Bulletin of the Technical Committee on

Data Engineering

December 1997 Vol. 20 No. 4

IEEE Computer Society

Letters

Special Issue on Data Reduction Techniques

Announcements and Notices

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems.

Letter from the Editor-in-Chief

1 Technical Committee Election

The Technical Committee on Data Engineering has held an election for a new TC Chair. The candidates for TC Chair were Betty Salzberg and Erich Neuhold. **The new TCDE Chair is Betty Salzberg**. Betty will begin her tenure as of January, 1998. I look forward to working with Betty to make this transition as seamless as possible.

I would like to thank the people who made this election possible. This includes, of course, both Betty and Erich for agreeing to run. The members of our nominating committee were Amit Sheth (chair), Nick Cercone, and Ron Sacks-Davis. I thank them for their efforts in planning and executing this election. I think it is also important to thank those TC members who took the time to vote in the election. Participation as voters is an important role that is too frequently not given sufficient importance.

Finally, I want to thank Rakesh Agrawal for serving over the last five years as TCDE Chair. Without Rakesh's efforts, there would not be a Technical Committee on Data Engineering. Rakesh revived an organization that was completely dormant. His efforts resulted in the revival of the TC as a going concern. He recruited other members of the executive committee as well as myself as editor. Rakesh took the lead in this effort and made it happen.

This Issue

Data reduction has become an area of increasing interest. Classically, the database community was interested in histograms to aid the optimization process. But more recently, OLAP and data mining have exploited it as a way of efficiently generating approximate answers to queries- in some cases making feasible queries that would have been prohibitive otherwise. The current issue is a report from an informal workshop held last summer in New Jersey. Hence the title of the sole paper in the issue. I agree with Joe Hellerstein that the issue is particularly timely as more and more people recognize the importance of data reduction to effective exploitation of databases. I also want to thank Joe for his role as issue editor. This required a very active involvement by Joe in order to orchestrate the production of a report by the workshop participants. The report gives a very broad survey of the area in a readily digestible form.

Changing Editorial Staff

As I did last month, I continue to make changes to the editorial staff of the Bulletin as two-year terms for editors expire. The current issue is the last one that will be done by Joe Hellerstein. I want to thank Joe for putting together two very successful issues of the Bulletin. The Bulletin depends on this kind of hard work, and we are indeed grateful that Joe agreed to join this effort.

I also am happy to announce a new editor for the Bulletin, Amr El Abbadi. Amr joined the faculty at the University of California at Santa Barbara in 1987 and is now an associate professor. His interests and expertise include availability, fault tolerance, replication, data placement and access, and heterogeneous systems. I look forward to working with Amr over the next couple of years.

David Lomet
Microsoft Research

Letter from the Special Issue Editor

In the last few years there has been an increasing interest in reducing large sets of data down to a "big picture". This desire has been expressed in many quarters, by people working on such apparently disparate problems as query cost estimation, data mining, decision support, OLAP, compression, information retrieval and data visualization. A host of techniques have been applied to the broad problem of data reduction, each having its pros and cons in various settings.

I became interested in unifying these ideas because of confluences occuring in supposedly separate research projects in the database group at Berkeley (data visualization, indexing, cost estimation and online query processing.) Looking for guidance, I went to the ACM-SIGMOD conference in the spring of 1997 with hopes of rounding up a variety of people for a workshop, perhaps to be held at the following year's conference. H.V. Jagadish was sufficiently enthused by this idea that he organized and hosted a meeting of minds at AT&T Labs just a few weeks later. A broad selection of database researchers converged on New Jersey, some traveling expressly for the meeting. As the list of authors in this edition indicates, it was a mix of veterans and junior researchers, academics and industrialists, with a broad set of skills and applications in mind.

The meeting began with a collective "brain dump", during which each participant spoke about their research and applications in the area. There was clearly a large set of different techniques available, and my original ideas of a grand unifying theory of data reduction were being replaced by an understanding of the area's breadth and richness. After discussing the variety of both techniques and applications, a collective decision was made to try describing the techniques in a less application-specific way, very roughly characterizing their utility (i.e. "thumbs up" or "thumbs down") for different classes of data. The resident expert on a technique was entitled to do the characterizing, with (of course) kibbitzing from the crowd as a whole. The end result of the meeting was a table of applicability which is presented at the end of the article in this issue, and serves as a reduced representation of the day's discussion.

As the day wore on, it was suggested that the conversation merited further and wider exposition in some publication, and the group agreed to write the article that is presented herein. Jagadish provided a skeleton for the sections of the article that roughly describes the different classes of data we considered; this skeleton is described in his introductory section. Each remaining section is an overview of a technique and its history, along with expanded commentary about the application to these classes. The different sections were written by different participants (or pairs of participants), who were free to express their understanding of the issues without editorial modification. My job as editor was greatly simplified by the efforts of Jagadish and also of Ken Ross, who helped to massage the various pieces of text into one document.

I hope that this issue of *Data Engineering Bulletin* — while different from the typical collection of research papers and reports — serves its intended purpose: to be a single convenient reference for techniques in data reduction, and to promote cross-pollination of ideas between researchers who are experts in these different areas. Up to now there have been isolated linkages across the various application areas, e.g. the late Gennady Antoshenkov's use of indexes for selectivity estimation, or a larger variety of researchers' use of compression techniques for indexing and data mining. I believe that much can yet be achieved by this kind of non-traditional application of methodologies across applications, and I hope that this bulletin has some small effect in promoting this activity.

Joseph M. Hellerstein
U.C. Berkeley

The New Jersey Data Reduction Report

Daniel Barbará William DuMouchel Christos Faloutsos Peter J. Haas
Joseph M. Hellerstein Yannis Ioannidis H. V. Jagadish Theodore Johnson
Raymond Ng Viswanath Poosala Kenneth A. Ross Kenneth C. Sevcik*

1 Introduction

There is often a need to get quick approximate answers from large databases. This leads to a need for *data reduction*. There are many different approaches to this problem, some of them not traditionally posed as solutions to a data reduction problem. In this paper we describe and evaluate several popular techniques for data reduction.

Historically, the primary need for data reduction has been internal to a database system, in a cost-based query optimizer. The need is for the query optimizer to estimate the cost of alternative query plans cheaply – clearly the effort required to do so must be much smaller than the effort of actually executing the query, and yet the cost of executing any query plan depends strongly upon the numerosity of specified attribute values and the selectivities of specified predicates. To address these query optimizer needs, many databases keep summary statistics. Sampling techniques have also been proposed.

More recently, there has been an explosion of interest in the analysis of data in warehouses. Data warehouses can be extremely large, yet obtaining answers quickly is important. Often, it is quite acceptable to sacrifice the accuracy of the answer for speed. Particularly in the early, more exploratory, stages of data analysis, interactive response times are critical, while tolerance for approximation errors is quite high. Data reduction, thus, becomes a pressing need.

The query optimizer need for estimates was completely internal to the database, and the quality of the estimates used was observable by a user only very indirectly, in terms of the performance of the database system. On the other hand, the more recent data analysis needs for approximate answers directly expose the user to the estimates obtained. Therefore the nature and quality of these estimates becomes more salient. Moreover, to the extent that these estimates are being used as part of a data analysis task, there may often be "by-products" such as, say, a hierarchical clustering of data, that are of value to the analyst in and of themselves.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Email addresses in order: dbarbara@isse.gmu.edu, dumouchel@research.att.com, christos@cs.cmu.edu, peterh@almaden.ibm.com, jmh@cs.berkeley.edu, yannis@di.uoa.gr, jag@research.att.com, johnsont@research.att.com, rng@cs.ubc.ca, poosala@research.bell-labs.com, kar@cs.columbia.edu, sevcik@cs.toronto.edu.

1.1 The Techniques

For many in the database community, particularly with the recent prominence of data cubes, data reduction is closely associated with aggregation. Further, since histograms aggregate information in each bucket, and since histograms have been popularly used to record data statistics for query optimizers, one may naturally be inclined to think only of histograms when data reduction is suggested. A significant point of this report is to show that this is not warranted. While histograms have many good properties, and may indeed be the data reduction technique of choice in many circumstances, there is a wealth of alternative techniques that are worth considering, and many of these are described below.

Following standard statistical nomenclature, we divide data reduction techniques into two broad classes: parametric techniques that assume a model for the data, and then estimate the parameters of this model, and non-parametric techniques that do not assume any model for the data. The former are likely, when well-chosen, to result in substantial data reduction. However, choosing an appropriate model is an art, and a parametric technique may not always do well with any given data set. In this paper we consider singular value decomposition and discrete wavelet transform as transform-based parametric techniques. We also consider linear regression models and log-linear models as direct, rather than transform-based, parametric techniques.

A histogram is a non-parametric representation of data. So is a cluster-based reduction of data, where each data item is identified by means of its cluster representative. Perhaps a more surprising inclusion is the notion of an index tree as a data reduction device. The central observation here is that a typical index partitions the data into buckets recursively, and stores some information regarding the data contained in the bucket. With minimal augmentation, it becomes possible to answer queries approximately based upon an examination of only the top levels of an index tree. If these top levels are cached in memory, as is typically the case, then one can view these top levels of the tree as a reduced form of data eminently suited for approximate query answering.

Finally, one way of reducing data is to bypass the data representation problem addressed in all the techniques above. Instead, one could just sample the given data set to produce a smaller reduced data set, and then operate on the reduced data set to obtain quick but approximate answers. This technique, even though not directly supported by any database system to our knowledge, is widely used by data analysts who develop and test hypotheses on small data samples first and only then do a major run on the full data set.

1.2 The Data Set

The appropriateness of any data reduction technique is centrally dependent upon the nature of the data set to be reduced. Based upon the foregoing discussion, it should be evident that there is a wide variety of data sets, used for a wide variety of analysis applications. Moreover, multi-dimensionality is a given, in most cases.

To enable at least a qualitative discussion regarding the suitability of different techniques, we devised a taxonomy of data set types, described below.

1.2.1 Distance Only

For some data sets, all we have is a distance metric between data points – without any embedding of the data points into any multi-dimensional space. We call these **distance only** data sets. Many data reduction (and indexing) techniques do not apply to such data sets. However, an embedding in a multi-dimensional space can often be obtained through the use of multi-dimensional scaling, or other similar techniques.

1.2.2 Multi-dimensional Space

The bulk of our concern is with data sets where individual data points can be embedded into an appropriate multi-dimensional attribute space. We consider various characteristics, in two main categories: intrinsic characteristics of each individual attribute, such as whether it is ordered or nominal, discrete or continuous; and extrinsic characteristics, such as sparseness and skew, which may apply to individual attributes or may be used to characterize the data set as a whole. We also consider dimensionality of the attribute space, which is a characteristic of the data set as a whole rather than that of any individual attribute.

1.2.3 Intrinsic Characteristics

We seem to divide the world strongly between ordered and unordered (or nominal) attributes. Unordered attributes can always be ordered by defining a hash label and sorting on this label. So the question is not as much whether the attribute is ordered by definition as whether it is ordered in spirit, that is, with useful semantics to the order. For example, a list of (customer) names sorted alphabetically is ordered by definition. However, for many reasonable applications, there is unlikely to be any pattern based on occurrence of name in the dictionary, and it is not very likely that queries will specify ranges of names. Therefore, for the purposes of data representation, such an attribute is effectively unordered. Similar arguments hold for account numbers, sorted numerically.

Ordered Attributes have values drawn from a finite or an infinite interval. The points in the data set may take discrete or continuous values. None of these details matter as far as data reduction techniques are concerned. There is a difference in language (and formal notation) between discrete and continuous domains. For convenience, we will use only the language of the discrete domain in what follows. The notation and language for a continuous attribute follows analogously, and is not given explicitly in this document. Some attributes, such as rank, may be ordered but have no metric associated with the order. We do not consider such attributes in this report.

Unordered Attributes can have values that are drawn from a flat or a hierarchical name space. The name space is said to be **flat** if there is no particular structure to the range of attribute values. A name space is said to be **hierarchical** if values within a sub-category are in some sense "closer" than values within the next higher level category, and so on, as in a file system hierarchy. Item stock unit numbers, library book classification numbers, and classification systems in general, all have this sort of property.

1.2.4 Extrinsic Attributes

Sparse Data Set A data set is said to be **sparse** if most points in the attribute space defined have no data points corresponding. Conversely, a data set is **dense** if most coordinate points in the attribute space have at least one data point defined. At least for ordered and hierarchical attributes, one can aggregate "ranges" of attribute values to change a sparse space into a less sparse (or more dense) data space. While such manipulations may be common in practice, we work with the data set as given to the data representation/reduction process, paying no heed to any pre-processing steps that may have been involved.

Skewed Data Set A data set is said to be **skewed** if the number of data points per attribute point has a high variance across the entire data space, but has a substantially lower variance in appropriately defined "local" regions. Note that this definition of skew applies only to ordered attributes. Note also that an over-fine disaggregation of attribute value will make it hard to observe skew – aggregation into appropriate size ranges is required. Finally, note that this definition is for "skew in frequency". There is a different notion of skew – "skew in value", where the attribute value for a small number of data points differs substantially from the attribute value for the bulk of the population. Skew in value implies skew in frequency over the attribute value range. However, the converse is not necessarily true.

1.2.5 Dimensionality

By default, we assume all data sets to be low dimensional, that is, represented in ten or fewer dimensions. A data set with more dimensions (attributes) is said to be **high dimensional**. There are many tricks that can be used to reduce the dimensionality of a given data set. We look at the data set after any such techniques have been applied.

1.3 Metrics

In this paper we focus purely on data reduction – the value of a hierarchical clustering of data, for example, to a data analyst or data miner, is not considered, except in so far as it results in less data storage and quick approximate answers to queries. Thus, the primary metric applied to a data reduction technique is how accurate it can be in response to queries as a function of the storage space consumed or as a function of the time taken to respond. In most cases, the time to respond is closely related to the storage consumed.

There are secondary metrics as well. Data often changes, and we may care how easy it is to maintain the data reduced storage structure incrementally in the face of additions and deletions. Some data reduction techniques may cause a complete recomputation, and this is clearly not desirable.

Finally, progressive resolution refinement may sometimes be useful. We may want to produce an approximate answer very rapidly, and then progressively improve the approximation with time. A few data reduction techniques may permit this sort of refinement.

1.4 Outline of Paper and Acknowledgment of Contribution

The paper is organized into sections, one per technique. In each section, the technique is first described and then its applicability to the different types of data sets is explored. Finally, our summary conclusions are presented in Section 10.

Section 2 on the Singular Value Decomposition, and Section 3 on the discrete Wavelet transform were primarily written by Christos Faloutsos. Section 4 on Linear Regression was primarily written by Daniel Barbará. Section 5 on Log-Linear Models was primarily written by Bill DuMouchel. Section 6 on Histograms was primarily written by Vishy Poosala and Yannis Ioannidis. Section 7 on Clustering was primarily written by Raymond Ng. Section 8 on Index Trees was primarily written by Ken Sevcik and Joe Hellerstein. Section 9 was primarily written by Peter Haas. The introduction and conclusion sections were primarily written by H. V. Jagadish. The paper as a whole was edited, smoothed, and formatted by Joe Hellerstein and Ken Ross.

2 Singular Value Decomposition (SVD)

The first proposed method is based on the so-called *Singular Value Decomposition (SVD)* of the data matrix. SVD is a popular and powerful operation, and it has been used in numerous applications, such as statistical analysis (as the driving engine behind the *Principal Component Analysis* [Jol86]), text retrieval under the name of *Latent Semantic Indexing* [Dum94], pattern recognition and dimensionality reduction as the Karhunen-Loeve (KL) transform [DH73], and face recognition [TP91]. SVD is particularly useful in settings that involve least-squares optimization such as in linear regression, dimensionality reduction, and matrix approximation. See [Str80] or [PTVF96] for more details. The latter citation also gives 'C' code.

Example 1: Table 1 provides an example of the kind of matrix that is typical in warehousing applications, where rows are customers, columns are days, and the values are the dollar amounts spent on phone calls each day. Alternatively, rows could correspond to patients, with hourly recordings of their temperature for the past 48 hours, or companies, with stock closing prices over the past 365 days. Such a setting also appears in other contexts. In information retrieval systems rows could be text documents, columns could be vocabulary terms, with the (i, j) entry showing the importance of the j-th term for the i-th document.

day / **customer**	We 7/10/96	Th 7/11/96	Fr 7/12/96	Sa 7/13/96	Su 7/14/96
ABC Inc.	1	1	1	0	0
DEF Ltd.	2	2	2	0	0
GHI Inc.	1	1	1	0	0
KLM Co.	5	5	5	0	0
Smith	0	0	0	2	2
Johnson	0	0	0	3	3
Thompson	0	0	0	1	1

Table 1: Example of a (customer-day) matrix

To make our discussion more concrete, we will refer to rows as "customers" and to columns as "days". The mathematical machinery is applicable to many different applications, such as those mentioned in the preceding paragraph, including ones where there is no notion of a customer or a day, as long as the problem involves a set of vectors or, equivalently, an $N \times M$ matrix $\mathbf{X}$.

2.1 Description

2.1.1 Preliminaries

We shall use the following notational conventions from linear algebra:

- Bold capital letters denote matrices, *e.g.*, $\mathbf{U}$, $\mathbf{X}$.
- Bold lower-case letters denote *column* vectors, *e.g.*, $\mathbf{u}$, $\mathbf{v}$.
- The "$\times$" symbol indicates matrix multiplication.

The SVD is based on the concepts of eigenvalues and eigenvectors:

Definition 2.1: For a square $n \times n$ matrix $\mathbf{S}$, the unit vector $\mathbf{u}$ and the scalar λ that satisfy

$$\mathbf{S} \times \mathbf{u} = \lambda \times \mathbf{u} \tag{1}$$

are called an eigenvector and its corresponding eigenvalue of the matrix $\mathbf{S}$.

2.1.2 Intuition behind SVD

Before we give the definition of SVD, it is best that we try to give the intuition behind it. Consider a set of points as before, represented as an $N \times M$ matrix $\mathbf{X}$. In our running example, such a matrix would represent for N customers and M days, the dollar amount spent by each customer on each day. It would be desirable to group similar customers together, as well as similar days together. This is exactly what SVD does, automatically! Each group corresponds to a "pattern" or a "principal component", *i.e.*, an important grouping of days that is a "good feature" to use, because it has a high discriminatory power and is orthogonal to the other such groups.

Figure 1 illustrates the rotation of axis that SVD implies: suppose that we have M=2 dimensions; then our customers are 2-d points, as in Figure 1. The corresponding 2 directions (x' and y') that SVD suggests are shown. The meaning is that, if we are allowed only k=1, the best direction to project on is the direction of x'; the next best is y', *etc.*See Example 2, for more details and explanations.

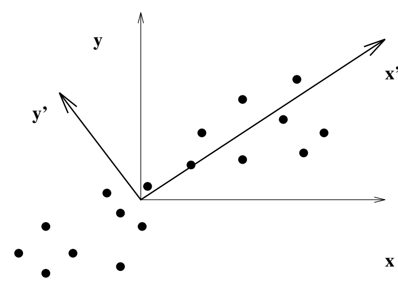

Figure 1: Illustration of the rotation of axis that SVD implies: the "best" axis to project is x'.

2.1.3 Definition of SVD

The formal definition for SVD follows:

Theorem 2.1 (SVD): Given an $N \times M$ real matrix $\mathbf{X}$ we can express it as

$$\mathbf{X} = \mathbf{U} \times \mathbf{\Lambda} \times \mathbf{V}^t \tag{2}$$

where $\mathbf{U}$ is a column-orthonormal $N \times r$ matrix, r is the rank of the matrix $\mathbf{X}$, $\mathbf{\Lambda}$ is a diagonal $r \times r$ matrix and $\mathbf{V}$ is a column-orthonormal $M \times r$ matrix.

Proof: See [PTVF96, p. 59]. □

Recall that a matrix $\mathbf{U}$ is called *column-orthonormal* if its columns $\mathbf{u}_i$ are mutually orthogonal unit vectors. Equivalently: $\mathbf{U}^t \times \mathbf{U} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Also, recall that the rank of a matrix is the highest number of linearly independent rows (or columns).

Eq. 2 equivalently states that a matrix $\mathbf{X}$ can be brought in the following form, the so-called *spectral decomposition* [Jol86, p. 11]:

$$\mathbf{X} = \lambda_1 \mathbf{u}_1 \times \mathbf{v}_1^t + \lambda_2 \mathbf{u}_2 \times \mathbf{v}_2^t + \ldots + \lambda_r \mathbf{u}_r \times \mathbf{v}_r^t \tag{3}$$

where $\mathbf{u}_i$, and $\mathbf{v}_i$ are column vectors of the $\mathbf{U}$ and $\mathbf{V}$ matrices respectively, and λ_i the diagonal elements of the matrix $\mathbf{\Lambda}$. Without loss of generality, we can assume that the eigenvalues λ_i are sorted in decreasing order. Returning to Figure 1, $\mathbf{v}_1$ is exactly the unit vector of the best x' axis; $\mathbf{v}_2$ is the unit vector of the second best axis, y', and so on.

Geometrically, $\mathbf{\Lambda}$ gives the strengths of the dimensions (as eigenvalues), $\mathbf{V}$ gives the respective directions, and $\mathbf{U} \times \mathbf{\Lambda}$ gives the locations along these dimensions where the points occur.

In addition to axis rotation, another intuitive way of thinking about SVD is that it tries to identify "rectangular blobs" of related values in the $\mathbf{X}$ matrix. This is best illustrated through an example.

Example 2: for the above "toy" matrix of Table 1, we have two "blobs" of values, while the rest of the entries are zero. This is confirmed by the SVD, which identifies them both:

$$\mathbf{X} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix} \tag{4}$$

or, in "spectral decomposition" form:

$$\mathbf{X} = 9.64 \times \begin{bmatrix} 0.18 \\ 0.36 \\ 0.18 \\ 0.90 \\ 0 \\ 0 \\ 0 \end{bmatrix} \times [0.58,\ 0.58,\ 0.58,\ 0,\ 0] + 5.29 \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.53 \\ 0.80 \\ 0.27 \end{bmatrix} \times [0,\ 0,\ 0,\ 0.71,\ 0.71]$$

Notice that the rank of the $\mathbf{X}$ matrix is r=2: there are effectively 2 types of customers: weekday (business) and weekend (residential) callers, and two patterns (*i.e.*, groups-of-days): the "weekday pattern" (that is, the group {'We', 'Th', 'Fr'}), and the "weekend pattern" (that is, the group {'Sa', 'Su'}). The intuitive meaning of the $\mathbf{U}$ and $\mathbf{V}$ matrices is as follows:

Observation 2.1: $\mathbf{U}$ can be thought of as the *customer-to-pattern* similarity matrix,

Observation 2.2: Symmetrically, $\mathbf{V}$ is the *day-to-pattern* similarity matrix.

For example, $v_{1,2} = 0$ means that the first day ('We') has zero similarity with the 2nd pattern (the "weekend pattern").

Observation 2.3: The column vectors $\mathbf{v}_j$ $(j = 1, 2, \ldots)$ of the $\mathbf{V}$ are unit vectors that correspond to the directions for optimal projection of the given set of points

For example, in Figure 1, $\mathbf{v}_1$ and $\mathbf{v}_2$ are the unit vectors on the directions x' and y', respectively.

Observation 2.4: The i-th row vector of $\mathbf{U} \times \mathbf{\Lambda}$ gives the coordinates of the i-th data vector ("customer"), when it is projected in the new space dictated by SVD.

For more details and additional properties of the SVD, see [KJF97] or [Fal96].

2.2 Distance-Only Data

SVD can be applied to any attribute-types, including un-ordered ones, like 'car-type' or 'customer-name', as we saw earlier. It will naturally group together similar 'customer-names' into customer groups with similar behavior.

2.3 Multi-Dimensional Data

As described, SVD is tailored to 2-d matrices. Higher dimensionalities can be handled by reducing the problem to 2 dimensions. For example, for the DataCube ('product', 'customer', 'date')('dollars-spent') we could create two attributes, such as 'product' and ('customer' × 'date'). Direct extension to 3-dimensional SVD has been studied, under the name of 3-mode PCA [KD80].

2.3.1 Ordered and Unordered Attributes

SVD can handle them all, as mentioned under the 'Distance-Only' subsection above.

2.3.2 Sparse Data

SVD can handle sparse data. For example, in the Latent Semantic Indexing method (LSI), SVD is used on very sparse document-term matrices. [FD92]. Fast sparse-matrix SVD algorithms have been recently developed [Ber92].

2.3.3 Skewed Data

SVD can handle skewed data. In fact, the more skewed the data values, the fewer eigenvalues that SVD will need to achieve a small error.

2.3.4 High-Dimensional Data

As mentioned, SVD is geared towards 2-dimensional matrices.

3 Wavelets

3.1 Description

The Discrete Wavelet Transform (DWT) is a signal processing technique that is well suited for data reduction. A *k-d signal* is a k-dimensional matrix (or, technically, *tensor*, or DataCube, in our terminology). For example, a 1-d signal is a vector (like a time-sequence); a 2-d signal is a matrix (like a grayscale image) *etc.*. The DWT is closely related to the popular Discrete Fourier Transform (DFT), with the difference that it typically achieves better lossy compression: for the same number of coefficients retained, DWT shows smaller error, on real signals. Thus, given a collection of time sequences, we can encode each one of them with its few strongest coefficients, suffering little error. Similarly, given a k-d DataCube, we can use the k-d DWT and keep a small fraction of the strongest coefficients, to derive a compressed approximation of it.

We focus first on 1-dimensional signals; the DWT can be applied to signals of any dimensionality, by applying it first on the first dimension, then the second, *etc.* [PTVF96].

Contrary to the DFT, there are more than one Wavelet transforms. The simplest to describe and code is the *Haar* transform. *Ignoring* temporarily some proportionality constants, the Haar transform operates on the whole signal (*e.g.*, time-sequence), giving the sum and the difference of the left and right part; then it focuses recursively on each of the halves, and computes the difference of their two sub-halves, *etc.*, until it reaches an interval with one only sample in it.

It is instructive to consider the equivalent, bottom-up procedure. The input signal $\vec{x}$ must have a length n that is a power of 2, by appropriate zero-padding if necessary.

1. Level 0: take the first two sample points x_0 and x_1, and compute their sum $s_{0,0}$ and difference $d_{0,0}$; do the same for all the other pairs of points (x_{2i}, x_{2i+1}). Thus, $s_{0,i} = C * (x_{2i} + x_{2i+1})$ and $d_{0,i} = C * (x_{2i} - x_{2i+1})$, where C is a proportionality constant, to be discussed soon. The values $s_{0,i}$ $(0 \leq i \leq n/2)$ constitute a 'smooth' (=low frequency) version of the signal, while the values $d_{0,i}$ represent the high-frequency content of it.

2. Level 1: consider the 'smooth' $s_{0,i}$ values; repeat the previous step for them, giving the even-smoother version of the signal $s_{1,i}$ and the smooth-differences $d_{1,i}$ $(0 \leq i \leq n/4)$

3. ... and so on recursively, until we have a smooth signal of length 2.

The Haar transform of the original signal $\vec{x}$ is the collection of all the 'difference' values $d_{l,i}$ at every level l and offset i, plus the smooth component $s_{L,0}$ at the last level L $(L = \log_2(n) - 1)$.

Following the literature, the appropriate value for the constant C is $1/\sqrt{2}$, because it makes the transformation matrix *orthonormal* (eg., see Eq. 8). An orthonormal matrix is a matrix which has columns that are unit vectors and that are mutually orthogonal. Adapting the notation (eg., from [Cra94] [VM]), the Haar transform is defined as follows:

$$d_{l,i} = 1/\sqrt{2}\ (s_{l-1,2i} - s_{l-1,2i+1}) \quad l = 0, \ldots, L, \quad i = 0, \ldots, n/2^{l+1} - 1 \tag{5}$$

with

$$s_{l,i} = 1/\sqrt{2}\ (s_{l-1,2i} + s_{l-1,2i+1}) \quad l = 0, \ldots, L, \quad i = 0, \ldots, n/2^{l+1} - 1 \tag{6}$$

with the initial condition:

$$s_{-1,i} = x_i \tag{7}$$

For example, the 4-point Haar transform is as follows. Envisioning the input signal $\vec{x}$ as a column vector, and its Haar transform $\vec{w}$ as another column vector ($\vec{w} = [s_{1,0}, d_{1,0}, d_{0,0}, d_{0,1}]^t$ - the superscript t denoting transposition), the Haar transform is equivalent to a matrix multiplication, as follows:

$$\begin{bmatrix} s_{1,0} \\ d_{1,0} \\ d_{0,0} \\ d_{0,1} \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 & -1/2 \\ 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{8}$$

The above procedure is shared among *all* the wavelet transforms: we start at the lowest level, applying two functions at successive windows of the signal: the first function does some smoothing, like a weighted average, while the second function does a weighted differencing; the smooth (and, notice, shorter: halved in length) version of the signal is recursively fed back into the loop, until the resulting signal is too short.

There are numerous wavelet transforms [PTVF96], some popular ones being the so-called Daubechies-4 and Daubechies-6 transforms [Dau92].

3.1.1 Discussion

The computational complexity of the above transforms is $O(n)$, as can be verified from Eq. 5-7. In addition to their computational speed, there is a fascinating relationship between wavelets, multiresolution methods (like quadtrees or the pyramid structures in machine vision), and fractals. The reason is that wavelets, like quadtrees, will need only a few non-zero coefficients for regions of the image (or the time sequence) that are smooth (*i.e.*, homogeneous), while they will spend more effort on the 'high activity' areas. It is believed [Fie93] that the mammalian retina consists of neurons which are tuned each to a different wavelet. Naturally occurring scenes tend to excite only few of the neurons, implying that a wavelet transform will achieve excellent compression for such images. Similarly, the human ear seems to use a wavelet transform to analyze a sound, at least in the very first stage [Dau92, p. 6] [WS93].

In conclusion, the Discrete Wavelet Transform (DWT) achieves even better energy concentration than the DFT and Discrete Cosine (DCT) transforms, for natural signals [PTVF96, p. 604]. It uses multiresolution analysis, and it models well the early signal processing operations of the human eye and human ear.

3.2 Distance-Only Data

In this case, DWT can only be applied after the data have been mapped to an k-dimensional space, with, e.g., Multidimensional scaling, or FastMap [FL95].

3.3 Multi-Dimensional Data

As mentioned, the DWT can be applied to an k-dimensional hyper-cube. In fact, it has been very successful in image compression [PTVF96], where a grayscale image is treated as a 2-d matrix.

3.3.1 Ordered and Unordered Attributes

DWT will give good results for ordered attributes, when successive values tend to be correlated (which is typically the case in real datasets). For unordered attributes (like "car-type"), DWT can still be applied, but it won't give the good compression we would like.

3.3.2 Sparse Data

DWT will work fine on sparse data - it will just have zero coefficients in the deserted regions of the address space.

3.3.3 Skewed Data

DWT should work well on skewed data because it is adaptable: it will have many non-zero coefficients for the portion of the address space that has large values, and near-zero coefficients for the rest.

3.3.4 High-Dimensional Data

As mentioned several times before, the definition of DWT can be trivially extended to arbitrary dimensionalities. However, although linear on the number of cells of the k-d matrix, notice that the number of cells itself grows exponentially with the number of dimensions k. This is the only point that may create efficiency problems. However, most of the competitors will run into similar problems, too (and, probably, sooner than DWT).

4 Regression

Regression is a popular technique that attempts to model data as a function of the values of a multi-dimensional vector. The simplest form of regression is that of *Linear Regression* [WW85], in which a variable Y is modeled as a linear function of another variable X, using Equation 9.

$$Y = \alpha + \beta X \tag{9}$$

The parameters α and β specify the line and are to be estimated by using the data at hand. To do this, one should apply the least squares criterion to the known values $Y_1, Y_2, ..., X_1, X_2, ...$. The least squares formulas for 9 yield the values of β and α as shown in Equations 10 and 11 respectively.

$$\beta = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{\sum(X - \bar{X})^2} \tag{10}$$

$$\alpha = \bar{Y} - \beta\bar{X} \tag{11}$$

where $\bar{X}$ and $\bar{Y}$ are the average values for the data points $X_1, X_2, ...$ and $Y_1, Y_2, ...$ respectively.

The extension of Linear Regression, called *Multiple Regression*, takes account of more than one independent variable X, allowing us to model Y as a linear function of a multidimensional vector. An example of a Multiple Regression model based on two dimensions is shown in Equation 12

$$Y = b_0 + b_1 X_1 + b_2 X_2 \tag{12}$$

Again, b_0, b_1 and b_2 must be estimated using the values at hand. The general procedure to do least square fitting for Multiple Regression can be found in [PTVF96].

It is also possible to use nonlinear functions to perform data regression. Equation 13 shows an example of a nonlinear regression between variables Y and X.

$$Y = b_0 + b_1 X + b_2 X^2 + b_3 X^3 \tag{13}$$

To estimate the parameters of (13), we could simply define the new variables:

$$\begin{aligned} X_1 &= X \\ X_2 &= X^2 \\ X_3 &= X^3 \end{aligned} \tag{14}$$

By the substitutions shown in Equation 14, Equation 13 becomes a linear model:

$$Y = b_0 + b_1 X_1 + b_2 X_2 + b_3 X_3 \tag{15}$$

that can be solved with the standard least square techniques. The method of redefining variables to make the model linear is quite general. For instance, terms like reciprocals ($\frac{1}{X}$ and cosines $(cos(\frac{X}{\pi}))$ can be easily redefined as linear terms. This technique does not work, however, if the nonlinearity is present in the parameters to be estimated.

A notorious case of nonlinearity that can be easily removed by taking logarithms is shown in Equation 16:

$$Y = b_0 X_1^b \tag{16}$$

The following substitutions

$$\begin{aligned} Y_1 &= log(Y) \\ \alpha &= log(b_0) \\ \beta &= b_1 \\ X_1 &= log(X) \end{aligned} \tag{17}$$

transform Equation 16 into the linear model shown in Equation 18.

$$Y_1 = \alpha + \beta X_1 \tag{18}$$

Other models are intractably nonlinear and cannot be subject to any transformation that renders them linear (e.g., the sum of exponential terms). For these models, it is sometimes possible to obtain least-square estimates by performing a lot of calculations on more complex formulae.

Let us describe now in which cases regression can be used to compress or characterize data.

4.1 Distance-Only Data

Clearly regression is useless with this kind of data, since the data is not embedded in any multi-dimensional space.

4.2 Multi-Dimensional Data

1. *Ordered:* This is the class of data for which regression applies more naturally. A simple example would be the case of modeling the amount of sales in a store as a function of the date.

2. *Unordered (Flat/Hierarchical)* Although unordered data can always be nominally ordered and thus subjected to regression, the model obtained by doing this may not be very meaningful. Alternatively, one could do the regression in using the range of a function whose domain is formed by the attribute values. An example of such a function is one to compute marginal values. Consider a dataset that maps "amount of sales" to the variables "store location" and "date". Clearly, the variable "store location" is not ordered. However, it is possible to create a variable X whose domain is formed by the cumulative sales for each store location, and use X as an ordered variable for the regression model.

4.2.1 Sparse Data

The sparseness of the data does not affect the applicability of regression. It is sometimes advisable to perform the regression only for the multidimensional points that are non-zero, to make the model fit the non-zero data better. For example, stores with amount of sales = 0 may be exceptional and fit a regression model poorly.

4.2.2 Skewed Data

Skewness is actually a good feature for regression. Skewed data is more likely to fit better in a model, given that the proper model is found.

4.2.3 High-Dimensional Data

High-dimensionality forces the usage of a multiregression model. The price of using a multiregression model of a high degree is performance. Given a model, the dataset that needs to be modeled might not fit in memory, forcing the estimation of the regression parameters to perform several passes over the data, thus slowing down the process. To alleviate this problem, one might choose to model portions of the dataset, by fixing the values of one or several of the dimensions in each portion. Each of the models will have a smaller degree and the corresponding datasets will be also smaller, perhaps small enough to fit in main memory, making the estimation a faster process. By doing this, however, one increases the number of estimations that need to be performed (one for each portion of the original dataset). Therefore there exists a tradeoff between the size of the portions modeled and the performance of the overall modeling process. If the portions of the dataset are too small, each individual modeling process runs fast, but there may be too many of them to be performed, thus offsetting the gains obtained by the individual runs. On the other hand, if the portions are too big, each individual modeling effort will run slowly, but there will be few models to be run. The tradeoff depends heavily on how sparse the data is: if only a few multidimensional cells are non-zero, then even a high degree portion of the data set (one with only a few of the dimensions fixed) may be small enough to fit in memory.

In the rest of this section we discuss three important aspects of the use of regression as a data reduction technique.

- *Accuracy:* Accuracy, of course, depends on how well the chosen model fits the real data. In practice, however, even with simple models such as linear regression (and multiregression) one can obtain a reasonable approximation to the dataset.

  One way of getting better accuracy progressively is by reducing the influence that outliers have in the model by giving them less weight in the least square regression. This method is known as biweight regression or robust regression; an example of this is the use of weighted least squares [WW85].

  The first thing to do is determine whether a data value is an outlier. That is usually done by measuring the difference between the real value and its estimation, as in Equation 19.

  $$d = Y - \hat{Y} \tag{19}$$

  It is customary to normalize d, dividing it by some overall measure of spread S, as shown in Equation 20. An example of S is the interquartile range of all deviations. (I.e., the difference between the 25th percentile and the 75th percentile of deviations).

  $$Z = \frac{Y - \hat{Y}}{3S} \tag{20}$$

  With Z, we can compute the *biweights*, shown in Equation 21

  $$w = \begin{cases} (1 - Z^2)^2 & \text{if } |Z| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

  Equation 21 effectively makes the weight of an outlier equal to 0. These weights are now used in the estimation formulas shown in Equations 22 and 23.

  $$\beta = \frac{\sum w(X - \bar{X})(Y - \bar{Y})}{\sum w(X - \bar{X})^2} \tag{22}$$

$$\alpha = \hat{Y} - b\hat{X} \tag{23}$$

where the averages $\hat{Y}$ and $\hat{X}$ are found by using Equations 24 and 25.

$$\hat{Y} = \frac{\sum wY}{\sum w} \tag{24}$$

$$\hat{X} = \frac{\sum wX}{\sum w} \tag{25}$$

The new estimation of α and β supports a more robust model. We can improve on it by re-calculating the deviations using Equation 20 and estimating new values of α and β using new weights. That should give an even better line. This process can be repeated until no substantial improvement can be obtained.

- *Progressive resolution refinement:* A way of obtaining progressively refined answers is to store the outliers of the model. A first cut of the answer consists of the estimated values for all the points requested. That answer can be polished by retrieving the real values of the outliers progressively replacing the estimated values for those data points. A technique similar to this has been successfully used in [BS97].

- *Incremental maintenance:* As new data gets incorporated in the dataset, the relevant model(s) need to be updated to reflect the effect of this data. The updating of the model can be achieved by using techniques similar to those described in [CR94] to update polynomial models for selectivity estimation. The techniques use a method called recursive least-square-error [You84] to avoid a lot of expensive recomputation.

5 Log-Linear Models

Log-linear modeling is a methodology for approximating discrete multidimensional probability distributions. The multi-way table of joint probabilities is approximated by a product of lower-order tables. For example, suppose the four categorical attributes A, B, C, and D can respectively assume the values $a = 1, ..., K_A; b = 1, ..., K_B; c = 1, ..., K_C$; and $d = 1, ..., K_D$. Then, if $p(a, b, c, d) = \mathrm{Prob}(A = a, B = b, C = c, D = d)$, one might assume a model of the form:

$$p(a, b, c, d) = \alpha_{ab}\beta_{ac}\gamma_{ad}\delta_{bcd} \tag{26}$$

For given matrices α, β, γ and three-dimensional array δ. The simplest log-linear model is that of independence, which in this example becomes $P(a, b, c, d) = \alpha_a\beta_b\gamma_c\delta_d$. The presence of multiple subscripts in the same array allows for greater dependency within the distributions of the associated attributes. The name "log-linear" is used for these models in the statistics literature because $\log p$ is assumed to be a linear combination of unknown parameters. The phrase "multiplicative model" is more common in the computer science literature. Such models have been discussed and used since the 1940s or earlier, but especially since the 1970s, when computer algorithms to fit them became widely available. Many text-book treatments of log-linear modeling are available, for example [Agr90] and [BFH75]. Sample references from the Computer Science literature are [KK69], [Pea88], and [Mal91].

Log-linear models use only categorical variables – continuous variables must be discretized first, and even then the modeling will not make use of the ordinal nature of the categories. The purpose of using this technique can be either data compression (since the several small arrays will take up less storage than the full multidimensional array) or data smoothing (since estimates of the small arrays will be less subject to sampling variation than elements of the full array), assuming that the full array was computed as observed proportions from a sample.

Using log-linear models involves two steps: choosing a general form (how many factors to use and what sets of attributes are associated with each factor) and then estimating the numerical values of the array elements for each factor (parameter estimation). An important result due to [Bir63] is that, given the results of step one, the parameter estimation problem only requires as input the marginal proportions corresponding to the combinations of attributes making up the factor arrays. By *marginal proportion* we mean the sum of the values of elements in the datacube corresponding to appropriate specified attributes, with all other attributes projected out. In the example of Equation 26, the parameters $\alpha, \beta, \gamma, \delta$ can all be estimated using just the marginals $p(a, b, +, +), p(a, +, c, +), p(a, +, +, d)$ and $p(+, b, c, d)$, where "+" denotes summation over the appropriate range. In statistical terms, the indicated marginals are *sufficient statistics* for the parameters, and no more information is needed to estimate the parameters efficiently, assuming that the model of Equation 26 holds. In addition, the computed approximation will fit the input marginal distributions exactly. Another application of the methodology occurs when only certain marginal tables are available, and it is required to extend the probability distribution to the complete array, as in [Mal89].

The estimation of the parameter arrays can sometimes, for certain assumptions of factor combinations called *decomposable models* or *graphical models*, be quite simple, involving just simple arithmetic products and ratios of the given marginal probabilities. In general, however, an iterative method will be required to obtain the maximum likelihood (maximum entropy) estimates for scaling factors to be applied to the marginals. The most common such method is *iterative proportional scaling*, generally attributed to [DS40], which is guaranteed to converge to a unique solution whenever the marginal arrays have all positive elements and are consistent with each other. One drawback of the standard iterative proportional scaling algorithm is that it requires storage and computation over the complete estimated probability array, which could be quite large. For example, if there are 20 attributes, the complete array could have 10^{10} cells, depending on the number of values each variable takes on. Even if the data base represents millions of entities, the vast majority of the cells will have zero count. In such situations, there is obviously a great advantage to choosing a decomposable model. Among others [Mal91] discusses how to search the set of decomposable models for a good fitting model. As in all such model choice problems, one must consider the usual tradeoff between parsimony and variance reduction on the one hand, and adequacy of representation on the other.

On the whole, log-linear modeling is a powerful and flexible technique that scales up well to many dimensions and has many favorable and well understood statistical properties. The user can specify that arbitrary marginal distributions be fit exactly, and be assured that all estimated probabilities remain within the unit interval. At the cost of discretizing continuous variables, it can be applied to any data type.

5.1 Distance-Only Data

Assuming that the distances can be rounded to a discrete number of values, log-linear models might be useful for data reduction, depending on the complexity of the attributes for labeling endpoints.

5.2 Ordered Data

There is no problem with using ordered categories with log-linear models, and some extensions of these models have been proposed to explicitly use the ordinal information, as in [Agr90] (Chapter 8).

5.3 Unordered Data

This type of data is the primary application for log-linear models.

5.4 Sparse Data

The log-linear methodology does not require dense data. However, as mentioned above, a very high-dimensional sample will usually have many cells with zero count in its multiway frequency table. Thus one may be limited for computational reasons to decomposable log-linear models. This may limit the adequacy of the representation of a sparse data set, depending on the complexity of the distributional dependencies.

5.5 Skewed Data

Since the user of log-linear models is free to choose discrete values to match the distribution of observed values, skewness of data values is not a problem. Skewness of frequencies (presence of some very large counts) is also not necessarily a problem, although such data may mean that simple log-linear models fit poorly.

5.6 High-Dimensional Data

As mentioned above, log-linear models scale up fairly well to ten or so dimensions for arbitrary models. Above that number, it may be necessary to restrict consideration to decomposable models, which have fewer and weaker dependency relations.

5.7 Accuracy

The ability to choose more complex log-linear models allows the user to tune the accuracy of the fit as desired. There is also a well-developed statistical theory providing measures of goodness of fit of models, hypothesis tests comparing models, and confidence limits for the estimated parameters.

5.8 Progressive Resolution Refinement

Once a log-linear model has been constructed, computing the answer to any point query is rather easy, involving simply the multiplication of a few numbers, so progressive resolution refinement is not of much value.

The method of iterative proportional scaling allows the result of fitting one log-linear model to be used as a starting point when fitting a more complex log-linear model - that is, a model inputting higher-dimensional marginals than the first model. This provides good control over the resolution of the model.

5.9 Incremental Maintenance

As new data are collected, the values of the marginal proportions used as input to the model fitting will change. As in the previous item, the fit from the previously computed data cube can be used to begin the iterative proportional scaling and hasten convergence compared to default initial values. However, in practice, the savings in computation will probably not be large in either situation, perhaps in the range of 10-50%.

6 Histograms

Histograms approximate the data in one or more attributes of a relation by grouping attribute values into "buckets" (subsets) and approximating true attribute values and their frequencies in the data based on summary statistics maintained in each bucket. For most real-world databases, there exist histograms that produce low-error estimates while occupying reasonably small space (of the order of 500 bytes in a catalog)[1]. Hence, they are the most commonly used form of statistics in practice (e.g., they are used in DB2, Informix, Ingres, Oracle, Microsoft SQL Server, Sybase, and Teradata). They are used mainly for selectivity estimation purposes within a query optimizer. They have also been used in query execution (e.g., for parallel-join load balancing [PI96]) and there is work in progress on using them for approximate query answering.

6.1 Definitions

In what follows, histograms are defined in the context of a single attribute. The extensions to multiple attributes can be found elsewhere [PI97].

The *domain* $\mathcal{D}$ of an attribute X in relation R is the set of all possible values of X and the (finite) *value set* $\mathcal{V}$ ($\subseteq \mathcal{D}$) is the set of values of X that are actually present in R. Let $\mathcal{V} = \{ v_i \colon 1 \leq i \leq D \}$, where $v_i < v_j$ when $i < j$. The *spread* s_i of v_i is defined as $s_i = v_{i+1} - v_i$, for $1 \leq i < D$. (We take $s_0 = v_1$ and $s_D = 1$.) In this section we only consider numerical attributes. A commonly used technique for constructing histograms on non-numerical attributes (such as string fields, etc.) is to use a function that converts these data types into floating point numbers before constructing a histogram[2]. The *frequency* f_i of v_i is the number of tuples $t \in R$ with $t.X = v_i$. Finally, the *area* a_i of v_i is equal to $v_i \times f_i$. The *data distribution* of X (in R) is the set of pairs $\mathcal{T} = \{ (v_1, f_1), (v_2, f_2), \ldots, (v_D, f_D) \}$. Typically, several real-life attributes tend to have *skewed* or highly *non-uniform* data distributions. The main characteristics of such distributions are unequal frequencies and/or unequal spreads.

A histogram on attribute X is constructed by partitioning the data distribution $\mathcal{T}$ into disjoint subsets called *buckets* and approximating the frequencies and values in each buckets in some common fashion. Typically, the frequencies are approximated by their average. The value domain is most often approximated by the entire set of values in the bucket's range (the *continuous value assumption*). A much more accurate approximation, however, and the one that has been used in recent research is one assuming that all the values in a bucket are separated by the same amount from their next neighbor (the *uniform spread assumption*). In all cases, histograms can be viewed as approximate data distributions of the underlying attributes and used in any estimation problem requiring those distributions. These definitions are illustrated in the following example.

Example 1: Consider a relation with schema `EMP(ename,salary)`. The following table shows how each parameter defined above is instantiated for this relation.

Quantity	**Set of values**					
Attribute Value $\{v_i\}$	10	60	70	120	140	190
Frequency $\{f_i\}$	110	90	20	30	70	80
Spread $\{s_i\}$	50	10	50	20	50	1

Figure 2 plots the data distribution, with attribute values on the x-axis and frequencies on the y-axis. Figure 3 corresponds to the approximate data distribution arising from a histogram using three buckets and making the uniform spread assumption.

[1] Nevertheless, one can construct data distributions that cannot be approximated well using a small number of buckets.
[2] For example, this technique is used in IBM's DB2-6000 system.

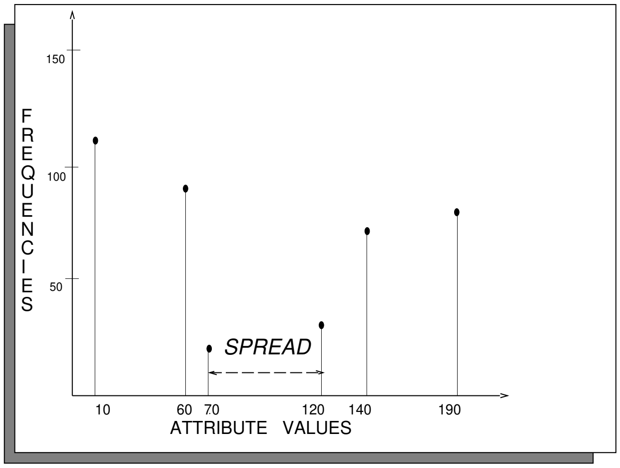

Figure 2: Data Distribution

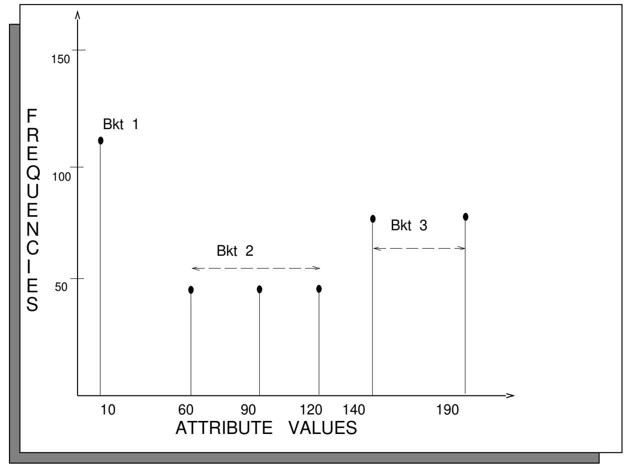

Figure 3: Approximated distribution

One of the key factors affecting the accuracy of the histograms is the *partitioning rule* employed in determining the buckets. In order to illustrate this, two well-known classes of histograms, the *equi-width* and *equi-depth* histograms are described next. Both these histograms group contiguous ranges of attribute values into buckets and assume that all attribute values within the range corresponding to a given bucket occur with equal probability. Theis difference lies in the exact choice of bucket boundaries chosen. In an *equi-width* histogram, the widths of all buckets' ranges are the same; in an *equi-depth* (or *equi-height*) histogram, the total number of tuples having the attribute values associated with each bucket is the same.

In [PIHS96], a set of key properties that characterize histograms have been identified, forming the basis for a taxonomy of histograms. These properties essentially determine the effectiveness histograms in approximating data distributions and are the following: the *sort parameter*, which determine the order in which the attribute-value/frequency pairs of the data distribution are grouped in the histogram; the *histogram class*, which determines the sizes of buckets allowed in the histogram (e.g., are singleton buckets mandatory?); the *source parameter*, which represents the quantity that the histogram should try to capture accurately; and the *partition constraint*, which is the mathematical rule that determines where exactly the histogram boundaries will fall based on the source parameter. Both the sort and the source parameters are functions of the attribute-value/frequency pairs in the data distribution. Examples include the attribute value itself, the frequency itself, and the area. The partition constraints include the following.

- *Equi-sum:* In an *equi-sum* histogram (with β buckets), the sum of the source values in each bucket is approximately the same and equal to $1/\beta$ times the sum of all the source values in the histogram.

- *V-Optimal:* The *V-Optimal* histogram on an attribute is the histogram with the least *variance* among all the histograms using the same number of buckets. Here, the variance of a histogram is the weighted sum of its source parameters values in each bucket, with the weights being equal to the number of values in that bucket.

- *MaxDiff:* In a *MaxDiff* histogram, there is a bucket boundary between two source parameter values that are adjacent (in sort parameter order) if the difference between these values is one of the $\beta - 1$ largest such differences.

- *Compressed:* In a *Compressed* histogram, the n highest source values are stored separately in n

singleton buckets; the rest are partitioned as in an equi-sum histogram. Often n is the number of source values that (a) exceed the sum of all source values divided by the number of buckets and (b) can be accommodated in a histogram with β buckets.

- *Spline-based:* In a *spline-based* histogram, the maximum absolute difference between a source value and the average of the source values in its bucket is minimized.

We refer to a histogram with c, u, and s as the partition constraint, source parameter, and sort parameter as the $c(s,u)$ histogram. Figure 4 provides an overview of the new combinations that were introduced in [PIHS96] together with the traditional combinations. Efficient sampling-based techniques exist for computing all classes of histograms and are given in [PIHS96].

SORT PARAMETER	SOURCE PARAMETER			
	SPREAD (S)	FREQUENCY (F)	AREA (A)	CUM. FREQ (C)
VALUE (V)	EQUI–SUM	EQUI–SUM V–OPTIMAL MAX–DIFF COMPRESSED	V–OPTIMAL MAX–DIFF COMPRESSED	V–OPTIMAL SPLINE BASED
FREQUENCY (F)		V–OPTIMAL MAX–DIFF		
AREA (A)			V–OPTIMAL MAX–DIFF	

Figure 4: Augmented Histogram Taxonomy.

Most of the work on histograms is in the context of evaluating their accuracy in estimating the result sizes of queries containing selections [Koo80, PSC84] and joins [IC93, Ioa93, IP95a]. Multi-dimensional histograms have also been studied in detail [MD88, PI97]. By building histograms on multiple attributes together, these techniques are able to capture dependencies between those attributes. Incremental maintenance techniques for histograms and samples have also been investigated [GMP97], as has the use of histograms in parallel-join load balancing [PI96]. Finally, there are several sources where one may find extensive discussions of histogram-based estimation techniques [Koo80, IP95b, MCS88, Poo97].

In the following sections, the effectiveness of histograms in approximating different kinds of data is studied.

6.2 Distance-Only Data

The current histogram techniques cannot approximate such data, because they rely on information about the placement of data in a multi-dimensional space. A possible solution would be to identify new choices for the parameters of the taxonomy that are based on distance and a new value domain approximation technique that does not rely on data placement.

6.3 Multi-Dimensional Data

1. *Ordered:* Histograms are well suited for approximating ordered data (discrete or continuous domains, finite or infinite intervals). Most of the research so far on the accuracy of histograms has focussed on ordered data and has shown that the most accurate classes of histograms for ordered data in fact preserve the order in grouping values into buckets.

2. *Unordered:* Histograms that do not use the attribute value as the sort parameter assume that there is no inherent ordering among the attribute values. Hence, these histograms can also be used to approximate unordered flat data. On the other hand, all techniques for approximating the value domain within a bucket rely on an inherent order among those values. As a result, a histogram on unordered data needs to keep track of all values falling within each bucket, which is clearly impractical for large value domains. In summary, it is unclear how histograms can be used to efficiently approximate unordered data. This discussion applies to hierarchical unordered data as well, with an important exception. Often the hierarchy structure can be used to group values into buckets (e.g., bucket per each week), in which case the values inside a bucket (often) need not be stored (e.g., days of the week) and the above problem disappears.

6.3.1 Sparse Data

Histograms have been shown in earlier work to be highly effective in approximating sparse and dense data [PIHS96]. Specifically, histograms making the uniform spread assumption work for both kinds of data while the older continuous value assumption works well only for dense data.

6.3.2 Skewed Data

Histograms have been shown to be most effective in approximating highly skewed data (frequency and value domain skews) as well as nearly uniform data. For high skews, there are a few significant attribute values (or frequencies) that can be captured accurately by the histograms using an appropriate choice for its sort and source parameters. For nearly uniform data, most histograms are likely to be highly accurate because the uniformity assumptions within the bucket will not result in high errors. Surprisingly, histograms perform relatively the poorest on data that is moderately skewed. For example, when several values have high but dissimilar frequencies, grouping them into a bucket will incur high errors because of the dissimilarities, and hence one needs more buckets to be accurate in this case.

6.3.3 High-Dimensional Data

Research has shown that multi-dimensional histograms are highly effective in approximating data in multiple (scalar) attributes of a relation. In all these studies, however, the number of attributes has been less than 5. More work needs to be done to effectively use histograms for very high dimensions. The same applies to approximating multi-dimensional data within a single attribute as well (e.g., polygons).

6.4 Aspects of histogram usage

- *Accuracy:* Although histograms are used in many systems, many of the histograms proposed in earlier works are not always effective or practical. For example, *equi-depth* histograms [Koo80, MD88, PSC84] work well for range queries only when the data distribution has low skew, while *V-Optimal(F,F)* histograms [IC93, Ioa93, IP95a] have only been proven optimal for equality joins and selections when a list of all the attribute values in each bucket is maintained. Earlier work has shown that the most accurate and practical histograms belong to the *V-Optimal(V,A)* and *MaxDiff(V,A)* classes [Poo97]. Briefly, these histograms group contiguous ranges of values into buckets and avoid grouping attribute values with highly different *area*s. These histograms have been shown to be highly accurate for both join and selection queries [Poo97].

- *Progressive resolution refinement:* A histogram on flat (non-hierarchical) data can not be used to provide different levels of resolution. On the other hand, histograms built on hierarchical data can be used at various levels of the hierarchy (if the buckets were also constructed based on the hierarchy) to provide progressive resolution refinement.

- *Incremental maintenance:* The common approach taken by all commercial systems is to periodically recompute the histogram from the updated data. This approach leads to inaccurate estimates from outdated histograms and can be quite expensive when used on a database with very large number of relations. Recent work has shown that some classes of histograms can be maintained efficiently and accurately using incremental techniques [GMP97]. These techniques make use of a (possibly disk-resident) *backing sample*, which is also incrementally maintained as a uniformly random representative of the underlying data. The histograms are kept in main-memory and are updated frequently to preserve their accuracy while the sample is accessed very infrequently - basically when the histogram becomes too inaccurate.

6.5 End-biased histograms (Outliers)

End-biased histograms are a special case of histograms that have only singleton buckets - i.e., each bucket has a single attribute-value/frequency pair. As a result, the value and its frequency are accurately captured. Obviously, due to limited space, not all values in the relation can be stored in this manner. Even these limited number of buckets, however, often provide highly accurate estimates - either directly or by supplementing other statistics. Hence, these outlier isolation techniques can be employed alongside any of the data reduction techniques described in this report.

There has been lot of work on storing and using outlier information in databases. Some of the research on histogram-based join result size estimation has shown the benefits of storing values with extreme frequencies. The class of *end-biased histograms* contains a few high-frequency values and a few low-frequency values in singleton buckets and the rest in a single large bucket [Ioa93].[3]. These histograms are less expensive to construct than the general class of histograms, occupy less space, and often offer equally high accuracies for join queries. A few commercial systems also employ singleton buckets for selectivity estimation purposes. For example, DB2 stores a small number of attribute values with the highest frequencies in the relation. For a highly skewed relation with a few very high frequencies, this information by itself may be enough to provide accurate estimates. When enhanced with a usual histogram on the remaining data, the combined set of statistics has been shown to be highly accurate. These combined statistics are in fact also used in DB2 and are known as *Compressed* histograms [PIHS96].

In the following sections, the effectiveness of using singleton buckets is discussed. In practice, these statistics are almost always used in conjunction with other forms of statistics, which eliminates most of the deficiencies below.

6.5.1 Distance-Only, Ordered, Unordered Data

End-biased histograms are not affected by these properties of the data because they store each value individually in a singleton bucket and do not assume anything about the relation between various data values.

6.5.2 Sparse Data

End-biased histograms are very well suited for sparse data because there are fewer values that need to be captured. On the other hand, they are incapable of approximating dense data because of the

[3] Storing lowest frequency values in singleton buckets is useful if the distribution has several high frequencies that are equal and a few much smaller frequencies. But, often, one only stores high frequencies in singleton buckets.

limited number of (singleton) buckets.

6.5.3 Skewed Data

End-biased histograms are most effective in approximating highly skewed data (both frequency and value domain skews). The reason is that skewed data often contains very few values that cause the skew (e.g., values with extreme frequencies) which can be captured accurately using the singleton buckets. These histograms, however, fail to capture lower to medium skews because of the large number of significant values to capture.

6.5.4 High-Dimensional Data

End-biased histograms function the same in capturing significant values in data of any dimensionality.

6.6 Aspects of end-biased histogram usage

- *Accuracy:* Although singleton buckets are used in some commercial systems, their accuracy has not been studied much in the literature. It has been shown that end-biased histograms are quite accurate in estimating join results sizes, particularly when the data is skewed [IP95a]. On the other hand, since they do not approximate the entire data distribution, they can not be used effectively for estimating the result sizes of selection predicates. There are two ways to increase the accuracy of end-biased histograms: adding more singleton buckets, and carefully choosing the values to be preserved in singleton buckets (which need not always be the high frequency values).

- *Progressive resolution refinement:* End-biased histograms directly provide the finest partitioning of data (into individual values) and hence can not provide further resolution.

- *Incremental maintenance:* Gibbons and Matias have designed efficient techniques with theoretical bounds on accuracy to incrementally maintain the highest frequency values in a database relation [GM96].

7 Clustering Techniques

In the past 30 years, cluster analysis has been widely studied in statistics. The objective is to identify *clusters* embedded in the data. Intuitively, a cluster is a collection of data objects that are "similar" to one another, thus legitimizing the treatment of all the objects collectively as one group. Similarity is expressed in terms of a distance function, which is typically, though not necessarily, a metric. In other words, for each pair of data objects p_1, p_2, the distance $D(p_1, p_2)$ is known. In addition to a distance function, there is a separate "quality" function that measures the "goodness" of a cluster. One example of a quality function is the centroid distance, i.e., the average over $\{D(p_1, c) | p_1 \in Cl\}$, where c is the centroid of all the objects in cluster Cl. Another example is the diameter, i.e., the maximum over $\{D(p_1, p_2) | p_1, p_2 \in Cl\}$.

Even though similarity between objects and goodness of clusters can be defined, it is much harder to define "similar enough" and "good enough". The fundamental question here is: how many *natural* clusters there are in the given dataset. The answer to this question are typically highly subjective and remains an open issue in cluster analysis [KR90]. Existing clustering algorithms deal with this issue in one of two ways.

7.1 Overview of Existing Algorithms

7.1.1 Statistical Methods

The first way is to avoid answering the question entirely by giving a complete clustering of the dataset. That is, if there are n objects in the dataset, a clustering algorithm of this type specifies how to group the objects in 1, 2, $\ldots, n$ clusters. Clustering algorithms of this type are called *hierarchical* methods. They are either agglomerative (i.e., "bottom-up" in computer science jargon), or divisive (i.e., "top-down"). Given n objects to be clustered, an agglomerative method begins with n clusters (i.e., all objects are apart). In each step, based on the distance and quality functions, it chooses two clusters to merge. This process continues until it puts all objects into one group. Conversely, a divisive method begins by putting all objects in one cluster. In each step, it picks a cluster to split into two. This process continues until it produces n clusters. While hierarchical methods have been successfully applied to many biological applications (e.g., for producing taxonomies of animals and plants, and classification of diseases [KR90]), they are well known to suffer from the weakness that they can never undo what was done previously. Once an agglomerative method merges two objects, these objects are always in one cluster. And once a divisive method separates two objects, these objects are never re-grouped into the same cluster. More importantly, for large datasets, producing all n clusters is excessive and computationally prohibitive.

The second way to answer the question of how many natural clusters there are in the dataset is to ask the human user – not the clustering algorithm – to determine that number and to feed the number as input to the algorithm. Given the number, denoted as k, a *partitioning* clustering method tries to find the best k partitions of the n objects, i.e., each object is assigned to exactly one group. [4] However, the task of finding the best k partitions amounts to solving a nonconvex discrete optimization problem. Exhaustive enumeration of all partitions appears to be the only way to find the global optimal solution. Thus, the development of partitioning methods focuses on heuristics that try to strike as good a balance as possible between efficiency and finding solutions close to the global optimum. The k-means and k-medoids algorithms are well-known examples of partitioning methods. They have found many successful application areas, including social studies (e.g., for classification of statistical findings), manufacturing (e.g., garment) and chemical analysis.

7.1.2 Databases Methods

In recent years, some database researchers have re-visited the clustering problem. For them, there is the additional focus that the datasets may be a lot larger than the typical sizes used for statistical clustering (i.e., hundreds of thousands, if not millions, versus only hundreds or thousands). Furthermore, because the data may be mainly disk-resident, there is also the emphasis of minimizing I/O cost.

Based on randomized search, CLARANS can be viewed as an extension to the k-medoids algorithm [NH94, KR90]. It is highly tunable, depending on how much CPU time the user can afford. Focusing techniques based on spatial access methods (e.g., R* trees, Voronoi diagrams) are later developed to reduce the I/O cost required by CLARANS [EKX95]. By employing a balanced tree structure called CF tree, BIRCH makes explicit and takes full advantage of the amount of available buffering space [ZRL96]. A single scan of the dataset gives a basic clustering, and additional scans can be used to improve the quality further. Relying on the parameters of the size of the neighborhood and the minimum number of data points in the neighborhood, DBSCAN connects regions of sufficiently high densities into clusters [EKXS96]. As such, it does a better job finding elongated clusters than most of the algorithms mentioned above. It uses an R* tree to achieve good performance. Finally, STING is

[4] There are a few methods that tolerate a limited degree of overlap between clusters. See [KR90] for more details.

a hierarchical cell structure that stores statistical information (e.g., density) about the objects in the cells [WYM97]. Thus, with only one scan of the dataset, clustering can be achieved by using the stored information but without recourse to the individual objects.

7.1.3 Machine Learning Methods

There are a few clustering methods developed in the machine learning community. They are mostly probability-based approaches [Fis87]. And typically, they make the assumption that the probability distributions on different attributes are independent of each other. In practice, this is often too strong an assumption, because correlation may exist between attributes. In fact, as far as data reduction is concerned, correlation is exactly what is being searched for.

7.2 Distance-Only Data

As discussed above, all clustering methods require the specification of a distance function. Distance-only datasets, thus, pose no additional problem to clustering methods. Many clustering methods (e.g., k-medoids, CLARANS) can even handle non-metric distance functions.

In our evaluation of all data reduction methods, we measure (i) the accuracy and space/time tradeoff, and check whether (ii) progressive resolution refinement and (iii) incremental maintenance are supported. As far as clustering methods are concerned, the first criterion is the key. Once clustering methods are considered to be applicable and provide good accuracy, the other two criteria of progressive resolution refinement and incremental maintenance are automatic. For example, CLARANS, BIRCH and DBSCAN all provide various parameters for tuning and incremental maintenance. Thus, in the sequel, we only focus on the first criterion.

7.3 Multi-Dimensional Data

7.3.1 Ordered and Unordered Attributes

1. *Ordered:*
   For ordered datasets, clustering algorithms should work well. This is because the underlying order provides a natural distance function for the clustering algorithms to use. For instance, if age is the ordered attribute under consideration, then the distance between A and B is the difference between the ages of A and B.

2. *Unordered (Flat/Hierarchical):*
   For flat datasets, clustering algorithms do not work. This is because clustering algorithms require the existence of a distance function. If equality is the only meaningful comparison operation for the dataset, there is not enough discrimination of the objects for the algorithms to reason with. Consequently, there is only one trivial clustering structure: the equivalent classes of the objects, i.e., objects are in the same cluster if and only if they are equal to each other.

   For hierarchical datasets, the answer to the question of whether clustering algorithms work well is not as clear-cut as in the ORDERED and FLAT cases. On the one hand, the underlying domain is unordered and provides no distance function for the clustering algorithms. On the other hand, the structure of the hierarchy can be used to provide candidate distance functions. For example, the distance between two objects p_1 and p_2 can be defined by the path length between p_1 and anc, and the path length between p_2 and anc, where anc is the smallest common ancestor of p_1 and p_2, and an ancestor is the smallest if it is farthest away from the root. For some applications, candidate distance functions of this nature provide reasonable clustering quality; for others, they do not.

7.3.2 Sparse Data

For sparse datasets, clustering algorithms should work well. The sparsity is translated to discrimination between objects. Clustering algorithms should be the most effective when the sparsity is localized corresponding to distinct clustering structures. In this case, clustering algorithms that can be tuned are the most preferred because the great discrimination between some objects renders a coarse-grained, but efficient, analysis to be sufficient.

For dense datasets, the embedded clustering structures are less distinct and crisp. Clustering algorithms still work, but their effectiveness is weakened. One of the reasons is that most clustering algorithms produce groups that do not overlap. For DENSE datasets, this requirement is restrictive. Algorithms that allow overlap perform better under this circumstance.

7.3.3 Skewed Data

For datasets that are skewed in the value domain, clustering algorithms should work very well. This situation is very similar to the localized sparsity scenario discussed above. As for datasets that are skewed in the frequency domain, clustering algorithms are not affected. This is because all data objects having the same attribute value behave identically as far as clustering is concerned. Thus, it is sufficient to pick one representative per attribute value to participate in the clustering.

7.3.4 High-Dimensional Data

For high dimensional datasets, the situation for clustering algorithms is mixed. First, from an effectiveness point of view, the algorithms are not affected by the dimensionality – so long as the distance function has already captured the relationships that may exist between the dimensions. From an efficieny standpoint, in theory, a larger number of dimensions only implies a larger cost in computing the distance function. Thus, clustering algorithms should scale linearly with the number of dimensions. However, in practice, the situation is not as rosy, particularly for those algorithms that rely on various kinds of indexing to facilitate processing. For instance, for algorithms relying on trees (e.g., BIRCH [ZRL96] and DBSCAN [EKXS96]) the $O(logn)$ factor degrades to $O(n)$ as the dimensionality increases. Similarly, for algorithms using a grid structure (e.g., STING [WYM97]), processing is exponential with respect to the number of dimensions.

8 Index Trees

8.1 Descriptions and References

Index trees of various types are widely used to organize and access large data sets. Typically they are used to speed up selection queries on one-dimensional data sets ordered on a single key attribute. *B+-trees* are the most common and significant form of index tree for disk-resident one-dimensional data [BM72, Com79]. For data sets of higher dimension (i.e., those organized and accessed on the basis of values of two or more attributes in combination), a variety of other types of disk-based index trees have been developed and used. The most common example is the R-tree [Gut84] and its variants: the R*-tree [BKSS90] and R+-tree [SRF87]. Other multidimensional search trees include quadtrees [FB74], k-D-B-trees [Rob81], hB-trees [LS90], and TV-trees [LJF94]. Multidimensional data can also be transformed into unidimensional data using a space-filling curve [Jag90]; after transformation, a

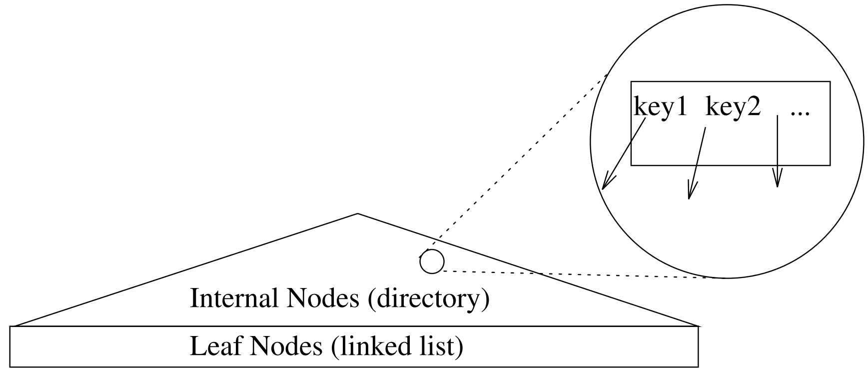

Figure 5: Sketch of a database index tree.

B+-tree can be used to index the resulting unidimensional data. A survey of multidimensional indexes is given by Gaede and Gunther [GG97].

8.2 A Generalized Picture of Index Trees

The canonical rough picture of a database index tree appears in Figure 5. It is typically a balanced tree, with high fanout. The internal nodes are used as a directory. The leaf nodes contain pointers to the actual data, and are stored as a linked list to allow for partial or complete scanning.

Within each internal node is a series of keys and pointers. In the typical application of index trees, the keys are used to guide tree traversal for finding all data items satisfying a selection predicate q. Traversal starts at the root node, and for each pointer on the node, if the associated key is found to be *consistent* with q — i.e., the key does not rule out the possibility that data stored below the pointer may match q — then traversal continues in the subtree below the pointer. This is repeated recursively down all consistent subtrees until all the matching data are found.

In B+-trees, queries are in the form of range predicates (e.g., "find all i such that $c_1 \leq i \leq c_2$"), and keys logically delineate a range in which the data below a pointer is contained. If the query range and a pointer's key range overlap, then the two are consistent and the pointer is traversed. In R-trees, queries are in the form of region predicates (e.g., "find all i such that (x_1, y_1, x_2, y_2) overlaps i"), and keys delineate the bounding box in which the data below a pointer is contained. If the query region and the pointer's key box overlap, the pointer is traversed.

Note that in the above description the only restriction on a key is that it must logically describe the set of data stored below it, so that the consistency check does not miss any valid data. In B+-trees and R-trees, keys are essentially "containment" predicates: they describe a contiguous region in which all the data below a pointer are contained. Containment predicates are not the only possible key constructs, however. For example, the key "purple, cardinality = 516" is perfectly acceptable, indicating that there are 516 data items below the pointer, all of which are all purple. In general, keys on a node may "overlap", i.e., two keys on the same node may hold simultaneously for some data item in the data set.

In essence, a database index tree is *a hierarchy of clusterings of a dataset, in which each cluster has a label that holds for the data contained in the cluster*. The grouping of data into clusters may be controlled by a variety of tree insertion and node splitting algorithms. The main variations among index trees are the way they represent keys, and the algorithms for insertion and node splitting. Although index trees have typically been used for accelerating selection queries, their structure is amenable for use in data reduction, as we shall see below.

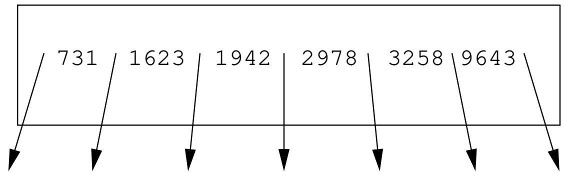

Figure 6: The root of a B+-tree

8.3 Data vs. Space Partitioning

Multidimensional index trees recursively subdivide a underlying k-dimensional space. The root node corresponds to the entire space. Each internal node corresponds to a portion of the space of its parent, and that portion is further subdivided among the children of the node. The leaf nodes consist of pointers to individual records (or the records themselves) that lie in the region of k-space corresponding to the node.

Data partitioning index trees divide the space based on the specific records or groups of records that are loaded or inserted into the tree. Examples of data partitioning trees include R-trees, TV-trees and hB-trees. *Space partitioning* trees divide the space along predetermined lines of division that are independent of the particular records represented in the trees (e.g., recursive binary splitting of the attribute ranges in each dimension). Examples of space partitioning trees include the various versions of quad trees, and k-D-B-trees. For both data and space partitioning trees, the splitting is propagated to a sufficiently deep level in each part of the tree that the leaf nodes can hold all the data or pointers to data assigned to them. Most space-parititoning trees are not balanced, which renders them less useful for disk-based storage; typically they are mapped to another representation when saved to disk.

Though there are many variations of index trees for both one-dimensional and multi-dimensional data, they all have shared properties. The Generalized Search Tree (GiST) [HNP95] is a template index tree that provides a common basis for describing and easily implementing a variety of index trees. In our discussion here, we focus on properties that tend to be shared by many if not all index tree variants. For one-dimensional data, we assume B+-trees as the prototypical index tree; for multidimensional data, we choose R-trees (or the very similar R*-trees).

8.4 Using Index Trees for Representing Aggregate or Summary Data

Index trees are used primarily for their benefits in organizing and supporting access to data. However, it is also possible to make direct use of the nodes of index trees to obtain and exploit aggregate or summary information about a data set; this has been observed by numerous researchers and implemented in some database products [ACD+88, Ant93b, Aok97]. The information stored in an index node can be used for such purposes as providing approximate answers to queries or making choices in query optimization.

As an example, consider looking only at the upper levels of an existing hierarchical index tree. They reveal a great deal of distributional information about the data. Typically, assuming branching factors found in practice in modern database systems, the root index node alone provides information equivalent to an approximately equi-depth histogram of one hundred to two hundred buckets. To see this consider the small example pictured in Figure 6. Assume that a B+ tree contains 10,000 records with keys in the range 1 to 99,999. Assuming that the root node is as shown in Figure 6 we can conclude that the data in the tree can be approximated by an equi-depth histogram of seven buckets, each containing 10000/7 items, split at the values 1, 731, 1623, 1942, 2978, 3258, 9643.

More and more detailed aggregate and summary information can be obtained by examining lower and lower levels of the index tree, which involves reading more and more index tree nodes. In essence, an index tree can be thought of as a *hierarchical histogram*. This is complicated somewhat if keys are allowed to overlap as they are in R-trees; this becomes analogous to a hierarchical histogram in which the "buckets" overlap.

A great deal of distribution information can be obtained from an index tree without traversing it very deeply. During query processing, it is common for blocks corresponding to the uppermost nodes in the tree to be consistently found in main memory, while those at lower levels usually require one block read from disk each for access.

Note that much summary and aggregate information is available from index trees with no modification beyond how they are used to organize and access data. However, some minor extensions that involve only modest overhead can increase the accuracy and precision of the summary and aggregate information that can be extracted from the index tree structure:

- One can store a count with each pointer that represents the precise number of data records in all the descendents of the node pointed to. Trees with such counts are said to be *ranked* [Knu73]. Ranked trees with non-overlapping keys truly are hierarchical histograms, and allow for arbitrary refinement of buckets by traversing pointers. The advantages of ranking do not come without cost. The space requirement for the counters associated with each pointer can reduce tree fanout by a significant factor, and maintenance of counters on a complete path from root to leaf is required on each record insertion or deletion. The overhead of handling insertion and deletion can be ameliorated by using *pseudo-ranking* [Ant93b], which allows counters to diverge a certain amount from the accurate values.

- While counters are the most natural "decoration" that one can add to index entries, there is no reason one can not store additional statistics in the entries, though additional statistics can consume space and further reduce the tree fanout. Generalized Search Trees [HNP95] support arbitrary keys of this nature, and Aoki's extensions to them [Aok97] extend the idea of psuedo-ranking to support extensible "divergence control" for arbitrary statistics.

8.5 Indexes and Histograms

It is asserted above that indexes can be viewed as hierarchical histograms, but not all flavors of histograms can be conveniently supported in index trees. Typically, index trees are balanced, meaning they are a hierarchy of roughly equi-depth histograms. This may or may not be appropriate for data reduction. In order to coerce index trees to serve as non-equi-depth histograms, one must tolerate index trees that are unbalanced either in height or node occupancy.

8.6 Distance-Only Data

Index trees provide no special advantage over other methods for dealing with distance-only data. Distance-only data can be stored using a multidimensional index tree only after the data is converted to positional data by embedding the points in a space of sufficiently high dimension.

8.7 Multi-Dimensional Data

8.7.1 Ordered and Unordered Attributes

Multidimensional index trees rely on the ordering of the attribute values in each dimension. Unordered attribute domains must have some ordering (perhaps an arbitrary one) imposed on them before they can be represented in a multidimensional index tree structure.

1. *Ordered:*

   Given an ordered attribute domain, the domain is usually normalized or mapped (preserving order) onto a canonical range, such as the unit interval. Some types of index trees rely on recursive binary splitting of the unit interval in order to keep the occupancy of each multi-dimensional sub-range of the space below some prespecified limit (e.g., reflecting the capacity of a single page of storage). In a manner similar to that used for B+-trees, occupancy information (either exact or approximate) can be associated with each pointer to a hyperrectangular area in the multi-dimensional index tree structure, see e.g., [WYM97].

2. *Unordered (Flat/Hierarchical)*

   The Generalized Search Tree was designed expressly to handle "flat" or "encapsulated" data. In GiSTs, user-defined operations may be implemented to choose an insertion location for new data, to partition data when nodes fill, and to label pointers with keys. A domain expert need not require ordering or hierarchical structure of the data *per se* to implement these operations, and the index itself can remain oblivious to the domain semantics. Moreover, it may be possible for the tree to automatically organize itself based on observing *queries*, to see which data items are often fetched together. Clustering such items into subtrees increases the efficiency of selection, and (more interestingly for our purposes here) *provides a query-driven notion of data reduction* that requires no semantic understanding of the data set. The application of GiSTs to such encapsulated flat data is an active area of research, but as of yet the results are preliminary. In the remainder of this section we focus on the more traditional multi-dimensional search trees, whose properties are currently better understood.

   Hierarchical relationships of values in each dimension of a multi-dimensional space are not conventionally represented within index trees. However, for index trees in which split points can be selected flexibly, placing the split points at major breaks in the hierarchy of values would have the advantage of tending to localize in storage elements that are more closely related in the hierarchy.

8.7.2 Sparse Data

Index trees generally handle sparse data well, in that the structure of a tree for a specific set of data adapts automatically to the distributional characteristics of the data. In some cases, problems arise if two or more attributes are so highly correlated that the effective dimensionality of the embedding space is reduced.

8.7.3 Skewed Data

As with sparse data, index trees generally adapt well to skewed data by either allowing deeper development of the tree in locally dense regions of the space, or by placing split boundaries in the dense areas to retain some balance in the populations associated with various tree nodes.

8.7.4 High-Dimensional Data

Some index trees have been developed explicitly to address high dimensional problems (e.g., TV-trees [LJF94] and X-trees [BKK96]). The efficacy of these structures remains in doubt, especially in light of recent results on the hardness of indexing high-dimensional space [HKP97]. Most known successful approaches involve projecting (based on some heuristics) to a space of lower, more manageable dimensionality.

9 Sampling

The notion that a large set of data can be represented by a small random sample of the data elements goes back to the end of the nineteenth century and has led to the development of a large body of survey-sampling techniques [Coc77, SSW92, Sud76]. Over the past fifteen years, there has been increasing interest in the application of sampling ideas to database management systems (DBMS's). Some existing and proposed applications of sampling include the following.

- *Query optimization* Given a query in an object-relational DBMS, the query optimizer estimates the cost of alternative query execution plans and attempts to select a low-cost plan. Current optimizers estimate costs based on summary statistics about the base relations; these statistics are stored in the system catalog. Specifically, the catalog statistics are used to estimate the sizes of intermediate query results via "selectivity formulas," and the resulting "selectivity estimates" are then substituted into cost formulas to yield the final cost estimates. Sampling is currently used in DBMS's such as DB2 V2 and Oracle 7 SQL Server to estimate a variety of catalog statistics from samples of the base relations, and there is ongoing research into sampling-based methods for estimating such key catalog statistics as quantiles and "column cardinality" (the number of distinct values of an attribute in a relation); see [GMP97, HNSS95, PIHS96]. Sampling is much less expensive than exact computation of catalog statistics from entire relations; such cost reduction is important since catalog statistics must be recomputed periodically as the database changes over time. Even when the catalog statistics are exact, selectivity estimates can be highly inaccurate because the selectivity formulas incorporate assumptions, such as lack of statistical correlation between attributes of a relation, that are in fact violated by the data. Unreliable selectivity estimates can lead to inaccurate cost estimates, which in turn can cause the optimizer to select an expensive query execution plan. In an effort to avoid these problems, a number of researchers have considered approaches in which selectivities and costs are estimated directly from a sample; see, for example, [GGMS96, HNSS96, HS92, HS95, HOD91, LNS90, LNSS93, NS90]. Several authors have outlined complete sampling-based approaches to query optimization [Ant93a, SBM93, Wil91].

- *Parallel processing of queries* Balancing the workload between processors is a critical objective of any parallel query-processing algorithm. Typically, records are assigned to processors based on the attribute values of the records. The goal is to determine a rule that assigns approximately the same number of records to each processor. Sampling can be used to estimate the distribution of attribute values and hence obtain good assignment rules. The parallel join-algorithms in [DNSS92] and the algorithms for efficient loading of parallel grid files in [LRS93], for example, use sampling in this manner.

- *Support for auditing* Various types of auditing require retrieval of a random sample of the records in a database or, in relational DBMS's, a random sample from the tuples in the output relation of a query. Examples of auditing applications include financial records auditing, fissile materials auditing, statistical quality control, and epidemiological studies. Other applications, such as market research and secure statistical DBMS's also require retrieval of random record sets; see [Olk93, Section 1.6] for further examples and references. Olken [Olk93, Section 1.5.2] has made the case that the most efficient approach to obtaining a random sample of records is to incorporate sampling into the DBMS, thereby avoiding both unnecessary record fetches and the overhead of passing data across the application/DBMS interface. Techniques for obtaining simple random samples (SRS's) from databases are developed in [Ant92, OR86, OR89, OR93, ORX90].

- *Approximate answers to aggregation queries* The answer to an aggregation query consists of a small set of summary statistics, such as COUNT, AVERAGE, or MAXIMUM, that is computed from a specified set of records. Sampling provides a means of obtaining quick, approximate answers to a variety of aggregation queries. Sampling techniques for aggregation queries in object-relational DBMS's have been studied in [HOD91, HOT88, HOT89, ODT+91]. These techniques also have been studied in the context of online-aggregation systems [Haa96, Haa97, HHW97]. In such a system, the user can observe the progress of an aggregation query and control execution on the fly; the records observed so far are viewed as a random sample of the set of all records in the database. Online application processing (OLAP) systems compute many aggregate statistics of interest, and several OLAP products now support sampling-based estimation; see, for example, [Inf97].

- *Data mining* Data mining algorithms typically are applied to extremely large data sets. Several authors have suggested that certain data mining algorithms can yield satisfactory approximate results when applied to a random sample of the data [Cat92, JL96, KM94].

There are many different types of samples. If a sample of n records is drawn from a set of N records ($N > n$) such that all possible samples of size n are equally likely, then the sample is a *simple random sample without replacement* (SRSWOR) of size n. If records are sampled randomly and uniformly from the record set, but a sampled record is replaced before the next sample is drawn, then we obtain a *simple random sample with replacement* (SRSWR). Sometimes the record set is grouped into M mutually disjoint "clusters" and a SRS of m clusters ($m < M$) is obtained. In this case the selected records form a *cluster* sample. For example, records in a database system usually are retrieved a page at a time, and the records obtained by retrieving a SRSWOR of pages form a cluster sample. A related type of sample is obtained by partitioning the data set into mutually disjoint "strata" and then obtaining a SRS from each stratum. The selected records then form a *stratified* sample. In a "shared-nothing" parallel DBMS, for example, a stratum might correspond to the records stored at a specified processing node; see [SN92] for a discussion of why, in parallel DBMS's, stratified sampling usually is preferable to simple random sampling. Other types of samples abound [Coc77, DC72, SSW92, Sud76]; we focus on simple, cluster, and stratified samples since these are the most common types of reduced data sets found in DBMS's.

Sampling is well-suited to the progressive refinement of a reduced data set: to "refine" the data set further, simply take more samples. Note, however, that if the sample is a SRSWOR, then adding new records to the sample may require checking for duplicates, which can become expensive as the sample size becomes large. Of course, for purposes of data reduction one usually is interested in small samples, and the hashing method given in [EN82] can be used to update a small to medium sized SRSWOR efficiently.

The ease of producing and maintaining a random sample depends on the available *sampling frame*, that is, the available mechanism for randomly accessing elements of the record set. For example, if records are stored in a B^+-tree or a hash file, then SRS's can be obtained using the techniques described in [Ant92, OR89] or [ORX90], respectively. In many file systems, pages of records are stored in contiguous blocks called *extents*, and a main memory data structure called an *extent map* provides access to the extents and the pages within the extents. This data structure can be exploited to efficiently obtain a SRSWR of pages by repeatedly generating a random number between 1 and the number of pages and then using the extent map to retrieve the corresponding page [DNSS92]. If a SRS of records (rather than pages) is required, then extent-map sampling can be combined with an *acceptance/rejection* (A/R) technique as described, for example, in [Olk93]. The idea behind A/R sampling is to accept a sampled page with a probability equal to the number of records on the page divided by the maximum number of records on a page; otherwise the page is rejected. If the page is accepted, then a record is selected from the page randomly and uniformly. Suppose that there are M pages with n_m records on page p_m $(1 \le m \le M)$ and consider a specified record r on a specified page p_m. Then, at each sampling step,

$P\{r$ included in sample$\}$

$$\begin{aligned}
&= P\{\text{page } p_m \text{ selected}\} \\
&\quad \times P\{\text{page } p_m \text{ accepted} \mid \text{page } p_m \text{ selected}\} \\
&\quad \times P\{\text{record } r \text{ selected} \mid \text{page } p_m \text{ selected and accepted}\} \\
&= \frac{1}{m} \times \frac{n_m}{n^*} \times \frac{1}{n_m} \\
&= \frac{1}{mn^*},
\end{aligned}$$

where $n^* = \max_{1 \le m \le M} n_m$. Thus, all of the records in the file have the same inclusion probability (namely $1/mn^*$) at each sampling step. Since successive records are selected independently, we obtain a SRSWR. The efficiency of the algorithm can be improved by first generating the (random) number of records to be selected from each page and then retrieving only those pages from which at least one record is to be selected. Moreover, the algorithm can easily be modified to produce a SRSWOR. The A/R approach described above lies at the heart of most algorithms for producing samples from complex data structures and from output relations in object-relational DBMS's. For many sampling algorithms, the cost of obtaining a sample is proportional to the size of the sample, and not the size of the database; this is in contrast to other data reduction techniques that require at least one complete pass through the data. (Sometimes, as in the case of histograms [GMP97, PIHS96], sampling can be combined with another data reduction technique, yielding an approximate reduction of the data that is relatively cheap to obtain.) It is also relatively inexpensive to update a sample as the underlying data changes; see [GMP97, OR92] for some updating methods.

The adequacy of sampling as a data-reduction technique depends crucially on how the sample is to be used. We focus on perhaps the most common use of a sample: estimation of the answer to an aggregation query. The simplest example of such an estimation problem is as follows: given a set $R = \{ r_1, r_2, \ldots, r_N \}$ of N records, estimate the quantity $\theta(f) = (1/N)\sum_{i=1}^{N} f(r_i)$ based on a SRSWR of size $n < N$, where f is a specified real-valued function. (If n is sufficiently small with respect to N, as is typically the case, then the distinction between an SRSWR and a SRSWOR is unimportant.) An unbiased estimate $\widehat{\theta}_n(f)$ of the unknown quantity $\theta(f)$ is obtained by averaging the function f over the n records in the sample. Denote by z_p the $(p+1)/2$ quantile of the standard normal distribution and denote by $\sigma^2(f)$ the variance of the function f over all of the records in the database:

$$\sigma^2(f) = \frac{1}{N}\sum_{i=1}^{N}(f(r_i) - \theta(f))^2.$$

The standard Central Limit Theorem asserts that when n is large the distribution of the estimator $\widehat{\theta}_n(f)$ is approximately normal with mean $\theta(f)$ and variance $\sigma^2(f)/n$. It follows that, for a sample size of

$$n = \left(\frac{z_p\sigma(f)}{\theta(f)\epsilon}\right)^2 \tag{27}$$

records, the estimator $\widehat{\theta}_n(f)$ estimates $\theta(f)$ to within a factor of $1 \pm \epsilon$ with probability approximately equal to $100p\%$. This approximate result is valid when ϵ is small. Conservative sample-size formulas can be derived from inequalities developed by Hoeffding [Hoe63]. For example, suppose that $l \le f(r_i) \le u$

for $1 \leq i \leq N$ and set

$$w_p^2 = \frac{1}{2} \ln \left(\frac{2}{1-p} \right)$$

for $0 < p < 1$. Then a sample size of

$$n = \left(\frac{w_p(u-l)}{\theta(f)\epsilon} \right)^2 \tag{28}$$

is sufficient to ensure that $\widehat{\theta}_n(f)$ estimates $\theta(f)$ to within a factor of $1 \pm \epsilon$ with probability greater than or equal to p; there are no restrictions on ϵ save that $\epsilon > 0$. The power of sampling-based estimation derives from the fact that the sample-size formulas do not depend explicitly on the size N of the database so that, if that data are well-behaved, the sampling fraction required to achieve reasonable precision can be extremely small when N is very large. For example, given a set of $N = 10^8$ records such that $f(r_i) = i$ for $1 \leq i \leq 10^8$, it follows from 27 that a sample size of approximately 220 records (0.0002% of the database) is sufficient to ensure that, with 99% probability, $\widehat{\theta}_n(f)$ estimates $\theta(f)$ to within 10%. Similarly, it follows from 28 that a sample size of 1060 records is sufficient to ensure that, with probability at least 99%, $\widehat{\theta}_n(f)$ estimates $\theta(f)$ to within 10%. In practice, either two-phase or sequential procedures can be used to estimate $\sigma^2(f)$ and control the sample size automatically; see, for example, [HS92, HOD91]. Similarly, *a priori* bounds on the function f often are available in practice, so that 28 can be used to determine the required sample size. The above calculations also can be turned around to yield estimates of the precision of $\widehat{\theta}_n$ for a specified sample size n. For example, fix n (with n relatively large) and p, and denote by $S_n^2(f)$ the variance of the function f over the records in the sample. It follows from 27 that the random interval

$$I_n = \left[\widehat{\theta}_n(f) - \frac{z_p S_n(f)}{\sqrt{n}}, \widehat{\theta}_n(f) + \frac{z_p S_n(f)}{\sqrt{n}} \right]$$

contains the point $\theta(f)$ with probability approximately equal to p. The interval I_n is called a (large sample) $100p\%$ *confidence interval* for $\theta(f)$; the width of the interval indicates the precision of the estimator $\widehat{\theta}_n$. In a similar manner, a conservative confidence interval can be derived from 28.

The basic methodology outlined above has been extended in several different directions.

- For SRS's, central limit theorems (and hence sample-size formulas and confidence intervals) have been established for large classes of summary statistics other than population averages, for example, population moments [Cra46, Chapter 28], maximum likelihood estimators [Cra46, Chapter 33], and U-statistics [Hoe48]. Moreover, the "delta method" can be used to derive new central limit theorems from old. The idea is that if $\widehat{\theta}_n$ estimates θ and the distribution of $\widehat{\theta}_n$ is approximately normal with mean θ and standard deviation σ, then the distribution of $f(\widehat{\theta}_n)$ is approximately normal with mean $f(\theta)$ and standard deviation $\sigma f'(\theta)$ for any function f that is continuously differentiable and positive at the point θ. With appropriate modifications, the delta method extends to the case in which $\widehat{\theta}_n$ is a k-tuple of estimators for some $k > 1$.

- For samples with a more complex structure, such as cluster or stratified samples, point estimators and confidence intervals are available for population sums of the form $\mu(f) = \sum_{i=1}^{N} f(r_i)$ and (via the delta method) smooth functions of such sums, e.g., ratios, averages, and central moments such as variance and skewness. The idea is as follows. Let R be a collection of records as above and S be a sample of records from R (not necessarily simple). Suppose that the inclusion probability π_i for record r_i is known *a priori* for each i. Then it is not hard to show that the estimator

$$\widehat{\mu}_n(f) = \sum_{r_i \in S} \frac{f(r_i)}{\pi_i}$$

is unbiased for $\mu(f)$, provided that each π_i is positive. See [SSW92] for a comprehensive discussion of such "Horvitz-Thompson" estimators and their associated confidence intervals.

- Estimation methods also are available when the summary statistic of interest is computed from the tuples in the output relation formed by executing a relational query over a set of base relations. One method is to materialize a SRS of the tuples in the output relation (using A/R techniques as described above and in [Olk93]) and then compute the estimate of the summary statistic. An alternative method is to maintain a SRS from each base relation, execute the query on the sample base-relations to obtain a sample version of the output relation, and then compute the summary statistic over the tuples in the sample version of the output relation. Two advantages of the latter approach are that it is easier to obtain a SRS of each base relation than to obtain a SRS of the output relation using A/R sampling, and the base-relation samples can be reused for subsequent aggregation queries. A potential difficulty is that in many cases, such as when the output relation is a join of two or more base relations, the tuples in the sample version of the output relation are not mutually independent (as in a SRS from the output relation). This difficulty has been at least partially overcome: formulas for estimators, large sample confidence intervals, and conservative confidence intervals corresponding to a variety of complex aggregation queries can be found in [Haa96, Haa97]. These formulas explicitly take into account the statistical dependence between the tuples in the sample version of the output relation. Procedures that exploit existing indexes on the base relations also are developed in order to handle the case in which straightforward execution of a query on the sample base-relations yields a sample version of the output relation that contains too few tuples.

The summary statistics discussed above can be estimated accurately from a small sample. Other summary statistics, however, are inherently difficult to estimate. Roughly speaking, these are "needle-in-a-haystack" type statistics, which cannot be estimated accurately unless one or more members of a very small subset of the records are included in the sample; the probabilities of such inclusion typically are extremely small. An example of such a statistic is $\max_{1\le i\le N} f(r_i)$, where $R = \{ r_1, r_2, \dots, r_N \}$ is a set of records as before and f is a real-valued function. If, say, $f(r_1) \gg f(r_i)$ for $i > 1$, then an estimate of the maximum function value will be extremely inaccurate unless record r_1 is included in the sample. To provide acceptably accurate estimates of such summary statistics, a hybrid approach is needed in which the sample is supplemented by additional information. Some examples of the hybrid approach are given in subsequent sections; development of hybrid methods is an active area of research.

We conclude by discussing the accuracy of sampling methods in the context of some specific types of data.

9.1 Distance-Only Data

Assuming that an efficient sampling frame is available, there is no particular difficulty in producing and maintaining a sample of distance-only data elements. The applicability of sampling for estimation of summary statistics, however, depends heavily on the type of statistic desired. Suppose, for example, that the statistic of interest is the average distance θ between data elements in the population and that a SRSWR $\{ X_1, X_2, \dots, X_n \}$ of $n > 1$ data elements is available. Then θ can be estimated by

$$\widehat{\theta}_n = \binom{n}{2}^{-1} \sum_{i=1}^{n} \sum_{j=i+1}^{n} d(X_i, X_j),$$

where d is the distance function. The estimator $\widehat{\theta}_n$ is a U-statistic [Hoe48], and therefore is unbiased and consistent for θ. (An estimator $\widehat{\theta}_n$ is consistent for a parameter θ if $\widehat{\theta}_n$ converges to θ as n

increases.) Moreover, there is a well-developed methodology for obtaining confidence intervals for $\widehat{\theta}_n$. Other summary statistics such as the average distance of a data element to its nearest neighbor can be much harder to estimate. If the data is partitioned into clusters and entire clusters can be sampled, then summary statistics defined in terms of clusters (such as the average distance between the points in a cluster) can be estimated using the methods described previously.

9.2 Multi-Dimensional Data

One strength of sampling as a data reduction technique is that multidimensional data, especially with statistical correlation between the attributes, can be handled gracefully. For example, the storage requirement for a d-dimensional histogram typically increases exponentially in d, while the corresponding storage requirement for a fixed-size sample increases only linearly. We consider various types of multidimensional data below.

9.2.1 Ordered and Unordered Attributes

As indicated above, one common use of a sample is to estimate some aggregate quantity that is computed by applying a real-valued function f to individual records or k-tuples of records. Thus, there is not much difference between ordered and unordered attributes in terms of estimation. On the other hand, whether or not the attributes are ordered can influence the way in which the data is stored, and hence the sampling frame. For example, data values having a linear ordering can be stored in a B^+ tree or a ranked B^+ tree, so that SRS's can be obtained using the methods in [Ant92, OR89].

9.2.2 Sparse Data

Depending on how the data is stored, sparseness of data may or may not have a detrimental effect on sample-based estimates. If, for example, there is an index on the data elements, then it is straightforward to compute and maintain a sample of these elements, and estimates can be computed using standard techniques. If, however, a sample must be obtained by randomly selecting attribute values, testing to see if there are one or more data points having those attribute values, and then retrieving such a data point if it exists (perhaps with an A/R step to ensure equal inclusion probabilities), then it is extremely expensive to form a sample. Typically, however, there will be data structures that permit efficient retrieval of data points, and this structure also can be used to obtain a sample.

Another form of sparseness occurs when the summary statistic of interest is computed over a very small qualifying subset of the records. (This is the needle-in-a-haystack problem again.) In this case, the sample needs to be augmented with additional information. For example, if there is a combined index on the attributes of interest, then additional samples from the qualifying subset can efficiently be obtained by sampling from the index. As another example, Haas and Swami [HS95] describe a method for estimating the selectivity of a join in which the sample is augmented with frequency counts for certain join-attribute values that are frequent in some relations and infrequent in other relations. Regression techniques [SSW92, Part II] can provide an effective means of combining information in the sample with other available information; see also [RKM90, Kuk93].

9.2.3 Skewed Data

Data that is skewed in frequency but not in value does not cause problems when a sample is used to estimate summary statistics that are sums, averages, or smooth functions of sums and averages. On the other hand, it can be extremely hard to estimate statistics such as the number of distinct values of a specified attribute when the data is skewed in frequency. Some distinct-value estimation procedures that can deal with moderate skew are discussed in [HNSS95, HS96]; a drawback of these procedures is that the sample size required for a specified degree of accuracy depends on the size of the data set.

Even sums or averages can be hard to estimate when the data is skewed in value. For example, consider a set of 10^6 records and a function f such that $f(r_1) = 10^9$ and $f(r_i) = 0$ for $1 < i \leq 10^6$, so that the average of the function f over all of the records is equal to 1000. Unless the sample contains record r_1, the usual estimate of the population average will be equal to zero. On the average, about 500,000 records must be sampled before r_1 is encountered. Even when the degree of skew is not as extreme, estimates of sums and averages can be highly variable and the actual probability that a confidence interval contains the population parameter of interest can be much less than the nominal probability. Several approaches have been developed in an attempt to handle this situation. One approach is to try and redefine the original estimation problem so that the summary statistic of interest is resistant to skew. For example, rather than trying to estimate the average of a function over a set of records, we can try to estimate the median value of the function instead. Alternatively, the sample can be supplemented with a small set of records having highly nonstandard function values. These values can be combined with the sample-based estimate in a manner similar to that in [HS95]. Finally, for data sets with moderate skew, "corrected" confidence intervals with improved coverage properties can in principle be computed using an extension (to the setting of discrete data values) of the "second-order pivotal transformations" discussed in [Gly82].

9.2.4 High-Dimensional Data

One potential difficulty caused by high-dimensional data is the large amount of space required to store a sample of a given size. If storage is limited, then the size of the sample may be too small to provide sufficiently accurate estimates. As mentioned previously, this problem occurs with other data reduction methods.

Another potential problem is that observations may be expensive to compute; that is, the function f that is to be applied to a record r_i might be expensive to evaluate if r_i is of extremely high dimension. For example, the "record" r_i might in fact be an entire document and f might require a complex pattern-matching operation as part of its evaluation. If it is possible to quickly rank a small set of records in approximate order of increasing value of f without actually evaluating f itself, then ranked set sampling techniques can be used to estimate averages, quantiles, and other summary statistics using many fewer function evaluations than are required by simple random sampling; see, for example, [DC72, SS88] and references therein. In a similar vein, Luo et al. [LSS97] provide estimation methods that require accurate evaluations of f on a small subset of the sample and cheap, inaccurate measures of f on the remainder of the sample.

10 Conclusions

Database technology, as a field, may have matured in contexts such as banking and payroll, where providing complete accuracy and consistency are central requirements. With the emphasis today on data warehousing and data analysis, there is a pressing need for quick approximate answers from very large data sets.

Data reduction is invaluable in this context, and we believe is going to be widely used in databases of the future. There already exist a rich variety of data reduction techniques, many of which have been described above, with different characteristics and different areas of applicability. A summary of our findings is given in Table 2.

Most techniques do best given a low-dimensional, roughly uniform, dense data set with all attributes ordered. (There are a few exceptions – clustering does not work too well on dense data, and some

Data Type	SVD	Wavelet	Regression	Log-Linear	Histogram	Clustering	Index Tree	Sampling
Distance Only	N	N	N	N	D	Y	M	Y
Unordered Flat	Y	N	N	Y	D	N	M	Y
Unordered Hierarchical	Y	M	N	Y	M	M	M	Y
Sparse	B	F	F	F	F	B	F	D
Skewed	F	F	B	F	F	F	F	D
High Dimensional	N	F	W	W	M	D	W	W
Secondary Metrics								
Progressive Resolution	Y	Y	Y	N	M	D	Y	Y
Incremental Computation	N	Y	M	N	M	M	Y	Y

Y = Yes ; N = No ; M = Maybe;
F = Fine ; B = Better ; W = Worse ;
D = Depends (on further specification, could be better or worse).

Table 2: Applicability of data reduction techniques to different types of data

techniques, such as log-linear, do not make any use of the ordering). We take this as the base case, and mark a technique "Fine" if it does approximately as well in a stress-case as in the base case. Most of the other other entries in the table are self-explanatory.

Entries marked "Depends" have a more complex dependence, and the reader is referred to the corresponding section above to get more details. For instance, clustering can give progressive resolution, if hierarchical clustering is used, but cannot if it is a one-level clustering. Entries marked "Maybe" are ones for which no definitive answer could be agreed upon, and are typically indicative of areas that could benefit from additional research.

Acknowledgments

The authors would like to thank David Lomet and Nick Koudas for their input on this article. We also acknowledge the support provided by AT&T in providing a forum to bring so many of us together for this venture.

Faloutsos is on leave from the University of Maryland, and Ioannidis is on leave from the University of Wisconsin. Faloutsos' work was partially funded by the National Science Foundation under grants No. EEC-94-02384, IRI-9205273 and IRI-9625428. Hellerstein's work was supported in part by NASA grant 1996-MTPE-00099, NSF grant IRI-9703972, and support from Informix and the California MICRO program. Ioannidis' work was supported in part by the National Science Foundation under Grants IRI-9700799 and IRI-9157368, and by grants from DEC, IBM, HP, AT&T, Oracle, and Informix. Ng's work was partially supported by NSERC Grants OGP0138055 and NCE IRIS-2 Grants HMI-5 and IC-5. Sevcik's work was supported in part by grants from the Centre for Advanced Studies, IBM Canada, and the Natural Sciences and Engineering Research Council of Canada.

References

[ACD+88] M. J. Anderson, R. L. Cole, W. S. Davidson, W. D. Lee, P. B. Passe, G. R. Ricard and L. W.

Youngren, Index Key Range Estimator. U. S. Patent 4,774,657, IBM Corp., Armonk, NY, Sep. 1988. Filed June 6, 1986.

[Agr90] A. Agresti. *Categorical Data Analysis.* Wiley-Interscience 1990.

[Ant92] G. Antoshenkov. Random sampling from pseudo-ranked B^+ trees. In *Proc. 19th Intl. Conf. Very Large Data Bases*, pages 375–382. Morgan Kaufmann, 1992.

[Ant93a] G. Antoshenkov. Dynamic query optimization in Rdb/VMS. In *Proc. Eleventh Intl. Conf. Data Engrg.*, pages 538–547. IEEE Computer Society Press, 1993.

[Ant93b] Gennady Antoshenkov. Query Processing in DEC Rdb: Major Issues and Future Challenges. *IEEE Data Engineering Bulletin* 16(4):42–45, 1993.

[Aok97] P. M. Aoki. Generalizing "Search" in Generalized Search Trees. *Proc. 14th Int'l Conf. on Data Engineering*, Orlando, FL, Feb. 1998. To appear.

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Comm ACM*, 18(9):509–517, September 1975.

[Ber92] Michael W. Berry. Large-scale sparse singular value computations. *The International Journal of Supercomputer Applications*, 6(1):13–49, Spring 1992.

[BFH75] Y. Bishop, S. Fienberg, and P. Holland. *Discrete Multivariate Analysis: Theory and Practice.* MIT Press, 1975.

[Bir63] M. Birch. Maximum Likelihood in Three-way Contingency Tables. *J. Roy. Statist. Soc.*, B25:220–233, 1963.

[BKK96] S. Berthold, D. Keim, and H. P. Kriegel. X-Tree: An Indexing Structure for High Dimensional Data. *Proc. 22nd VLDB*, pages 10–21, August 1996. Mumbai, India.

[BKSS90] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An Efficient and Robust Access Method For Points and Rectangles. In *Proc. ACM-SIGMOD International Conference on Management of Data*, pages 322–331, Atlantic City, May 1990.

[BM72] Rudolf Bayer and Edward M. McCreight. Organization and Maintenance of Large Ordered Indices. *Acta Informatica*, pages 173–189, January 1972.

[BS97] D. Barbará and M. Sullivan. Quasi-Cubes: A space-efficient way to support approximate multidimensional databases. Technical Report, Department of Information and Software Systems Engineering, George Mason University, 1997.

[Cat92] J. Catlett. Peepholing: Choosing attributes efficiently for megainduction. In *Proc. Ninth Intl. Work. Machine Learning*, pages 49–54. Morgan Kaufmann, 1992.

[Coc77] W. G. Cochran. *Sampling Techniques.* Wiley, New York, third edition, 1977.

[Com79] D. Comer. The Ubiquitous B-Tree. *Computing Surveys*, 11(2):121–137, June 1979.

[CR94] C.M. Chen and N. Roussopoulos. Adaptive Selectivity Estimation Using Query Feedback. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data, Minneapolis, Minnesota*, May 1994.

[Cra46] H. Cramér. *Mathematical Methods of Statistics.* Princeton University Press, 1946.

[Cra94] Richard E. Crandall. *Projects in Scientific Computation.* Springer-Verlag New York, Inc., 1994.

[Dau92] Ingrid Daubechies. *Ten Lectures on Wavelets.* Capital City Press, Montpelier, Vermont, 1992. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

[DC72] T. R. Dell and J. L. Clutter. Ranked set sampling theory with order statistics background. *Biometrics*, 28:545–555, 1972.

[DH73] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[DNSS92] D. DeWitt, J. F. Naughton, D. A. Schneider, and S. Seshadri. Practical skew handling algorithms for parallel joins. In *Proc. 19th Intl. Conf. Very Large Data Bases*, pages 27–40. Morgan Kaufmann, 1992.

[DS40] W. Deming and F. Stephan. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. *Annals Math. Stat.*, 11:427–444, 1940]

[Dum94] Susan T. Dumais. Latent semantic indexing (lsi) and trec-2. In D. K. Harman, editor, *The Second Text Retrieval Conference (TREC-2)*, pages 105–115, Gaithersburg, MD, March 1994. NIST. Special publication 500-215.

[EKX95] M. Ester, H.P. Kriegel and X. Xu. (1995) *Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification*, Proc. Fourth International Symposium on Large Spatial Databases.

[EKXS96] M. Ester, H.P. Kriegel, J. Sander and X. Xu. (1996) *A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, Proc. Second International Conference on Knowledge Discovery and Data Mining, pp. 226–231.

[EN82] J. Ernvall and O. Nevalainen. An algorithm for unbiased random sampling. *Comput. J.*, 25:45–47, 1982.

[Fal96] Christos Faloutsos. *Searching Multimedia Databases by Content.* Kluwer Academic Inc., 1996. ISBN 0-7923-9777-0.

[FD92] Peter W. Foltz and Susan T. Dumais. Personalized information delivery: an analysis of information filtering methods. *Comm. of ACM (CACM)*, 35(12):51–60, December 1992.

[Fie93] D.J. Field. Scale-invariance and self-similar 'wavelet' transforms: an analysis fo natural scenes and mammalian visual systems. In M. Farge, J.C.R. Hunt, and J.C. Vassilicos, editors, *Wavelets, Fractals, and Fourier Transforms*, pages 151–193. Clarendon Press, Oxford, 1993.

[FB74] R. A. Finkel and J. L. Bentley. Quad-Trees: A Data Structure For Retrieval On Composite Keys. *ACTA Informatica*, 4(1):1–9, 1974.

[Fis87] D. Fisher. (1987) *Acquisition via Incremental Conceptual Clustering*, Machine Learning, 2, 2.

[FL95] C. Faloutsos and K. Lin. FastMap: a Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. In*Proc. 1995 ACM SIGMOD Intl. Conf. Management of Data*, pages 163-174.

[GG97] V. Gaede and O. Gunther. Multidimensional Access Methods. *ACM Computing Surveys*, 1997. To appear.

[GGMS96] S. Ganguly, P. B. Gibbons, Y. Matias, and A. Silberschatz. Bifocal sampling for skew-resistant join size estimation. In *Proc. 1996 ACM SIGMOD Intl. Conf. Management of Data*, pages 271–281. ACM Press, 1996.

[Gly82] P. W. Glynn. Asymptotic theory for nonparametric confidence intervals. Technical Report 63, Department of Operations Research, Stanford University, Stanford, CA, 1982.

[GM96] Phillip Gibbons and Yossi Matias. Space efficient maintenance of top sellers list in large databases. Unpublished manuscript, Bell Labs, 1996.

[GMP97] Phillip B. Gibbons, Yossi Matias, and Viswanath Poosala. Fast incremental maintenance of approximate histograms. *Proc. of the 23rd Int. Conf. on Very Large Databases*, August 1997.

[Gut84] A. Guttman. R-Trees: A Dynamic Index Structure For Spatial Searching. In *Proc. ACM-SIGMOD International Conference on Management of Data*, pages 47–57, Boston, June 1984.

[Haa96] P. J. Haas. Hoeffding inequalities for join-selectivity estimation and online aggregation. IBM Research Report RJ 10040, IBM Almaden Research Center, San Jose, CA, 1996.

[Haa97] P. J. Haas. Large-sample and deterministic confidence intervals for online aggregation. In *Proc. Ninth Intl. Conf. Scientific and Statist. Database Management*, pages 51–63. IEEE Computer Society Press, 1997.

[HHW97] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *Proc. 1997 ACM SIGMOD Intl. Conf. Managment of Data*. ACM Press, 1997. To appear.

[HKP97] Joseph M. Hellerstein, Elias Koutsoupias, and Christos H. Papadimitriou. On the Analysis of Indexing Schemes. In *Proc. 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 249–256, Tucson, May 1997.

[HNP95] J. M. Hellerstein, J. F. Naughton, and A. Pfeffer. Generalized Search Trees for Database Systems (Extended Abstract). In *Proc. 21st International Conference on Very Large Data Bases*, Zurich, September 1995.

[HNSS95] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *Proc. 21st Intl. Conf. Very Large Data Bases*, pages 311–322. Morgan Kaufmann, 1995.

[HNSS96] P. J. Haas, J. F. Naughton, S. Seshadri, and A. N. Swami. Selectivity and cost estimation for joins based on random sampling. *J. Comput. System Sci.*, 52:550–569, 1996.

[HOD91] W. Hou, G. Ozsoyoglu, and E. Dogdu. Error-constrained COUNT query evaluation in relational databases. In *Proc. 1991 ACM SIGMOD Intl. Conf. Managment of Data*, pages 278–287. ACM Press, 1991.

[Hoe48] W. Hoeffding. A class of statistics with asymptotically normal distribution. *Ann. Math. Statist.*, 19:293–325, 1948.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, 58:13–30, 1963.

[HOT88] W. Hou, G. Ozsoyoglu, and B. Taneja. Statistical estimators for relational algebra expressions. In *Proc. Seventh ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Sys.*, pages 276–287. ACM Press, 1988.

[HOT89] W. Hou, G. Ozsoyoglu, and B. Taneja. Processing aggregate relational queries with hard time constraints. In *Proc. 1989 ACM SIGMOD Intl. Conf. Managment of Data*, pages 68–77. ACM Press, 1989.

[HS92] P. J. Haas and A. N. Swami. Sequential sampling procedures for query size estimation. In *Proc. 1992 ACM SIGMOD Intl. Conf. Managment of Data*, pages 1–11. ACM Press, 1992.

[HS95] P. J. Haas and A. N. Swami. Sampling-based selectivity estimation using augmented frequent value statistics. In *Proc. Eleventh Intl. Conf. Data Engrg.*, pages 522–531. IEEE Computer Society Press, 1995.

[HS96] P. J. Haas and L. Stokes. Estimating the number of classes in a finite population. IBM Research Report RJ 10025, IBM Almaden Research Center, San Jose, CA, 1996.

[IC93] Yannis Ioannidis and Stavros Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. *ACM TODS*, 1993.

[Inf97] Informix Corporation. *Technical Brief: Informix Metacube Explorer*, 1997. http://www.informix.com/informix/products/techbrfs/metacube.

[Ioa93] Yannis Ioannidis. Universality of serial histograms. *Proc. of the 19th Int. Conf. on Very Large Databases*, pages 256–267, December 1993.

[IP95a] Yannis Ioannidis and Viswanath Poosala. Balancing histogram optimality and practicality for query result size estimation. *Proc. of ACM SIGMOD Conf*, pages 233–244, May 1995.

[IP95b] Yannis Ioannidis and Viswanath Poosala. Histogram-based solutions to diverse database estimation problems. *IEEE Data Engineering Bulletin*, 18(3):10–18, December 1995.

[Jag90] H. V. Jagadish. Linear Clustering of Objects With Multiple Attributes. In *Proc. ACM-SIGMOD International Conference on Management of Data*, pages 332–342, Atlantic City, May 1990.

[JL96] G. H. John and P. Langley. Static versus dynamic sampling for data mining. In *Proc. Second Intl. Conf. Knowledge Discovery and Data Mining*, pages 367–370. AAAI Press, 1996.

[Jol86] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[KD80] P. M. Kroonenberg and J. De Leeuw. Principal Component Analysis of Three-Mode Data By Means of Alternating Least Squares Algorithms. *Psychometrika*, 45:69-97, 1980.

[KJF97] F. Korn, H.V. Jagadish and C. Faloutsos. Efficiently Supporting ad Hoc Queries in Large Datasets of Time Sequences. In *Proc. ACM-SIGMOD International Conference on Management of Data*, pages 289–300, Tucson, 1997.

[KM94] J. Kivinen and H. Mannila. The power of sampling in knowledge discovery. In *Proc. Thirteenth ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Sys.*, pages 77–85. ACM Press, 1994.

[Knu73] D. E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley Publishing Co., 1973.

[Koo80] R. P. Kooi. *The optimization of queries in relational databases*. PhD thesis, Case Western Reserver University, Sept 1980.

[KK69] H. Ku and S. Kullback. Approximating discrete probability distributions. *IEEE Trans. Inform. Theory*, IT-15:444–447, 1969.

[KR90] L. Kaufman and P.J. Rousseeuw. (1990) *Finding Groups in Data: an Introduction to Cluster Analysis*, John Wiley & Sons.

[Kuk93] A. Y. C. Kuk. A kernel method for estimating finite population distribution functions using auxilliary information. *Biometrika*, 80:385–392, 1993.

[LJF94] King-Ip Lin, H. V. Jagadish, and Christos Faloutsos. The TV-tree: An Index Structure for High-Dimensional Data. *VLDB Journal* 3(4):517–542, September 1994.

[LNS90] R. J. Lipton, J. F. Naughton, and D. A. Schneider. Practical selectivity estimation through adaptive sampling. In *Proc. 1990 ACM SIGMOD Intl. Conf. Managment of Data*, pages 1–11. ACM Press, 1990.

[LNSS93] R. J. Lipton, J. F. Naughton, D. A. Schneider, and S. Seshadri. Efficient sampling strategies for relational database operations. *Theoret. Comput. Sci.*, 116:195–226, 1993.

[LRS93] J. Li, D. Rotem, and J. Srivastava. Algorithms for loading parallel grid files. In *Proc. 1993 ACM SIGMOD Intl. Conf. Managment of Data*, pages 347–356. ACM Press, 1993.

[LS90] D. B. Lomet and B. Salzberg. The hB-Tree: A Multiattribute Indexing Method. *ACM Transactions on Database Systems*, 15(4):625-58, December 1990.

[LSS97] M. Luo, S. L. Stokes, and T. W. Sager. Estimation of the CDF of a finite population using a calibration sample. *Environ. Ecol. Statist.*, 1997. To appear.

[Mal89] F. Malvestuto. Computing the maximum-entropy extension of discrete probability distributions. *Comput. Statist. Data Anal.*, 8:299–311, 1989.

[Mal91] F. Malvestuto. Approximating Discrete Probability Distributions with Decomposable Models. *Trans. Systems, Man, Cybernetics*, 21(5):1287–1294, 1991.

[MCS88] M. V. Mannino, P. Chu, and T. Sager. Statistical profile estimation in database systems. *ACM Computing Surveys*, 20(3):192–221, Sept 1988.

[MD88] M. Muralikrishna and David J Dewitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. *Proc. of ACM SIGMOD Conf*, pages 28–36, 1988.

[NH94] R. Ng and J. Han. (1994) *Efficient and Effective Clustering Method for Spatial Data Mining*, Proc. 1994 VLDB, pp. 144-155.

[NS90] J. F. Naughton and S. Seshadri. On estimating the size of projections. In *Proc. Third Intl. Conf. Database Theory*, pages 499–513. Springer-Verlag, 1990.

[ODT+91] G. Ozsoyoglu, K. Du, A. Tjahjana, W. Hou, and D. Y. Rowland. On estimating COUNT, SUM, and AVERAGE relational algebra queries. In D. Dimitris Karagiannis, editor, *Database and Expert Systems Applications, Proceedings of the International Conference in Berlin, Germany, 1991 (DEXA 91)*, pages 406–412. Springer-Verlag, 1991.

[Olk93] F. Olken. *Random Sampling from Databases.* Ph.D. Dissertation, University of California, Berkeley, CA, 1993. Available as Tech. Report LBL-32883, Lawrence Berkeley Laboratories, Berkeley, CA.

[OR86] F. Olken and D. Rotem. Simple random sampling from relational databases. In *Proc. 12th Intl. Conf. Very Large Data Bases*, pages 160–169, 1986.

[OR89] F. Olken and D. Rotem. Random sampling from B^+ trees. In *Proc. 15th Intl. Conf. Very Large Data Bases*, pages 269–277, 1989.

[OR92] F. Olken and D. Rotem. Maintenance of materialized views of sampling queries. In *Proc. Eighth Intl. Conf. Data Engrg.*, pages 632–641. IEEE Computer Society Press, 1992.

[OR93] F. Olken and D. Rotem. Sampling from spatial datatbases. In *Proc. Ninth Intl. Conf. Data Engrg.*, pages 199–208. IEEE Computer Society Press, 1993.

[ORX90] F. Olken, D. Rotem, and P. Xu. Random sampling from hash files. In *Proc. 1990 ACM SIGMOD Intl. Conf. Managment of Data*, pages 375–386. ACM Press, 1990.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kauffman, Palo Alto, 1988.

[PI96] Viswanath Poosala and Yannis Ioannidis. Estimation of query-result distribution and its application in parallel-join load balancing. *Proc. of the 22nd Int. Conf. on Very Large Databases*, September 1996.

[PI97] Viswanath Poosala and Yannis Ioannidis. Selectivity estimation without the attribute value independence assumption. *Proc. of the 23rd Int. Conf. on Very Large Databases*, August 1997.

[PIHS96] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proc. 1996 ACM SIGMOD Intl. Conf. Managment of Data*, pages 294–305. ACM Press, 1996.

[Poo97] Viswanath Poosala. *Histogram-based estimation techniques in databases.* PhD thesis, Univ. of Wisconsin-Madison, 1997.

[PSC84] Gregory Piatetsky-Shapiro and Charles Connell. Accurate estimation of the number of tuples satisfying a condition. *Proc. of ACM SIGMOD Conf*, pages 256–276, 1984.

[PTVF96] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, The Art of Scientific Computing.* Cambridge University Press, Cambridge, MA, 1996.

[RKM90] J. N. K. Rao, J. G. Kovar, and H. J. Mantel. On estimating distribution functions and quantiles from survey data using auxilliary information. *Biometrika*, 77:365–375, 1990.

[Rob81] J.T. Robinson. The K-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes. *Proceedings ACM SIGMOD*, pages 10–18, 1981.

[SBM93] K. D. Seppi, J. W. Barnes, and C. N. Morris. A Bayesian approach to database query optimization. *ORSA J. Comput.*, 5:410–419, 1993.

[Sch81] M. Scholl. New File Organizations Based on Dynamic Hashing. *ACM Transactions on Database Systems*, 6(1):194-211, March 1981.

[SN92] S. Seshadri and J. F. Naughton. Sampling issues in parallel database systems. In *Advances in Database Technology- EDBT '92, 3rd Intl. Conf. Extending Database Technology*, Lecture Notes in Computer Science, pages 328–343. Springer-Verlag, 1992.

[SRF87] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-Tree: A Dynamic Index For Multi-Dimensional Objects. In *Proc. 13th International Conference on Very Large Data Bases*, pages 507–518, Brighton, September 1987.

[SS88] S. L. Stokes and T. W. Sager. Characterization of a ranked-set sample with application to estimating distribution functions. *J. Amer. Statist. Assoc.*, 83:374–381, 1988.

[SSW92] C.-E. Särndal, B. Swensson, and J. Wretman. *Model Assisted Survey Sampling*. Springer-Verlag, New York, 1992.

[Str80] Gilbert Strang. *Linear Algebra and its Applications*. Academic Press, 1980. 2nd edition.

[Sud76] S. Sudman. *Applied Sampling*. Academic Press, New York, 1976.

[TP91] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[VM] Brani Vidakovic and Peter Mueller. *Wavelets for Kids*. Duke University, Durham, NC. ftp://ftp.isds.duke.edu/pub/Users/brani/papers/.

[Wil91] D. E. Willard. Optimal sample cost residues for differential database batch query problems. *J. ACM*, 38:104–119, 1991.

[WS93] Kuansan Wang and Shihab Shamma. Spectral shape analysis in the central auditory system. *NNSP*, September 1993.

[WW85] R.J. Wonnacott and T.H. Wonnacott. *Introductory Statistics*. John Wiley, New York, 1985.

[WYM97] Wei Wang, Jiong Yang, and R. Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining . *Proc. 23rd VLDB*, pages 186–195, August 1997. Athens, Greece.

[You84] P. Young. *Recursive estimation and time-series analysis*. Springer-Verlag, New York, 1984.

[ZRL96] T. Zhang, R. Ramakrishnan and M. Livny. (1996) *BIRCH: an Efficient Data Clustering Method for Very Large Databases*, Proc. 1996 SIGMOD, pp. 103–114.

1998 VLDB Conference
Conference Announcement and Call for Papers
24th International Conference on Very Large Data Bases
New York City, U.S.A.
August 24-27, 1998

General Chairs:
Alexandros Biliris, AT&T Labs.
Inderpal S. Mumick, Savera Systems

Program Chairs:
Americas & Australia:
Jennifer Widom
Stanford University
Computer Science Dept.
Palo Alto, CA 94305
USA
widom@db.stanford.edu
Europe, Asia & Africa:
Oded Shmueli
Computer Science Department
Technion
Technion City, Haifa 32000
Israel
oshmu@cs.technion.ac.il

Industrial Program Chairs:
Dennis Shasha
COINS
New York University
New York, N.Y. 10012-1185
U.S.A.
shasha@cs.nyu.edu

Patrick Valduriez
INRIA Rocquencourt
B.P. 105
78153 Le Chesney Cedex
France
Patrick.Valduriez@inria.fr

Panels Chairs:
Klaus Dittrich, Universität Zurich
dittrich@ifi.unizh.ch
Yannis Ioannidis, Uni. Wisconsin
yannis@cs.wisc.edu

Tutorials Chair:
H.V. Jagadish, AT&T Labs.
jag@research.att.com

Exhibit Chairs:
Nelson Mattos, IBM Database Technology Institute
mattos@us.ibm.com
Andy Witkowski, Oracle
awitkows@us.oracle.com

Awards Committee Chair:
Jeffrey Ullman, Stanford University

Proceedings Chair:
Ashish Gupta, Junglee
agupta@shal.stanford.edu

For 24 years, VLDB has served the database community as its primary truly international conference. The current trend in VLDB is towards a forum where researchers and practitioners exchange ideas and experiences with excellent quality ensured by rigorous refereeing. The 1998 VLDB Conference will continue and strengthen this trend. The research paper track will be divided into three sections: "regular" section for standard research papers, "vision" section for papers on speculative and futuristic topics, and an "experience" section to promote papers describing novel and challenging database system implementation aspects or novel applications of database technology. In addition to the research track, there will be an industrial track, panels, tutorials, and exhibits. Due to location in New York city, we are especially interested in papers, tutorials, panels, and exhibits that may be of interest to the financial community.

Topics of Interest

The topics of interest in the 1998 VLDB Conference include but are not limited to: Active Databases, Authorization and Security, Concurrency Control and Recovery , DBMS Architectures, Data Mining and Knowledge Discovery, Data Models and Database Design, Data Warehousing, Database Benchmarks and Standards, Database Constraint Management, Database Evolution and Migration, Database Languages, Database Reengineering, Databases and Electronic Commerce, Engineering Databases, Financial Applications of Databases, Heterogeneous and Federated Databases, Interaction of Database and IR Systems, Internet/Intranet Services and Databases, Knowledge Base Management Systems, Logic and Databases, Mobility and Databases, Multidimensional Databases, Multimedia Databases, On-line Analytical Processing, Object-Oriented Database Systems, Optimization and Performance, Parallel and Distributed Databases, Real-time Databases, Replication, Caching, and View Materialization, Scientific and Statistical Databases, Spatial Databases, Temporal Databases, Text Databases, User Interfaces and Information Visualization, View Management, Visual Query Languages and Tools, Workflow Systems.

Paper Submission – Research Track

For all three sections of the research track – "regular", "vision", and "experience" – **six** copies of original papers should be submitted to the program co-chair for the geographic region in which the authors of the paper reside. Submitted papers should not exceed 20 pages, double-spaced and typeset with minimum 11 point font; over-length papers may be rejected without review.

Industrial Track

The industrial track consists of short papers from industry to report on new developments in DBMS technology. Papers should describe experiences using or building some aspect of database technology, or in dealing with large volumes of data in an industrial setting, and contain either positive or negative practical advice about the techniques described. Six copies of original papers should be submitted to the industrial program co-chair for the geographic region in which the authors of the paper reside. Submitted papers should not exceed 4 pages, double-spaced and typeset with min. 11 point font; over-length papers may be rejected without review.

Panels and Tutorials

Panel and tutorial proposals should be submitted to the respective chairs. Panels should address exciting new and controversial issues and must be debate-oriented rather than a series of short presentations. Tutorial proposals must clearly identify the intended audience, which should be considerably broader than the database research community.

Exhibits

The conference will showcase exhibits from the industry. To propose an exhibit, please send by email a short proposal to the exhibit chairs by May 1, 1998.

Important Dates

Paper, panel, tutorial, and industrial submissions:	
E-mail information/abstract:	16 February 1998 (firm deadline)
Hard-copy due:	23 February 1998 (firm deadline)
Notification of acceptance:	8 May 1998
Conference:	24-27 August 1998

For more information, please monitor the WWW site: http://www.research.att.com/conf/vldb98/

... co-located with VLDB-98 !

Knowledge Discovery and Data Mining

The Fourth International Conference (KDD-98)

New York City, August 27-31, 1998

Sponsored by the American Association for Artificial Intelligence
In Cooperation with IEEE Technical Committee on Data Engineering

`http://www-aig.jpl.nasa.gov/kdd98/`
(paper submission deadline: March 17, 1998)

Because of the continuing rapid growth of databases, techniques for extracting knowledge automatically are becoming more and more necessary. Knowledge Discovery and Data Mining is an area of common interest to researchers in machine learning, databases, statistics, and data visualization. As an interdisciplinary field, KDD has made important strides by facilitating the cross-fertilization of ideas. This year KDD-98 will be co-located with VLDB-98 (the Conference on Very Large Databases) in order to stimulate interactions between researchers. Topics of interest include, but are not limited to the following:

Theory and Foundational Issues
Representation of data and knowledge
Search, retrieval, and discovery
Structured, unstructured and multimedia data

Methods, Algorithms & Tools
Complexity, efficiency, and scalability
Implementations for scalable platforms
Large/high-dimensional datasets
Use of prior domain knowledge
Disk-based algorithms
Database architectures and primitives
Integration with data warehousing, OLAP
Unification with database querying

KDD Process and Human Interaction
Models of the knowledge discovery process
Preprocessing and data engineering
Data and knowledge visualization
Evaluating subjective relevance and utility
Use of discovered knowledge
Interactive data exploration and discovery
Privacy and security

Applications
Business, science, medicine and engineering
Text, image, audio, sensor, numeric, categorical or mixed format data
Discovery useing the internet

Organizing Committee

General Chair: Gregory Piatetsky-Shapiro, Knowledge Stream Partners, gps@kstream.com
Program Chair: Rakesh Agrawal, IBM Almaden Research Center, ragrawal@almaden.ibm.com
Program Chair: Paul Stolorz, Jet Propulsion Laboratory, pauls@aig.jpl.nasa.gov
Publicity Chair: Foster Provost, Bell Atlantic Science and Technology, foster@nynexst.com
Tutorial Chair: Padhraic Smyth, University of California at Irvine, smyth@ics.uci.edu
Panel Chair: Willi Kloesgen, GMD, Germany, kloesgen@gmd.de
Workshops Chair: Ronny Kohavi, SGI, ronnyk@engr.sgi.com
Exhibits Chair: Ismail Parsa, Epsilon, iparsa@epsilon.com
Local Arrangements Chair: Kyusoek Shim, Bell Laboratories, shim@research.bell-labs.com
Sponsorship Chair: Ramasamy Uthurusamy, General Motors, samy@iss.gm.com
Conference Webmaster: Alexander Gray, Jet Propulsion Laboratory, agray@aig.jpl.nasa.gov

For full details, including calls for papers, workshops, tutorials, panels, and exhibits, please see:
`http://www-aig.jpl.nasa.gov/kdd98/`

ADVANCE PROGRAM

ICDE '98

DATA ENGINEERING

14th International Conference on

Data Engineering

February 23 - 27, 1998

Adam's Mark Hotel, Orlando, Florida, USA

Sponsored by the IEEE Computer Society with support from ORACLE Corporation

IEEE

ORACLE Enabling the Information Age™

THE CONFERENCE

Data Engineering deals with the use of engineering techniques and methodologies in the design, development and assessment of information systems for different computing platforms and application environments. The 14th International Conference on Data Engineering will continue in its tradition of being a premier forum for presentation of research results and advanced data-intensive applications and discussion of issues on data and knowledge engineering. The mission of the conference is to share research solutions to problems of today's information society and to identify new issues and directions for future research and development work.

TECHNICAL PROGRAM HIGHLIGHTS

Research Paper Sessions:

• Semistructured Data • Indexing in Data Warehousing • Mobile and Disconnected Database Operations • Management of Objects • Concurrency Control • Server-Based Database Systems • Recovery and Buffer Management • Parallel Database Systems • Cooperation and Workflow Management • Video and Image Databases• Visualization of Multimedia Data • Mining of Association Rules • Association Rules and Dependencies • Architectural Issues in Data Mining • Query Processing in Spatial Databases • Access Structures • Spatial and Temporal Databases • Queries and Views

Keynote Speakers:

• Andrew Mendelsohn (ORACLE Corporation): "Databases: The Next Generation"
• Hector Garcia-Molina (Stanford University): "Safeguarding and Charging for Digital Information"

Panel Discussions:

• Data Warehousing Lessons from Experience • WWW and Internet - did we (the database community) miss the boat? • Directions in Database Research (The ICDE ritual continues) • Legacy Systems

Industry Program:

• Data Warehouses • Data Mining • Database Engines • Object-Oriented Systems

Demonstrations of research prototypes will also be featured

LOCAL ARRANGEMENTS

The Adam's Mark hotel is conveniently situated just moments from all of Orlando's renowned attractions and, best of all, the Adam's Mark is connected to one of Florida's largest and most fashionable shopping centers, the Florida Mall. Guests can stroll through the art-deco styled lobby into the hustle-bustle of over 200 shops and restaurants, making the hotel location unlike any other in the area. Visit the ICDE98 home page for registration details.

ORGANIZING COMMITTEE

General Chair: Philip Yu, IBM T.J. Watson Research Center
Program Co-Chairs: Susan D. Urban, Arizona State University
Elisa Bertino, University of Milano
Steering Committee Chair: Joseph E. Urban, Arizona State Univ.
Industry Program Chair: Surajit Chaudhuri, Microsoft Research
Panel Program Chair: Calton Pu, Oregon Graduate Institute
Tutorial Program Chair: Tamer Ozsu, University of Alberta

TUTORIAL PROGRAM

1. **Workflow Management: Technology, Experiences, and Research Issues:** Amit Sheth and John Miller (University of Georgia), February 23, 1998, (full day).

2. **Dissemination-Based Information System:** Michael Franklin (University of Maryland) and Stan Zdonik (Brown University), Morning, February 23, 1998, (half day).

3. **Uncertainty Management in Database and Knowledge Base Systems:** Laks Lakshmanan (Concordia University) and Fereidon Sadri (University of North Carolina at Greensboro), Afternoon, February 23, 1998, (half day).

4. **Distributed Multimedia Information Systems:** Leana Golubchik (University of Maryland) and John Lui (Chinese University of Hong-Kong), February 24, 1998, (full day).

5. **Recent Trends in Transaction Processing and Client Server Computing:** C. Mohan (IBM, Almaden), February 24, 1998, (full day).

6. **Decision Support Technologies: OLAP, Data Warehousing, and Data Mining:** Surajit Chaudhuri (Microsoft Research) and Umesh Dayal (Hewlett-Packard), Morning, February 24, 1998, (half day).

7. **Databases and Mobile Computing:** Margaret Dunham (Southern Methodist University), Afternoon, February 24, 1998, (half day).

ICDE98 will be preceded by the 8th International Workshop on Research Issues in Data Engineering: Continuous-Media Databases and Applications, February 23-24, 1998. For details visit http://www.cs.buffalo.edu/ride98.html

INFORMATION RESOURCES

The Advance Program and other information concerning conference and hotel registration are available from the ICDE98 home page: **http://gump.bellcore.com:8000/icde98/**

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398