The History of Histograms (abridged)

Yannis Ioannidis

Department of Informatics and Telecommunications, University of Athens
Panepistimioupolis, Informatics Buildings
157-84, Athens, Hellas (Greece)
yannis@di.uoa.gr

Abstract

The history of histograms is long and rich, full of detailed information in every step. It includes the course of histograms in different scientific fields, the successes and failures of histograms in approximating and compressing information, their adoption by industry, and solutions that have been given on a great variety of histogram-related problems. In this paper and in the same spirit of the histogram techniques themselves, we compress their entire history (including their "future history" as currently anticipated) in the given/fixed space budget, mostly recording details for the periods, events, and results with the highest (personally-biased) interest. In a limited set of experiments, the semantic distance between the compressed and the full form of the history was found relatively small!

1 Prehistory

The word '*histogram*' is of Greek origin, as it is a composite of the words 'isto-s' ($\iota\sigma\tau os$) (= 'mast', also means 'web' but this is not relevant to this discussion) and 'gram-ma' ($\gamma\rho\alpha\mu\mu\alpha$) (= 'something written'). Hence, it should be interpreted as a form of writing consisting of 'masts', i.e., long shapes vertically standing, or something similar. It is not, however, a word that was originally used in the Greek language[1]. The term 'histogram' was coined by the famous statistician Karl Pearson[2] to refer to a "common form of graphical representation". In the Oxford English Dictionary quotes from "Philosophical Transactions of the Royal Society of London" Series A, Vol. CLXXXVI, (1895) p. 399, it is mentioned that "[The word 'histogram' was] introduced by the writer in his lectures on statistics as a term for a common form of graphical representation, i.e., by columns marking as areas the frequency corresponding to the range of their base.". Stigler identifies the lectures as the 1892 lectures on the geometry of statistics [69].

The above quote suggests that histograms were used long before they received their name, but their birth date is unclear. Bar charts (i.e., histograms with an individual 'base' element associated with each column) most likely predate histograms and this helps us put a lower bound on the timing of their first appearance. The oldest known bar chart appeared in a book by the Scottish political economist William Playfair[3] titled "The Commercial and Political Atlas (London 1786)" and shows the imports and exports of Scotland to and from seventeen countries in 1781 [74]. Although Playfair was skeptical of the usefulness of his invention, it was adopted by many in the following years, including for example, Florence Nightingale, who used them in 1859 to compare mortality in the peacetime army to that of civilians and through those convinced the government to improve army hygiene.

From all the above, it is clear that histograms were first conceived as a visual aid to statistical approximations. Even today this point is still emphasized in the common conception of histograms: Webster's defines a

Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003

[1] To the contrary, the word '*history*' is indeed part of the Greek language ('istoria' - $\iota\sigma\tau o\rho\iota\alpha$) and in use since the ancient times. Despite its similarity to 'histogram', however, it appears to have a different etymology, one that is related to the original meaning of the word, which was 'knowledge'.

[2] His claim to fame includes, among others, the chi-square test for statistical significance and the term 'standard deviation'.

[3] In addition to the bar chart, Playfair is probably the father of the pie chart and other extremely intuitive and useful visualizations that we use today.

histogram as "a bar graph of a frequency distribution in which the widths of the bars are proportional to the classes into which the variable has been divided and the heights of the bars are proportional to the class frequencies". Histograms, however, are extremely useful even when disassociated from their canonical visual representation and treated as purely mathematical objects capturing data distribution approximations. This is precisely how we approach them in this paper.

In the past few decades, histograms have been used in several fields of informatics. Besides databases, histograms have played a very important role primarily in image processing and computer vision. Given an image (or a video) and a visual pixel parameter, a histogram captures for each possible value of the parameter (Webster's "classes") the number of pixels that have this value (Webster's "frequencies"). Such a histogram is a summary that is characteristic of the image and can be very useful in several tasks: identifying similar images, compressing the image, and others. Color histograms are the most common in the literature, e.g., in the QBIC system [21], but several other parameters have been proposed as well, e.g., edge density, texturedness, intensity gradient, etc. [61]. In general, histograms used in image processing and computer vision are accurate. For example, a color histogram contains a separate and precise count of pixels for each possible distinct color in the image. The only element of approximation might be in the number of bits used to represent different colors: fewer bits imply that several actual colors are represented by one, which will be associated with the number of pixels that have any of the colors that are grouped together. Even this kind of approximation is not common, however.

In databases, histograms are used as a mechanism for full-fledged compression and approximation of data distributions. They first appeared in the literature and in systems in the 1980's and have been studied extensively since then at a continuously increasing rate. In this paper, we concentrate on the database notion of histograms, discuss the most important developments on the topic so far, and outline several problems that we believe are interesting and whose solution may further expand their applicability and usefulness.

2 Histogram Definitions

2.1 Data Distributions

Consider a relation R with n numeric attributes X_i ($i = 1..n$). The *value set* $\mathcal{V}_i$ of attribute X_i is the set of values of X_i that are present in R. Let $\mathcal{V}_i = \{ v_i(k)\colon 1 \leq k \leq D_i \}$, where $v_i(k) < v_i(j)$ when $k < j$. The *spread* $s_i(k)$ of $v_i(k)$ is defined as $s_i(k) = v_i(k+1) - v_i(k)$, for $1 \leq k < D_i$. (We take $s_i(D_i) = 1$.) The *frequency* $f_i(k)$ of $v_i(k)$ is the number of tuples in R with $X_i = v_i(k)$. The *area* $a_i(k)$ of $v_i(k)$ is defined as $a_i(k) = f_i(k) \times s_i(k)$. The *data distribution* of X_i is the set of pairs $\mathcal{T}_i = \{ (v_i(1), f_i(1)), (v_i(2), f_i(2)), \ldots, (v_i(D_i), f_i(D_i)) \}$.

The *joint frequency* $f(k_1, .., k_n)$ of the value combination $< v_1(k_1), .., v_n(k_n) >$ is the number of tuples in R that contain $v_i(k_i)$ in attribute X_i, for all i. The *joint data distribution* $\mathcal{T}_{1,..,n}$ of $X_1, .., X_n$ is the entire set of *(value combination, joint frequency)* pairs.

In the sequel, for 1-dimensional cases, we use the above symbols without the subscript i.

2.2 Motivation for Histograms

Data distributions are very useful in database systems but are usually too large to be stored accurately, so histograms come into play as an approximation mechanism. The two most important applications of histogram techniques in databases have been selectivity estimation and approximate query answering within query optimization (for the former) or pre-execution user-level query feedback (for both). Our discussion below focuses exactly on these two, especially range-query selectivity estimation as this is the most popular issue in the literature. It should not be forgotten, however, that histograms have proved to be useful in the context of several other database problems as well, e.g., load-balancing in parallel join query execution [65], partition-based temporal join execution [68] and others.

2.3 Histograms

A *histogram* on an attribute X is constructed by partitioning the data distribution of X into β (≥ 1) mutually disjoint subsets called *buckets* and approximating the frequencies and values in each bucket in some common fashion. This definition leaves several degrees of freedom in designing specific histogram classes as there are several possible choices for each of the following (mostly orthogonal) aspects of histograms [67]:

Partition Rule: This is further analyzed into the following characteristics:

- **Partition Class:** This indicates if there are any restrictions on the buckets. Of great importance is the *serial* class, which requires that buckets are non-overlapping with respect to some parameter (the next characteristic), and its subclass *end-biased*, which requires at most one non-singleton bucket.
- **Sort Parameter:** This is a parameter whose value for each element in the data distribution is derived from the corresponding attribute value and frequencies. All serial histograms require that the sort parameter values in each bucket form a contiguous range. Attribute value (V), frequency (F), and area (A) are examples of sort parameters that have been discussed in the literature.
- **Source Parameter:** This captures the property of the data distribution that is the most critical

in an estimation problem and is used in conjunction with the next characteristic in identifying a unique partitioning. Spread (S), frequency (F), and area (A) are the most commonly used source parameters.

- **Partition Constraint:** This is a mathematical constraint on the source parameter that uniquely identifies a single histogram within its partition class. Several partition constraints have been proposed so far, e.g., *equi-sum, v-optimal, maxdiff,* and *compressed*, which are defined further below as they are introduced. Many of the more successful ones try to avoid grouping vastly different source parameter values into a bucket.

Following [67], we use *p(s,u)* to denote a serial histogram class with partition constraint p, sort parameter s, and source parameter u.

Construction Algorithm: Given a particular partition rule, this is the algorithm that constructs histograms that satisfy the rule. It is often the case that, for the same histogram class, there are several construction algorithms with different efficiency.

Value Approximation: This captures how attribute values are approximated within a bucket, which is independent of the partition rule of a histogram. The most common alternatives are the *continuous value assumption* and the *uniform spread assumption*; both assume values uniformly placed in the range covered by the bucket, with the former ignoring the number of these values and the later recording that number inside the bucket.

Frequency Approximation: This captures how frequencies are approximated within a bucket. The dominant approach is making the *uniform distribution assumption*, where the frequencies of all elements in the bucket are assumed to be the same and equal to the average of the actual frequencies.

Error Guarantees: These are upper bounds on the errors of the estimates a histogram generates, which are provided based on information that the histogram maintains.

A multi-dimensional histogram on a set of attributes is constructed by partitioning the joint data distribution of the attributes. They have the exact same characteristics as 1-dimensional histograms, except that the partition rule needs to be more intricate and cannot always be clearly analyzed into the four other characteristics as before, e.g., there is no real sort parameter in this case, as there can be no ordering in multiple dimensions [66].

3 The Past of Histograms

First Appearance

To the best of our knowledge, the first proposal to use histograms to approximate data distributions within a database system was in Kooi's PhD thesis [47]. His proposal was an immediate loan from statistics of the simplest form of histogram, with the value set being divided into ranges of equal length, i.e., the so called *equi-width* histograms. Hence, in terms of the taxonomy of Section 2.3, the entry point for histograms into the world of databases was the serial class of *equi-sum(V,S)*, where the equi-sum partition constraint requires that the sums of the source-parameter values (spreads in this case) in each bucket are equal. Within each bucket, values and frequencies were approximated based on the *continuous value assumption* and the *uniform distribution assumption*, respectively.

Equi-width histograms represented a dramatic improvement over the uniform distribution assumption for the entire value set (i.e., essentially a single-bucket histogram), which was the state of the practice at the time. Hence, they were quickly adopted by the Ingres DBMS in its commercial version, and later on by other DBMSs as well.

First Alternative

A few years after Kooi's thesis, the first alternative histogram was proposed, changing only the source parameter [62]. Instead of having buckets of equal-size ranges, the new proposal called for buckets with (roughly) the same number of tuples in each one, i.e., the so called *equi-depth* or *equi-height* histograms. In terms of the taxonomy, these are the *equi-sum(V,F)* histograms. There was ample evidence that equi-depth histograms were considerably more effective than equi-width histograms, hence, many commercial vendors switched to those in the years following their introduction. Equi-depth histograms were later presented in their multi-dimensional form as well [58].

Optimal Sort Parameter

After several years of inactivity on the topic of histograms, interest in it was renewed in the context of studying how initial errors in statistics maintained by the database propagate in estimates of the size of complex query results [36]. In particular, it was shown that, under some rather general conditions, in the worst case, errors propagate exponentially in the query size (i.e., in the number of joins), removing any hope for high-quality estimates for large multi-join queries.

The first results that led towards new types of histograms were derived in an effort to obtain statistics that would be optimal in minimizing/containing the propagation of errors in the size of join results [37]. The basic mathematical tools used were borrowed from majorization theory [55]. The focus was on a rather restricted class of equality join queries, i.e., single-join queries or multi-join queries with only one attribute participating in joins per relation (more generally, with a 1-1 functional dependency between each pair of join attributes of each relation). For this query class, and

under the assumption that the value set is known *accurately*, it was formally proved that the optimal histogram was serial and had frequency as the sort parameter[4].

Ten years ago

The above result might have not had the impact it did if it had remained true only for the restricted query class it was first proved for. Soon afterwards, however, in VLDB'93, it was generalized for arbitrary equality join queries, giving a strong indication that the most effective histograms may be very different from those that were used until that point [34].

To the best of our knowledge, histograms with frequency as the sort parameter represented the first departure from value-based grouping of buckets, not only within the area of databases, but overall within mathematics and statistics as well. Furthermore, their introduction essentially generalized some common practices that were already in use in commercial systems (e.g., in DB2), where the highest frequency values were maintained individually and accurately due to their significant contribution to selectivity estimates. Such a practice is an instance of a special case of a histogram in the *end-biased* partition class, with frequency as the sort parameter: the highest sort-parameter values are maintained in singleton buckets. Although less accurate than general serial histograms, in several cases, end-biased histograms proved quite effective.

New Partition Constraints

The results on the optimality of frequency as the sort parameter left open two important questions. First, which partition constraints are the most effective, i.e., which ones among all possible frequency-based bucketizations? Second, which histograms are optimal when the value set is not accurately maintained but is approximated in some fashion?

The answer to the first question came in the form of the *v-optimal* histograms, which partition the data distribution so that (roughly) the variance of source-parameter values within each bucket is minimized [38].

Unfortunately, the second question had no analytical answer, but extensive experimentation led to the formation of the space of histogram characteristics that we use as the basic framework for our discussion in this paper (Section 2.3) [67]. In addition to the *equi-sum* and *v-optimal* partition constraints, it introduced several possible new ones as well, which similarly to v-optimal had as a goal to avoid grouping together in the same bucket vastly different source-parameter values. Among them, we distinguish *maxdiff*, which places bucket boundaries between adjacent source-parameter values (in sort-parameter order) whose difference is among the largest, and *compressed*, which puts the highest source values in singleton buckets and partitions the rest in equi-sum fashion. Overall, the new partition constraints (i.e., v-optimal, maxdiff, compressed) were shown to be the most effective in curbing query-result-size estimation errors.

The same effort pointed towards several possibilities for the sort and source parameters, i.e., value, spread, frequency, area, cumulative frequency, etc., with frequency and area being the best source parameters. Interestingly, the best sort parameter proved to be the value and not the frequency, as the original optimality results would suggest, indicating that, if values are not known accurately, having buckets with overlapping value ranges does not pay off for range queries.

The most effective of these histograms have actually been adopted by industrial products (see Section 4). Furthermore, in addition to selectivity estimation for various relational and non-relational queries, these histograms have proved to be very effective in approximate query answering as well [39].

Since the specification of the above space of histograms, there have been several efforts that have studied one or more of its characteristics and have proposed alternative, improved approaches. For each characteristic, we outline some of the most notable pieces of work on it in a separate subsection below. Unless explicitly mentioning the opposite, the discussion is about 1-dimensional histograms.

Alternative Partition Constraints

In addition to the partition constraints that were introduced as part of the original histogram framework [67], a few more have been proposed that attempt to approach the effectiveness of v-optimal, usually having a more efficient construction cost. Among them, we note one that uses a simplified form of the *optimal knot placement problem* [18] to identify the bucket boundaries, which are where the 'knots' are placed [46]. The simplification consists of using only linear splines that are also allowed to be discontinuous across bucket boundaries. This is combined with interesting alternatives on the value and frequency approximation within each bucket.

Multi-Dimensional Partition Rules

The first introduction of multi-dimensional histograms was by Muralikrishna and DeWitt [58], who essentially described 2-dimensional equi-depth histograms. Space was divided in the same way it is done in a Grid-file, i.e., recursively cutting the entire space into half-spaces by using a value of one of the dimensions as a boundary each time, the dimension and the value being chosen in a way prespecified at the beginning of the process [58]. Buckets were non-overlapping (the multi-dimensional version of the *serial* partition class) on

[4]These were called simply *serial histograms* at the time, but the term was later generalized to imply non-overlapping ranges of any sort parameter, not just frequency, which is how we use the term in this paper as well.

the space of the multi-dimensional values (the multi-dimensional version of value as the sort parameter), the boundaries chosen with equi-sum as the partition constraint and frequency as the source parameter.

It was not until several years later that any new partition rules were proposed [66], this time taking advantage of the generality of the histogram taxonomy [67]. The most effective family of such rules was *MHIST-2*, which starts from the entire joint data distribution placed in a single bucket and, at each step, splits the space captured by one of the buckets it has formed into two subspaces, until it has exhausted its budget of buckets. The split is made in the bucket and along the dimension that is characterized as most "critical", i.e., whose marginal distribution is the *most in need of partitioning*, based on the (1-dimensional) partition constraint and source parameter used. In combination with the most effective partition constraints and source parameters (i.e., v-optimal or maxdiff with frequency or area), MHIST-2 represented a dramatic improvement over the original multi-dimensional equi-depth histograms.

Since MHIST, there have been several other interesting partition rules that have been proposed. One of them is *GENHIST* [31], which was originally proposed in the context of multi-dimensional real-valued data, but its applicability is broader. The main characteristic of GENHIST is that it allows buckets to overlap in the space of multi-dimensional values: the algorithm starts from a uniform grid partitioning of the space and then iteratively enlarges the buckets that contain high numbers of data elements. This has two effects: first, the density of data in each bucket decreases, thus making the overall density smoother; second, the buckets end up overlapping, thus creating many more distinct areas than there are buckets per se. The data distribution approximation within each area is a combination of what all the overlapping bucket that form the area indicate. This results in a small number of buckets producing approximations with low errors.

Another alternative is the *STHoles Histogram* [11], which takes, in some sense, a dual approach to GENHIST: instead of the region covered by a bucket increasing in size and overlapping with other buckets, in STHoles, this region may decrease in size due to the removal of a piece of it (i.e., opening a hole) that forms a separate, child bucket. This creates buckets that are not solid rectangles, and is therefore capable of capturing quite irregular data distributions.

Identifying effective multi-dimensional partition rules is by no means a closed problem, with different approaches being proposed continuously [23].

Value Approximation Within Each Bucket

Given a specific amount of space for a histogram, one of the main tradeoffs is the number of buckets versus the amount of information kept in each bucket. A small amount of information within each bucket implies gross local approximations but also more buckets. Finding the right balance in this tradeoff to optimize the overall approximation of the data distribution is a key question.

With respect to approximating the set of values that fall in a 1-dimensional bucket, there have been essentially two approaches. Under the traditional *continuous value assumption*, one maintains the least amount of information (just the min and max value), but nothing that would give some indication of how many values there are or where they might be. Under the more recent *uniform spread assumption* [67], one also maintains the number of values within each bucket and approximates the actual value set by the set that is formed by (virtually) placing the same number of values at equal distances between the min and max value. A different version of that has also been proposed that does not record the actual average spread within a bucket but one that reduces the overall approximation error in range queries by taking into account the popularity of particular ranges within each bucket [46]. There have been several studies that show each general technique superior to the other, an indication that there may be no universal winner.

The two main approaches mentioned above have been extended for multi-dimensional buckets as well, maintaining the min and max value of each dimension in the bucket. Under the continuous value assumption nothing more is required, but under the uniform spread assumption, the problem arises of which distinct (multi-dimensional) values are assumed to exist in the bucket. If d_i is the number of distinct values in attribute X_i that are present in a bucket and $v'_i(k)$ is the k'th approximate value in dimension i (obtained by applying the uniform spread assumption along that dimension), then a reasonable approach is to assume that all possible combinations $< v'_1(k_1), .., v'_n(k_n) >$, $1 \leq k_i \leq d_i$, exist in the bucket [66].

There has also been an interesting effort that introduces the use of *kernel estimation* into the 1-dimensional histogram world [10] to deal specifically with real-valued data. Roughly, it suggests choosing the points of considerable change in the probability density function as the bucket boundaries (in a spirit similar to the maxdiff partition constraint) and then applying the traditional kernel estimation method for approximating the values within each bucket. This has also been generalized for the multi-dimensional case [31].

Frequency Approximation Within Each Bucket

With respect to approximating the set of frequencies that fall in a bucket, almost all efforts deal with the traditional *uniform distribution assumption*. Among the few exceptions is one that is combined with the linear spline partition constraint mentioned above and

uses a *linear spline-based* approximation for frequencies as well [46]. It records one additional data item per bucket to capture linearly growing or shrinking frequencies at the expense of fewer buckets for a fixed space budget. Likewise, another exception uses equally small additional space within each bucket to store cumulative frequencies in a 4-level tree index [13]. Contrary to the previous effort, however, it is combined with some of the established partition constraints, i.e., v-optimal and maxdiff.

Efficient and Dynamic Constructions

Although estimation effectiveness is probably the most important property of histograms (or any other compression/estimation method for that matter), construction cost is also a concern. With respect to this aspect, histograms may be divided into two categories: *static* histograms and *dynamic/adaptive* histograms.

Static histograms are those that are traditionally used in database systems: after they are constructed (from the stored data or a sample of it), they remain unchanged even if the original data gets updated. Depending on the details of the updates, a static histogram eventually drifts away from what it is supposed to approximate, and the estimations it produces may suffer from increasingly larger errors. When this happens, the administrators ask for a recalculation, at which point the old histogram is discarded and a new one is calculated afresh. An important consideration for static histograms is the cost of each calculation itself, which is mostly affected by the partition constraint. Most such constraints (e.g., equi-sum, maxdiff, compressed) have straightforward calculations that are efficient. This is not the case, however, for what has been shown to be the most effective constraint, i.e., v-optimal, whose straightforward calculation is in general exponential in the number of source-parameter values. A key contribution in this direction has been the proposal of a dynamic-programming based algorithm that identifies the v-optimal histogram (for any sort and source parameter) in time that is quadratic in the number of source-parameter values and linear in the number of buckets, thus making these histograms practical as well [42]. Subsequently, several (mostly theoretical) efforts have introduced algorithms that have reduced the required running time for calculating these optimal histograms, eventually bringing it down to linear overall and achieving similar improvements for the required space as well [30]. Dynamic-programming algorithms have also been proposed for constructing the optimal histograms for (hierarchical) range queries in OLAP data [44]. For the multidimensional case, optimal histogram identification is NP-hard, so several approximate techniques have been proposed [59].

Another interesting development has been the proposal of algorithms to identify optimal *sets* of histograms (as opposed to individual histograms), based on an expected workload [40]. This effort focuses on v-optimal histograms, but is equally applicable to other partition constraints as well.

Even with the existence of efficient calculation algorithms, however, static histograms suffer from increasing errors between calculations. Moreover, in a data stream environment, static histograms are not an option at all, as there is no opportunity to store the incoming data or examine it more than once. Hence, several works have proposed various approaches to dynamic/adaptive/self-tuning histograms, which change as the data gets updated, while remaining competitive to their static counterparts. Among these, we note one for equi-depth and compressed histograms [26], one for v-optimal histograms [27], and one for (linear) spline-based histograms [46]. There is also an effort focusing on data streams, where a *sketch* on the (joint) data distribution of the stream is maintained, from which an effective multidimensional histogram may be constructed [72]; the STHoles histogram is used for experimentation with the method, but in principle, it could be applied to other histogram classes as well.

Another approach to dynamic construction that has been examined in the past consists of query feedback mechanisms that take into account actual sizes of query results to dynamically modify histograms so that their estimates are closer to reality. In essence, this is histogram adaptation at query time instead of at update time. The main representatives in this category are the *ST-histograms* [2] and their descendant *STHoles histograms* [11], which employ a sophisticated partition rule as well. These techniques are independent of the particular characteristics of the initial histograms, which may be constructed in any way, e.g., they could be equi-depth histograms. In addition to their dynamic nature, a key advantage of these approaches is their low cost.

The LEO system [70] generalizes these efforts as it uses result sizes of much more complicated queries to modify its statistics, including join and aggregate queries, queries with user-defined functions, and others. Interestingly, LEO does not update the statistics in place, but puts all feedback information into separate catalogs, which are used in combination with the original histograms at estimation time.

Error Guarantees

Most work on histograms deals with identifying those that exhibit low errors in some estimation problem, but not with providing, together with the estimates, some information on what those errors might be. The first work to address the issue [42] suggests storing in each bucket the maximum difference between the actual and the approximate (typically, the average) frequency of a value in the bucket and using that to

provide upper bounds on the error of any selectivity estimates produced by the histogram for equality and range selection queries. An interesting alternative focuses on optimizing top-N range queries and stores additional information on a per-histogram rather than on a per-bucket basis [20].

Other Data Types

As mentioned earlier, most work on histograms has focused on approximating numeric values, in one or multiple dimensions (attributes). Nevertheless, the need to approximation is much broader, and several efforts have examined the use of histograms for other data types as well.

With respect to spatial data, the canonical approaches to 2-dimensional histograms do not quite work out as these are for point data and do not extend to objects that are 2-dimensional themselves. Furthermore, frequency is usually not an issue in spatial data, as spatial objects are not repeated in a database. Several interesting techniques have been presented to address the additional challenges, which essentially are related to the partition rule, i.e., how the spatial objects are grouped into buckets. Some form buckets by generalizing conventional histogram partition constraints while others do it by following approaches used in spatial indices (e.g., R-trees). The *MinSkew Histogram* [5] is among the more sophisticated ones and divides the space by using binary partitionings (recursively dividing the space along one of the dimensions each time) so that the overall *spatial skew* of all buckets is minimized. The latter captures the variance in the density of objects within each bucket, so it follows, in some sense, the spirit of the v-optimal histograms. The *SQ-Histogram* [3] is an interesting alternative, dividing the space according to the Quad-tree rule (which is more restrictive than arbitrary binary partitionings) and, in addition to spatial proximity, taking into account proximity in the size as well as the complexity (number of vertices) of the polygons that are placed in the same bucket. Both approaches are quite effective, with SQ being probably the overall winner. Spatial histograms, i.e., MinSkew histograms, have also been extended to capture the velocity of object movement, thus becoming able to approximate spatio-temporal data as well [17].

The recent interest in XML could not, of course, leave untouched XML file approximation, XML query result size estimation, and other related problems. The semi-structured nature of XML files does not lend itself to histogram-based approximation, as there is no immediate multi-dimensional space that can be bucketized but one needs to be formed from some numeric XML-file characteristics. In the StatiX approach [22], information in an XML Schema is used to identify potential sources of structural skew and then 1-dimensional histograms are built for the most problematic places in the schema, approximating the distributions of parent ids for different elements. In the XPathLearner approach [49], (first-order) Markov Histograms are used [1], where the frequencies of the results of traversing all paths of length 2 are stored in two 2-dimensional histograms. The dimensions always represent the 'from' and 'to' nodes of the paths in the XML graph; in the first histogram, both nodes/dimensions are for XML tags, whereas in the second histogram, the 'from' node/dimension is an XML tag and the 'to' node/dimension is a value. Assuming enough memory, frequencies are maintained accurately for all tag-to-tag pairs (accurate histogram), as there are very few. To the contrary, <tag,value> pairs are placed in a histogram that is based on a 2-dimensional version of the compressed partition constraint, with frequency as the source parameter. Another approach for estimating XML-query result sizes builds *position histograms* on a 2-dimensional space as well, only here the two dimensions are directly or indirectly related to the numbering of each node in a preorder traversal of the XML graph [79]. Finally, histograms have also been used in combination with or as parts of other data structures for XML approximations. The *XSketch* is quite an effective graph-based synopsis that tries to captured both the structural and the value characteristics of an XML file [63, 64]. Histograms enter the picture as they are used at various parts of an XSketch to capture statistical correlations of elements and values in particular neighborhoods of the XSketch graph.

In addition to XML graphs, histograms have also been proposed to capture the degrees of the nodes in general graphs as a way to compare graphs between them and grade their similarity [60].

Unconventional Histograms

Throughout the years, there have been a few interesting pieces of work that do not quite follow the general histogram taxonomy or histogram problem definitions. One of them suggests the use of the Discrete Cosine Transform (DCT) to compress an entire multi-dimensional histogram and store its compressed form [48]. It employs a very simple multi-dimensional partition rule (a uniform grid over the entire space), divides the space into a large number of small buckets, and then compresses the bucket information using DCT. This appears to save on space but also estimation time, as it is possible to recover the necessary information through the integral of the inverse DCT function.

There is also a promising line of work that combines histograms with other techniques to produce higher-quality estimations than either technique could do alone. In addition to several such combinations with sampling, a particularly interesting technique tries to overcome the 'curse of dimensionality' by identifying the critical areas of dependence and independence

among dimensions in multi-dimensional data, capturing them with a statistical interaction model (e.g., log-linear model) which can then form the basis for lower-dimensional MHIST histograms to approximate the overall joint data distribution [19].

Finally, there is a very interesting departure from the convention that histograms are built on base relations and estimations of the data distributions of intermediate query results are obtained by appropriate manipulations of these base-relation histograms [12]. It discusses the possibility of maintaining histograms on complex query results, which proves to be quite effective in some cases. This work uses the main SQL Server histograms (essentially maxdiff - see Section 4) to demonstrate the proposed approach, but the overall effort is orthogonal to the particular histogram class. As the number of potential complex query histograms is much larger than that of base-relation histograms, the corresponding database design problem of choosing which histograms to construct is accordingly more difficult as well. Fortunately, a workload-based algorithm proves adequate for the task.

4 Industrial Presence of Histograms

Histograms have not only been the subject of much research activity but also the favorite approximation method of all commercial DBMSs as well. Essentially all systems had equi-width histograms in the beginning and then eventually moved to equi-depth histograms. In this section, we briefly describe the currently adopted histogram class for three of the most popular DBMSs.

DB2 employs compressed histograms with value as the sort parameter and frequency as the source parameter [50]. Users may specify the numbers of singleton and non-singleton buckets desired for the most-frequent values and the equi-depth part of a compressed histogram, respectively, with the default being 10 and 20. A departure from our general descriptions above is that DB2 stores cumulative frequencies within non-singleton buckets. Histogram construction is based on a reservoir sample of the data. DB2 exploits multi-dimensional cardinality information from indices on composite attributes (whenever they are available) to obtain some approximate quantification of any dependence that may exist between the attributes, and uses this during selectivity estimation. Otherwise, it assumes attributes are independent. The learning capabilities of LEO [70] play a major role in how all available information is best exploited for high-quality estimation.

Oracle still employs equi-depth histograms [78]. Its basic approach to multi-dimensional selectivities is similar to that of DB2, based on exploiting any available information from composite indices. In addition to that, however, it offers dynamic sampling capabilities to obtain on-the-fly dependence information for rather complex predicates whenever needed (selections and single-table functions are already available, while joins will be in the next release). It also takes into account the dependencies that exist between the attributes of a cube's dimensions' hierarchies during roll-up and provides estimates at the appropriate hierarchy level. Finally, the next release will employ learning techniques to remember selectivities of past predicates and use them in the future.

SQL Server employs maxdiff histograms with value as the sort parameter and essentially average frequency (within each bucket) as the source parameter [9]. It permits up to 199 buckets, storing within each bucket the frequency of the max value and (essentially) the cumulative frequency of all values less than that. Histogram construction is typically based on a sample of the data. Composite indices are used in a similar fashion as in the other systems for obtaining multi-dimensional selectivity information.

Note that all commercial DBMSs have implemented strictly 1-dimensional histograms. Except for some incidental indirect information, they essentially still employ the *attribute value independence assumption* and have not ventured off to multi-dimensional histograms.

5 Competitors of Histograms

The main technique that has competed against histograms in the past decade is *wavelets*, which is very important for image compression and has been introduced into the database world in the late 90's [7, 56]. Wavelets have been used extensively for approximate answering of different query types and/or in different environments: multidimensional aggregate queries (range-sum queries) in OLAP environments [75, 76], aggregate and non-aggregate relational queries with computations directly on the stored wavelet coefficients [14], and selection and aggregate queries over streams [28]. As with histograms, there have also been efforts to devise wavelet-based techniques whose approximate query answers are provided with error guarantees [24], as well as to construct and maintain the most important wavelet coefficients dynamically [57].

Sampling is not a direct competitor to histograms, as it is mostly a runtime technique, and furthermore, the literature on sampling is extremely large, so it is impossible to analyze the corresponding highlights in the limited space of this paper. However, we should emphasize that sampling is often a complementary technique to histograms, as static (and even several forms of dynamic) histograms are usually constructed based on a sample of the original data [15, 26, 62].

There are also several specialized techniques that have been proposed and compete with histograms on specific estimation problems. These include techniques for selectivity estimation of select-join queries [71] or spatial queries [8], using query feedback to modify stored curve-fitting/parametric information for bet-

ter selectivity estimation [16], selectivity estimation for alphanumeric/string data in 1-dimensional [43, 45] and multi-dimensional environments [41, 77], identification of quantiles [6, 53, 54] and their dynamic maintenance with a priori guarantees [29], approximate query answering for aggregate join queries [4], select-join queries [25], and within the general framework of *on-line aggregation* [33, 32, 51], computing frequencies of high-frequency items in a stream [52], and others. Despite their suboptimality compared to some of these techniques on the corresponding problems, histograms remain the method of choice, due to their overall effectiveness and wide applicability.

6 The Future of Histograms

Despite the success of histograms, there are several problems whose current solutions leave enough space for significant improvement and several others that remain wide open, whose solution would make the applicability of histograms much wider and/or their effectiveness higher. We have recently discussed various problems of both types [35], some addressing specific histogram characteristics from the existing taxonomy while others being cast in a slightly more general context. In this section, we focus on three of the open problems, those that we believe are the most promising and sense as being the furthest away from any past or current work that we are aware of.

Histogram Techniques and Clustering

Abstracting away the details of the problem of histogram-based approximation, one would see some striking similarities with the traditional problem of clustering [73]: the joint data distribution is partitioned into buckets, where each bucket contains similar elements. Similarity is defined based on some distance function that takes into account the values of the data attributes and the value of the frequency if there is any variation on it (e.g., if it is not equal to 1 for all data elements). The buckets are essentially clusters in the traditional sense, and for each one, a very short approximation of the elements that fall in it is stored.

Despite the similarities, the techniques that have been developed for the two problems are in general very different, with no well-documented reasoning for many of these differences. Why can't the histogram techniques that have been developed for selectivity estimation be used for clustering or vice versa? From another perspective, why can't the frequency in selectivity estimation be considered as another dimension of the joint data distribution and have the problem be considered as traditional clustering? What would the impact be of using stored approximations developed for one problem to solve another? In general, given the great variety of techniques that exist for the two problems, it is crucial to obtain an understanding of the advantages and disadvantages of each one, its range of applicability, and in general, their relative characteristics when mutually compared. A comprehensive study needs to be conducted that will include several more techniques than those mentioned here. The "New Jersey Data Reduction Report" [7] has examined many techniques and has produced a preliminary comparison of their applicability to different types of data. It can serve as a good starting point for verification, extrapolation, and further exploration, not only with respect to applicability, but also precise effectiveness trade-offs, efficiency of the algorithms, and other characteristics.

Bucket Recognition and Representation

The goal of any form of (partition-based) approximation, e.g., histogram-based and traditional clustering, is to identify groups of elements so that all those within a group are similar with respect to a small number of parameters that characterize them. By storing approximations of just these parameters, one is able to reconstruct an approximation of the entire group of elements with little error. Note that, in the terms of the histogram taxonomy, these parameters should be chosen as the source parameter(s), to satisfy the proximity-expressing partition constraint.

How do we know which parameters are similar for elements so that we can group them together and represent them in terms of them? This is a typical question for traditional pattern recognition [73], where before applying any clustering techniques, there is an earlier stage where the appropriate dimensions of the elements are chosen among a great number of possibilities. There are several techniques that make such a choice with varying success depending on the case.

It is important, however, to emphasize that, in principle, these parameters may not necessarily be among the original dimensions of the data elements presented in the problem but may be derivatives of them. For example, in several histogram-based approximations as we have described them above, proximity is sought directly for frequencies but not for attribute values, as attention there is on their spreads. (Recall also the success of area as a source parameter, which is the product of frequency with spread.) The frequencies in a bucket are assumed constant and require a smaller amount of information to be stored for their approximation than the attribute values, which are assumed to follow a linear rule (equal spread). Hence, conventional histogram-based approximation, under the uniform distribution and uniform spread assumptions, implies clustering in the derived space of frequency and spread. In principle, however, not all data distributions are served best with such an approach.

To increase the accuracy of histogram approximations, there should be no fixed, predefined approximation approach to the value dimensions and the frequencies. It should not necessarily even be the same

for different buckets. Histograms should be flexible enough to use the optimal approximation for each dimension in each bucket, one that would produce the best estimations for the least amount of information. Identifying what that optimal approximation is, is a hard problem and requires further investigation.

Histograms and Tree Indices

The fact that there is a close relationship between approximate statistics kept in databases, especially histograms, and indices has been recognized in the past in several works [7]. If one considers the root of a B+ tree, the values that appear in it essentially partition the attribute on which it is built into buckets with the corresponding borders. Each bucket is then further subdivided into smaller buckets by the nodes of the subsequent level of the tree. One can imagine storing the appropriate information next to each bucket specified in a node, hence transforming the node into a histogram, and the entire index into a so called *hierarchical histogram*. This may adversely affect index search performance, of course, as it would reduce the out-degree of the node, possibly making the tree deeper. Nevertheless, although this idea works against the main functionality of an index, its benefits are non-negligible as well, so it has even been incorporated into some systems.

We believe that hierarchical histograms and, in general, the interaction between approximation structures and indices should be investigated further, as there are several interesting issues that remain unexplored as analyzed below. Consider again a B+ tree whose nodes are completely full. In that case, the root of the tree specifies a bucketization of the attribute domain that corresponds to an equi-depth histogram, i.e., each bucket contains roughly an equal number of elements under it. Similarly, any node in the tree specifies an equi-depth bucketization of the range of values it leads to.

The main issue with B+ trees being turned into hierarchical equi-depth histograms is that the latter are far from optimal overall on selectivity estimation [67]. Histograms like v-optimal and maxdiff are much more effective. What kind of indices would one get if each node represented bucketizations following one of these rules? Clearly, the trees would be unbalanced. This would make traditional search less efficient on the average. On the other hand, other forms of searches would be served more effectively. In particular, in a system that provides approximate answers to queries, the root of such a tree would provide a higher-quality answer than the root of the corresponding B+ tree. Furthermore, the system may move in a progressive fashion, traversing the tree as usual and providing a series of answers that are continuously improving in quality, eventually reaching the leaves and the final, accurate result.

Returning to precise query answering, note that typically indices are built assuming all values or ranges of values being equally important. Hence, having a balanced tree becomes crucial. There are often cases, however, where different values have different importance and different frequency in the expected workloads [46]. If this query frequency or some other such parameter is used in conjunction with advanced histogram bucketization rules, some very interesting trees would be generated whose average search performance might be much better than that of the B+ tree.

From the above, it is clear that the interaction between histograms and indices presents opportunities but also several technical challenges that need to be investigated. The trade-off between hierarchical histograms that are balanced trees with equi-depth bucketization and those that are unbalanced with more advanced bucketizations requires special attention. The possibility of some completely new structures that would strike even better trade-offs, combining the best of both worlds, cannot be ruled out either.

7 Conclusions

Histograms have been very successful within the database world. The reason is that, among several existing competing techniques, they probably represent the optimal point balancing the tradeoff between simplicity, efficiency, effectiveness, and applicability for data approximation/compression. Research-wise most of the basic problems around histograms seem to have been solved, but we believe there are still better solutions to be found for some of them. Moreover, as outlined in the previous section, there are some untouched foundational problems whose solution may require significant changes in our overall perspective on histograms. As much as the past ten years have been enjoyable and productive in deepening our collective understanding of histograms and applying them in the real world, we believe the next ten will be even more exciting and really look forward to them!

8 Personal History

Our personal history with histograms has been strongly influenced by Stavros Christodoulakis. It all started during the "Query Optimization Workshop", which was organized in conjunction with SIGMOD'89 in Portland, when Stavros argued that optimizing very large join queries did not make any sense, as the errors in the selectivity estimates would be very large after a few joins. Wanting to prove him wrong due to a personal interest in large query optimization, we started collaborating with him on the error propagation problem, work that led to results that justified Stavros' fears completely [36]. During this effort, we were initiated by Stavros into the wonderful world of majorization theory, Schur functions, and all the other

mathematical tools that much of our subsequent histogram work would be based on. Further collaboration with Stavros resulted in the identification of "serial histograms" and the first realization of their significance [37], which was the springboard for the VLDB'93 paper. For all these, we want to express our sincere gratitude to Stavros for revealing an exciting research path that was hiding many treasures along the way.

The second person who has marked significantly our involvement with histograms is Vishy Poosala. As a PhD student at Wisconsin, Vishy took the original "serial histogram" results, dived deep into them, and pushed them in many different directions, an effort that eventually led us to several interesting results that have played an important role in the success of histograms. For the long and fruitful collaboration we have had, both before and after his PhD degree, many thanks are due to Vishy as well.

Acknowledgements: For this present paper, we would like to thank Minos Garofalakis, Neoklis Polyzotis, and again Vishy Poosala for several useful suggestions and for verifying that the error in the approximation of the history of histograms presented is small.

References

[1] Aboulnaga A., Alameldeen A., Naughton J.: Estimating the Selectivity of XML Path Expressions for Internet Scale Applications. VLDB Conf. (2001) 591-600

[2] Aboulnaga A., Chaudhuri S.: Self-tuning Histograms: Building Histograms Without Looking at Data. SIGMOD Conf. (1999) 181-192

[3] Aboulnaga A., Naughton J.: Accurate Estimation of the Cost of Spatial Selections ICDE 2000 123-134

[4] Acharya S., Gibbons P., Poosala V., Ramaswamy S.: Join Synopses for Approximate Query Answering. SIGMOD Conf. (1998) 275-286

[5] Acharya S., Poosala V., Ramaswamy S.: Selectivity Estimation in Spatial Databases. SIGMOD Conf. (1999) 13-24

[6] Alsabti K., Ranka S., Singh V.: A One-Pass Algorithm for Accurately Estimating Quantiles for Disk-Resident Data. VLDB Conf. (1997) 346-355

[7] Barbará D., et al.: The New Jersey Data Reduction Report. Data Engineering Bulletin 20:4 (1997) 3-45

[8] Belussi A., Faloutsos C.: Estimating the Selectivity of Spatial Queries Using the 'Correlation' Fractal Dimension. VLDB Conf. (1995) 299-310

[9] Blakeley J., Kline N.: personal communication. (2003)

[10] Blohsfeld B., Korus D., Seeger, B.: A Comparison of Selectivity Estimators for Range Queries on Metric Attributes. SIGMOD Conf. (1999) 239-250

[11] Bruno N., Chaudhuri S., Gravano L.: STHoles: A Multidimensional Workload-Aware Histogram. SIGMOD Conf. (2001) 294-305

[12] Bruno N., Chaudhuri S., Gravano L.: Exploiting Statistics on Query Expressions for Optimization SIGMOD Conf. (2002) 263-274

[13] Buccafurri F., Rosaci D., Saccà D.: Improving Range Query Estimation on Histograms. ICDE (2002) 628-238

[14] Chakrabarti K., Garofalakis M., Rastogi R., Shim K.: Approximate Query Processing Using Wavelets. VLDB Journal 10:2-3 (2001) 199-223

[15] Chaudhuri S., Motwani R., Narasayya V.: Random Sampling for Histogram Construction: How Much is Enough? SIGMOD Conf. (1998) 436-447

[16] Chen C., Roussopoulos N.: Adaptive Selectivity Estimation Using Query Feedback. SIGMOD Conf. (1994) 161-172

[17] Choi Y.-J., Chung C.-W.: Selectivity Estimation for Spatio-Temporal Queries to Moving Objects. SIGMOD Conf. (2002) 440-451

[18] De Boor C.: A Practical Guide to Splines. Springer (1994)

[19] Deshpande A., Garofalakis M., Rastogi R.: Independence is Good: Dependency-Based Histogram Synopses for High-Dimensional Data. SIGMOD Conf. (2001) 199-210

[20] Donjerkovic D., Ramakrishnan R.: Probabilistic Optimization of Top N Queries. VLDB Conf. (1999) 411-422

[21] Flickner M., et al.: Query by Image and Video Content: The QBIC System. IEEE Computer 28:9 (1995) 23-32

[22] Freire J., Haritsa J., Ramanath M., Roy P., Simeon J.: StatiX: Making XML Count. SIGMOD Conf. (2002) 181-191

[23] Furtado P., Madeira H.: Summary Grids: Building Accurate Multidimensional Histograms. DASFAA Conf. (1999) 187-194

[24] Garofalakis M., Gibbons P.: Wavelet Synopses with Error Guarantees. SIGMOD Conf. (2002) 476-487

[25] Getoor L., Taskar B., Koller D.: Selectivity Estimation Using Probabilistic Models. SIGMOD Conf. (2001) 461-472

[26] Gibbons P., Matias Y., Poosala V.: Fast Incremental Maintenance of Approximate Histograms. VLDB Conf. (1997) 466-475

[27] Gilbert A., Guha S., Indyk P., Kotidis Y., Muthukrishnan S., Strauss M.: Fast, Small-Space Algorithms for Approximate Histogram Maintenance. ACM STOC (2002) 389-398

[28] Gilbert A., Kotidis Y., Muthukrishnan S., Strauss M.: Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries. VLDB Conf. (2001) 79-88

[29] Gilbert A., Kotidis Y., Muthukrishnan S., Strauss M.: How to Summarize the Universe: Dynamic Maintenance of Quantiles. VLDB Conf. (2002) 454-465

[30] Guha S., Indyk P., Muthukrishnan S., Strauss M.: Histogramming Data Streams with Fast Per-Item Processing. ICALP Conf. (2002) 681-692

[31] Gunopulos D., Kollios G., Tsotras V., Domeniconi C.: Approximating Multi-Dimensional Aggregate Range Queries Over Real Attributes. SIGMOD Conf. (2000) 463-474

[32] Haas P., Hellerstein J.: Ripple Joins for Online Aggregation. SIGMOD Conf. (1999) 287-298

[33] Hellerstein J., Haas P., Wang H.: Online Aggregation. SIGMOD Conf. (1997) 171-182

[34] Ioannidis Y.: Universality of Serial Histograms. VLDB Conf. (1993) 256-267

[35] Ioannidis Y.: Approximations in Database Systems. ICDT (2003) 16-30

[36] Ioannidis Y., Christodoulakis S.: On the Propagation of Errors in the Size of Join Results. SIGMOD Conf. (1991) 268-277

[37] Ioannidis Y., Christodoulakis S.: Optimal Histograms for Limiting Worst-Case Error Propagation in the Size of Join Results. ACM TODS 18:4 (1993) 709-748

[38] Ioannidis Y., Poosala V.: Balancing Histogram Optimality and Practicality for Query Result Size Estimation. SIGMOD Conf. (1995) 233-244

[39] Ioannidis Y., Poosala V.: Histogram-Based Approximation of Set-Valued Query-Answers. VLDB Conf. (1999) 174-185

[40] Jagadish H. V., Jin H., Ooi B. C., Tan K.-L.: Global Optimization of Histograms. SIGMOD Conf. (2001) 223-234

[41] Jagadish H. V., Kapitskaia O., Ng R., Srivastava D.: Multi-Dimensional Substring Selectivity Estimation. VLDB Conf. (1999) 387-398

[42] Jagadish H. V., Koudas N., Muthukrishnan S., Poosala V., Sevcik K., Suel T.: Optimal Histograms with Quality Guarantees. VLDB Conf. (1998) 275-286

[43] Jagadish H. V., Ng R., Srivastava D.: Substring Selectivity Estimation. PODS Symposium (1999) 249-260

[44] Koudas N., Muthukrishnan S., Srivastava D.: Optimal Histograms for Hierarchical Range Queries. PODS Symposium (2000) 196-204

[45] Krishnan P., Vitter J., Iyer B.: Estimating Alphanumeric Selectivity in the Presence of Wildcards. SIGMOD Conf. (1996) 282-293

[46] König A., Weikum G.: Combining Histograms and Parametric Curve Fitting for Feedback-Driven Query Result-Size Estimation. VLDB Conf. (1999) 423-434

[47] Kooi R.: The Optimization of Queries in Relational Databases. PhD Thesis, Case Western Reserve University (1980)

[48] Lee J.-H., Kim D.-H., Chung C.-W.: Multi-Dimensional Selectivity Estimation Using Compressed Histogram Information. SIGMOD Conf. (1999) 205-214

[49] Lim L., Wang M., Padmanabhan S., Vitter J., Parr R.: XPathLearner: An On-Ling Self-Tuning Markov Histogram for XML Path Selectivity Estimation. VLDB Conf. (2002) 442-453

[50] Lohman G.: personal communication. (2003)

[51] Luo G., Ellmann C., Haas P., Naughton J.: A Scalable Hash Ripple Join Algorithm. SIGMOD Conf. (2002) 252-262

[52] Manku G. S., Motwani R.: Approximate Frequency Counts over Data Streams. VLDB Conf. (2002) 346-357

[53] Manku G. S., Rajagopalan S., Lindsay B.: Approximate Medians and Other Quantiles in One Pass and with Limited Memory. SIGMOD Conf. (1998) 426-435

[54] Manku G. S., Rajagopalan S., Lindsay B.: Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Datasets. SIGMOD Conf. (1999) 251-262

[55] Marshall A., Olkin L.: Inequalities: Theory of Majorization and its Applications. Academic Press (1986)

[56] Matias Y., Vitter J., Wang M.: Wavelet-Based Histograms for Selectivity Estimation. SIGMOD Conf. (1998) 448-459

[57] Matias Y., Vitter J., Wang M.: Dynamic Maintenance of Wavelet-Based Histograms. VLDB Conf. (2000) 101-110

[58] Muralikrishna M., DeWitt D.: Equi-Depth Histograms for Estimating Selectivity Factors for Multi-Dimensional Queries. SIGMOD Conf. (1988) 28-36

[59] Muthukrishnan S., Poosala V., Suel T.: On Rectangular Partitionings in Two Dimensions: Algorithms, Complexity, and Applications. ICDT (1999) 236-256

[60] Papadopoulos A., Manolopoulos Y.: Structure-Based Similarity Search with Graph Histograms. DEXA Workshop (1999) 174-179

[61] Pass G., Zabih R.: Comparing Images Using Joint Histograms. Multimedia Systems 7 (1999) 234-240

[62] Piatetsky-Shapiro G., Connell C.: Accurate Estimation of the Number of Tuples Satisfying a Condition. SIGMOD Conf. (1984) 256-276

[63] Polyzotis N., Garofalakis M.: Statistical Synopses for Graph-Structured XML Databases. SIGMOD Conf. (2002) 358-369

[64] Polyzotis N., Garofalakis M.: Structure and Value Synopses for XML Data Graphs. VLDB Conf. (2002) 466-477

[65] Poosala V., Ioannidis Y.: Estimation of Query-Result Distribution and its Application in Parallel-Join Load Balancing. VLDB Conf. (1996) 448-459

[66] Poosala V., Ioannidis Y.: Selectivity Estimation Without the Attribute Value Independence Assumption. VLDB Conf. (1997) 486-495

[67] Poosala V., Ioannidis Y., Haas P., Shekita E.: Improved Histograms for Selectivity Estimation of Range Predicates. SIGMOD Conf. (1996) 294-305

[68] Sitzmann I., Stuckey P.: Improving Temporal Joins Using Histograms. DEXA Conf. (2000) 488-498

[69] Stigler S.: The History of Statistics: The Measurement of Uncertainty before 1900. Harvard University Press (1986)

[70] Stillger M., Lohman G., Markl V., Kandil M.: LEO - DB2's LEarning Optimizer. VLDB Conf. (2001) 19-28

[71] Sun W., Ling Y., Rishe N., Deng Y.: An Instant and Accurate Size Estimation Method for Joins and Selection in a Retrieval-Intensive Environment. SIGMOD Conf. (1993) 79-88

[72] Thaper N., Guha S., Indyk P., Koudas N.: Dynamic Multidimensional Histograms. SIGMOD Conf. (2002) 428-439

[73] Theodoridis, S., Koutroumbas K.: Pattern Recognition. Academic Press, 2nd edition (2003)

[74] Tufte E.: The Visual Display of Quantitative Information. Graphics Press (1983)

[75] Vitter J., Wang M.: Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. SIGMOD Conf. (1999) 193-204

[76] Vitter J., Wang M., Iyer B.: Data Cube Approximation and Histograms via Wavelets. CIKM Conf. (1998) 96-104

[77] Wang M., Vitter J., Iyer B.: Selectivity Estimation in the Presence of Alphanumeric Correlations. ICDE (1997) 169-180

[78] Witkowski A.: personal communication. (2003)

[79] Wu Y., Patel J., Jagadish H. V.: Using Histograms to Estimate Answer Sizes for XML Queries. Information Systems 28:1-2 (2003) 33-59