STATISTICAL DATABASES

A statistical database management system (SDBMS) is one that can model, store, and manipulate data in a manner well suited to the needs of users who want to perform statistical analyses on the data. Statistical databases have some special characteristics and requirements that are not supported by existing commercial database management systems. For example, while basic aggregation operations like SUM and AVG are part of SQL, there is no support for other commonly used operations like variance and co-variance. Such computations, as well as more advanced ones like regression and principal component analysis, are usually performed using statistical packages and libraries, such as SAS (1) and SPSS (2).

From the end-user's perspective, whether the statistical calculations are being performed in the database or in a statistical package can be quite transparent, especially from a functionality viewpoint. However, once the datasets to be analyzed grow beyond a certain size, the statistical package approach becomes infeasible, due to either its inability to handle large volumes of data, or the unacceptable computation times which make interactive analysis impossible. With the increasing sophistication of data collection instrumentation, and the cheap availability of large volume and high speed storage devices, most applications are today collecting data at unprecedented rates. In addition, an increasing number of applications today want the ability to perform interactive and on-line analysis of these data in real time, such as "what-if" analysis in forecasting. The emergence of multiple gigabyte corporate data warehouses (3), with on-line analytical processing (OLAP) (4–6) and data mining (7–8) type of analyses on them, is a good example of this trend. Hence, there is an increasing need for supporting statistical functions directly inside the database management system. This is precisely the goal of statistical database management, in addition to common requirements of regular DBMSs, such as data privacy, user-friendly query languages, consistency, and integrity.

In this article we provide a general overview of statistical database management, with specific focus on the various technical issues and proposed approaches. By its very nature, the treatment here is brief, and many details have been omitted. References to the original sources are provided, and the reader is invited to refer to them for further details.

REQUIREMENTS OF STATISTICAL DATABASES

SDB applications involve complex data sets from many fields, such as biology and physics, and different operations on them. They have requirements that far exceed the capabilities of current commercial DBMSs. In this section, we discuss issues including additional requirements of SDBs over the basic features of regular DBMSs, data model, query language, concurrency control, data integrity, backup/recovery, and physical storage. In addition, another important requirement of SDBs, which is to support data analysis, shall be described briefly.

Data Model

A data model provides an abstraction for representing the structure and semantics of data collected from the real world. The most popular data model is the relational model, where data are organized into a tabular format. This model is not adequate for storing data used for statistical purposes, because SDBs usually contain data that are not appropriate to store as a two-dimensional table, such as temporal or spatial data. Moreover, SDBs must store complex data objects such as points in space, images, and sequences. Representing these data objects in terms of relational tables loses a lot of semantics. Hence, data models that support complex structures of data directly and naturally are needed for SDBs.

Query Language

Every type of database needs some query language defined over the data model for accessing and manipulating data. SDBs require a more adequate query language than SQL, which has been shown to be a fairly good language for accessing data from traditional DBs. As an example, consider a query like "*find a subsequence of length* k *starting from position* n" from a sequence-structured data field. This type of query is not directly supported by SQL and is hard to obtain even if we add more features to SQL due to the relational nature of data storage. In general, complex data objects are composed of multiple structures which require a set of operations of their own.

SDB applications come from different disciplines, e.g. biology, earth science, and physics, and require highly specialized operations on their own specialized data. Usually, each of those data domains needs special operations on it.

Concurrency Control

For business-oriented DBs, such as banking or accounting, concurrency control is very important and is the main source of overhead for the whole system. In SDBs, concurrency control is not that important an issue and has different requirements due to four reasons. First, data are not updated frequently. Second, if data are to be updated, it is usually done by a single person or single program. In such a case, the update does not need to be visible right away and thus can be made on a simple locked version of the database. Third, even if multiple users are to update the database at the same time, they often access different parts of the DBs. Finally, long transactions are very common for SDBs, where a scientist may embark on a lengthy analysis.

For these reasons, concurrency control technique currently in use are not appropriate for SDBs. There is a need to support multiple versions of datasets, and to keep track of the correspondence between them. In addition, there is the need to support a master version, where different parts can be accessed by multiple users. Support for long transactions is another important requirement. Finally, for the massive amount of data stored in SDBs, the overhead of controlling concurrency is not acceptable. Fortunately, SDB applications do not have very strict consistency requirements.

Integrity Constraints

Since data in SDBs tend to have complex structure, it often results in complex constraints. Currently commercial DBMSs do not support such complex types of constraints. In general, a typical SDBMS should capture more semantics and the integrity constraints implied by these semantics. This would help users by saving the chore of expressing them explicitly.

Recovery

Mechanisms for backup and recovery are needed in any database with updates, especially if it has long-running activities. For SDBs, where long transactions are common, backing up to the last transaction is not acceptable since too much work will be lost. Mechanisms developed for supporting long transactions should be used. Some type of versioning mechanism with incremental checkpointing is desirable.

Physical Database Organization

Relational database implementations typically organize rows of relations as records in files and provide additional access paths, such as B-trees and hash tables, to access data efficiently. This approach is not sufficient for SDBs. The main reason is that alternate ways of clustering data provide more efficient access to statistical data. For example, SDBs often need to access a few columns from a table for, say, doing some aggregate operations. The "row-wise" type of storage is not efficient for such applications. The entire record must be read even though we only need a few attributes. Hence, for efficient performance, a SDBMS should provide various options of physical organization and appropriate query optimization techniques.

Data Analysis Requirements

The most important purpose of a SDBMS is to provide tools and mechanisms for users to analyze data faster and more easily. Conventional DBMSs are not capable of providing this in an efficient manner. Attempts have been made to feed data from traditional DBs to statistical packages to do the job, but this has been proven not to be a good approach, especially when the data sizes grow beyond some threshold. Moreover, the *bookkeeping* problem, namely the problem of keeping track of sets of analyses and intermediate results, has not been solved very successfully. More research needs to be done along these two dimensions.

STATISTICAL DATA MODELS AND METADATA

A data model in general means a notation for describing data, and a set of operations to manipulate them (9).

SDBs and commercial DBs differ in the nature of the raw data and the operations desired on them. Statistical data are more abstract, and operations on SDBs have different semantics than business data. Usually SDBs are analyzed by creating aggregate data from raw data. Aggregate data can be of many forms, such as a cross-table histogram, which are typically not supported by RDBMSs. Moreover, the relational model is also not suitable to handle such types of data. The main reason is the multidimensionality of statistical data (10–12). Thus, new data structures and operations are needed to handle this. Examples would be the data cube operator (6,11), and the Aggregate Data Structure (ADaS) (12). Another direction is to extend the relational data model to have set-valued relations and new operators (13). More information on operators for SDBMSs can be found in Refs. 14–16.

In addition, statistical objects, e.g., tables and histograms, have two types of attributes (12,17–19): (a) summary attributes representing the result of applying aggregation on raw data, and (b) descriptive attributes which describe associated summary attributes and are also called metadata. Great benefit can be gained from easy access to metadata, e.g., many analysis tasks prefer aggregates and descriptions rather than individual records. Thus, facilitating access to metadata is very important in SDBMSs. The reader is referred to Refs. 20–24 for more information on this.

Because of limited space, we shall briefly introduce the main statistical data models proposed so far in terms of the notations for describing data. Further details are provided in Ref. 25.

Subject

One of the early models was SUBJECT (17), introduced by Chan and Shoshani. The authors distinguished two types of abstraction in order to organize the statistical information, namely category attribute and summary attribute. The main advantage of modeling the semantics of category and summary attributes is the capability of automatic aggregation, i.e., the system is able to infer the attributes (category attribute) over which an aggregation should be applied (26).

These semantic concepts are hidden from the user and are represented as a graph. There are two kinds of nodes, namely cross-product nodes (X-node) and cluster nodes (C-node). The nodes can be connected by edges to form an acyclic graph. C-nodes represent a collection of items. X-nodes represent composite keys of category attributes. *Clustering* and *cross product* can be understood as two different types of relationships among categories in a SDB.

Cluster nodes are used to represent a hierarchy of parameters, each of which is a category attribute. This is a way to represent complex category attributes. Cluster nodes also represent the collection of summary attributes under the node labeled *variables.*

This graph structure is used to support a menu-driven interface. The user need not know the types of nodes, but the system can use them to provide automatic aggregation. The graph can either be browsed by moving up and down the nodes, or searched directly with keywords. The sharing of nodes provides the capability to use the same clusters across data sets, and to avoid confusion of names. The main advantage of this representation is that the user can be shown the content of the DB by gradually revealing additional detail when requested (17). As SUBJECT can model almost every statistical table, it was widely followed by statistical database researchers.

Semantic Association Model (SAM)

Su (19) and co-workers proposed SAM, which was designed for modeling both scientific-statistical databases and business-oriented databases. In SAM, each part of the real world can be modeled by a network of interrelated concepts. SAM distinguishes two types of concepts, i.e., atomic and nonatomic concepts.

An atomic concept is a nondecomposable, observable physical object, abstract object, event, or any data element that the user regards as an information unit. Its meaning is assumed to be understood, and thus need not to be redefined. An atomic concept can be represented by a simple data type such as integer, real, or character, or by a complex data object (CDT), which is a structured data type corresponding to an abstract data element from the user's point of view. A nonatomic concept is a physical object, an abstract object, or an event whose meaning is described in terms of other atomic and/or nonatomic concepts.

The grouping of atomic or nonatomic concepts to describe another nonatomic concept is called an association. Based on different structural properties, operational characteristics, and semantic constraints, Su classified associations into seven types: membership association, aggregation association, generalization association, interaction association, composition association, cross product association, and summarization association. He used a network representation, in which labeled nodes and arcs are used to describe the seven association types. Each nonatomic concept of an association type is labeled by the name of the association type. If the same nonatomic concept is seen by different users as having different semantic properties, the concept will be labeled with more than one association type. The conceptual description of a database can be given in terms of any number and structure of these association types, depending on the semantic complexity of the database.

Graphical Approach for Statistical Summaries (GRASS)

GRASS (27,28) is an extension of the SUBJECT model. It uses a directed, connected, and acyclic graph to represent the model. GRASS gives the statistical user an easy and immediate instrument to help understand the structure of the statistical database at a logical level. GRASS introduced five types of nodes, which are *marked* to distinguish types, and *labeled* to distinguish each node within the limits of the same type. The marks which distinguish the node types are S, T, A, C, and t_n. Their semantic descriptions follow:

- An *S node* represents the conceptual relation which exists between different nodes (of the S or T type) on a lower level of aggregation.
- A *T node* represents the summary data physically present in the database; the label expresses the event described and the type of data itself.
- An *A node* and a *C node* represent, respectively, the concepts of *aggregation* (or *cross product*) and *category attribute* (or *clustering*).
- A t_n *node* represents one of the assumable values within the limits of a data domain for a category attribute definition.

A T node is a *root* with respect to both the part of the graph made up of the S nodes, and the table trees formed by nodes C, A, and t_n.

If G is a GRASS graph, the connection rules between different nodes of G can be described as follows:

- R_1: A minimal graph consists of the following chain: S $\rightarrow$ T $\rightarrow$ A $\rightarrow$ C $\rightarrow$ t_n. All the edges are oriented toward a T-node.
- R_2: An S-node can be connected with one or more S-nodes and/or one or more T-nodes.
- R_3: A T-node can be connected with one or more S-nodes and/or one or more A-nodes.
- R_4: An A-node can be connected with one or more T-nodes, with two or more C-nodes or with one or more C-nodes and one or more A-nodes.
- R_5: A C-node can be connected with one or more A-nodes and with two or more t_n nodes, which have to be instances of the same domain.
- R_6: A t_n node can be connected with only one C-node.

We can have the same labels for different nodes (A and C) as long as the nodes are not common to the same table tree, i.e., they do not have the same T node as root.

Conceptual Statistical Model (CSM)

The CSM model was proposed by Batini and Di Battista (29) using two different data models for elementary and summary data. The authors justify this choice, as opposed to others such as Su (19), since from their point of view, this allows a cleaner description and comparison of the two types of data during the design process. Furthermore, they have provided the statistical model with new specific representation structures that allow more powerful modeling of aggregations. The following is a brief description of CSM.

First, CSM uses two different but complementary data models for the description of elementary and summary data, namely the ER model by Chen and redefinition of the GRASS model (27,28). Second, a conceptual schema is used, which consists of an effective tradeoff between top-down and bottom-up activities. Typical top-down activities are the design of the draft elementary schema, of the skeleton statistical schema, and the aggregation subschemas. A bottom-up activity can be the design of the statistical schema through incremental merging of aggregation subschemas. Third, seven representation structures are defined in the CSM that consist of various types of abstractions. As for other models, a conceptual schema in CSM can be represented by a diagram, in which each of the representation structures is labeled by a unique symbol. Symbols are connected by edges, according to their definition. The seven structures and their corresponding symbols are described briefly as follows:

- Class of objects (S) is defined as a set of objects of the real world that share common properties and are involved in some aggregation of interest. Classes in the statistical schema may be identical to the classes in the elementary schema or derived from them as the result of some manipulation.
- Category attribute (C) is a property of a class used in some statistical aggregation. It can be seen as a correspondence between a class and a set of values, called domain.
- Statistical classification (X) describes the relationship between a class of objects involved in some aggregation and the set of category attributes used in the aggregation.
- Class of data (D) can be built using three mechanisms, namely starting from a partition of a class of object, starting from another class of data, or starting from two or more classes of data. For example, starting from the class PERSON by SEX and AGE RANGE, we can build the class of data *number of persons by sex and age range,* counting the objects of each subset.
- Data view (V) allows the grouping together of classes of data with homogeneous characteristics. For instance, we can group classes of data referring to unemployed people, using a data view DATA_ON_LABOR_FORCES. A data view can also be defined from other data views.
- An aggregate (A) defined from category attributes $A_1, \ldots, A_n$, with domains $D_1, \ldots, D_n$, is a category attribute with domain $D_1 \times D_2 \ldots \times D_n$. Using aggregates, we may see a category attribute as expressed in terms of its component properties. In addition, aggregates are useful to express classifications defined over common attributes in a compact way.
- Grouping (*) is used to group the elements of the domain of an attribute according to some common property. This is similar to association in the SAM model.

Statistical Object Representation Model (STORM)

SUBJECT is useful to model actual statistical tables in print. However, it is unsatisfactory as a data model for a statistical database shared by many users. This is mainly because the same data may be described in different manners. As a simple example, columns and rows in a statistical table can be, in general, exchangeable without harm. However, SUBJECT cannot reflect this fact because it depends on the physical structure of the table.

For these reasons, later works on SDB tend to represent logical models separately from the physical structure of a statistical table. STORM (30,31), proposed by Rafanelli and Shoshani, is one such model. It is an enhancement of GRASS. While STORM adopts a graph representation resembling SUBJECT, it introduces conditions which make the description of statistical data clearer. These conditions can be summarized as follows:

1. There is only a single variable node in the graph tree and it points to the root X-node, i.e., only one summary attribute is allowed for the tree.
2. A single X-node is pointed to by the variable node.
3. Multiple C-nodes or X-nodes can point to an X-node.
4. Only a single type of C-node or an X-node can point to another C-node.
5. Condition 4 implies that a C-node is allowed only if it clusters categories belonging to a single domain. For example, a sex category and an age category must respectively belong to different C-nodes.

In summary, when modeling statistical data in a complex statistical table, we must decompose it into elementary statistical files.

STATISTICAL QUERY LANGUAGES

Most traditional databases have been developed for commercial business applications, which involve extensive decision-making activities. Such database management systems (DBMS) are not suitable for SDBs, which require extensive use of statistical analysis techniques. Statistical DBMSs (SDBMSs) are expected to provide users with rich internal modeling tools, and powerful and easy-to-use query languages to define and manipulate statistical data. In this section, we shall introduce statistical query languages following the taxonomy by Tansel (25). Most query languages can be evaluated based on the following criteria: data and metadata definition, data manipulation, interface to statistical packages, and the expressive power of the language.

For metatada definition we consider:

- Objects definable in the language
- Data description such as units of measure, missing values, data quality information, and universe description
- Footnotes
- Keywords
- Textual description
- Temporal data and time dimension, item editing specifications, and data structuring capabilities.

The most common SDB objects are summary tables and tabular representation of aggregated data. All statistical packages provide some type of summary table output formatting facilities. However, these are quite limited and mostly at an elementary level. There are many SDBMSs which include summary tables as a modeling object and provide powerful and easy-to-use summary table processing languages. STBE is a good example of such a language (32–34).

For data manipulation capabilities, we consider aggregation, subsetting and sampling, metadata manipulation, and handling time dimension explicitly.

The interface to the statistical package is another important aspect of a SDB query language. Basic statistical operations such as *avg, min, max, sum, count,* and *standard deviation* are included in almost all commercial DBMSs. However, sophisticated ones like correlation and principal component analysis are not, and hence the calculations are usually inefficient. The expressive power of a language can be determined by the possible objects and operations derivable from that language. Moreover, the language should be easy to use, and the syntax as well as functionalities should be well defined.

The following taxonomy of SDBMS query languages has been developed in Ref. 35:

- SDBMS built on top of CDBMS: the majority of the systems in this category are relational systems which are HSDB (36) on Model 204 (37), Ghosh's extension to SQL (38), System K on SQL/DS (39), GRAFSTAT (40) on DB2(SQL/DS), STRAND (41) on INGRES (42–44).
- Another approach is to use a Generalized Interface system that links together available CDBMS, statistical packages and graphics software, using a high-level interface language. Such systems are PASTE (45), SIBYL (46), GPI (47), and PEPIN-SICLA (48).
- Separately developed SDBMS: Systems in this category generally use relations as data modeling tools and an algebra- or calculus-based language. We further group them into six categories with respect to their data model and query languages:
  1. Relational data model (49) and relational query languages. These systems have been developed within the framework of the relational data model. They provide new internal (file) organization techniques and conceptual modeling tools suitable for SDBMS as well as well-defined aggregation operations in their query languages. Examples are RAPID (50) and CAS SDB (51) which use relational algebra, ABE (52–54) which uses relational calculus, SIR/SQL (55), GENISYS (56), and CANTOR (57) which use SQL, JANUS (58) and the algebra of Ref. 59 which uses tables and relational algebra like complete information (60). The July system uses a universal relation interface (61). Summary data model generates summary tables from the existing ones (62).
  2. Network and hierarchical data model. Examples are SIR/DBMS (55), TPL, and TPLDCS (63) and BROWSE (64).
  3. Formal extensions of relational model. Examples are SSDL (65), Klug's work (52–54) and extensions of Ozsoyoglu et al. (15,66).
  4. SDBMS with graphical user interfaces. These systems provide graphical, two-dimensional, and diagrammatic query languages in contrast to the traditional query languages in which query statements are coded. Examples are SUBJECT (17), GUIDE (67), ABE (52–54), STBE (32,34), SEEDS online code book (68), ALDS data editor (69), GRASP (70), and GRASS (27,28).
  5. Natural language based user interface: LIDS 86 (71).
  6. Query languages which calculate aggregates from temporal data. Examples are TQUEL (72), HQUEL (73,74), TBE (75), Tansel's extension to relational algebra (76), temporal data model of Shoshani and Segev (77–79), and the query language of TEER (80).

In this section, we have presented a taxonomy to classify the statistical query languages in the literature. The taxonomy is based on that of Ref. 35. The bibliographies on these systems have also been provided. Statistical query language and SDBMS systems have been studied since the early 1980s. However, there are still issues that need to be addressed like embedding security specification in the query language or a good query language for both temporal and spatial data. A full treatment of these issues is still open to investigation.

PHYSICAL DATABASE DESIGN AND INDEX STRUCTURES FOR STATISTICAL OPERATIONS

Physical data structures and access methods are critical for efficient query processing. A traditional DBMS often stores rows of relation tables as records in files and provides additional index mechanisms over the various columns of the relations. Some of the classical ones are VSAM (81–84), B-tree (85–87), GRID-file (88), and linear hashing (89), while some newer ones are BANG file (90), and R-tree (91). These data structures along with their access methods have significantly reduced the time required to access data from physical storage. However, SSDB applications need additional types of data organization and access algorithms, since most of the data structures mentioned are aimed at speeding up the processing of relational operators, primarily the join operator. For SSDBs, the best physical clustering of the data may not be according to records in a file. The main reason is that in SSDB applications, the access requirements would benefit from other ways of clustering data. For instance, spatial applications typically have local access operators such as "find the points that are close to me." For such applications, physical locality of data according to their spatial locality would reduce the amount or I/O from secondary storage. In this section, we briefly present TBSAM (92) as an example of a typical SSDB physical design and access method.

TBSAM (tree based statistics access method) is designed to efficiently process a class of aggregate queries such as:

Calculate ⟨*set-of-aggregates*⟩ of all data items
such that ⟨*boolean qualification*⟩

Here, aggregates are some overall characteristics of all the qualifying data items. Examples of such aggregates include descriptive and order statistics. This class of queries arises very naturally in applications such as scientific data analysis, planning, and forecasting.

TBSAM is based on the *B+_tree,* and it exploits all the benefits of a B+_tree's dynamic nature. It provides facilities for efficient evaluation of the arithmetic mean and higher moments of one or more attributes. The B+_tree index structure provides an ordering of the tuples of a relation on the index attribute. The aim is the efficient retrieval of a tuple, given the value of its index attribute. However, there is no proviso for retrieving a tuple whose rank in the order is specified instead of its index attribute value. This is the basic operation required in finding the median and other order statistics for a set of data items. This operation is supported in a natural and efficient manner by TBSAM. TBSAM can be used for performing statistical sampling on a relational database.

TBSAM is a dynamic index, and thus can support insertion/deletion/modification of tuples in the relation. These operations can be performed very naturally, and the cost is almost the same as that for the B+_tree.

Each *non-root* node of TBSAM of order m contains k keys, where $\lceil m/2 \rceil \leq k \leq m$. In addition, it also contains $k + 1$ pointers P_j, $0 \leq j \leq k$, that point to its subtrees. P_i points to the root of the subtree containing all keys K_j such that $K_i \leq K_j < K_{i+1}$. The leaf nodes point to data pages that contain tuples of the relation being indexed by the tree. Figure 1 shows a TBSAM index created on the key attribute K. The index provides an ordering of tuples on the key attribute.

Each node of the TBSAM index contains some information in addition to the keys and pointers. Beside K_i and P_i, there is a structure called S_i storing *statistics metadata* for the ith subtree. Statistics metadata is metadata that facilitates the processing of statistical queries such as *count*(i)—the number of data tuples in the ith subtree, or *Sum_Age*(i, p)—the sum of all the age values to the pth power, of all tuples in the ith subtree. The choice for the statistics metadata is not arbitrary. It is based on the aggregate property which all of the chosen quantities exhibit. This property of the tree can be used to efficiently process queries requiring order statistics (i.e., median and quantiles), and aggregates on the specified attributes.

Various types of statistical queries can be facilitated by the use of the TBSAM index. Examples of such query types are descriptive statistics, order statistics, or statistical sampling. Statistical sampling from relational databases has been discussed in detail by Olken (93). However, the approach taken is quite different. The query language has to be extended by adding a set of sampling operators.

In conclusion, data management systems for SSDBs need to have a rich set of physical organization options to provide efficient performance. TBSAM is only one of many examples of physical storage structures that an SSDB should adopt. A multiplicity of choices makes the problem of query optimiza-

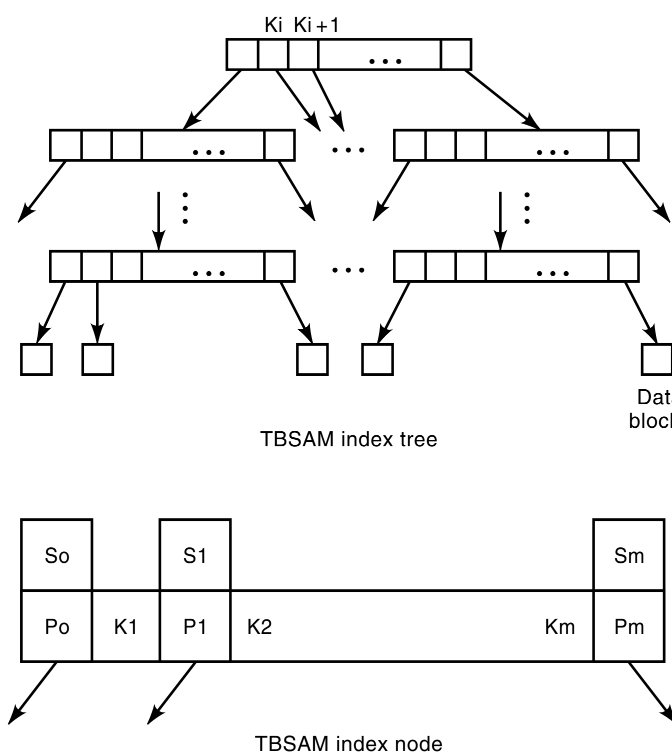

Figure 1. A sample TBSAM index structure.

tion much harder. The management of these options and the way they can interact is a great challenge to the designers of such systems. For more information, the reader is referred to Refs. 94 and 95.

QUERY PROCESSING AND OPTIMIZATION

A large portion of statistical data is either spatial or temporal data. Traditional relational DBMSs are not designed to deal with such types of data. The pure tables of relational databases are not capable of efficiently storing or helping retrieving such data. For example, with temporal data, a natural query would be to compute aggregates or moving averages and joins along the time dimension. With spatial data, the query users want to ask would be to get the average over some neighborhood around a given point. Standard techniques are not good enough for such queries. Of course, the issues involve not only the query processing techniques themselves, but also the query language and the physical storage structure of choice. In this section, we survey some query processing and optimization techniques that have been designed specially for temporal and spatial data.

The query processing and optimization techniques for relational databases have been well studied for a long time and are described in most of the basic database texts. The algorithms typically reorder the operations to be performed (join, select, group, etc.), build the optimal or suboptimal query processing tree, and then, depending on the physical data storage structures, choose the best possible strategy to query the data. Basically, the most expensive operation is the join operation and the principal focus has been on optimizing the join operation.

With respect to temporal data, most queries are asked along the time dimension. Usually, additional operators are needed, such as "overlaps," "starts," "equal," "during," and "finishes." Moreover, a traditional type of query can be issued based on some constraints on time, for example, join on a particular attribute where the join values are equal at the same time. For that reason, many new operations have been proposed on temporal data. We introduce a few of them here. "Temporal theta join" is made up of the conjunction of two sets of predicates, the time join predicate and the non-time join predicate. In "TE-join," or "Temporal Equijoin," two tuples (or rows) in two join relations (tables) are joined if their time intervals intersect. Intuitively, this is like "join them if they exist at the same time". A "T-join" causes the concatenation of tuples from the operand relations only if their time intervals intersect. No predicate on non-time attributes is specified. Semantically, T-join is just a TE-join with a null predicate on the non-time attributes. "Time Union Join" is characterized by a union operation on the time intervals. An "Event-join" groups several temporal attributes of an entity into a single relation. A good treatment on temporal join operation can be found in Ref. 25.

After new operators have been specified, there must be new strategies to process the temporal queries in an efficient way. The factors affecting processing optimization are physical data organization, indexing methods, metadata, architecture of the query processor, and how good the estimation of selectivities is. Query execution strategies for "TE-join" are given in Ref. 96, and for "Event join" in Ref. 97. Cliff Leung et al. presented a strategy for executing temporal semi joins in Ref. 98.

With respect to spatial data, the same situation exists, namely, tables are too simple to represent points or geometric objects in space. Queries like "find a shortest path from A to B" are not easy to perform in RDBMSs. Algorithms to answer such questions deterministically or approximately have been proposed (99,100). In addition, query processing techniques and indexing structures for spatial joins have also been developed by many researchers, e.g., Gunther (101) and Faloutsos et al. (102). Aref et al. (103) proposed several optimization strategies for *spatial queries.*

To summarize, statistical data and the queries on them have special characteristics, which are different from business data, and which require more complicated data structures, database operations, and query processing techniques. Most research on this topic has focused on some particular aspects of the problem, and thus the area is still quite open for research.

SAMPLING AND ITS ROLE IN STATISTICAL DATABASES

Obtaining information, whether from a database or the real world, requires time and effort. In many cases, we do not need the exact answers to our questions. In fact, in some cases even the concept of an exact answer may be undefined. For example, the quantity "number of hamburgers sold by McDonald's on December 5, 1997" may seem well defined, until we try to actually compute it. If we consider only the sales of McDonald's in a single time zone, then the calculation may be possible, but what if we cross time zones, especially as we consider the world-wide sales of McDonald's. Similarly, what exactly is meant by a "hamburger"? Do we include chicken burgers and fish filet sandwiches, and do we include mutton burgers popular in south-east Asia? However, answers to such questions are routinely required by the McDonald's corporation. In this case, an approximate answer or an estimate with a high degree of confidence is usually sufficient. Since estimates with high degrees of accuracy can be computed from samples of data, rather than scanning the entire data set, this approach can lead to a few orders of magnitude of savings in computational costs, at the expense of little or no loss of accuracy.

For large administrative, marketing, forecasting, and scientific databases, retrieval costs can be significant. For example, social security and tax record databases contain tens to hundreds of millions of records. High energy physics datasets often contain hundreds of gigabytes of data. Even if retrieval costs were negligible, sampling would still be important in order to reduce sample post-processing costs. Some of these costs may arise from extensive computation on each record (93). Finally, even if computing were free, sampling would still be important for those applications which require physical inspection of the real world objects which correspond to the sampled database records. Examples include sampling to audit financial databases (104,105), inspection of components for quality control (106,107), and medical examination of sampled patient data for epidemiological studies.

Random sampling (108) is typically used to support statistical analysis of a dataset, either to estimate parameters of interest (109,110) or for hypothesis testing. Applications include control systems, scientific investigations, product quality control, and policy analyses. Note that the accuracy of estimates from sampling is typically a function of the size of the sample, with little dependence on the population size. Hence, sampling is most advantageous when done from large databases.

Given that answering queries from a data sample is often much better than from the entire database, the next question is how to create a sample in the most cost-effective manner. The traditional approach in statistical analysis has been to use a library function to do sampling from a data set, either in a file or from a database. In either case, the entire data set must be read at least once. Recent research efforts (93,111) have shown that building sampling functionality into a database management system, and hence performing it as close as possible to the data source, gives substantial computational benefits. The efficiency gained arises from the reduction in the amount of data to be retrieved for sampling queries, and from exploiting the indices and access methods used in the DBMS. Instead of completely processing a database query and then sampling the result, the sampling and query operators can be interchanged, so that sampling is done prior to query processing. In a series of papers Olken and Rotem developed this idea, showing how to do sampled querying for a single relational operator (112) sampling from B+-trees (111), and sampling from hash files (93).

Sampling can also be used to provide estimates of the answers to aggregate queries, in applications where such estimates are adequate, and where the cost in time or money to fully evaluate the query may be excessive. Morgenstein (113) discusses the estimation procedures for various aggregate queries such as count, with some initial description of the use of sampling. Hou et al. (109) discussed the construction of statistical estimators for arbitrary relational expressions for COUNT aggregates, and their use in real-time applications (110). Sampling may also be used to estimate database parameters used by the query optimizer to choose query evaluation plans. Willard (114) discusses the determination of asymptotically optimal sample size for estimating the selectivity of a selection query, while Srivastava et al. (25) showed how to maintain these selectivities to specified degrees of accuracy. Lipton and Naughton (115) discuss the use of sampling to estimate the size of transitive closures. Denning (116) proposed the use of sampling as a means of providing security for individual data, while permitting access to statistical aggregates.

STATISTICAL DATABASE SECURITY

An important goal of statistical databases is to provide answers to aggregate queries. However, at the same time it must be ensured that sensitive information about individuals is not leaked. The problem becomes especially hard if we consider the fact that a series of aggregate queries, each of which by itself does not reveal sensitive information, can be used to infer sensitive information. This has been called the statistical database inference problem, and mechanisms to safeguard against it are called inference control mechanisms.

The data security problem is quite complex, as is illustrated by the following example from Ref. 117:

> A jeweler not only sells gems but also handles for his customers the jewels he has sold to them. In addition, to repair and service, the jeweler must sometimes keep his customers' jewels in his safe, especially if these are regular and important customers. He knows that the safety of the jewels is only as good as the safety of the safe. To this end he must have well-made safes with elaborate protection mechanisms and combinations. Quite often he has to transport the jewels from one place to another. In this situation, he must devise a secure procedure for transportation. Although safe may be included in the transportation, there is still a need for special vehicles, e.g. armored cars, to carry the safes. Furthermore, he must take great precautions when the jewels are moved from one vehicle to another. The security problem is compounded when one day the jeweler realizes that in an effort to safeguard the jewels, he has to keep them hidden, thereby interfering with one of the most important objectives of jewels, namely their display. He then ponders the question whether it is possible to simultaneously display jewels and ensure their security. He investigates the possibility of developing security mechanisms which are part of the jewels themselves, and hence fulfill both the objectives.

There is a straightforward analogy between jewels and data, and the jeweler and database administrator. In general, all security controls for data are divided into two classes, namely external and internal. External methods include personnel security, building security, physical security, etc.: that is, issues outside the computer system. Internal methods are usually divided into four categories (118):

- Access controls regulate which users may enter the system, and subsequently which data sets an active user may read or write,
- Flow controls regulate the dissemination of values among the data sets accessible to a user,
- Inference controls protect statistical databases by preventing questioners from deducing confidential information by posing carefully designed sequences of statistical queries and correlating the responses, and
- Data encryption attempts to prevent unauthorized disclosure of confidential information in transit or storage by encoding it.

We now give an example to illustrate the difficult nature of the problem of database inference.

Example: Consider a company's database with employee information given in Table 1. This database operates under the policy that salaries of individuals are confidential information, and should not be revealed. However, averages must have returned since this is a statistical database. To achieve this goal, the database does not return answers to queries like "what is the salary of the employee whose name is Jill?" Furthermore, it does not even answer aggregate queries where it determines that the average is being computed over a single record. Hence, it refuses to answer a query like "what

Table 1. An Example Table of a Company's Database

Name	Gender	Department	Salary
John	Male	Mathematics	20,000
Todd	Male	Computer Science	30,000
Jane	Female	Mathematics	26,000
Jill	Female	Computer Science	32,000

is the average salary of female employees who work for the Computer Science department?" Let the individual salaries of the four employees be s_1, s_2, s_3, and s_4, respectively. As the following sequence of queries shows, it is rather straightforward to determine each individual's salary.

- What is the average salary of female employees? → returns 29,000 = $(s_3 + s_4)/2$
- What is the average salary of male employees? → returns 25,000 = $(s_1 + s_2)/2$
- What is the average salary of mathematics employees? → returns 23,000 = $(s_1 + s_3)/2$
- What is the average salary of computer science employees? → returns 31,000 = $(s_2 + s_4)/2$

From the four equations above, each of s_1, s_2, s_3, and s_4 can be calculated.

For a good treatment of issues in statistical database security, the reader is referred to Refs. 119–127.

OTHER ISSUES

In addition to the principal technical areas discussed previously, the statistical database research community has explored a number of other issues as well, including data visualization, data integration, and statistical expert systems. Brief overview of these areas are provided.

- *Data visualization:* A data record can be considered a point in multidimensional space, where each attribute represents a dimension of the space. This can be used as a basis to build an interactive data visualization system where the user can browse in the multidimensional space. For example, Hinterberger (25,128) describes the use of data densities as a basis for developing a storage structure that is efficient for data management and visualization.
- *Data integration:* A statistical database can be thought of as a collection of data sets, where each set gives summary information about a certain population of objects, and is obtained by applying some aggregation function to a collection of observations. In a large statistical database one usually encounters several summary tables that share the same summary variable and the same population, but make use of different sets of category attributes, and possibly come from distinct data sources. Such summary tables are called homogeneous summary tables. Management of a statistical database containing homogeneous summary tables leads to problems in data integration. Details of this are provided in Malvestuto (129).
- *Statistical expert systems:* A statistical expert system is a program which can act in the role of an expert statistical consultant (25,130). It can give expert advice on how to design a study, what data to collect to answer the research questions, and how to analyze the data collected. Thus, the system advises on data analysis, carries it out, and discusses the results and further analysis directions with the analyst. The development of statistical expert systems has been motivated by a number of factors (25), including (a) very large data volume, (b) an effort to overcome the shortage of expert statisticians, (c) a desire to help the statistically unsophisticated researcher arrive at the right conclusion, and (d) a wish, from a statistical expert, for a second opinion.

CONCLUSION

The number of applications that collect vast amounts of data, and require interactive real-time analysis capabilities on it, is on the rise. A large part of these analyses are the computation of statistical parameters from the data set. In this environment, the standard approach of statistical analysis to load part of the data from a file or database into a statistical package, and then perform analysis on it will not work due to efficiency and flexibility reasons. The overall goal of research in statistical database management has been to make this analysis an integral part of the data management system itself. The focus of the research community has been on developing techniques to make this happen. In this article we have provided a brief overview of the issues in statistical database management, and encourage the interested reader to follow up details from the references. The edited collection of papers by Michalewicz (25) is a very good starting point for this exploration.

BIBLIOGRAPHY

1. SAS Institute, *SAS/INSIGHT User's Guide: Version 6,* 3 ed., Cary, NC: SAS Institute, 1995.
2. SPSS, Inc., *SPSS Reference Guide,* Englewood Cliffs, NJ, Chicago, IL: Prentice-Hall, and SPSS, Inc., 1990.
3. W. Inmon, *Building the Data Warehouse,* New York: Wiley, 1990.
4. G. Colliat, OLAP, relational, and multidimensional database systems, *SIGMOD Rec. (ACM Special Interest Group Manage. Data),* **25** (3): 64–69, 1996.
5. A. Shoshani, OLAP and statistical databases: Similarities and differences, *PODS '97. Proc. 16th ACM SIG-SIGMOD-SIGART Symp. Principles Database Syst.,* ACM, ed., 1997, Tucson, Arizona. New York: ACM Press, 1997, pp. 185–196.
6. V. Harinarayan, A. Rajaraman, and J. D. Ullman, Implementing data cubes efficiently, *SIGMOD Record (ACM Special Interest Group on Management of Data),* **25** (2): 205–216, 1996.
7. H. Mannila, Data mining: Machine learning, statistics, and databases, in P. P. Svensson and J. C. J. C. French (eds.), *Proc. 8th Int. Conf. Sci. Stat. Database Syst., Stockholm, Sweden, 1996* (1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA), Los Alamitos, CA: IEEE Computer Society Press, 1996, pp. 2–11.
8. J. Han, Data mining techniques, *SIGMOD Record (ACM Special Interest Group on Management of Data),* **25**: 545, June 1996.
9. J. D. Ullman, *Principles of Database Systems,* Rockville, MD: Computer Science Press, 1982.
10. A. Shoshani and H. K. T. Wong, Statistical and scientific database issues, *IEEE Trans. Softw. Eng.,* **SE-11**: 1040–1047, 1985.
11. J. Gray, et al., Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals, *Proc. 7th Int. Conf. Data Eng.,* New Orleans, Louisiana, 1996, pp. 152–159.
12. A. Bezenchek, M. Rafanelli, and L. Tininini, A data structure for representing aggregate data, in P. P. Svensson and J. C. J. C. French (eds.), *Proc. 8th Int. Conf. Sci. Stat. Database Syst., Stockholm, Sweden*, Los Alamitos, CA: IEEE Computer Society Press, 1996, pp. 22–31.

13. G. Özsoyoğlu, Z. M. Özsoyoğlu, and V. Matos, Extending relational algebra and relational calculus with set-valued attributes and aggregate functions, *ACM Trans. Database Syst.,* **12** (4): 566–592, 1987.
14. J. E. Gentle and J. Bell, Special data types and operators for statistical data, *IEEE Comput. Soc. Tech. Commun.,* **7** (1): 1984.
15. S. N. Khoshafian, D. M. Bates, and D. J. DeWitt, Efficient support of statistical operations, *IEEE Trans. Softw. Eng.,* **11** (10): 1058–1070, 1985.
16. M. Rafanelli and R. F. L. Mefisto, A functional model for statistical entities, *IEEE Trans. Knowl. Data Eng.,* **5**: 670–681, 1993.
17. P. Chan and A. Shoshani, SUBJECT: A directory driven system for large statistical databases, *Proc. LBL Workshop Stat. Database Manage.,* Lawrence Berkeley Lab, Berkeley, CA, 1981.
18. M. Rafanelli and F. L. Ricci, Proposal of a logical model for statistical database, *Proc. 2nd Int. Workshop Statistical Database Manage.,* Los Altos, CA, 1983.
19. S. Su, SAM*: A semantic association model for corporate and scientific-statistical databases, *J. Inf. Sci.,* **29** (2–3): 151–199, 1983.
20. J. L. McCarthy, Metadata management for large statistical databases, *Proc. 8th Conf. Very Large Databases,* Mexico City, Morgan Kaufman, San Mateo, CA, 1982, pp. 234–243.
21. G. M. van den Berg and E. de Feber, Definition and use of metadata in statistical data processing, in *Proc. 6th Int. Conf. Stat. Sci. Manage.,* Ascona, Switzerland, 1992, pp. 290–306.
22. J. P. Kent and S. M., Some thoughts about a metadata management system, *Proc. 9th Int. Conf. Sci. Stat. Databases,* Olympia, WA, 1997, Piscataway, NJ: IEEE Press, pp. 155–164.
23. A. Westlake, A simple structure for statistical meta-data, *Proc. 9th Int. Conf. Sci. Stat. Databases,* Olympia, WA, Piscataway, NJ: IEEE Press, 1997, pp. 186–195.
24. S. P. Ghosh, *Statistical Metadata,* vol. 8. New York: Wiley, 1988.
25. Z. Michalewicz, (ed.), *Statistical and Scientific Databases,* Chichester, UK: Ellis Horwood, 1991.
26. A. Shoshani, Statistical databases: Characteristics, problems, and some solutions, *Proc. 8th Int. Conf. Very Large Data Bases (VLDB),* 1982, pp. 208–222.
27. A. Balsami, M. Rafanelli, and F. L. Ricci, Grass: A logical model for statistical databases, Tech. Rep., Rome, 1982.
28. M. Rafanelli and F. L. Ricci, A visual interface for browsing and manipulating statistical entities, *Proc. 5th Int. Conf. Sci. Stat. Database Manage.,* 1990, pp. 163–182.
29. G. DiBattista and C. Batini, Design of statistical databases: A methodology for the conceptual step, *J. Inf. Syst.,* **13** (4): 407–422, 1988.
30. M. Rafanelli and A. Shoshani, STORM: a statistical object representation model, *IEEE CS Tech. Com. Database Eng. Bull.,* **13**: Sept. 1990.
31. M. Rafanelli and A. Shoshani, STORM: A statistical object representation model, *Proc. 5th Int. Conf. Sci. Stat. Database Manage.,* 1990, pp. 14–29.
32. Z. M. Özsoyoğlu and G. Özsoyoğlu, Summary-table-by-example: A database query language for manipulating summary data, *Proc. Int. Conf. Data Eng.,* Los Angeles, CA, Los Alamitos, CA: IEEE Computer Society Press, 1984, pp. 193–202.
33. G. Özsoyoğlu, V. Matos, and Z. M. Özsoyoğlu, Query processing techniques in the summary-table-by-example database query language, *ACM Trans. Database Syst.,* **14** (4): 526–573, Dec. 1989.
34. G. Ozsoyoglu and W. A. Abdul-Qader, Human-factors study of two screen-oriented query languages: STBE and QBE, *Inf. Softw. Techn.,* **34** (1): 1992.
35. G. Ozsoyoglu and Z. M. Ozsoyoglu, Statistical database query languages, *IEEE Trans. Softw. Eng. (TSE),* **11**: 1071–1081, 1985.
36. H. Ikeda and Y. Kobayashi, Additional facilities of a conventional dbms to support interactive statistical analysis, *Proc. LBL Workshop Stat. Database Manage., Lawrence Berkeley Lab,* Berkeley, CA, 1981.
37. C. C. Computer Corporation of America, *File manager's technical reference manual, model 204 database management system.*
38. S. P. Ghosh, Statistical relational tables for statistical database management, *IEEE Trans. Softw. Eng.,* **12**: 1106–1116, 1986.
39. D. Maier and C. Cirilli, SYSTEM/K: A knowledge based management system, *Proc. 2nd Int. Workshop Stat. Database Manage.,* Los Altos, CA, 1983.
40. D. M. Stein, A database interface to an integrated data analysis and plotting tool, *Proc. 3rd Int. Workshop Stat. Database Manage.,* Luxembourg, GD Luxembourg, 1986.
41. R. Johnson, Modeling summary data, *Proc. ACM SIGMOD Conf.,* Ann Arbor, Michigan, 1981, pp. 93–97.
42. M. Stonebraker et al., The design and implementation of INGRES, *ACM Trans. Database Syst.,* **1** (3): 189–222, Sept. 1976.
43. C. J. Date, *A Guide to Ingres,* Reading, MA: Addison-Wesley, 1987.
44. M. Stonebraker (ed.), *The Ingres Papers,* Reading, MA: Addison-Wesley, 1986.
45. S. Weiss and P. Weeks, PASTE: A tool to put application systems together easily, *Proc. 2nd Int. Workshop Stat. Database Manage.,* Los Altos, CA, 1983.
46. S. Heiler and R. Bergman, SIBYL: An economist's workbench, *Proc. 2nd Int. Workshop Stat. Database Manage.,* Los Altos, CA, 1983.
47. L. Hollabaugh and L. Reinwald, GPI: A statistical package/database interface, *Proc. LBL Workshop Stat. Database Manage.,* Lawrence Berkeley Lab, Berkeley, CA, 1981.
48. P. Boufares et al., La version sm90 du sgbd relationnel pepin, *Journees SM90,* 1985.
49. E. F. Codd, A relational model of data for large shared data banks, *CACM: Commun. ACM,* **13** (6): 377–387, 1970.
50. M. T. Turner, R. Hammond, and P. Cotton, A DBMS for large statistical databases, *Proc. Int. Conf. Very Large Data Bases,* Rio de Janeiro, 1979, p. 319.
51. S. Johji and H. Sato, Statistical database research project in Japan and the cas sdb project, *Proc. 2nd Int. Workshop Stat. Database Manage.,* Los Altos, CA, 1983.
52. A. Klug, ABE—a query language for constructing aggregates-by-example, *Proc. LBL Workshop Stat. Database Manage.,* Berkeley, CA, 1981.
53. A. Klug, Access paths in the 'ABE' statistical query facility, *19 CM SIGMOD Conf. Manage. Data,* Orlando, FL, 1982.
54. A. Klug, Investigating access paths for aggregates using the ABE statistical query facility, *IEEE Data Eng. Bulletin,* **5**: 52–55, 1982.
55. G. Anderson et al., An integrated research support system for interpackage communication and handling large volume output from statistical database analysis operation, *Proc. 2nd Int. Workshop Stat. Database Manage.,* Los Altos, CA, 1983.
56. S. M. Dintelman and A. T. Maness, An implementation of a query language supporting path expressions, *Proc. 1982 ACM SIGMOD Int. Conf. Manage. Data,* Orlando, Florida, 1982, pp. 87–93.
57. I. Karasolo and P. Sevenson, An overview of CANTOR—a new system for data analysis, *Proc. 2nd Int. Workshop Stat. Database Manage.,* Los Altos, CA, 1983.
58. J. Klensin, A statistical database component of a data analysis and modeling system: Lessons from eight years of user experi-

ence, *Proc. 2nd Int. Workshop Stat. Database Manage.,* Los Altos, CA, 1983.

59. E. Fortunato et al., An algebra for statistical data, *Proc. 3rd Int. Workshop Stat. Database Manage.,* Luxembourg, GD Luxembourg, 1986.
60. C. Chan and Z. Michalewicz, A query language capable of handling incomplete information and statistics, *Proc. 3rd Int. Workshop Stat. Database Manage.,* Luxembourg, GD Luxembourg, 1986.
61. A. D'attri and F. Ricci, Interpretation of statistical queries to relational databases, *Proc. 4th Int. Working Conf. Stat. Sci. Database Manage.,* Roma, Italy, Springer-Verlag Lecture Notes In Computer Science, 1988.
62. M. Chen, L. McNamee, and M. Melkanoff, A model of summary data and its applications, in *Proc. 4th Int. Working Conf. Stat. Sci. Database Manage.,* Roma, Italy, Springer-Verlag Lecture Notes In Computer Science, 1988.
63. W. Weiss, P. Weeks, and M. Byrd, Must we navigate through databases? *Proc. LBL Workshop Stat. Database Manage.,* Lawrence Berkeley Lab, Berkeley, CA, 1981.
64. G. G. Hendrix et al., Developing a natural language interface to a complex system, *ACM Trans. Database Syst.,* **3** (2): 105–147, 1978. Also published as Proc. 3rd Conf. Very Large Databases, Tokyo, Japan, Morgan Kaufman, San Mateo, CA: 1977, pp. 292–300.
65. W. Brown, S. Navathe, and S. Su, Complex data types and a data manipulation language for scientific and statistical databases, *Proc. 2nd Int. Workshop Stat. Database Manage.,* Los Altos, CA, 1983.
66. G. Ozsoyoglu and M. Ozsoyoglu, Features of ssdb, *Proc. 2nd Int. Workshop Stat. Database Manage.,* Los Altos, CA, 1983.
67. H. K. T. Wong and I. Kuo, GUIDE: Graphical user interface for database exploration, *Proc. 8th Conf. Very Large Databases,* San Mateo, CA: Morgan Kaufman, Mexico City: McLeod and Villasenor, 1982.
68. D. Merrill et al., Distributed data management in a minicomputer network, *Proc. 2nd Int. Workshop Stat. Database Manage.,* Los Altos, CA, 1983.
69. J. Thomas and D. Hall, Alds project: Motivation, statistical database management issues, perspectives, and directions, *Proc. 2nd Int. Workshop Stat. Database Manage.,* Los Altos, CA, 1983.
70. T. Catarci and G. Santucci, GRASP: A graphical system for statistical databases, *Proc. 5th Int. Conf. Sci. Stat. Database Manage.,* Charlotte, NC, 1990, pp. 148–162.
71. H. Sato, A data model, knowledge base and natural language processing for sharing a large statistical database, *Lecture Notes In Computer Science,* Rome, Italy, New York: Springer Verlag, 1988.
72. R. T. Snodgrass, The temporal query language TQuel, *Symp. Principles Database Syst.,* 1984, pp. 204–213.
73. A. U. Tansel and M. E. Arkun, HQUEL, A query language for historical relational databases, *Proc. 3rd Workshop Stat. Sci. Databases,* Luxembourg, 1986.
74. A. U. Tansel, *A historical query language,* Tech. Rep., Bernard M. Baruch College, CUNY, New York, 1987. Revised version of *TANSEL86A: HQUEL, A Query Language for Historical Relational Databases.*
75. A. Tansel, M. E. Arkun, and G. Ozsoyoglu, Time-by-example query language for historical databases, *IEEE Trans. Softw. Eng.,* **SE-15**: 464–478; **15**: Apr. 1989.
76. A. U. Tansel, A statistical interface for historical relational databases, *Proc. IEEE CS Int. Conf. No. 3 Data Eng.,* Los Angeles, 1987.
77. A. Shoshani and K. Kawagoe, Temporal data management, *Proc. 22nd Int. Conf. Very Large Data Bases (VLDB),* 1986, pp. 79–90.
78. A. Segev and A. Shoshani, Logical modeling of temporal data, *19 ACM SIGMOD Conf. Manage. Data,* 1987.
79. A. Shoshani and A. Segev, Modeling temporal semantics, in M. L. Colette Rolland, Francois Bodart (eds.), *IFIP TC 8/WG 8.1 Working Conf. Temporal Aspects Inf. Syst.,* Sophia-Antipolis, France, 1987.
80. R. Elmasri and V. Kouramajian, A temporal query language based on conceptual entities and roles, *Lecture Notes in Computer Science, Springer-Verlag,* **645**: 1992.
81. D. G. Keehn and J. O. Lacy, VSAM data set design parameters, *IBM Syst. J.,* **13** (3): 186–212, 1974.
82. Anonymous, Correction: VSAM data set design parameters, *IBM Syst. J.,* **13** (4): 352, 1974.
83. B. Musteata and R. Lesser, *VSAM Techniques: System Concepts and Programming Procedures,* QED Inf. Sci., 1987.
84. J. Ranade, *VSAM: Performance, Design, and Fine-Tuning,* New York: Macmillan, 1987.
85. D. J. Taylor and J. P. Black, A locally correctable B-tree implementation, *Comput. J.,* **29** (3): 269–276, 1986.
86. B. C. Desai, P. Goyal, and F. Sadri, Composite B-tree: an access aid for query processing and integrity enforcement, *Comput. J.,* **35** (3): A215–A225, 1992.
87. G. Held and M. Stonebraker, B-trees re-examined, *Commun. Assoc. Comput. Mach.,* **21** (2): 139–143, Feb. 1978.
88. H. M. Blanken et al., The generalized grid-file: Description and performance aspects, *Proc. IEEE Int. Conf. Data Eng.,* Los Angeles, CA, 1990, p. 380.
89. P. A. Larson, Linear hashing with separators—A dynamic hashing scheme achieving one-access retrieval, *ACMTDS: ACM Trans. Database Syst.,* **13** (3): 366–388, 1988.
90. M. Freeston, The BANG file: A new kind of grid file, *ACM SIGMOD Int. Conf. Manage. Data,* San Francisco, 1987, pp. 260–269.
91. T. Brinkhoff, H.-P. Kriegel, and B. Seeger, Efficient processing of spatial joins using R-trees, Tech. Rep., Washington, DC, 1993.
92. J. Srivastava, J. S. Tan, and V. Y. Lum, TBSAM: An access method for efficient processing of statistical queries, *IEEE Trans. Knowl. Data Eng.,* **1**: 414–423, 1989.
93. F. Olken, D. Rotem, and P. Xu, Random sampling from hash files, *Proc. 1990 ACM SIGMOD Int. Conf. Manage. Data,* **19**: New York, 1990, pp. 375–386.
94. B. R. Iyer and D. Wilhite, Data compression support in databases, in J. Bocca, M. Jarke, and C. Zaniolo (eds.), *20th Int. Conf. Very Large Data Bases,* 1994, Santiago, Chile proceedings, San Mateo, CA: Morgan Kaufmann, 1995, pp. 695–704.
95. M. A. Bassiouni, Data compression in scientific and statistical databases, *IEEE Trans. Softw. Eng.,* **11**: 1047–1058, 1985.
96. H. Gunadhi and A. Segev, Query processing algorithms for temporal intersection joins, *Proc. Int. Conf. Data Eng. (ICDE),* Kobe, Japan, 1991, pp. 336–344.
97. A. Segev and H. Gunadhi, Event-join optimization in temporal relational databases, *Proc. 15th Int. Conf. Very Large Data Bases,* Amsterdam, The Netherlands, 1989, pp. 205–215.
98. T. C. Leung and R. Muntz, Query processing for temporal databases, *Proc. 6th Int. Conf. Data Eng.,* Los Angeles, California, 1990, pp. 200–208.
99. S. Shekhar, A. Kohli, and M. Coyle, Path computation algorithms for advanced traveller information system, *Proc. 9th Int. Conf. Data Eng.,* Vienna, Austria, 1993, pp. 31–39.
100. T. Brinkoff, H. Kriegel, and R. Schneider, Query processing in spatial database systems, *Proc. 9th Int. Conf. Data Eng.,* Vienna, Austria, 1993, pp. 40–49.
101. O. Gunther, Efficient computation of spatial joins, *Proc. 9th Int. Conf. Data Eng.,* Vienna, Austria, 1993, pp. 50–59.
102. C. Faloutsos and Y. Rong, A spatial access method using fractals, *Proc. 7th Int. Conf. Data Eng.,* Kobe, Japan, 1991, pp. 152–159.

103. W. G. Aref and H. Samet, Optimization strategies for spatial query processing, *Proc. 17th Int. Conf. Very Large Data Bases,* San Mateo, CA: 1991, pp. 81–90.
104. H. Arkin, *Handbook of Sampling for Auditing and Accounting,* New York: McGraw-Hill, 1984.
105. D. A. Leslie, A. D. Teitlebaum, and R. J. Anderson, *Dollar Unit Sampling,* Belmont, CA: Copp Clark Pitman., 1979.
106. D. Montgomery, *Introduction to Statistical Quality Control,* New York: Wiley, 1985.
107. H. L. et al., *Frontiers in Statistical Quality Control 2,* Wurzburg, Germany: Physica-Verlag, 1984.
108. W. G. Cochran, *Sampling Techniques,* New York: Wiley, 1977.
109. W.-C. Hou, G. Ozsoyoglu, and B. K. Taneja, Statistical estimators for relational algebra expressions, *Proc. ACM SIGMOD Int. Conf. Manage. Data,* 1988, pp. 288–293.
110. W.-C. Hou, G. Ozsoyoglu, and B. K. Taneja, Processing aggregate relational queries with hard time constraints, *Proc. ACM SIGMOD Int. Conf. Manage. Data,* 1989, pp. 68–77.
111. F. Olken and D. Rotem, Random sampling from b+_tree files, *Proc. 15th Int. Conf. Very Large Data Bases,* Amsterdam, The Netherlands, 1989, pp. 269–277.
112. F. Olken and D. Rotem, Simple random sampling from relational databases, *Proc. 12th Int. Conf. Very Large Data Bases,* 1986.
113. J. Morgenstein, *Computer Based Management Information Systems Embodying Answer Accuracy as a User Parameter,* PhD thesis, Univ. California, Berkeley, 1980.
114. D. Willard, Sampling algorithms for differential batch retrieval problems, *Proc. ICALP-84,* New York: Springer-Verlag, 1984.
115. R. J. Lipton and J. F. Naughton, Estimating the size of generalized transitive closures, *Proc. 15th Int. Conf. Very Large Data Bases,* 1989.
116. D. E. Denning, Secure statistical databases with random sample queries, *ACM Trans. Database Syst.,* **5** (3): 291–315, 1980.
117. D. Hsiao and R. Baum, Information secure systems, *Advances in Computers,* **14**: 231–272, 1976.
118. D. Denning and P. Denning, Data security, *ACM Trans. Database Syst.,* **11** (3): 227–249, 1979.
119. D. Denning, *Cryptography and Data Security,* Reading, MA: Addison-Wesley, 1982.
120. T. S. Hsu and M. Y. Kao, Security problems for statistical databases with general cell suppressions, *Proc. 9th Int. Conf. Sci. Stat. Databases,* Olympia, WA, IEEE Press, Piscataway, NJ: 1997, pp. 155–164.
121. D. Gusfield, A graph theoretic approach to statistical data security, *SIAM J. Comput.,* **17** (3): 552–571, 1988.
122. M.-Y. Kao, Data security equals graph connectivity, *SIAM J. Discrete Math.,* **9** (1): 87–100, 1996.
123. M.-Y. Kao and D. Gusfield, Efficient detection and protection of information in cross tabulated tables. I. Linear invariant test, *SIAM J. Discrete Math.,* **6**: 460–476, 1993.
124. N. R. Adam and J. C. Wortmann, Security-control methods for statistical databases: A comparative study, *ACM Comput. Surveys,* **21**: 515–556, 1989.
125. F. M. Malvestuto, M. Moscarini, and M. Rafanelli, Suppressing marginal cells to protect sensitive information in a two-dimensional statistical table (extended abstract), in *PODS '91. Proc. 10th ACM SIGACT-SIGMOD-SOGART Symp. Principles Database Syst.: 1991,* Denver, Colorado, in ACM (ed.), *Journal of Computer and Systems Sciences,* vol. 51(2), New York: ACM Press, 1991, pp. 252–258.
126. F. M. Malvestuto, A universal-scheme approach to statistical databases containing homogeneous summary tables, *ACM Trans. Database Syst.,* **18** (4): 678–708, 1993.
127. M.-Y. Kao, Total protection of analytic-invariant information in cross-tabulated tables, *SIAM J. Comput.,* **26** (1): 231–242, 1997.
128. H. Hinterberger, The growth of differently structured grid file directories, 1990.
129. F. M. Malvestuto and M. Moscarini, Query evaluability in statistical databases, *IEEE Trans. Knowl. Data Eng.,* **2**: 425–430, 1990.
130. D. Hand, Emergent themes in statistical expert systems, *Knowl., Data, Comput.-Assisted Decisions,* 1990.

JAIDEEP SRIVASTAVA
HUNG Q. NGO
University of Minnesota

STATISTICAL DESIGN. See CAD FOR MANUFACTURABILITY.

STATISTICAL ESTIMATION. See ESTIMATION THEORY.

STATISTICAL PATTERN RECOGNITION. See PATTERN RECOGNITION.