Selectivity Estimation of Range Queries Based on Data Density Approximation via Orthonormal Series

Feng Yan[†] Wen-Chi Hou[‡] Zhewei Jiang[‡] Qiang Zhu

Department of Mathematics[†]

Department of Computer Science[‡]

Southern Illinois University at Carbondale

Carbondale, IL 62901, USA

Dept. of Computer and Information Sci.

University of Michigan - Dearborn

Dearborn, MI 48128, USA

Abstract

Selectivity estimation is an integral part of query optimization. In this paper, we propose to approximate data density functions of relations and use the approximations to estimate selectivities of range queries. A data density function here is approximated by a partial sum of an orthonormal series. Compared with histogram-based approaches, such as the wavelet, DCT, and kernel-spline methods, our approach completely avoids building and compressing histograms and hence incurs no such overheads. Experimental results have shown that our approach yields better estimates than others, especially in high dimensional spaces. Moreover, our estimator can be easily derived, efficiently stored, and dynamically updated.

Keywords: Selectivity estimation, Range queries, Data density function, Range query, Query optimization

1 Introduction

The main task of query optimization is to select an efficient query execution plan. The selection is usually based on the cost estimates of alternative plans, which in turn are based on the selectivity estimates of operators. Selectivity estimation is an integral part of query optimization.

Selectivity estimation plays an even more important role in today's database systems for several reasons. First, as queries become more complex, accurate estimates are more difficult to derive because estimation errors can propagate exponentially ([11]). Finding an accurate selectivity estimation method is now more important than ever. Second, many modern databases are very large and perhaps distributed over a network (e.g., a multidatabase). The cost of query execution plans could vary drastically ([33]). Inaccurate estimation may result in choosing a plan that turns out to be very costly. Other applications that require accurate selectivity estimation include the design and management of data warehouses and materialized views.

Various selectivity estimation methods for range queries have been proposed in the literature. Excellent surveys of these techniques appeared in ([4, 21, 27]). Among them, histogram methods ([3, 7, 13, 22, 25, 27]) have attracted the most attention due to their simplicity and accuracy. However, as the dimensionality of data increases, the space requirement of the histogram increases exponentially. Several methods, such as the wavelet transform ([23]), discrete cosine transform (DCT) ([15]), and kernel methods ([7, 14]), have been proposed to reduce the storage requirements of histograms without sacrificing much accuracy.

In this paper, we develop a nonparametric selectivity estimation approach for range queries based on the data density approximation method inspired by Rao ([28]). The data density function represents the actual data distribution of a relation. To store such a potentially complex distribution, we propose to approximate a data density function by an orthonormal series, specifically the cosine series. Similar to the wavelet, DCT, and kernel methods, we attempt to use a small number of coefficients to represent the data distribution. Unlike the wavelet, DCT, and kernel methods, which compress or approximate histograms, we approximate data distributions directly. We develop an algorithm to estimate selectivities using approximate data density functions. This novel method is applicable to all ordinal, continuous and noncontinuous, attributes. Another important property of the method is that the estimator can be updated easily and remains accurate even when the underlying data has undergone substantial changes. It possesses desirable properties of a selectivity estimation scheme for today's database systems; that is, it is efficient, accurate, adaptive, and easy to maintain.

The rest of the paper is organized as follows. In Section 2, we review selectivity estimation methods for range queries in the literature. In Section 3, we describe the mathematical foundation of our approach. In Section 4, we discuss the theory of data density approximation using orthonormal series and derive the estimator. In Section 5, we develop a selectivity estimation scheme using the derived data density estimator. In Section 6, we present the dynamic maintenance (or update) scheme for the estimator. In Section 7, we compare the estimator construction, estimation, and update complexities of the proposed method with some notable approaches, such as the wavelet, DCT, and kernel-spline methods. In Section 8, we report the experimental results. Section 9 summarizes our contributions.

2 Literature Survey

In this research, we focus on ordinal attributes, including both discrete and continuous ones. Estimation techniques on metric domains can be classified into two approaches: parametric and nonparametric ones. The parametric approach ([2, 3, 20, 29]) approximates a distribution with a model distribution function, such as a uniform, a Pearson family, or a Zipf distribution ([32]). In order to choose an appropriate model function, a priori knowledge about the actual distribution is needed. This approach can behave poorly if the model function does not fit the actual distribution well.

The nonparametric approach, which includes curve fitting, sampling, and histogram methods, uses mathematical and statistical techniques to estimate selectivities without imposing any assumptions on the distributions of data.

Curve fitting methods ([4, 17, 30]) use polynomial functions to approximate attribute value distributions. Statistics are usually collected beforehand and used to compute the coefficients of an optimal polynomial that minimizes the sum of squared estimation errors on attribute values. Special care must be taken to avoid oscillation and rounding errors ([15]).

Various sampling methods ([9, 10, 19, 16]) have also been used to estimate query result sizes. Although sampling methods can yield accurate estimates (assuming sufficient samples are taken), the cost of sampling can be considerable because frequent accesses to relations on the secondary storage may be needed ([4]).

Histogram methods ([3, 7, 13, 22, 25, 27]) have attracted the most attention due to their simplicity and accuracy. However, the size of a histogram can grow exponentially with the increase in dimensionality of data, rendering these methods prohibitively costly in high dimensions. To reduce storage space, techniques, such as the wavelet decomposition ([23]), discrete cosine transform (DCT) ([15]), and kernel methods ([7, 14]), have been proposed. It is known that both DCT and wavelet methods are able to compress a large number of histogram buckets into a smaller number of coefficients with little information loss. It has also been shown that compressed histograms compared favorably to uncompressed histograms ([23]). However, for the wavelet, a portion of the histogram needs to be reconstructed for selectivity estimation; considerable overheads are incurred. Kernel methods ([7, 14]) smooth discontinuities at histogram edges through spline interpolations. Although this measure may aid in the accuracy of approximation, it requires more space to store statistics. Specifically, the kernel-spline method ([7, 14]), a curve fitting histogram-based method, uses 4^d coefficients for each bucket (knot) of a d-dimensional histogram. The space consumption can be a serious problem in high dimensional spaces.

Dynamic update of coefficients (or statistics) is also an important issue. Although the DCT is able to update its coefficients efficiently, updates are performed in batch. Therefore, the DCT may not reflect the distribution changes in a timely manner. As for the wavelet method, it maintains an auxiliary data structure to help update the coefficients. The maintenance scheme is complex and may not always be able to promote the best candidates to replace insignificant coefficients after deletions and updates ([12]). Recently, Garofalkis et al. ([6]) have used probabilistic wavelet synopses to improve the accuracy of deterministic wavelet methods. Since it needs to reconstruct all points inside the range of a range query, it may not be feasible for range queries over large domains or continuous domains. Furthermore, it is also very difficult to maintain the synopses incrementally (dynamically) ([6]). The kernel-spline method can update its coefficients incrementally only when the locations of knots are fixed. When the underlying distribution has undergone substantial changes, reconstruction of the estimator is necessary.

3 Definitions and Background

In this section, we introduce the mathematical foundation of our approach.

3.1 Desirable Properties

In general, there are four desirable properties of a selectivity estimation scheme ([16, 18]): (1) *efficiency* – a selectivity estimate ought to be obtainable at negligible computational cost; (2) *accuracy* – a selectivity estimate should not differ significantly from its true value in most cases; (3) *adaptivity* – no assumption on the distribution or dependency of the underlying data is made; (4) *maintainability* – the maintenance of stored statistics and parameters should be easy and can be performed dynamically.

3.2 Attribute Value Normalization

Let T be a relation with attributes $X_1, \ldots, X_d$. Considering a relation as a sample and tuples of a relation as observations, we can view $X = (X_1, \ldots, X_d)$ as a d-variate random variable. For each $1 \leq i \leq d$, denoted by x_i is a value of attribute X_i. Let R be the set of all real numbers and R^d be the set of all d-dimensional real vectors. Let $x = (x_1, \ldots, x_d) \in R^d$ and $y = (y_1, \ldots, y_d) \in R^d$. We say $x \leq y$ if $x_i \leq y_i$ for all $i = 1, \ldots, d$.

Attributes have different domains. To simplify the notation and algorithm implementation, we normalize the attribute domains so that they all have the same domains $[l, r] \subset R$, where $l < r$, as follows. Let max_i and min_i be the maximal and minimal values, assumed known, of attribute X_i, respectively. If the minimum and maximum are not known in advance, let min_i and max_i be a small and a large enough values, respectively, such that few or no tuple can have an X_i value smaller than min_i or greater than max_i.

A value x_i of X_i can be normalized as:

$$x_i^z = \begin{cases} l, & x_i \leq min_i \\ \frac{x_i - min_i}{max_i - min_i}(r - l) + l, & min_i < x_i < max_i \\ r, & x_i \geq max_i \end{cases} \qquad (3.1)$$

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.

3.3 Queries

Traditional range queries have a qualification on a single attribute. Here, we extend the definitions to allow conjunctive qualifications on multiple attributes in range queries. Let $a = (a_1, \ldots, a_d) \in [l, r]^d$, $b = (b_1, \ldots, b_d) \in [l, r]^d$, and $a_i \leq b_i$ for all $i = 1, \ldots, d$. We have the following types of queries:

- *Range queries*: denoted by $Q(a, b)$, have a qualification that is a conjunction of range predicates $a_i < X_i \leq b_i$, $1 \leq i \leq d$.

- *General range queries*: have a qualification that is a conjunction of range predicates $a_i \oplus_i X_i \otimes_i b_i$, $1 \leq i \leq d$, where both $\oplus_i$ and $\otimes_i \in \{<, \leq\}$.

In this research, we consider single-attribute range queries as well as multi-attribute ones.

3.4 Data Density Functions

To describe the frequencies of attribute values, we introduce the data density function, which is the derivative of the empirical distribution function, of a set of attributes. First, we consider the simplest case - a relation with only one attribute X. Considering X as a discrete random variable, the distribution of X can be naturally represented by its frequency function. Indeed, if we look at the set of values $\{a_1, \ldots, a_n\}$ (a_i's need not be distinct) of X one by one, we can represent its distribution by a function

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} \delta(x - a_i), \quad \forall x \in R, \tag{3.2}$$

where δ is the Dirac Delta (pulse) function ([5]), describing the occurrence of individual X values. The δ function has the properties: $\delta(0) = 1$, $\delta(x) = 0$ if $x \neq 0$, and for any continuous function ψ defined on $(l, r]$,

$$\int_l^r \psi(u)\delta(x - u)\, du = \psi(x), \quad \forall x \in (l, r]. \tag{3.3}$$

We call the above function $\hat{f}(x)$ the *data density function* [1] of X. Note that we define $\hat{f}$ on the whole real line R to facilitate selectivity estimation.

Now let us consider the multi-attribute relation case. Let $X = (X_1, \ldots, X_d)$ be the set of d attributes in a relation, and $A \subset (l, r)^d$ be a set of n tuples, where $l \leq X_i \leq r$, $A = (a_1, \ldots, a_n)'$ and $a_i = (a_{i1}, \ldots, a_{id})$ for $i = 1, \ldots, n$. The data density function $\hat{f}(x_1, \ldots, x_d)$ is defined as

$$\hat{f}(x_1, \ldots, x_d) = \frac{1}{n}\sum_{i=1}^{n}\prod_{j=1}^{d} \delta(x_j - a_{ij}), \qquad \forall x = (x_1, \ldots, x_d) \in R^d. \tag{3.4}$$

Note that attributes $X_1, \ldots, X_d$ are not necessarily independent.

A data density function can be viewed as a histogram with the finest partition possible; however, it is a function rather than a collection of frequencies. It represents the exact data distribution of a relation. While having such a good property, the storage of such a function can consume significant space. Therefore, in this paper, we attempt to find an estimator for $\hat{f}(X)$, denoted by $\hat{f}_m(X)$, which shall take less space to store, yet yield a good approximation to $\hat{f}(X)$.

3.5 Sample Probability

To lay down the groundwork for selectivity estimation, let us introduce the *sample probability* $P\{B\}$ of a set $B \subset R^d$ as the ratio of the number of samples falling in the set B to the sample size n. In

[1] Strictly speaking, we should call $\hat{f}$ a generalized data density function since, in mathematical terms, it is a generalized function.

other words, $P\{B\}$ is the probability that a sample falls in the set B. From now on, we shall use P to denote the sample probability.

Let $a = (a_1, \ldots, a_d) \in R^d$, $b = (b_1, \ldots, b_d) \in R^d$, and $a \leq b$. The sample probability $P\{(a, b]\}$ that X belongs to the hyper-interval

$$(a, b] = \{x = (x_1, \ldots, x_d) \in R^d : a_1 < x_1 \leq b_1, \ldots, a_d < x_d \leq b_d\}$$

can be represented as an integral of its data density:

$$P\{(a, b]\} = \int_{a_1}^{b_1} \cdots \int_{a_d}^{b_d} \hat{f}(x_1, \ldots, x_d)\, dx_d \cdots dx_1. \tag{3.5}$$

3.6 Selectivity

The *instance selectivity* $\sigma(a, b)$ of a range query $Q(a, b)$ is given by the number of result tuples divided by the total number of tuples in the relation[2], whereas the *distribution selectivity* $\hat{\sigma}_m(a, b)$ is the probability that a tuple is in the hyper rectangle $(a, b]$, with respect to a data density approximation function $\hat{f}_m(x)$ ([1]) (to be discussed in Section 4). Note that the instance selectivity indeed represents the actual data distribution $\hat{f}(x)$ of the relation, while the distribution selectivity is an estimate of it based on $\hat{f}_m(x)$. They are equivalent if $\hat{f}_m(x) = \hat{f}(x)$.

4 Data Density Approximation Using Orthonormal Series

As mentioned earlier, the storage of a potentially complex data density function can consume considerable space. In this section, we shall develop a method to approximate the data density function such that it can be stored efficiently. It is noted that we intend to derive an approximate data density function directly from tuples of a relation without first generating the data density function itself. In Section 5, we discuss the use of the derived approximate data density function to estimate selectivities.

In the field of numerical analysis, a generalized function like $\hat{f}(x)$ is often represented or approximated by its Fourier series with respect to some orthonormal basis. Such a representation usually yields a very good approximation, requires only small amount of storage space, and can also be implemented easily.

[2]If there are multiple operand relations in the query, it is assumed to be their Cartesian product.

4.1 1-Dimensional Case

Suppose $\{V_1, \ldots, V_n\}$ are sample tuples from a one-attribute relation T. Let $L^2[l, r]$ be the family of all the functions satisfying $\int_l^r f^2(x)\,dx < \infty$. Let $\{\phi_i(x) : i \geq 0\}$ be an orthonormal basis of $L^2[l, r]$. The term "orthonormal" means

$$\int_l^r \phi_i(x)\phi_j(x)\,dx = \begin{cases} 0, & i \neq j \\ 1, & i = j. \end{cases} \tag{4.1}$$

Assume an appropriate m has been chosen (to be discussed in Section 4.3). The orthonormal series estimator of a data density function $\hat{f}(x)$ is

$$\hat{f}_m(x) = \sum_{i=0}^{m-1} \hat{\beta}_i \phi_i(x), x \in [l, r], \tag{4.2}$$

where the coefficients $\hat{\beta}_i$ are given by

$$\hat{\beta}_i = \frac{1}{n} \sum_{j=1}^{n} \phi_i(V_j). \tag{4.3}$$

For simplicity of discussion, we leave out the details of the derivation of this estimator. Interested readers are referred to ([17]) for more information. The function $\hat{f}_m(x)$ can be fully represented by these numbers $\{\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_{m-1}\}$, and be readily reconstructed by these coefficients following Equation (4.2).

After a careful evaluation of several potential orthonormal bases, we chose the cosine series, whose discretized counterpart, the discrete cosine transformation, has been widely used in image and signal processing because of its excellent ability to approximate mathematical functions. For simplification of notation and implementation of the cosine transform, we have chosen to normalize all the attribute domains to (0, 1). From now on, we shall assume all domains are so normalized without explicitly mentioning it. The cosine series, which contains infinite numbers of functions, is

$$\{1, \sqrt{2}\cos(\pi x), \sqrt{2}\cos(2\pi x), \sqrt{2}\cos(3\pi x) \ldots\}. \tag{4.4}$$

That is, we have chosen $\phi_0 = 1$, and $\phi_i(x) = \sqrt{2}\cos i\pi x$, where $i = 1, 2, \ldots$. Since

$$\int_0^1 2\cos(j\pi x)\cos(k\pi x)\,dx = \int_0^1 [cos((j+k)\pi x) + cos((j-k)\pi x)]\,dx, \tag{4.5}$$

we can see that $\int_0^1 2\cos(j\pi x)\cos(k\pi x)\,dx = 1$ if $j = k$, and $\int_0^1 2\cos(j\pi x)\cos(k\pi x)\,dx = 0$ if $j \neq k$. Thus, the cosine series is orthonormal.

Note that $\hat{\beta}_0$ is always 1. The cosine series guarantees that the distribution selectivity of $Q(0,1)$ is always 1.

To illustrate the use of this approximation, we use a 1-dimensional (one-attribute) normalized relation as an example in the following.

Example 4.1 Assume there are six tuples in the one-attribute relation and their attribute values are 0.32, 0.33, 0.12, 0.66, 0.90, and 0.80, denoted by $V_1, ..., V_6$, respectively. Suppose m has been set to 2. By applying the formula (4.3), we can calculate the coefficients of the data density estimator as follows:

$$\hat{\beta}_1 = \frac{1}{6}\sum_{j=1}^{6}\phi_1(V_j) = \frac{1}{6}\sum_{j=1}^{6}[\sqrt{2}\cos\pi V_j] = -0.063,$$

$$\hat{\beta}_2 = \frac{1}{6}\sum_{j=1}^{6}\phi_2(V_j) = \frac{1}{6}\sum_{j=1}^{6}[\sqrt{2}\cos 2\pi V_j] = 0.0951.$$

Here, only three real numbers $\{1, -0.063, 0.0951\}$ need to be stored. From Equation (4.2), the estimator of the data density function can be written as

$$\hat{f}_3(x) = 1 - 0.063\cos\pi x + 0.0951\cos 2\pi x.$$

4.2 Multi-Dimensional Case

Suppose $\{V_1, \ldots, V_n\}$ are sample tuples from a relation T with d attributes. Each V_k can be written as a vector $(V_{k1}, \ldots, V_{kd})$. Let $i^d = (i_1, \ldots, i_d)$ be a composite index denoting the subscript of a variable in a d-dimensional situation, e.g., $\hat{\beta}_{i^d} = \hat{\beta}_{(i_1,\ldots,i_d)}$, and $\{\phi_j(x) : j \geq 0\}$ be an orthonormal basis of $L^2[l, r]$. Suppose $x = (x_1, \ldots, x_d) \in [l, r]^d$. Then

$$\{\phi_{i^d}(x) = \phi_{i^d}(x_1, \ldots, x_d) = \prod_{j=1}^{d}\phi_{i_j}(x_j) : i \geq 0\} \tag{4.6}$$

is an orthonormal basis of $L^2([l, r]^d; R)$. To see this, let $j^d = (j_1, \ldots, j_d)$ and $k^d = (k_1, \ldots, k_d)$ be two composite indexes, it follows

$$\int_l^r \cdots \int_l^r [\prod_{i=1}^{d}\phi_{j_i}(x_i)][\prod_{i=1}^{d}[\phi_{k_i}(x_i)]\, dx_1 \cdots dx_d = \prod_{i=1}^{d}[\int_l^r \phi_{j_i}(x_i)\phi_{k_i}(x_i)\, dx_i]. \tag{4.7}$$

Equation (4.7) has value 1 if $j_i = k_i$, for all $i = 1, ..., d$, and 0 otherwise.

The coefficients of the data density function estimator are given by

$$\hat{\beta}_{i^d} = \frac{1}{n}\sum_{k=1}^{n}\phi_{i^d}(V_k) \tag{4.8}$$

where $\hat{\beta}_{i^d} = \hat{\beta}_{(i_1,\ldots,i_d)}$. The estimator function $\hat{f}_m(x)$ of the data density function $\hat{f}(x)$ is given by

$$\hat{f}_m(x) = \sum_{i_1=0}^{m-1} \cdots \sum_{i_d=0}^{m-1} \hat{\beta}_{(i_1,\ldots,i_d)} \prod_{j=1}^{d} \phi_{i_j}(x_j), \tag{4.9}$$

where $x = (x_1, \ldots, x_d) \in [l, r]^d$. Note that only the m^d coefficients $\{\hat{\beta}_{(0,\ldots,0)}, \ldots, \hat{\beta}_{(m-1,\ldots,m-1)}\}$ need to be stored in the d-dimensional case.

4.3 Significant Coefficients

In general, the larger the m value, the higher the accuracy of the estimator. Also, the greater the kurtosis of the distribution, the larger the m value is needed to achieve a desired accuracy. So far, we have assumed that an estimator has m^d coefficients. Here, we present a scheme that uses only a portion of these m^d coefficients to estimate selectivities without much accuracy loss.

Cosine transform is known to have excellent energy compaction properties, where most of the signal information tends to be concentrated in a few low-frequency components of the transform. Here, low-frequency components refer to those $\cos i\pi x$ functions that have small i values. Therefore, it is possible to store only the low frequency coefficients and yet not lose significant accuracy.

To filter out high frequency coefficients, the triangular method ([15]) can be applied. In this method, a coefficient $\hat{\beta}_{(i_1,\ldots,i_d)}$ is retained if $i_1 + \cdots + i_d \leq m - 1$. It is noted that the number of coefficients whose indexes satisfy $i_1 + \cdots + i_d \leq m - 1$ is $K = \begin{pmatrix} m+d-1 \\ d \end{pmatrix}$, which is less than m^d. Moreover, given an m, the indexes of such subset of coefficient $(i_1, \ldots, i_d)$ are uniquely determined. To see this, we can define a 1-1 mapping ψ_d from $\{(i_1, \ldots, i_d) : i_1 + \cdots + i_d \leq m - 1\}$ to $\{j : 0 \leq j \leq \begin{pmatrix} m+d-1 \\ d \end{pmatrix}\}$ by:

$$\psi_d(i_1, \ldots, i_d) = \begin{cases} i_1, & d = 1, \quad 0 \leq i_1 \leq m - 1 \\ \phi_{d-1}(i_1, \ldots, i_{d-1}) + \begin{pmatrix} \sum_{j=1}^{d} i_j + d - 1 \\ d \end{pmatrix}, & i_1 + \cdots + i_d \leq m - 1. \end{cases} \tag{4.10}$$

Therefore, the composite indexes need not be stored with the coefficients, unlike the wavelet method. Thus, our method consumes only half the space of the wavelet method for each coefficient, as does the DCT method with the triangular sampling scheme.

4.4 Data Density Approximation Algorithm

Let S be the subset of composite indexes (subscripts) of coefficients derived from the triangular method. The following algorithm shows the derivation of a data density estimator with such set of coefficients.

Algorithm DDEO($S, V[n]$): Data Density Estimator Using Orthonormal Series
Input: (1) S: a pre-determined subset of the m^d coefficients' composite indexes $\{i^d = (i_1, \ldots, i_d), 0 \leq i_j \leq m-1\}$, where m is a pre-determined number; (2) $V_1, \ldots, V_n$: n tuples from relation T with d attributes, where $V_i = (V_{i1}, \ldots, V_{id})$.
Output: $\hat{\beta}_{(i_1,\ldots,i_d)}$ of the data density estimator $\hat{f}_m(x)$, where $(i_1, \ldots, i_d) \in S$.
Method:

```
1  $\hat{\beta}_{(i_1,\ldots,i_d)} \leftarrow 0$ for all $i^d = (i_1, \ldots, i_d) \in S$;
2  for $k \leftarrow 1$ to $n$ do
3      $\hat{\beta}_{(i_1,\ldots,i_d)} \leftarrow \hat{\beta}_{(i_1,\ldots,i_d)} + \prod_{j=1}^{d} \phi_{i_j}(V_{kj})$      if $i^d = (i_1, \ldots, i_d) \in S$;
4  for all $i^d = (i_1, \ldots, i_d) \in S$ do
5      $\hat{\beta}_{(i_1,\ldots,i_d)} \leftarrow \hat{\beta}_{(i_1,\ldots,i_d)}/n$;
```

5 Selectivity Estimation

In this section, we discuss how to estimate selectivities using the estimator $\hat{f}_m(X)$ derived in the last section.

5.1 Selectivity for 1-Dimensional Case

Let X be a random variable such that $X \in [l, r] \subset R$, and $\hat{f}(x)$ be the data density function of X. Let $Q(a, b)$, $l \leq a < X \leq b \leq r$, be a range query. Then, the instance selectivity $\sigma(a, b)$ of the range query $Q(a, b)$ is, by definition (Sections 3.3 and 3.4), $\int_a^b \hat{f}(x)\, dx$, while the distribution selectivity (with respect to $\hat{f}_m(X)$) is

$$\hat{\sigma}(a, b) = \int_a^b \hat{f}_m(x)\, dx = \sum_{i=0}^{m-1} \hat{\beta}_i(\Phi_i(b) - \Phi_i(a)), \tag{5.1}$$

which is an estimate of $\sigma(a, b)$. The integral

$$\Phi_i(x) = \int_l^x \phi_i(u)\, du \tag{5.2}$$

is usually easy to evaluate with a suitable choice of a basis $\{\phi_i(x)\}$, such as the cosine series that we have chosen in the previous section. Note that the right hand side of (5.1) may be negative; thus, we let $\hat{\sigma}(a, b) = 0$ in that case.

5.2 Selectivity for Multi-Dimensional Case

Assume $a = (a_1, \ldots, a_d) \in [l, r]^d$, $b = (b_1, \ldots, b_d) \in [l, r]^d$, and $a_i \le b_i$ for all $i = 1, \ldots, d$. Then, following Equations (3.5) and (4.9), an estimate of the probability of a tuple falling in the hyper interval $(a, b]$ is

$$P_m\{(a, b]\} = \int_{a_1}^{b_1} \cdots \int_{a_d}^{b_d} \hat{f}_m(x_1, \ldots, x_d)\, dx_d \cdots dx_1 = \sum_{i_1=0}^{m-1} \cdots \sum_{i_d=0}^{m-1} \hat{\beta}_{(i_1,\ldots,i_d)} \prod_{j=1}^{d} (\Phi_{i_j}(b_j) - \Phi_{i_j}(a_j)).$$

Suppose the underlying probability density function f of the relation T is smooth, e.g., a normal density function. Then, as shown in ([8]), with an appropriate m (usually much smaller than n), as $n \to \infty$, the convergence rate of the mean square error is given by

$$\int_{[l,r]^d} |P_m\{(l, x]\} - P\{(l, x]\}|^2\, dx = \int_{[l,r]^d} |\int_l^x \hat{f}_m(u)\, du - \int_l^x \hat{f}(u)\, du|^2\, dx = o(n^{-1}).$$

That is, when n and m are large enough, the difference between the distribution selectivity $P_m\{(l, x]\}$ and the instance selectivity $P\{(l, x]\}$ is approximately $o(n^{-1/2})$, In other words, the convergence rate of the error is a constant 0.5, which is independent of dimension d. It is a desirable property of this estimator.

Range queries

Suppose a range query has a qualification that is a conjunction of range predicates $a_i < X_i \le b_i$ for $1 \le i \le d$, then its distribution selectivity is given by

$$\hat{\sigma}(a, b) = \begin{cases} P_m\{(a, b]\} & \text{if non-negative,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

General range queries

Suppose a general range query has a qualification that is a conjunction of range predicates $a_i \oplus_i X_i \otimes_i b_i$, $1 \le i \le d$, where $\oplus_i$ and $\otimes_i \in \{<, \le\}$. Assume $\oplus_i$ is $\le$ for $i \in \{i_1, ..., i_k\} \subseteq \{1, 2, ..., d\}$, and $\otimes_i$ is $<$ for $i \in \{j_1, ..., j_m\} \subseteq \{1, 2, ..., d\}$.

Now we can approximate this type of general range query as follows. Choose a vector $\xi = (\xi_1, \ldots, \xi_d)$ with very small values $\xi_i > 0$ if $i \in \{i_1, \ldots, i_k\}$; $\xi_i = 0$, otherwise. Choose another vector $\gamma = (\gamma_1, \ldots, \gamma_d)$ with very small values $\gamma_i > 0$ if $i \in \{j_1, \ldots, j_m\}$; $\gamma_i = 0$, otherwise. Let

$a'_i = a_i - \xi_i$ for all i and $a' = (a'_1, \ldots, a'_d)$. Let $b'_i = b_i + \gamma_i$ for all i and $b' = (b'_1, \ldots, b'_d)$. Then, the distribution selectivity of the range query[3] $Q(a', b')$ is given by

$$\hat{\sigma}(a', b') = \begin{cases} P_m\{(a', b']\} & \text{if non-negative,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

Since all ξ_i's and γ_i's are zeros or very small positive values, $\hat{\sigma}(a', b')$ is a good estimate for the distribution selectivity of the original general range query using a and b.

5.3 Selectivity Estimation Algorithm

As discussed above, we can convert a general range query into a range query. Therefore, in the following we present only the algorithm for computing the distribution selectivity of a range query. Let Φ_{i^d} be the partial integrals of the orthonormal basis ϕ_{i^d} (c.f. Equation (5.2)).

Algorithm Selectivity($S, a, b, \hat{\beta}_{()}$): Computing the Selectivity of a Range Query
Input: (1) S : a subset of the m^d coefficients' composite indexes $\{i^d : i^d = (i_1, \ldots, i_d), 0 \le i_j \le m-1\}$, where m is a pre-determined number; (2) $a = (a_1, \ldots, a_d)$, $b = (b_1, \ldots, b_d)$, $l \le a_i \le b_i \le r$: range query boundaries; (3) $\{\hat{\beta}_{i^d} : i^d = (i_1, \ldots, i_d) \in S\}$: the coefficients of the estimator $\hat{f}_m(x)$.
Output: *selectivity estimation* $\hat{\sigma}(a, b)$.
Method:

```
1 sel ← 0;
2 for all $i^d = (i_1, \ldots, i_d) \in S$ do
3     sel ← sel + $\hat{\beta}_{(i_1,\ldots,i_d)} \prod_{j=1}^{d} (\Phi_{i_j}(b_j) - \Phi_{i_j}(a_j))$;
4 if sel $\ge 0$ return sel;
5 else return 0;
```

When a dimension j is not involved in the query, it implies that the whole range of X_j, i.e., [0,1], satisfies the query. Since $\int_0^1 \phi_j(x)\, dx = 0$, any term with index $i_j, j \neq 0$, vanishes. Therefore, instead of using the entire set of coefficients in S in the above algorithm, we can use only a subset of S, denoted as $\bar{P}_j(S) = \{i^d : i^d = (i_1, \ldots, i_{j-1}, 0, i_{j+1}, \ldots, i_d), i^d \in S\}$, to compute the selectivity.

[3]Note that it is possible that $a_i - \xi_i < l$ and/or $b_i + \gamma_i > r$. In such cases, since $\hat{f}(x)$ is defined on R^d, we extend the definition of the basis function $\phi_{i_j}(x)$ as follows: $\phi_{i_j}(x) = \phi_{i_j}(l)$ if $x < l$ and $\phi_{i_j}(x) = \phi_{i_j}(r)$ if $x > r$.

6 Dynamic Maintenance

It is desirable that an estimator can remain accurate after the underlying relation has experienced substantial changes. It turns out that our estimator can be updated easily and dynamically.

6.1 Dynamic Update of Coefficients

From Equation (4.8), it is observed that a coefficient of our estimator is just the average of the basis functions. Therefore, the coefficients can be easily updated as follows:

(i) *Insertion*

If a tuple $x = (x_1, \ldots, x_d)$ is inserted into the relation T, then the coefficients of the estimator can be updated following Equation (4.8) as

$$\hat{\beta}_{i^d} \leftarrow \frac{n}{n+1}\hat{\beta}_{i^d} + \frac{\phi_{i^d}(x)}{n+1}. \tag{6.1}$$

(ii) *Deletion*

If a tuple $x = (x_1, \ldots, x_d)$ is deleted from the relation T, then the coefficients can be updated following Equation (4.8) as

$$\hat{\beta}_{i^d} \leftarrow \frac{n}{n-1}\hat{\beta}_{i^d} - \frac{\phi_{i^d}(x)}{n-1}. \tag{6.2}$$

(iii) *Update*

An update operation can be treated as a deletion followed by an insertion.

Note that the resulting coefficients, updated incrementally as above, are the same as if they were derived by applying the DDEO algorithm directly to the updated relation.

The entire process of update does not involve using any auxiliary data structures such as histograms. Moreover, since the coefficients are updated incrementally, the estimator remains accurate even if the updates do not follow the same distribution as the underlying data.

6.2 Comparisons

Let us now compare our maintenance scheme with others. The DCT constructs a temporary histogram to store the updates and then later on updates the coefficients in batch. While it basically performs the same computations as ours, updates are not done in a timely manner.

As for the wavelet, besides the estimator, it maintains an auxiliary set of coefficients that were not significant when the estimator was built ([24]). After the relation has experienced substantial changes, previously significant coefficients may now become insignificant, and vice versa. The wavelet algorithm replaces insignificant coefficients of the estimator with candidates from the auxiliary histogram. Although this scheme works for insertions, it does not work well for deletions (and hence updates) because the auxiliary histogram may not contain all the previously insignificant coefficients that now become significant. Besides the storage overhead, the maintenance scheme is also more complex than ours.

The kernel-spline needs only to update the knots affected by the updates. However, this is valid only if the updates follow the same distribution as the relation. Otherwise, the estimator has to be reconstructed to be accurate.

In summary, our dynamic update scheme is simple and easy to implement. More importantly, with the dynamic update scheme, our estimator remains accurate even if the underlying data distribution has experienced substantial changes.

7 Complexity Analysis

In this section, we compare the density estimator with three well-known histogram-based compression or approximation methods - the (Haar) wavelet, DCT, and kernel-spline, based on the estimator construction, estimation, and update costs.

Each method stores a set of statistics for selectivity estimation. The basic unit of statistics here is a coefficient for the density method, a coefficient for the DCT, a coefficient and its index for the wavelet, and 4^d real numbers (for each knot) for the kernel-spline. For simplicity, we shall call all these basic units of statistics a coefficient. Readers are advised to note that each unit of statistics, now a coefficient, may require a different amount of space for a different method, that is, 1 real number for the density and DCT estimators, 2 reals for the wavelet, and 4^d reals for the kernel-spline. Let m be the number of coefficients stored for a method, and H the size of a histogram (i.e., the number of buckets or knots in the histogram). Usually, $H >> m$ for the density, DCT, and wavelet methods, but $H = m$ for the kernel-spline method. Let n be the number of tuples in the relation. We summarize the complexities in Table 1 and explain them in details in the following subsections.

7.1 Estimator Construction Complexity

The only space needed to construct a density estimator is that for storing its m coefficients. As for the wavelet, DCT, and kernel-spline methods, a histogram needs to be constructed first, from which

Table 1: Space/Time Complexity Comparison

Method	Construction Space	Construction Time	Estimation Time	Update Time
Wavelet	$O(H+m)$	$O(H)$	$O(HlogH)$	$O(logH)$
DCT	$O(H+m)$	$O(Hm)$	$O(m)$	$O(m)$
Kernel	$O(Hm)$	$O(H)$	$O(m)$	$O(1)$
Density	$O(m)$	$O(mn)$	$O(m)$	$O(m)$

coefficients are derived. Consequently, they all require $O(H+m)$ space. The size of histograms in the experiments ranges from 256 to 16 million buckets for different settings, while m ranges from tens to thousands. Clearly, our density estimator uses much less space than the others to construct.

It takes $O(nm)$ time to construct a density estimator as m coefficients need to be updated for each input tuple. As for the other methods, a histogram needs to be built first and hence a $O(n)$ time is required in the first place. Then, it takes additional $O(H)$ time for the wavelet and kernel-spline methods to derive coefficients from the histograms, and $O(Hm)$ time for the DCT.

7.2 Estimation Complexity

The DCT and density estimators perform the same computations to derive estimates and thus have the same complexity of $O(m)$. The kernel-spline derives estimates in a similar way and thus also has a $O(m)$ complexity. The wavelet has a $O(log(H))$ time complexity for a point query. But, for a range query, it needs to reconstruct the portion of a histogram that is relevant to the query and thus has a complexity of $O(Hlog(H))$. Consequently, the wavelet estimator can be very slow if the relevant portion of the histogram is large.

7.3 Update Complexity

It takes $O(m)$ time for a density estimator to update its coefficients, following Equation (6.1) or (6.2). For a DCT estimator, updates are first stored in a temporary histogram and then later added to the original one by performing the same computations as the density estimator. Consequently, it also has the complexity $O(m)$. As for the wavelet, it takes $O(logH)$ time to update its coefficients. The kernel-spline needs only to update the coefficients of affected knots. Therefore, if the data distribution does not change, an update can be accomplished in $O(1)$ time.

8 Experimental Results

In this section, we report the experimental results of the proposed density estimator and the three well-known methods. All algorithms are implemented in the C++ programming language and tests are run on a PC with 933 MHz Pentium III CPU and 512 MB memory.

We performed experiments on synthetic data as well as real data. We compare the methods based on the four metrics: 1) estimator construction speed, 2) estimation speed, 3) update speed, and 4) estimation accuracy. We briefly summarize the results as follows.

- Estimator construction speed. The density estimator generally runs the fastest, especially in the multi-dimensional cases.
- Estimation speed. The density and DCT estimators are the fastest, especially in multi-dimensional cases. The wavelet estimator is the slowest because it needs to reconstruct the portion of a histogram that is relevant to the range query.
- Update speed. The kernel-spline is the fastest while the wavelet is the slowest, especially in the multi-dimensional cases.
- Estimation accuracy. The density estimator performs the best, slightly better than the DCT. Recall that the density estimator always represents the best limit of the DCT, and yet without worrying about the construction and partition of the underlying histogram. The kernel-spline is the worst.

In the following, we discuss the results in details.

8.1 Results on Synthetic Data Sets

The purpose of these experiments is to study the behaviors of these methods under different circumstances. We run 1,000 range queries with selectivity ranging from 0 to 1.

8.1.1 Synthetic Data Sets

We generate attribute values using the TPC-D benchmark data ([31]), which is also used in ([23]). The synthetic relation LINEITEM has attributes OORDERDATE, LSHIPDATE, LRECEIPT-DATE, ... etc, whose values are generated as follows:

$$LSHIPDATE = OORDERDATE + random(4000),$$
$$LRECEIPTDATE = LSHIPDATE + random(4000),$$

where OORDERDATE is uniformly distributed between 0 and $2,000$, and $random(num)$ returns a random number between 0 and num. The attribute values generated are confined to pre-chosen ranges for each respective attribute. Other attributes are generated in the same way.

The size of the relations is set to 10^6 $(= n)$ tuples. To generate n d-dimensional tuples, we first randomly generate B base tuples, numbered from 1 to B. Then, we replicate the base tuples based on the Zipf distribution ([32]), which is commonly used to model the frequencies of events in various fields. Given a tuple i, $1 \leq i \leq B$, the frequency of it appearing in the relation, denoted as $f_z(i)$, is

$$f_z(i) = \frac{1}{i^z} / \sum_{j=1}^{B} \frac{1}{j^z}, \qquad z \geq 0, \tag{8.1}$$

where z is the Zipf distribution parameter. In other words, Zipfianlly distributed frequencies are assigned to randomly chosen cells in a joint frequency distribution matrix. When $z = 0$, it has a uniform distribution. The larger the z value, the more kurtosis the distribution has.

In general, the higher the dimension, the greater the number of coefficients (or the larger the space) is needed to derive reasonable estimates. For example, as shown later in Section 8.1.5, 30 coefficients (a real number each) are sufficient for a density estimator to yield good estimates in the one-dimensional experiments, but it would take 400 and 3,000 coefficients to yield reasonable ones in the two- and three-dimensional experiments, respectively. In the following, we shall use these sample space budgets, i.e., 30, 400, and 3,000 real numbers for the one-, two-, and three-dimensional cases, respectively, to compare the estimator construction, estimation, and update speeds.

8.1.2 Estimator Construction Speed

As shown in Table 2, there is not much difference in the one-dimensional case for all these estimators. But as the dimension increases, the wavelet and DCT become much slower, comparing their 7,025 and 8,043 seconds, respectively, to our 19.4 seconds for the three-dimensional case. The slowness of the wavelet and DCT is due to the exponential growth of the histogram size, 256^d buckets, recalling that the wavelet and DCT have $O(H)$ and $O(Hm)$ complexities, respectively. Although the kernel-spline estimator also has a $O(H)$ complexity, its histogram size (or the number of knots) only increases moderately, from 30/4 knots in the one-dimensional case, $400/4^2$ the two-dimensional case, to $3{,}000/4^3$ knots the three-dimensional case, recalling that 4^d reals are used to store a knot

Table 2: Estimator Construction Speed

second	1-dimension	2-dimension	3-dimension
Wavelet	0.45	5.57	7,025
DCT	0.45	1.75	8,043
Kernel	0.61	1.55	-
Density	0.41	4.80	19.4

Table 3: Estimation Speed

millisecond	1-dimension	2-dimension	3-dimension
Wavelet	0.05	6.59	11,434
DCT	0.04	0.463	3.447
Kernel	1.21	28.273	-
Density	0.04	0.463	3.477

(bucket). Consequently, the kernel estimator is faster to construct for a given space budget. On the other hand, since it uses so much space for a knot, it can only model fewer knots (buckets) for the same space budget, resulting in poor estimation (to be discussed in Section 8.1.5). Therefore, we shall not discuss the kernel-spline estimator beyond two dimensions due to its poor accuracy in multi-dimensional cases.

8.1.3 Estimation Speed

As shown in Table 3, the wavelet is comparable to the density and DCT estimators in one-dimensional small-histogram case. But, as the dimension increases, the wavelet becomes much slower, recalling that the wavelet has a $O(HlogH)$ complexity, while the others have a $O(m)$. As shown in the three-dimensional case, while the wavelet took 11,434 milliseconds, the density estimator took only 3.477 milliseconds. The DCT and density estimators use the same amount of time because they perform the same computations. The kernel-spline estimator is also very slow because it involves complex computations.

8.1.4 Update Speed

As shown in Table 4, the DCT and data density estimators have the same speed and are the fastest to update their coefficients. They basically perform the same computations except that the former

Table 4: Estimator Update Speed

millisecond	1-dimension	2-dimension	3-dimension
Wavelet	0.20	0.72	7.21
DCT	0.014	0.246	1.966
Kernel	0.004	0.025	-
Density	0.014	0.246	1.966

performs updates in batch, while the latter on the fly.

The wavelet ($O(logH)$) is slower than the data density and DCT estimators. As the number of dimensions increases, the size of the underlying histogram increases quickly, so does the cost of its update. The cost involves the updates of both the largest m coefficients and the auxiliary coefficients. The kernel estimator runs the fastest as it needs only to update the coefficients of the affected knot.

8.1.5 Estimation Accuracy

We use the average relative error as the criterion for accuracy, as did Matias et al. ([23]). The relative error is defined as $|\hat{\sigma} - \sigma|/\sigma$, where σ is the instance (or true) selectivity and $\hat{\sigma}$ is the distribution (or estimated) selectivity. We measure the errors based on the same amount of space used for storing the statistics. Since all values are stored as 4-byte real numbers, we shall alternatively use the numbers of reals as the space measure. The coefficients of the DCT and data density estimators are chosen by the triangular sampling scheme, while the coefficients of the wavelet method are chosen by the largest-absolute-values criterion.

Figures 1 and 2 show the results of 1-dimensional experiments with Zipf parameters 0.5 and 1.0, respectively. It is observed that the kernel-spline method performs much worse than the other methods. One reason is that the kernel-spline is generally not suitable for non-smooth distributions, which the experimental data exhibit. Another reason is that it uses a considerable amount of space, 4^d reals, for each knot. It is also observed that the wavelet does not perform as well as the DCT and our density method. For example, the wavelet has an error of 11.22% with 40 real numbers (Figure 1) while the DCT and the density estimator have only 1.23% and 1.22% errors, respectively. One reason for the wavelet's underperformance is that it requires twice as much space as the DCT and the data density estimator to store a coefficient. The DCT and the data density estimator have almost identical results, indicating that a histogram with 256 buckets is already fine enough

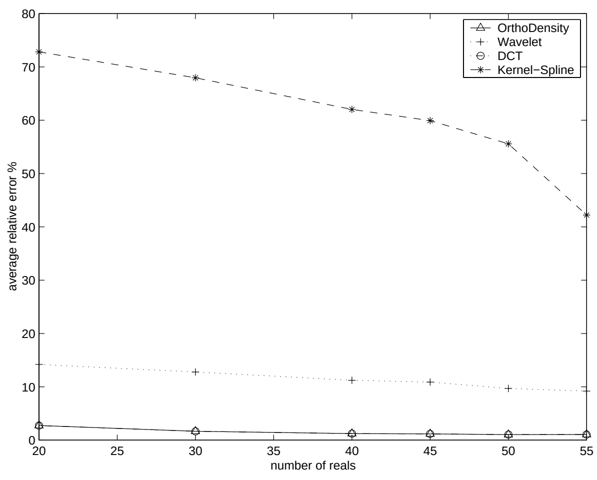

Figure 1: Performance on 1-dim Zipf distribution with $z = 0.5$

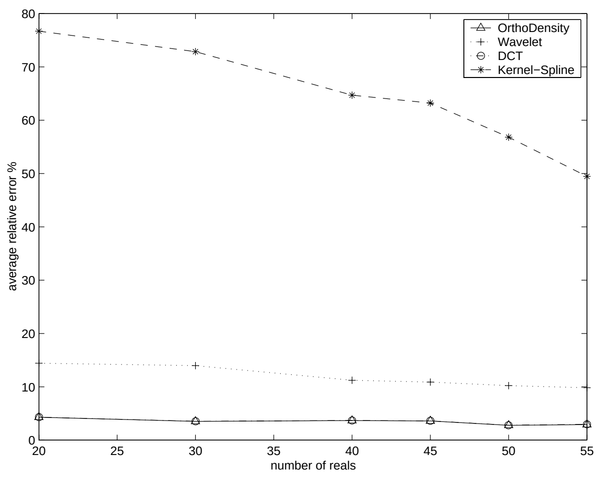

Figure 2: Performance on 1-dim Zipf distribution with $z = 1.0$

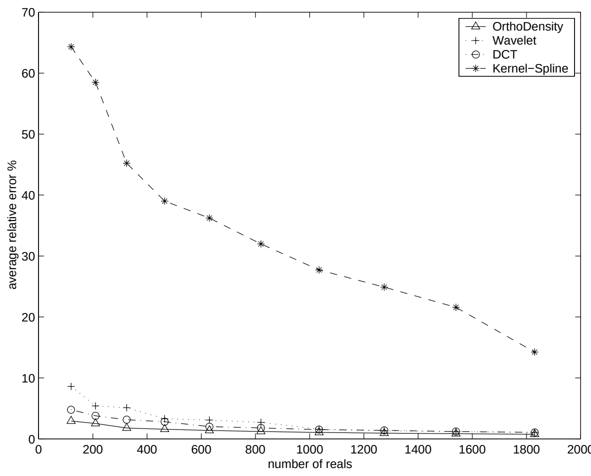

Figure 3: Performance on 2-dim Zipf distribution with $z = 0.5$

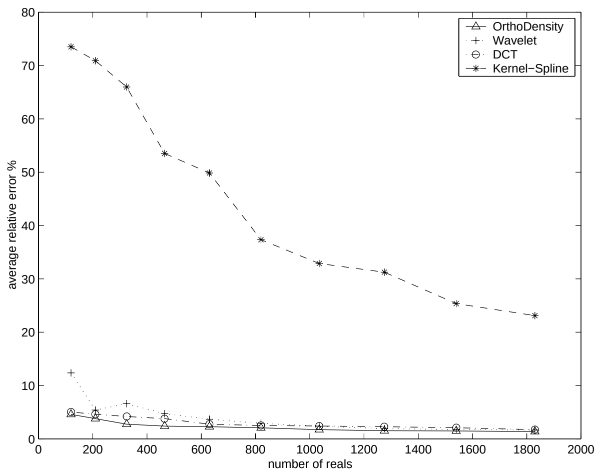

Figure 4: Performance on 2-dim Zipf distribution with $z = 1.0$

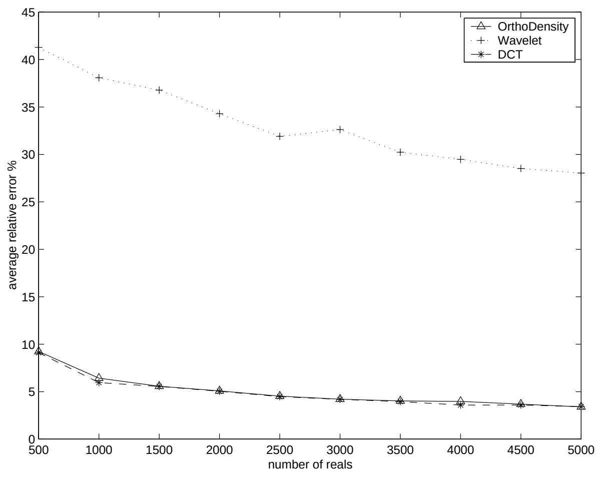

Figure 5: Performance on 3-dim Zipf distribution with $z = 0.5$

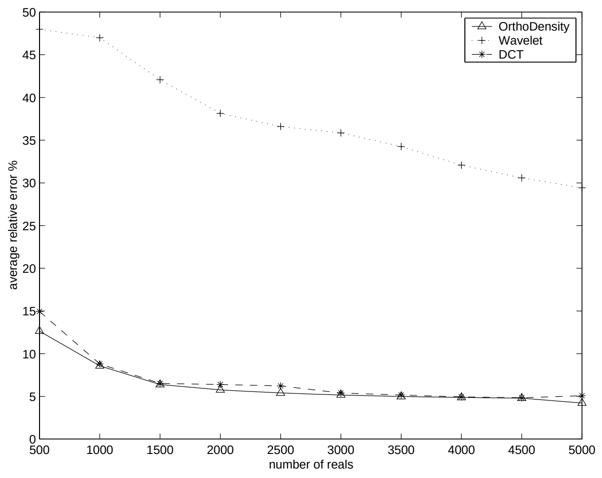

Figure 6: Performance on 3-dim Zipf distribution with $z = 1.0$

to capture all the essential information of the experimental data. Finer partitions (i.e., > 256 partitions) are not necessary and will not yield any notable improvement. Recall that our density estimator always represents the best limit of the DCT without concerning about the histogram sizes.

As expected, the larger the number of coefficients, the higher the accuracy. However, the rate of improvement decreases. This is due to the fact that newly added coefficients are less significant than the previously included ones. Note that energy is generally conserved in the first few terms of the DCT and the cosine transform, while in the largest-absolute-value terms of the wavelet.

Comparing the two figures, we observed that the higher the z value, the larger the relative error. In general, the more complex the data distribution (e.g., more distinct values, greater kurtosis), the more the coefficients are needed to achieve the same accuracy.

Figures 3 and 4 show the results of 2-dimensional data with Zipf parameters 0.5 and 1.0, respectively. Here, a histogram is divided into 256 partitions along each dimension, that is, a histogram composed of 256^2 buckets for both the DCT and the wavelet. As for the kernel-spline method, since each knot requires 16 ($=4^2$) reals, given a space budget S, one can store only $S/16$ knots. Again, the kernel-spline performs the worst. The situation can get even worse as the number of dimensions increases. Therefore, we shall leave out the kernel-spline method in the subsequent discussions due to its poor performance. It is also observed that the DCT and the data density estimator are indistinguishable and perform a little better than the wavelet method.

In Figures 5 and 6, we show the results of 3-dimensional experiments with histograms parti-

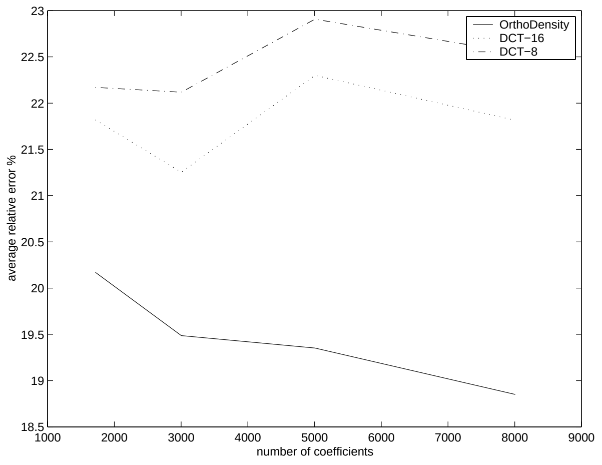

Figure 7: Performance on 6-dim Zipf distribution with $z = 0.5$

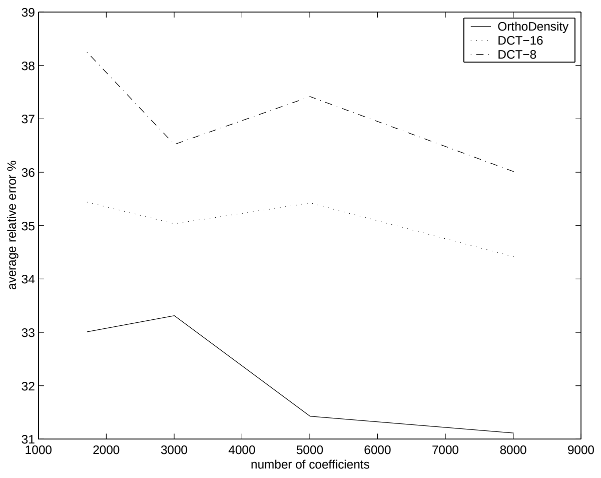

Figure 8: Performance on 6-dim Zipf distribution with $z = 1.0$

tioned into 16 million ($= (256)^3$) buckets. As the dimension increases, more coefficients are needed to achieve a desired accuracy. Again, the DCT and ours have indistinguishable performance but clearly outperform the wavelet. For example, with with z=0.5 and 5,000 real numbers used, the errors are 28.46%, 3.43%, and 3.40% for the wavelet, DCT, and density methods, respectively. Indeed, as the dimension increases, e.g., from 1 to 3 dimensions, the superiority of the DCT and our method (over the wavelet) becomes more evident. Therefore, we shall not discuss the wavelet any further in higher dimensional spaces.

In Figures 7 and 8, we consider the 6-dimensional case. Notice the changes of the scale on the error or Y axis. Due to the explosive growth of the bucket numbers in the histograms and the high computation cost of deriving the estimator of the DCT method, we are forced to choose coarser partitions that have only 16 (= 16 million buckets) and 8 partitions in each of the six dimensions, denoted as DCT-16 and DCT-8, respectively, in the figures. As shown, our method performed better than DCT. As observed, the finer the histograms, the closer the performance of the DCT to ours. However, using finer histograms in high dimensional cases could incur tremendous overheads in constructing the estimator. Note that our method is simple because it does not need to worry about the granularity of histograms and always gives the best results that the DCT could possibly achieve. Although we do not show the results of the wavelet, it is worth mentioning that the wavelet has errors greater than 50% in all the 6-dimensional experiments.

In summary, our density estimator yields the best accuracy. Note that our approach always represents the best limit of the DCT without worrying about the underlying histogram sizes. The kernel-spline method performed the worst in all the cases tested.

8.2 Results on Real Data Sets

In this section, we present the experimental results on the real data sets, *thyroid* and *cloud* ([14]). The thyroid data set contains the thyroid diagnoses obtained from the Garvan Institute, consisting of 9172 records from year 1984 to early 1987. We use the largest relation *thyroid*0387.*data* for our experiments, whose size is 755K bytes. Two continuous attributes, *age* and *TT4*, are used in the experiments with ranges [1, 100] and [1, 600], respectively. As shown in Figure 9, with only 5 real numbers, the density and DCT estimators have already achieved 2.24% and 2.56% errors, respectively. On the other hand, the wavelet and kernel-spline methods still have errors as high as 48.37% and 67.22%, respectively. With the number of reals being increased to 30, the errors of the wavelet and kernel estimators decrease to 12.97% and 25.84%, respectively, which are still much worse than our 0.91%. Similar results are observed in the 2-dimensional experiments, as shown in Figure 10. With 50 real numbers, the errors of the density, DCT, wavelet, and kernel-spline estimators are 5.86%, 6.21%, 23.08% and 72.43%, respectively.

Another data set, *cloud*, is obtained from CDIAC (Carbon Dioxide Information Analysis Center) for cloud analysis. We chose the file containing the ocean weather information for September, 1990 for the experiments. The data file contains 103,765 records. Two attributes, latitude *LAT* and longitude *LON*, are chosen for experiments. Their ranges are [-9,000, 9,000] and [0, 36,000], respectively. As shown in Figures 11 and 12, the density estimator outperforms the other methods in both experiments. In the one-dimensional experiment, our method yields a 6.0% relative error with only 20 real numbers, while the DCT, wavelet, and kernel-spline methods yield 6.3%, 41.1%, and 33.4% errors, respectively. With 100 real numbers, the relative errors of the DCT, wavelet and kernel-spline methods are 2.1%, 26.1%, and 22.1%, compared to our 1.4%. With the increase of the dimensions, more coefficients are required to achieve the desired accuracy. With 200 real numbers, the relative errors of the density, DCT, wavelet, and kernel-spline estimators are 34.1%, 34.3%, 58.2%, and 69.9%, respectively. When the number of real numbers used increases to 1,000, the relative errors of the density, DCT, wavelet, and kernel-spline estimators decrease to 11.3%, 11.5%, 32.4%, and 29.8%, respectively.

In summary, the experimental results on the real data sets are consistent with those on the synthetic data sets, that is, the density estimator has the best accuracy.

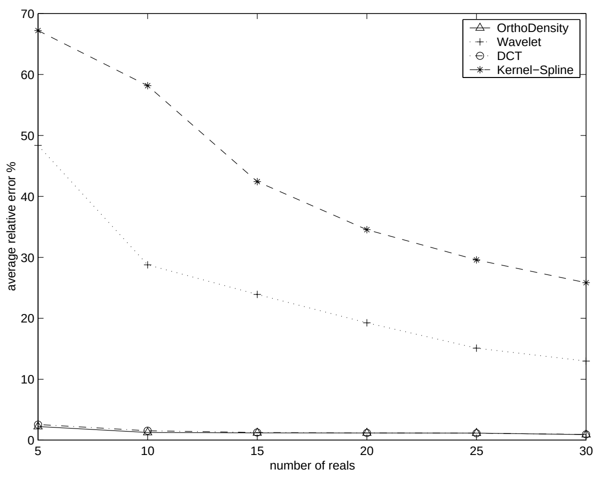

Figure 9: Performance on 1-dim Thyroid Data Set

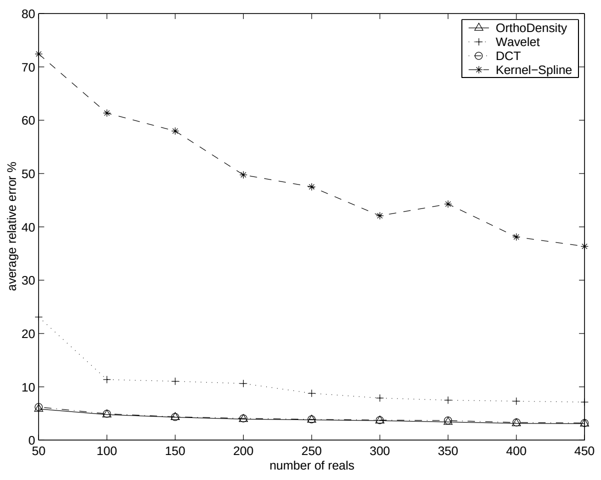

Figure 10: Performance on 2-dim Thyroid Data Set

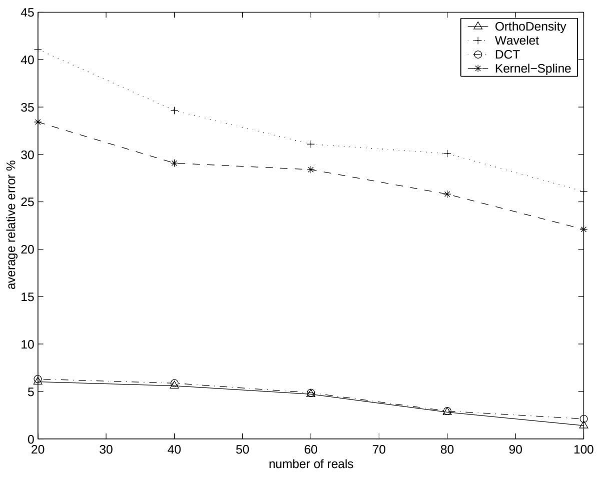

Figure 11: Performance on 1-dim Clouds Data Set

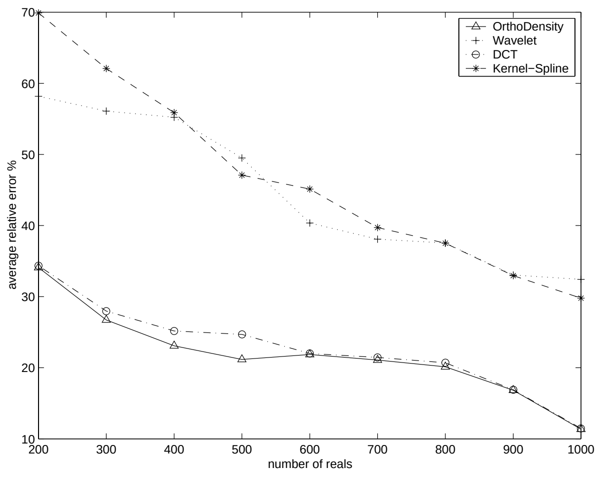

Figure 12: Performance on 2-dim Clouds Data Set

9 Conclusions

In this paper, we have developed a statistical selectivity estimation method based on the data density approximation by orthonormal series ([28]). The advantages of our method, compared with the wavelet, DCT, and kernel-spline methods, are summarized as follows:

- Our estimator takes the least amount of time to construct, especially in the multi-dimensional cases.
- Our estimator, as well as the DCT estimator, are the fastest to derive estimates.
- Our estimator can be easily and quickly updated. It can reflect distribution changes in a timely manner.
- Our method is the most accurate one. Although it is only slightly better than the DCT, it always represents the best limit of the DCT without concerning the size of underlying histograms.

In summary, our density estimator is very promising in meeting the requirements of selectivity estimation for query optimization in today's database systems.

References

[1] B. Blohsfeld, D. Korus and B. Seeger, A Comparison of Selectivity Estimators for Range Queries on Metric Attributes, in: Proc. ACM SIGMOD Conference (1999), 239 - 250.

[2] S. Christodoulakis, Estimating Record Selectivities, Inf. Syst. Vol. 8, No. 2, (1983), 105 - 115.

[3] S. Christodoulakis, Estimating Block Transfers and Join Sizes, in: Proc. ACM SIGMOD Conference (1983), 40 - 54.

[4] C. Chen and N. Roussopoulos, Adaptive Selectivity Estimation Using Query Feedback, in: Proc. ACM SIGMOD Conference (1994), 161 - 172.

[5] C. Cui, An Introduction to Wavelets (Academic Press, 1992).

[6] M. Garofalakis and P. Gibbons, Probabilistic Wavelet Synopses, ACM Transactions on Databases Systems, vol. 29, No. 1, March 2004, 43-90.

[7] D. Gunopulos, G. Kollios, V. Tsotras, and C. Domeniconi, Approximating Multi-dimensional Aggregate Range Queries Over Real Attributes, in: Proc. ACM SIGMOD Conference (2000), 463 - 474.

[8] P. Hall, Orthogonal Series Distribution Function Estimation, with Applications, Journal of the Royal Statistical Society, Series B, Vol. 45, No. 1 (1983), 81 - 88.

[9] W. Hou, G. Ozsoyoglu and E. Dogdu, Error-Constrained Count Query Evaluation in Relational Database, in: Proc. ACM SIGMOD Conference (1991), 279 - 287.

[10] P. Haas and A. Swami, Sequential Sampling Procedures for Query Size Estimation, in: Proc. ACM SIGMOD Conference (1992), 341 - 350.

[11] Y.E. Ioannidis and S. Christodoulakis, On the Propagation of Errors in the Size of Join Results, in: Proc. ACM SIGMOD Conference (1991), 268 - 277.

[12] K. Kan, D. Cheung, B. Kao, Maintenance of Partial-sum-based Histograms, in: Proc. DASFAA Conference (2003), 149 - 156.

[13] R. Kooi, The Optimization of Queries in Relational Database Systems, Ph. D. Thesis, Case Western University, 1980.

[14] F. Korn, T. Johnson and H. V. Jagadish, Range Selectivity Estimation for Continuous Attributes, in: Proc. SSDM Conference (1999), 244 - 253.

[15] J. Lee, D. Kim and C. Chung, Multi-dimensional Selectivity Estimation Using Compressed Histogram Information, in: Proc. ACM SIGMOD Conference (1999), 205 - 214.

[16] Y. Ling and W. Sun, A Supplement to Sampling Based Methods for Query Size Estimation in a Database System, in: Proce. ACM SIGMOD Conference (1992), 12 - 15.

[17] E. Lefons, A. Silvestri and F. Tangorra, An Analytical Approach to Statistical Databases, in: Proc. 9th VLDB Conference (1983), 260 - 274.

[18] E. Lefons, A. Merico, F. Tangorra, "Analytical Profiles Estimation in Database Systems", in: Information Systems, Vol. 20, No. 1 (1995), 1 -20.

[19] R. Lipton, J. Naughton and D. Schneider, Practical Selectivity Estimation through Adaptive Sampling, in: ACM SIGMOD Conference (1990), 1 - 11.

[20] C. Lynch, Selectivity Estimation and Query Optimization in Large Databases with Highly Skewed Distributions of Column Values, in: 14th VLDB Conference (1988), 240 - 251.

[21] M.V. Mannino, P. Chu and T. Sager, Statistical Profile Estimation in Database Systems, ACM Computing Surveys, Vol. 20, No. 3 (1988), 191 - 221.

[22] M. Muralikrishna and D.J. DeWitt, Equi-depth Histograms for Estimating Selectivity Factors for Multi-dimensional Queries, in: ACM SIGMOD Conference (1988), 28 - 36.

[23] Y. Matias, J. Vitter, and M. Wang, Wavelet-Based Histograms for Selectivity Estimation, in: ACM SIGMOD Conference (1998), 448 - 459.

[24] Y. Matias, J. Vitter, and M. Wang, Dynamic Maintenance of Wavelet-based Histograms, in: Proc. 26th VLDB Conference (2000), 101 - 110.

[25] V. Poosala and Y. Ioannidis, Selectivity Estimation Without the Attribute Value Independence Assumption, in: Proc. 23rd VLDB Conference (1997), 486–495.

[26] V. Poosala, Histogram-Based Estimation Techniques in Database Systems, Ph.D. Dissertation, University of Wisconsin-Madison, 1997.

[27] V. Poolsala, P. Hass, Y. Ioannidis, and E. Shekita, Improved Histogram for Selectivity Estimation of Range Predicates, in: Proc. ACM SIGMOD Conference (1996), 294 - 305.

[28] B.L.S. Prakasa Rao, Nonparametric Functional Estimation, Academic Press (1983).

[29] P. Selinger, M. Astrahan, D. Chamberlin, R. Lorie and T. Price, Access Path Selection in a Relational Database Management System, in: Proc. ACM SIGMOD Conference (1979), 23 - 34.

[30] W. Sun, Y. Ling, N. Rishe and Y. Deng, An Instant and Accurate Size Estimation Method for Joins and Selection in a Retrieval-Intensive Environment, in: Proc. ACM SIGMOD Conf. (1993), 79 - 86.

[31] TPC benchmark D (decision support), Standard Specification Revision 1.1, Transaction Processing Performance Council, 1995.

[32] G.K. Zipf, Human Behavior and the Principle of Least Effort (Addison-Wesley, Reading, MA, 1949).

[33] Q. Zhu and P.-Å. Larson, Solving Local Cost Estimation Problem for Global Query Optimization in Multidatabase Systems, Distributed and Parallel Databases, Vol. 6, No. 4 (1998), 373-420.