Title

STATISTICAL AND SCIENTIFIC DATABASE ISSUES

Permalink

https://escholarship.org/uc/item/94n8m5ks

Authors

Shoshani, A.
Wong, H.K.T.

Publication Date

1985-02-01

LBL-19841
Preprint

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA, BERKELEY

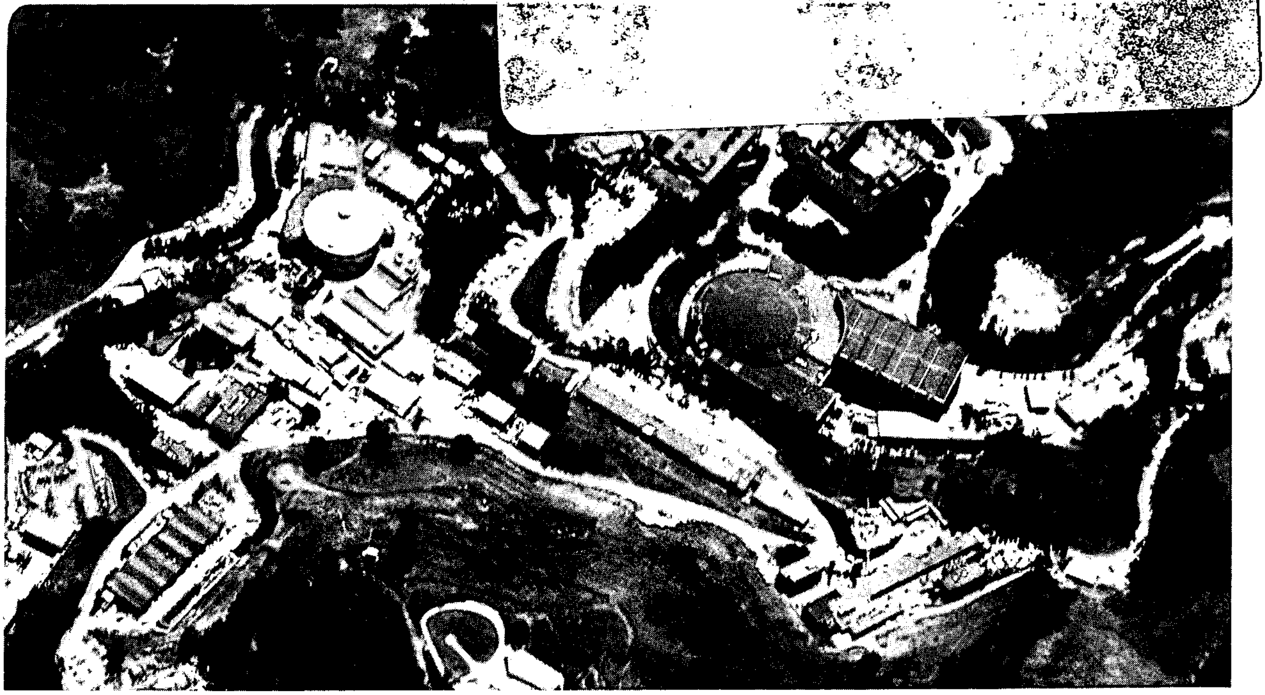
Information and Computing Sciences Division

Submitted to IEEE Transactions on
Software Engineering

STATISTICAL AND SCIENTIFIC DATABASE ISSUES

A. Shoshani and H.K.T. Wong

February 1985



Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

LBL-19841

Statistical and Scientific Database Issues

Arie Shoshani and Harry K.T. Wong

Computer Science Research Department
University of California
Lawrence Berkeley Laboratory
Berkeley, California 94720

February, 1985

This research was supported by the Applied Mathematics Sciences Research Program of the Office of Energy Research, U.S. Department of Energy under contract DE-AC03-76SF00098.

STATISTICAL AND SCIENTIFIC DATABASE ISSUES

Arie Shoshani and Harry K.T. Wong

Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

Abstract

The purpose of this paper is to summarize the research issues of Statistical and Scientific Databases (SSDBs). It organizes the issues into four major groups: physical organization and access methods, operators, logical organization and user interfaces, and miscellaneous issues. It emphasizes the differences between SSDBs and traditional database applications, and motivates the need for new and innovative techniques for the support of SSDBs. In addition to describing current work in this field, it discusses open research areas and proposes possible approaches to their solution.

I. Introduction

Over the last several years there has been growing interest in Statistical and Scientific Database (SSDB) research. This interest is due, in part, to the inadequacy of commercial database management systems to support statistical and scientific applications. A similar situation exists in other application areas such as CAD/CAM, VLSI design, and knowledge-based systems. The main reason for this situation is that most of the commercial systems available today were designed primarily to support transactions for business applications (such as marketing and banking), while other applications have different data characteristics and processing requirements.

Another reason for the interest is the large amount of data that exist in SSDB applications. A single scientific experiment can generate hundreds of megabytes of data within days. Many scientific simulations are not carried out to the desired granularity level, only because it is impractical to process and analyze the large amount of data that would be generated. Many practical databases collected for statistical purposes, such as trade data between countries, or the various census data, are too large to be handled with conventional data management techniques efficiently. But, in addition, there is a large amount of data that was not collected originally for experimental or statistical purposes, that has tremendous potential when used for statistical purposes. For example, routine patient records in hospitals can be used for statistical "cause and effect" studies. Business transactions can be statistically analyzed for policy setting and econometric models. For the most part, such sources of routine collections of data are left unused because adequate data management and analysis facilities do not exist.

SSDB applications differ from commercial applications both in the properties of the data and in the operations over the data. For example, SSDBs often contain sparse data (usually in a multi-dimensional matrix form) that need to be efficiently compressed. They often contain special data types such as vectors or time series. Similarly, statistical analysis requires different types of operations over the data. Typically, only a few variables are examined over a large number of the cases (as is the case in determining cross-correlation). This suggests a transposed file organization where all the values of a single variable are stored together, as opposed to the "horizontal" record structure found in most commercial systems. In addition to statistical operators, such as sampling and aggregation, the access of the data is of a different nature. For example, it is quite common to access a region in multi-dimensional space, such as finding materials with certain approximate properties, or cases that fit a statistical pattern. For such cases multi-dimensional data structures and search methods are desirable.

Statistical and scientific applications have many similar properties, which is the reason for exploring such applications together. Most scientific databases are eventually statistically analyzed. Consider, for example, a particle physics experiment, where particles are collided in order to generate sub-particles. Such collisions can occur millions of times, and the results captured by special detectors. The data from the detectors passes many stages of analysis. The first few stages reconstruct the tracks of sub-particles produced. Eventually, collections of tracks for certain sub-particles are analyzed statistically in order to characterize the collisions. In general, scientific data have more complexity than strictly statistical data (such as survey data), because in addition to the measured data, it contains data about the instruments, the environment of the experiment, and the configuration of the experiment. Configuration data can be quite complex, as is the case, for example, in describing a wing configuration for an airplane simulation.

It is not the purpose of this paper to present a survey of current work, or to characterize the properties and access requirements of SSDBs. Such work was previously presented in [1] and [2], for example. Rather, it's purpose is to summarize the areas of research. Most areas have only been partially addressed by current research. We found it convenient to group the research areas into four groups: physical organization and access methods, operators, logical organization and user interfaces, and miscellaneous issues. It is hoped that the delineation of issues in these areas will spur further research interest. Since our purpose is not to survey existing research work, but rather to delineate the research areas, we only mention here a few references that have extensive reference lists or a few representative papers that introduce new approaches.

II. On the Role of Multi-Dimensionality

Multi-dimensionality is a dominant feature in SSDBs. Many of the characteristics and requirements of SSDBs can be traced to this feature. It greatly affects both logical and physical data management design considerations, and deserves special support in systems that are intended to manage SSDBs. While in other applications the concept of an "entity" (e.g. employee, department, bank account) is common, it is more convenient in SSDBs to think about "cases," which are instances of an experiment, a simulation, or a survey. While cases can be given unique identification numbers (so that they can be thought of as instances of entities), they are commonly characterized by a set of parameters or categories. For example, trade data can be identified in terms of the exporting country, importing country, commodity, year, and month. Similarly, a corrosion experiment can be described in terms of the temperature, acidity, salinity, length of exposure, and the material used. Another common characterization of physical experiments is in terms of space and time. In all such examples there exists naturally a multi-dimensional space for which measured data is collected. Many issues discussed later involve multi-dimensionality.

One can dismiss multi-dimensionality as nothing more than a composite key that can be handled simply by defining it as such in the data definition section of the data management system. However, in reality, data definition alone is not sufficient; the system should provide the corresponding physical support, the query facilities and query optimization algorithms to handle multi-dimensionality properly. At the physical level special data structures for organizing multi-dimensional data are needed, as well as special access methods and compression techniques. At the logical level, special models to represent the multi-dimensional aspects of the data are necessary, and the corresponding user interfaces to express queries over them are needed. The difficulty of dealing with multi-dimensional spaces is further compounded by the fact that each dimension can itself have a complex (usually hierarchical) structure. For example, a trade commodity can be broken into categories of food, energy, clothing, etc. Each category can be further broken into sub-categories, such as energy into the sub-categories of oil, coal, and gas. This sub-categorization of a single dimension can continue into many levels, and sometimes overlaps may exist. Representing such complexity requires special facilities at the logical and user interface level. Finally, the query optimization of data structures involving multi-dimensional spaces has the additional complexity of dealing with special physical structures and operators over such data.

III. Physical Organisation and Access Methods

In this section we are concerned with physical organization techniques that take advantage of data properties and access patterns to provide efficient data storage and access.

1. Compression

There are several reasons for the need of compression in SSDBs, two of which are related to the multi-dimensionality property described in the previous section.

The first reason is that the multi-dimensional space created by the cross product of the values of the dimensions can be naturally sparse. For example, in the trade statistics database mentioned previously, where the dimensions are exporting country, importing country, material, year, and month, only a small number of the materials are exported from a given country to other countries. Similarly, in a physics experiment of colliding particles only a small percentage of the detectors would actually record a measurement, since only those in the path of sub-particles will have a measurable reading.

The second reason for compression is the need to compress the descriptors of the multi-dimensional space. Suppose, for example, that the trade database mentioned above is to be put into a relational database system. The five dimensions organized in this tabular form will create a great repetition of the values of each dimension. In fact, in the extreme, but often realistic case that the full cross product is stored, the number of times that each value of a given dimension repeats is equal to the product of the cardinalities of the remaining dimensions. In addition, the descriptors of some of the dimensions can be hierarchical to several levels as was explained in section II. This further complicates their storage in a compressed form.

It is interesting to note that the data in a multi-dimensional space can be reorganized to yield better compression. Multi-dimensional data can be thought of as being organized as an n-dimensional matrix. The rearrangement of rows and columns of the matrix can result in better clustering of the data into regions that are highly sparse or highly dense. Methods that take advantage of such clustering can thus become quite effective. Determining the optimal ordering of the data to achieve maximum compression is a difficult problem that probably cannot be solved without probabilistic and/or heuristic method.

Other reasons for compression in SSDBs result from the properties of the data values. Often the data values are skewed, where there are a few large values and many small values. Techniques such as front compression may be applied in such cases. When data values are large but close to each other, difference (or delta) encoding methods can be used. When certain values tend to appear repeatedly, techniques such as Huffman encoding that assign the smaller codes to the most frequent values, can be used. Such techniques as well as techniques to support multi-dimensional data are described in a compression survey paper appearing in this issue [3].

2. Efficient Access and Manipulation of Compressed Data

The access and manipulation of compressed data requires the decompression of the data, which is an inefficient process when only part of the data is needed. The problem is that a large amount of data needs to be decompressed in order to locate the desired data values. This is particularly true for operations over the data which require random access such as sampling. But even when the operators are over a large subset of the data set, such as aggregation operators, it is still necessary to locate and assemble the subsets of the data as specified by a restriction operator.

Traditional compression techniques are only concerned with the efficiency of compression. It is generally true that the gains achieved by compression in terms of the total amount of data that have to be accessed and manipulated are large enough to offset the decompression cost. However, it is still desirable to develop techniques that can access the data in their compressed form and that can perform logical operations on the compressed data. Such techniques should provide efficient indexing capabilities that access only that portion of the compressed data that is requested. Such techniques can range from the ability to find the location of every data item (which was not compressed out), to locating blocks of compressed data which are then decompressed. The later is important for efficient I/O access, while the former reduces, in addition, the CPU processing time.

3. Multi-Dimensional Data Structures

The property of multi-dimensionality makes multi-dimensional data structures, such as grid files, quad trees, or K-D trees (for references see [4]), attractive data structures for SSDBs. These data structures are especially effective when the data values in a region of the multi-dimensional space are needed together. There are many applications where SSDBs tend to be accessed in local clusters, such as searching for nearest neighbors of a given point in space. For example, in order to find a material with certain properties, we need to perform a nearest neighbor search in the multi-dimensional space representing these properties. Similarly, identifying a particle track in physical space requires locating the readings from detectors that are in the neighborhood of the path of the particle. Such applications suggest that data should be organized into cells along the dimensions of the data.

Algorithms for determining the cell boundaries and the indexing structures for accessing the cells efficiently should provide more efficient access than conventional indexing techniques. An example of work whose purpose is to optimize the selection of boundaries of multi-dimensional cells according to their access pattern, was recently reported in [5].

4. Alternative File Organizations

Statistical analysis typically involves only a few of the variables at each analysis step. When data is organized horizontally (all variables of each event stored together), then the access of a few of the variables requires the access of the entire data set. Partitioning of the data vertically (i.e. the values of a variable for all events stored together) is a more efficient method in such cases. Such a file organization, called a transposed file, has been used in several statistical database system (see references in [1]). A more extreme version of transposed files has been proposed in [6], where the transposition of files is carried to the bit level (i.e. even values within each attribute are further transposed into "vertical" bit partitions).

Such methods can be naturally enhanced with compression methods. Each partition of the transposed files can be compressed separately. The challenge is then to design access methods that can search and manipulate the compressed partitions efficiently.

The combination of the above methods for file organization are not well understood. An effective system will have to combine the ideas of transposed files, compression of multi-dimensional data, and multi-dimensional access methods, as well as more conventional indexing techniques (such as B-tree indexing). Accordingly, physical database design techniques which select the best file structure for a given application will also need to be developed.

5. Buffer Management

The effective use of the primary memory buffer (called buffer management) can greatly reduce the I/O from secondary storage. In SSDBs the buffer management is very important because of the large size of data that needs to be manipulated. Even for the simplest operations, such as linear aggregations (e.g., sum) it is necessary to devise methods that minimize reading data from secondary storage repeatedly. For such simple cases, it is sufficient to use simple methods, such as keeping running sums and counts. But other operations, such as transposing a large matrix, or multiplying matrices, require more complex buffer management methods. Several examples of such methods are discussed in a paper appearing in this issue on the efficient support of statistical operations [7].

Buffer management methods may also be required as a result of the file organizations discussed above. An example is tuple assembly of transposed files. Once the selection of data values from the transposed files have been performed, it may be necessary to assemble tuples (records) for output purposes. This may require scanning over large partitions of the data in a limited buffer, which poses a buffer management problem.

IV. Database Operators

Certain operators are particularly important for SSDBs. Some of the operators are not traditionally considered,part of the data management functions but are needed for SSDB data. Examples are sampling and multi-dimensional data structure operators such as transposition and aggregation. Lack of operators forces the user to program the operators repeatedly in an unnatural manner resulting often in large inefficiencies. In order to achieve better efficiency, all important operators should be supported directly by the underlying DBMS as close to the physical storage level as possible. For example, if the data is compressed, then the operators should operate directly on the compressed data, without having to decompress them first, and then compress the results. Also, the query optimizer should take the availability of these operators into account in generating query processing strategies.

1. Sampling

Sampling is often left for an application level program, such as a statistical analysis package. However, it is more efficient to perform sampling by the data access routines, since only the sampled data is passed to the application program. If at all possible, sampling should be done as early as possible in the query processing stage and as close to the physical storage level as possible, as stated earlier. For example, it is desirable to "push" the sampling operator as far down the query tree as possible as long as the result is statistically sound. Another sampling example is sampling from joined sets. It is desirable to develop methods that sample from each data set without the need to join the entire data sets before sampling can be done.

There are a large number of sampling techniques in use by data analysts, such as simple random sampling, stratified sampling, cluster sampling, probability sampling, subsampling, etc. The sampling may be with or without replacement and it may have a fixed sample size or fraction with respect to the SSDB. Accommodating all these techniques in the context of SSDB management is an important and challenging research direction.

2. Nearest Neighbor Search

Operators for nearest neighbor search are important for SSDBs. Many applications require such a search for finding matching cases. For example, in a hospital patient study, we may be interested in patients that have similar characteristics as a given patient being observed.

There are basically two types of nearest neighbor search. One is finding the closest point (or several closest points) to a given point. The other is finding all points within a certain "distance" of the given point. In both cases, techniques for finding elements in multi-dimensional cells are necessary, and then a "selection" criterion applied to the set retrieved.

The efficient support of nearest neighbor techniques requires appropriate data structures. The multi-dimensional data structures mentioned in Section III are designed to support such operators efficiently. However, other indexing techniques that support range queries well can also be useful for nearest neighbor operators.

3. Estimation and Interpolation

Data analysis on SSDBs often requires making estimates on missing data values. An example is the coal production and consumption where estimation techniques on missing data range from a simple guess to elaborate interpolation techniques. Scientific data usually describe continuous physical phenomena, but only a finite number of discrete data points are actually stored. As a result, data do not always exist for all desirable points. In such cases, estimation and interpolation routines could be incorporated into the data access mechanisms. Methods performing these operators efficiently are necessary. Such methods require efficient access of nearest neighbors.

4. Transposition

The output and display of scientific data often requires transposition of the data, because the data is stored according to certain dimensions and needed for output along other dimensions. Efficient transposition techniques need to be developed, under the constraint of a limited buffer space. Techniques exist for multi-dimensional matrix transposition, but they are optimized with respect to the number of instructions, and not to the more relevant database criterion of the number of I/O page fetches. Another important research issue for transposition is the direct manipulation of compressed data so that a minimal number of passes are made over the data without having to decompress the data and compress the results again.

5. Aggregation

Aggregation is another common operation in SSDB processing. An example of the application of the aggregation operation that is often used in SSDB applications is the generation of summary data from micro data by tabulation. Another example of aggregation is the "collapsing" of multi-dimensional data structures in order to remove a certain dimension. Most commercial DBMSs provide some aggregation capability, but it is inadequately supported both at the logical as well as the physical level as pointed out in the paper on query languages for SSDBs in this issue [8]. In addition to the usual aggregate functions provided by most DBMSs such as max, min, sum, average, and count, an SSDB system should provide more complex aggregation functions such as tabulation, binning, and median. Again, these functions should be implemented directly over compressed data and designed to run efficiently with a limited buffer size.

6. Relational Operations

Data analysts often require operations over multiple data sets, such as subsetting, intersecting, and combining of cases. Set operations such as subsetting, union, and intersection are needed. These operations are more complicated than those available in commercial DBMSs because of the additional meta-data information attached to the data, such as the different types of missing data and the reliability level of the data. These conditions require resolution before the set operations can be applied.

In addition, the traditional sort-merge and nested loop join methods may not be appropriate for large databases (which is often the case for SSDBs) because of the excessive I/0 and sorting costs. Hash join methods may be a more viable alternative because they have linear complexity with respect to I/O. Hash join methods essentially partition the two data sets that need to be joined by a hash function on the join attribute and the partitions are then joined pairwise. Multiple-pass hash join methods can be used to reduce the size of the partitions to fit the available memory. Evaluation of hash joins and other join methods is an important research topic.

7. Support of Other Statistical Operators

The emphasis of current SSDB research has been on the discovery of supplementary structures to conventional database systems, such as file structures, modeling constructs, and data management operators. This development trend still follows the traditional DBMSs paradigm. As a result, research on data management support of computational statistical operators is lacking. These operators are now considered the responsibility of the statistical packages, where they are supported less efficiently because the optimization opportunity of using the physical storage structures for these operations is not available. Efficient implementation of these operations should rely on the availability of the underlying physical storage structures, buffer management strategies, and optimization techniques. The companion paper in this issue on data management support of statistical operators motivates this problem and introduces the potential benefits of this approach [7].

V. Logical Modeling and User Interfaces

In this section we discuss issues relating to the logical representation of information and the query facilities available to the user.

1. Complex Data Types

Conventional data management systems support the concept of a set of entity instances in one form or another. Most existing modeling approaches such as the relational, hierarchical, network, and entity-relationship, support only simple data types for the attributes of the entities. In SSDBs there are many examples of the need to support complex data types, such as vectors, matrices, and time-series. These complex data types should be, in general, extendible so that the value of a complex data type can itself be complex. Consider, for example, the need to represent the boundaries of geographical regions. A boundary can be represented as a variable size vector, where each of its elements is a pair made of the longitude and latitude of the points making up the boundaries.

SSDBs are a good example of the need to support a large diversity of data types. The usual character and numeric data types are needed, including high precision floating point for numeric measurements. In addition to vectors and matrices mentioned above, more complex table representation of multi-dimensional spaces are needed. Graphs and histograms are common results of statistical analysis, and support for textual data for bibliographies, descriptions of experiments, and statistical collection methods are often needed. It is difficult to imagine a single system that can handle all these data types concurrently, but there are some attempts of using new techniques to extend existing systems, such as the use of "abstract data types." In the final analysis, the efficiency of such approaches will determine their success.

2. Semantically Rich Models

In addition to modeling the variety of complex data types discussed above, logical models for SSDBs need to be able to represent higher level semantic concepts. For example, modeling of multi-dimensional spaces where the dimensions themselves can be represented as a hierarchy of terms is a very desirable feature for some applications. Similarly, some statistical analysis operations can result in multiple data sets that require special modeling capabilities. For example, the results of a linear regression analysis may include a coefficient vector, a covariance matrix, and a residuals vector. The data model should be capable of representing the semantic fact that these data sets belong together.

Many SSDBs do not represent a homogeneous collection of data. In fact, they can be thought of as multiple databases. For example, a scientific experiment may be composed of the measured data, another data set which records the temperature and pressure conditions over time, and another data set that represents the calibrations of the different detectors. Each of these data sets can be thought of as a separate database, but they are all needed together when the data is analyzed. It is therefore necessary that the data model permits the description of the relationships between these data sets so that the analysis can be properly specified. In the above example, it is necessary for a given measurement to be interpreted according to the characteristics of the detector used, and the actual temperature and pressure at the time of the measurement. Such modeling capabilities should form the basis for efficient query languages and powerful user interfaces.

3. Temporal Data

Temporal data are crucial to SSDBs since quite often they represent a collection of information over time. This aspect may not be as important in commercial databases since usually only the most updated information is needed. Even in the case that historical information is kept in commercial applications, they are not typically accessed across the time dimension; rather one slice of the data for a certain point in time is accessed. Indeed, there are no commercial systems that explicitly model or support temporal data. However, recently there is renewed interest in modeling time for business applications [e.g., 9].

It is usually possible to model the time element of measurement data as one more dimension in the multi-dimensional space formed by other parameters. This is especially true if the measurements are taken at regular intervals over time. The difficulty of dealing with temporal data are most pronounced when events occur in discrete but non-regular sequences, such as the breakdown or the replacement of detectors. The other difficulty is that different data sets may vary at different rates. For example, the drift of a magnetic field may be taken every minute, while the measurements are taken every second. Similarly, the rate of collecting trade data varies for the different commodities. Even worse, the rate can vary for the same commodity over different time intervals and over different countries. During the data analysis process, it is necessary to correlate both kinds of measurements.

In general, modeling temporal data requires the integration of static data, data that change in regular intervals, and data that appear as irregular discrete events. From a logical point of view it is necessary to model the various types of temporal data, so that their semantics are clear to the user. It is also necessary to provide query facilities that permit users to specify conditions in the time dimension, as well as correlate data that may be varying at different rates.

4. Meta-Data

There is a large amount of information which describe the data in SSDBs that is kept in an ad-hoc fashion in log books, text files, or even hand-written notes. Examples are the failure logs of devices, the date and method used in generating a new analyzed data set, the identity of people who generated the data sets, the description of materials that were encoded in the database, the data units used, etc. Such information, which is referred to collectively as meta-data, can be quite complex and is just as important as the database itself for analysis purposes. In addition, such meta-data are particularly important for archival purposes.

In a typical statistical analysis process an analyst may generate a large number of subsets over the data. He/she may first take large samples that are further refined, select subsets of the database which may be successively reduced, and generate new data sets as a result of the analysis. This proliferation of data sets, and the information on who generated them, when they were generated, and why they were generated, needs to be kept track of. In general, these data sets have a hierarchical relationship between them. From the above comments it can be seen that supporting meta-data can require the full support of data management tools.

While in commercial systems there is a tendency to restrict meta-data to the data definition/directory facilities, the distinction between meta-data and data in SSDBs is not that obvious. It seems that the best solution for SSDBs is to blur the distinction between the two types of data, and to handle both types with the same data management facilities. As a result, the meta-data can be searched, modified and associated with the rest of the data in a single system.

5. User Interface Facilities

From the discussion above of special modeling facilities, complex data types, and the integration of data and meta-data, it follows that special user interfaces and query facilities are needed. But in addition to providing the ability to access the data, the query facilities need to provide special operators over the data. In Section IV above, we discussed several classes of operators that are especially important to SSDBs. The query facilities should provide easy to use constructs for such operators.

There are many approaches that researchers have proposed for user interfaces to SSDBs. They include extensions to current languages to support special operators, menu facilities to browse the hierarchies of multi-dimensional spaces, graphical user interfaces that help narrow down the area of interest, query-by-example methods, table manipulation methods, and new query languages. A survey of query facilities for SSDBs which discusses these approaches is given in another paper in this issue [8]. While the different approaches are important steps in achieving good user interface facilities for SSDBs, it is still unclear whether a single joint facility can emerge, or whether having a collection of tools for different purposes is the only practical approach.

VI. Miscellaneous Issues

1. Data Integrity and Quality

Because of the size of SSDBs, users often derive subsets from the original database and perform statistical analysis on the subsets. Subsets of the subsets are derived by other users, and so on. When a new subset is generated and manipulated, the integrity rules of the original set may not be valid for the new subset. The procedure by which new subsets are generated and their implied integrity rules need to be maintained by the system. Otherwise, further operations over the subsets may be incorrect. For example, suppose that a subset was selected, its missing values interpolated, and outliers removed. These facts have to be known for subsequent processing. Often analysts take an existing subset, assume its correctness and proceed to analyze it, perhaps getting incorrect results. Powerful integrity control mechanisms are needed in order to enforce the validity rules among the subsets. Facilities for expressing integrity constraints by the users as they generate subsets are also needed.

The data in SSDBs often come from many sources with a varying degree of quality and "believability." Using such data for data analysis should take this important aspect into account. For example, the accuracy and resolution of the instruments that generate data in a scientific experiment provide a vital clue to the data quality collected. Another example is the World Bank's economic data on developing countries where a significant amount of data is derived by estimates. This concept of data quality is not supported by commercial DBMSs. An SSDB system needs an efficient way of "tagging" the data with data quality flags, and integrating the concept of data quality into the data model, query language, and operators over the data.

The validation facilities in current DBMSs are not powerful enough to express the correctness conditions of incoming data. The validation of SSDB data may involve complex computations that check cross correlation of data values. Techniques that automate data validation both at the data collection and the data generation stages are necessary.

2. Analysis Process Management

Data analysis on SSDBs is an iterative and evolutionary process. A facility that records the analysis steps automatically would be very helpful to the analyst. Using this information, the different paths of the analysis can be pursued, and past steps of the analysis can be recovered without having to recreate the analysis over again. This kind of analysis process management facility represents a new concept of "logging" where the intermediate objects of temporary files, permanent files, and hard copies, as well as the analysis procedures are maintained. In addition to serving as a documentation aid, this facility can also be used to implement repetitive analysis processes. To support facilities of this nature, efficient storage structures are needed because of the large amount of data that need to be stored. Also, user interface techniques are needed for the users of this facility during the analysis process.

3. Intelligent Data Analysis

A large part of the data analysis activity is relatively well understood. This suggests that automatic data analysis may be a feasible possibility. Examples include the automation of data validity checking (such as outlier detection), the derivation of summary statistics, the suggestion of applicable statistical procedures to the analyst, etc. This kind of intelligent data analysis tools require representation and application of extensive knowledge about statistical data and the analysis process. The expert systems approach used in Artificial Intelligence applications seems to be a viable paradigm for this kind of activity. The paper in this issue on antisampling for estimation is an example of this approach [10].

4. Query Optimization

Query optimization is an important research area in conventional DBMSs. In the area of SSDBs, there are many useful file and data structures (such as transposed files, multi-dimensional data structures, etc.), operators (such as sampling, transposition, and aggregation, etc.), and query facilities (such as summary table query languages, graphical languages, etc.), that the query optimizer has to explore a very different (and sometimes much larger) solution space for strategies. Traditional query optimization techniques may have to be extensively expanded. A more thorough understanding of the complexity and behavior of these file structures, operators, and languages have to be gained before they can be integrated into a query optimizer.

5. Special Hardware

Since the cost of producing special purpose hardware is rapidly coming down, it is appropriate to consider whether specialized hardware could significantly benefit SSDBs. There is almost no research work to-date on this possibility, but this approach looks promising because of the special nature of the data and the operators in SSDBs.

There are several file organizations and operators that are intuitively appealing. The management of multi-dimensional spaces could benefit from specialized hardware that performs "array linearization." Such hardware would compute the position of points in multi-dimensional space from the positions of each dimension, and could access blocks of data accordingly. Another possibility is the use of special hardware for compression, so that sparse data can be accessed more efficiently in their compressed form.

Special hardware can also be used to process the data in parallel. Two possibilities come to mind. One is the processing of partitions of transposed files in parallel, and assembling the results. In the extreme case of file transposition to the bit level discussed in [6], the special hardware is quite simple since it deals with partitions that represent bit streams. File transposition hardware can also be designed in conjunction with compression so that each partition can be accessed in its compressed form.

The other possibility for parallel processing is for data that is organized into multi-dimensional cells. It is often the case that the data in each cell can be processed independently from the other cells, and thus in parallel. A case in point is simulation data that often requires only nearest neighbors data for their computations.

Finally, special hardware can be designed for special operators, such as transposition, aggregation, or matrix multiplication. Here again, performing such operations on compressed data is a challenge that is worth investigating. So far, the benefits of special hardware are only speculative, but SSDBs seem to offer the opportunities for such an approach.

6. Security

There is a large body of literature that deals with the security of statistical databases (see [11] for references). The interest in statistical database security stems from the desire of protecting information about individuals while releasing aggregate information about groups of individuals.

Statistical database security is a difficult problem. It has been demonstrated that simple techniques, such as restricting the number of individuals that qualify in an aggregate query to be above a chosen threshold, do not work. Thus a variety of other techniques were proposed. They include keeping track of previous queries, pre-partitioning the data sets into cells whose boundaries cannot be crossed by aggregate queries (this method is most popular for census data), estimating the aggregate values from samples over the data, and perturbing either the input or output values. A detailed discussion of these methods is beyond the scope of this paper. It is worth noting that such security issues do not usually arise in scientific databases, but they can be essential for some statistical applications.

Conclusions

In this paper we described the different issues that arise in the context of statistical and scientific database applications. The usual areas of physical organization, access methods, logical models, and user interfaces are discussed, but in addition needs that are unique to these applications are described. These include special operators, managing the statistical analysis process, special hardware, interpolation and estimation, and statistical security.

The purpose of this paper is not to survey existing work, but rather to bring up relevant issues even if little or no research on them yet exists. It is our hope that our discussion of such issues would bring to the attention of researchers the gap that exists between available commercial data management software and the tools needed to support statistical and scientific database applications. Such applications abound but they are often not visible because they are handled with special purpose software. In other cases, data that could be statistically analyzed yielding important benefits, are left idle (usually archived) because of the lack of appropriate tools or manpower for writing special purpose software. General purpose statistical and scientific data management tools could benefit both situations, by saving work to those who handle them with special purpose software, and by the ability to analyze important data that would be otherwise wasted.

Acknowledgement

This research was supported by the Applied Mathematics Sciences Research Program of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.

References

(1) Shoshani, A., "Statistical Databases: Characteristics, Problems, and Some Solutions", *Proceedings of the 8th International Conference on Very Large Data Bases (VLDB),* 1982, pp.208-222.

(2) Shoshani, A., Olken, F., Wong, H.K.T., "Characteristics of Scientific Databases", *Proceedings of the 10th International Conference on Very Large Data Bases (VLDB),* 1984, pp. 147-160.

(3) Bassiouni, M. A., "Data Compression in Scientific and Statistical Databases" in this issue of *IEEE Transactions on Software Engineering.*

(4) Samet, H., The Quadtree and Related Hierarchical Data Structures, *ACM Computing Surveys, 16,* 2, June 1984, pp. 187-260.

(5) Fushimi, S., Kitsuregawa, M., Nakayama, M., Tanaka, H., Moto-oka, T., "Algorithm and Performance Evaluation of Adaptive Multidimensional Clustering Technique," *ACM SIGMOD Proceedings of the International Conference on Management of Data,* 1985, pp. 308-318.

(6) Wong, H.K.T., Liu, H., Olken, F., Rotem, D., Wong, L., "Bit Transposed Files," *Proceedings of the 11th International Conference on Very Large Data Bases (VLDB),* 1985.

(7) Khoshafian, S.N., Bates, D.M., DeWitt, D.J., "Efficient Support of Statistical Operations", in this issue of *IEEE Transactions on Software Engineering.*

(8) Ozsoyoglu, G., Ozsoyoglu, Z.M., "Statistical Database Query Languages", in this issue of *IEEE Transactions on Software Engineering.*

(9) Shodgrass, R., Ahn, I., "A Taxonomy of Time in Databases," *ACM SIGMOD Proceedings of the International Conference on Management of Data,* 1985, pp. 236-246.

(10) Rowe, N.C., "Antisampling for Estimation: an Overview" in this issue of *IEEE Transactions on Software Engineering.*

(11) Denning, D.E., ***Cryptography and Data Security,*** Addison-Wesley, Reading, Mass, 1982.

This report was done with support from the Department of Energy. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the Department of Energy.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720