ran 100 queries with randomly chosen ranges on dimension attributes. The accuracy of estimation is measured by average relative errors.

1) Performance on Synthetic Datasets: Figures 2 and 3 show the estimation results of range-sum queries over two-dimensional data cube with Zipf parameters, 0.5 and 1.0, respectively. As shown in the figures, in general, the greater the number of coefficients used, the better the results for both techniques.

As shown in Figure 2, the cosine method performed better than the wavelet using the same number of coefficients for the slightly skewed distribution. The wavelet generated average errors ranging from 3.93% for 100 coefficients to 1.07% for $2,000$ coefficients, while ours from 1.02% to 0.33% for the same number of coefficients.

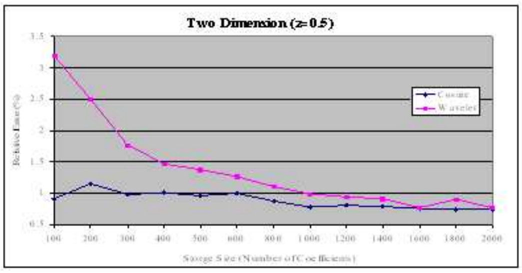

Fig. 2. Two-dimensional Queries, z=0.5

Notice that with only 100 coefficients, our estimates are already very accurate (around 1% error) in both cases. But the accuracy does not improve much when the number of coefficients increases further. This demonstrates the good energy compaction property of the cosine transform.

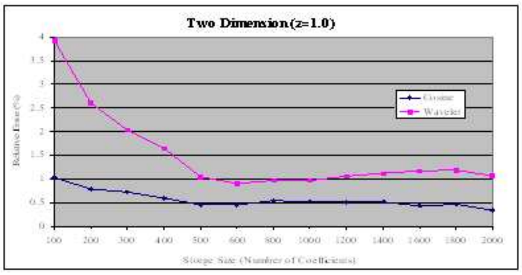

Fig. 3. Two-dimensional Queries, z=1.0

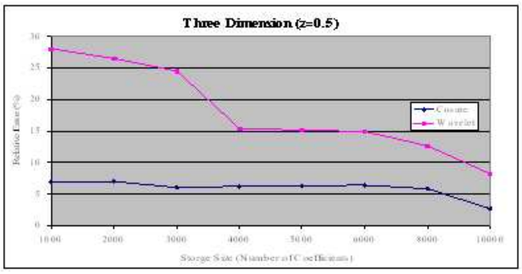

Fig. 4. Three-dimensional Queries, z=0.5

As mentioned earlier, the cosine transform stores only the coefficients, but the wavelet needs to store the indexes with the coefficients. Thus, for the same number of coefficients, the wavelet uses at least twice as much space as ours. This further demonstrates the efficiency of the cosine transform.

As the distributions become more skewed (i.e., $z = 1.0$), it becomes more difficult to capture the sharp changes in frequency and thus estimation accuracy degrades. Nevertheless, our method demonstrates an even larger performance edge over wavelet in the more skewed case ($z = 1.0$) than in the smoother case ($z = 0.5$).

Figures 4 and 5 show the results of range-sum queries with constrains on three-dimensional data cubes with Zipf parameters 0.5 and 1.0, respectively. In general, the higher the dimension, the greater the number of coefficients is needed to achieve a desired accuracy. This is mainly due to the increased number of frequency values to be approximated (or compressed) in higher dimensional spaces. Again, our method performed better than the wavelet method for the same number of coefficients. In Figure 4, wavelet generated average errors ranging from 28.04% for $1,000$ coefficients to 8.25% for $10,000$ coefficients, while ours from 6.9% to 2.67% for the

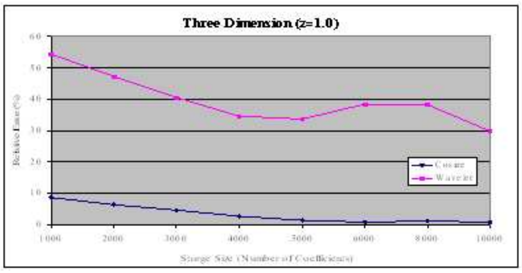

Fig. 5. Three-dimensional Queries, z=1.0

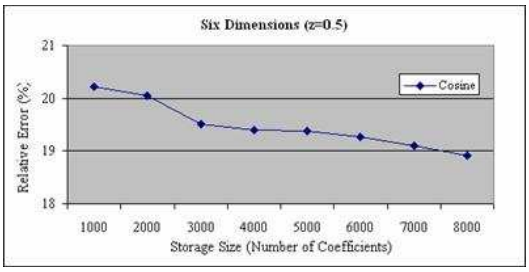

Fig. 6. Six-Dimensional Queries, z=0.5

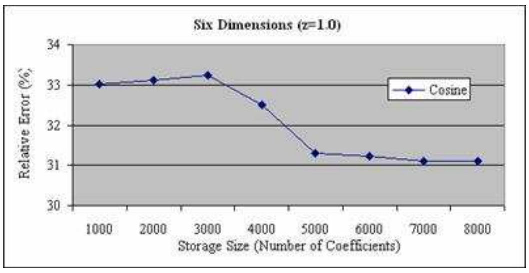

Fig. 7. Six-Dimensional Queries, z=1.0

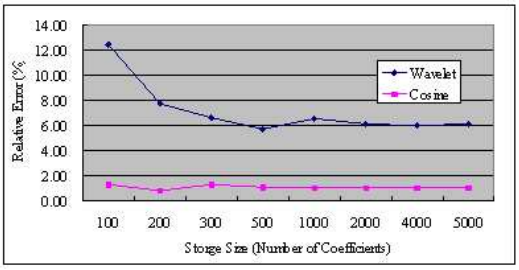

Fig. 8. Real Dataset: Two-dimension Queries

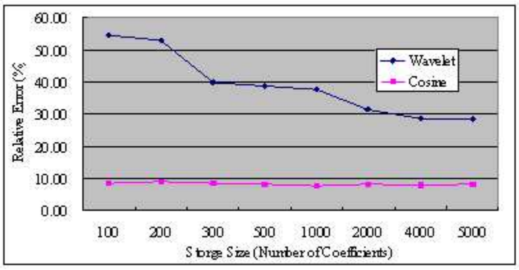

Fig. 9. Real Dataset: Three-dimension Queries

same number of coefficients. Wavelet's errors are about 4 times larger than ours.

As distributions become more skewed, the errors become greater like in the 2-dimensional cases. For example, at $z = 1.0$, the wavelet generated average errors ranging from 54.45% for $1,000$ coefficients to 29.88% for $10,000$ coefficients while our approach generated from 8.68% to 0.81% for the same number of coefficients. The wavelet's errors are about 6 to 37 times larger than ours.

In Figures 6 and 7, we show the results of six-dimensional queries. With each dimension partitioned into 16 regions, it results in a $16^6(= 16$ million)-bucket histogram. The wavelet generated large errors (e.g., $> 100\%$) for small numbers of coefficients (e.g., $1,000$, $2,000$, etc). Even for the largest number of coefficients we tested, i.e., $8,000$ coefficients, the errors are still very large, for example, 49.5% for $z = 0.5$ and 59.3% for $z = 1.0$. Due to the large errors of wavelet, we present only the results of cosine series in the following.

As a short summary, the cosine transform performs much better than the wavelet in accuracy. Nevertheless, all these two methods use much less space than a prefix-sum cube method, which requires at least as large space as the original data cube (1024^d cells).

2) A Real Dataset: Figures 8 and 9 show the results on the real dataset.

As in the synthetic experiments, our method performs better than the wavelet. For example, with only 100 coefficients, as shown in Figure 8, the errors of the wavelet and cosine methods are 12.45% and 1.68%, respectively.

For three-dimensional queries, as shown in Figure 9, with 100 coefficients, our error is already below 10% while the wavelet still has an error as high as 54%.

C. Update and Estimation Speeds

As mentioned earlier, the higher the number of dimensions, the greater the number of coefficients is required for an estimator to achieve an acceptable accuracy. We assume that the estimators have 400, $3,000$, $8,000$ coefficients for 2-dimensional 3-dimensional, and 6-dimensional cases, respectively, which we believe are generally large enough to yield reasonable accuracy.

Let m be the number of coefficients used in the estimators. It takes $O(m)$ time to update a cosine estimator, as shown in Eqs. (11) and (12). On the other hand, wavelet takes $O(logH)$ time to update the coefficients, where H is the size of the underlying histogram (or the number of cells in the data cube), recalling that wavelet is a histogram-based method. Note that the size of histogram generally increases exponentially with the number of dimensions (d) of the histogram, i.e., $H = |D|^d$, where $|D|$ is the size of each dimension attribute domain (assumed to be the same for all dimension attributes). Consequently, Table I shows wavelet is much slower in high dimensional cases.

TABLE I
UPDATE SPEED

Time (μs)	2-dimension	3-dimension	6-dimension
Wavelet	4.7	6,906	—
Cosine	132	1,250	3,002

TABLE II
ESTIMATION SPEED

Time (μs)	2-dimension	3-dimension	6-dimension
Wavelet	210	2,058	—
Cosine	72	389	1,321

The cosine method has a complexity of $O(2^d m)$ for a range-sum query estimation, as demonstrated in Eq. (9). The wavelet has a complexity of $O(2^d Hlog(H))$. Hence, the wavelet estimator can be very slow in high dimensions, as shown in Table II.

VIII. Conclusions

In this paper, we develop a nonparametric statistical range-sum query estimation approach, which is based upon the empiric distribution estimation by the cosine series. First, we derive an estimator for the empiric distribution of measure attribute values in a data cube, and then use the empiric distribution estimator to compute the range-sum query estimates. The empiric distribution estimator can be stored easily and updated efficiently. The experimental results have shown that our approach produced much more accurate estimates than the wavelet method. The proposed method is well suited for on-line approximate aggregate query estimation over data cubes. It is simple, accurate, efficient, and adaptive.

References

[1] W. Acharya and P. Gibbons and V. Poosala, Aqua: A Fast Decision Support System Using Approximate Query Answers, 1999, Proc. 25^{th} VLDB Conference.

[2] D. Barbara and M. Sullivan, Quasi-cubes: Exploiting approximation in multi-dimensional databases, 1997, SIGMOD Record, 26, 12-17.

[3] C. Chan and Y. Ioannidis, Hierarchical cubes for range-sum queries, 1999, Proc. VLDB, 675-686.

[4] S. Geffner and D. Agrawal and A. Abbadi and T. Smith, Relative prefix sums: an efficient approach for querying dynamic OLAP Data Cubes, 1999, Proc. ICDE, 328-335.

[5] S. Geffner and D. Agrawal and A. Abbadi, The dynamic data cubes, 2000, Proceeding of International Conference on Extending Database Technology (EDBT), 237-253.

[6] J. Gray and A. Bosworth and A. Layman and H. Pirahesh, Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals, 1996, Proc. ICDE Conference.

[7] C. Ho and R. Agrawal and N. Megiddo and R. Srikant, Range queries in OLAP data cubes, 1997, Proc. ACM SIGMOD Conference, 73-88.

[8] L. Lakshmanan and J. Pei and J. Han, Quotient cube: How to summarize the semantics of a data cube, 2002, Proc. 28^{th} VLDB Conference, 528-539.

[9] J. Lee and D. Kim and C. Chung, Multi-dimensional Selectivity Estimation Using Compressed Histogram Information, 1999, ACM SIGMOD, 205-214.

[10] S. Li and S. Wang, Semi-closed cube: An effective approach to trading off data cube size and query response time, Journal of Computer Science and Technology, 20(3), 367-372.

[11] Y. Matias and J. Vitter and M. Wang, Wavelet-based histograms for selectivity estimation, 1998, ACM SIGMOD Conference, 448-459.

[12] Y. Matias and J. Vitter and M. Wang, Dynamic Maintenance of Wavelet-Based Histograms, 2000, Proc 26^{th} VLDB Conference, 101-110.

[13] Y. Nievergelt, Wavelets Made Easy, 1999, Birkhauser.

[14] Y. Sismanis and N. Roussoupoulos and A. Deligiannakis and Y. Kotidis, Dwarf: Shrinking the petacube, 2002, Proc. ACM SIGMOD Conference, 464-475.

[15] J. Vitter and M. Wang and B. Lyer, Data cube approximation and histograms via wavelets, 1998, Proc. CIKM, 96-104.

[16] W. Wang and J. L. Feng, Condensed cube: An effective approach to reducing data cube size, 2002, Proceedings of the 18^{th} International Conference on Data Engineering.

[17] G. Zipf, Human behavior and the principle of least effort, 1949, Addison-Wesley.

[18] TPC, TPC benchmark D, decision support, 1995.

[19] BC, http://www.bls.census.gov/sipp/_ftp.html#sipp04, 2004.