Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

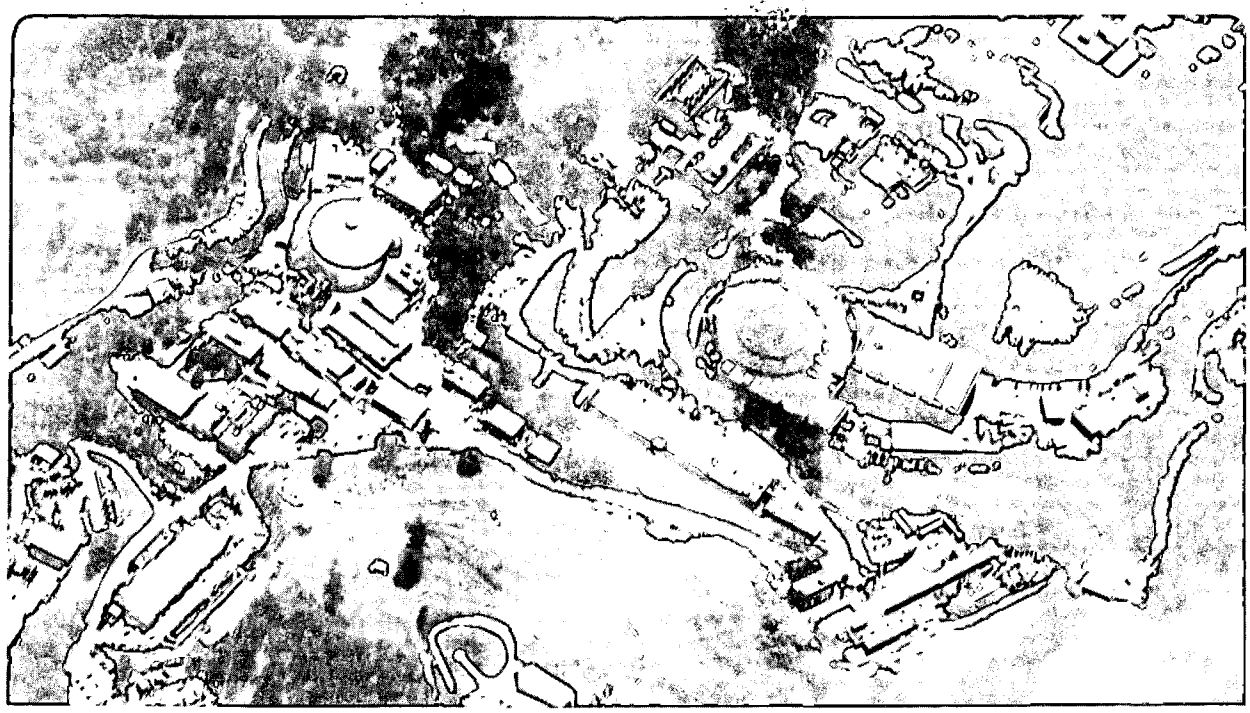
Computing Division

Presented at the AICA Annual Conference on
Information and Automatic Computation,
Rome, Italy, October 23-26, 1984

DATA MANAGEMENT ISSUES OF STATISTICAL DATABASES

A. Shoshani

August 1984



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

DISCLAIMER

LBL-18814

Data Management Issues of Statistical Databases

Arie Shoshani

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

August, 1984

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.

DATA MANAGEMENT ISSUES OF STATISTICAL DATABASES

Arie Shoshani

Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

SUMMARY

In this paper we describe the nature of statistical data bases and the special problems associated with them. We first describe the characteristics of statistical data bases in terms of data structures and usage. Then, we describe several problems unique to statistical databases, and when appropriate discuss some solutions or work in progress. The problems and solutions are organized into the following areas: physical organization, logical modeling, user interface, and the integration of statistical analysis and data management functions.

1. INTRODUCTION

Statistical data bases (SDBs) can be described in terms of the type of data they contain, and their use. SDBs are primarily collected for statistical analysis purposes. They typically contain both parameter data and measured data (or "variables") for these parameters. For example, parameter data consists of the different values for varying conditions in an experiment; the variables are the measurements taken in the experiment under these varying conditions. The data base is usually organized into "flat files" or tables.

The statistical analysis process involves the selection of records (or tuples) using selection conditions on the parameters, taking a random sample, or using a graphics device to point to the items desired. Several variables are then selected for analysis. The analysis may involve applying simple univariate statistical functions to the value sets of the variables (e.g. sum, mean, variance) or using more complex multivariate analysis tools (e.g. multiple regression, log-linear models).

The statistical analysis process may involve several steps. It includes phases of data checking, exploration, and confirmation. The purpose of data checking is to find probable errors and unusual but valid values (called "outliers" by statisticians), by checking histograms or integrity constraints across attributes. The purpose of data exploration is to get an impression of the distribution of variables and the relationships between them. This phase involves taking samples of the data, selecting records, and creating temporary data sets for use in graphical display and preliminary analysis. In the conformation phase, the analyst tests hypothesized distributions (which are based on the observations made in the exploratory phase) against the data base, or relationships between variables (cross tabulations). This process may then iterate several times until satisfactory results are achieved. A more detailed description of the statistical analysis process can be found in [Boral et al 82]. A compact description of data manipulation capabilities that are important for SDBs can be found in [Bragg 81].

At first glance it appears that the necessary data management functions can be supported by existing generalized data management systems. For example, one can view flat files as relations in a relational data management system, and the generation of subsets for analysis by using relational operators, such as "join" and "project". However, practice has shown that these data management systems have not been used for SDBs. Instead, one finds that statistical packages are used or special purpose software is developed for the particular data base in hand, or for a collection of data bases with similar characteristics. A case in point is the Census Data Base, which is collected and processed by special purpose software, distributed as flat files whose descriptions are quite complex, and therefore requires special purpose software for subsequent querying. An example of such a system which was designed specifically to manage geographically-based data is the Social, Economic, Environmental, Demographic Information System (SEEDIS), developed at at Lawrence Berkeley Laboratory [McCarthy et al 82].

Another approach to managing SDBs is by using statistical packages such as SAS [SAS 79] or SPSS [Nie et al 75]. While these packages have some data management capabilities their primary purpose is to provide statistical analysis tools to the analyst.

There are two main reasons for the fact that commercial data management systems have not been widely used for SDBs. The first reason is the storage and access inefficiency of these systems for SDBs. As will be discussed later, many SDBs have a high degree of data redundancy that can benefit from sophisticated compression techniques. The organization of the data into records (or tuples) makes retrieval inefficient in those cases where only a few attributes are needed for the analysis. Other data organization methods, such as organizing the data by columns instead of rows (called "transposed files") are usually more efficient [Teitel 77, Turner et al 79]. Most existing data management systems are designed for high volume interactive transactions with the possibility of concurrent access to the data. The large overhead required for the support of concurrent access is not necessary for SDBs. Analysts work with their particular subset of the data, and are willing to put up with occasional sequential access to the original data bases. A short discussion of additional reasons can be found in [Cohen & Hay 81].

The second reason stems from the lack of functionality and ease of use. Statistical functions available in commercial data management systems are quite limited, usually to simple aggregate univariate functions such as sum, maximum, or average. Most systems do not have facilities for supporting additional user-defined functions, although some provide an ability to create predefined functions in libraries. In addition, some query languages are quite complex when it comes to specifying aggregate functions. This is in part because of insufficient modeling of the SDBs, as will be discussed later.

Ease of use considerations are much more pragmatic. In order to perform statistical analysis, an analyst must eventually rely on more sophisticated statistical tools such as those found in statistical packages. This means that in order to use a data management system the analyst will need to become familiar with two systems, and the methods used to pass data between them. Often, the analyst will choose to stay with the essential statistical tools provided by the statistical package, and manage with the limited data management tools provided by them. However, for many applications more sophisticated data structures, such as networks, matrices, or vectors, and the operations to manipulate them are required. In such cases, the choice is between limited capabilities of a single system, or having to learn to use and interface the two systems. In a later section the problems of interfacing data

management systems and statistical packages are discussed.

The purpose of this paper is to describe the special characteristics and problems of SDBs. When appropriate, solutions that have been proposed in the literature are pointed out. The next section contains a discussion of the special characteristics of SDBs. The remaining sections describe problem areas requiring special attention. A expanded version of this paper can be found in [Shoshani 82].

2. CHARACTERISTICS

In this section we identify the characteristics that are common to SDBs in terms of the structure and use of the data. These characteristics are the basis for the discussion of problems described in the remainder of the paper.

2.1. Category and summary attributes

Most SDBs can be thought of as having two types of data: measured data on which statistical analysis is performed, and parameter data which describe the measured data. Why the distinction? In traditional data management, data is organized into record types, relations, or entities whose columns represent attributes. Both parameter data and measured data are described in terms of the attributes, and no distinction is made.

To illustrate the reasons for this distinction, consider Figure 1 which represents a simple data base in a table (relation) form. The first five attributes (oil type, state, county, year, month) represent the parameter data, and the last two (consumption, production) represent measured data. The attributes for the parameter data have been referred to in the literature as "category" attributes, since they contain categories for the measured data. The attributes for the measured data are referred to as "summary" attributes, since they contain data on which statistical summaries (and analysis) are applied. There are several points to note in Figure 1.

First, note that a combination of the category attribute values is necessary for each of the values of each summary attribute. That is, the category attributes serve as a composite key for the summary attributes. This relationship between category and summary attributes is key to some modeling techniques, as discussed in a later section on logical modeling.

Second, as can be readily seen from Figure 1, there is a great amount of redundancy in the values of the category attributes. In many data bases all possible combinations of the category attributes (i.e. the full cross product) exist. In such cases each value of a category attribute repeats as many times as the product of the cardinality of the remaining category attributes. This is the main reason for the organization of SDBs into matrix form. A matrix organization replaces the need to store the category values in the data base by representing them as positions of the columns and rows. Clearly, there is a need for efficient storage and access of category attributes. This issue is discussed later in the section on physical organization.

OIL TYPE	STATE	COUNTY	YEAR	MONTH	CONSUMPTION	PRODUCTION
Crude	Alabama	County 1	1977	Jan	500	800
Crude	Alabama	County 1	1977	Feb	700	300
"	"	"	"	.	1700	700
"	"	"	"	.	.	.
"	"	"	"	.	.	.
"	"	"	"	Dec	.	.
"	"	"	1978	Jan	.	.
"	"	"	"	.	.	.
"	"	"	"	.	.	.
"	"	"	"	.	.	.
"	"	"	"	Dec	.	.
"	"	"	1979	.	.	.
"	"	"	"	.	.	.
"	"	"	"	.	.	.
"	"	County 2	.	.	.	.
"	"	.	.	.	.	.
"	"	.	.	.	.	.
"	Alaska	.	.	.	.	.
"	.	.	.	.	.	.
"	.	.	.	.	.	.
Heating	.	.	.	.	.	.
"	.	.	.	.	.	.
"	.	.	.	.	.	.
"	.	.	.	.	.	.

FIGURE 1: An example of category and summary attributes

Third, the range of category attributes is usually small, from as little as two (e.g. "sex") to a few hundreds (e.g. oil type). In contrast, summary attributes often have large ranges since they usually represent numeric measures. Often, category attribute ranges are grouped together so as to have fewer categories, such as using "age groups" rather than "age". Also, category values are more descriptive in nature, and therefore tend to be character data (e.g. oil type), while summary values tend to be numeric. Often, codes are assigned to replace long text values, a practice that often forces users to remember these codes.

Of course, there are exceptions to the above observations. Indeed, it is not always obvious whether an attribute should be considered a category or a summary attribute. For example, in a population data base which contains the attributes: race, sex, age, income, and profession, it is impossible to tell *a priori* which are category attributes and and which are summary attributes. Statistics on age can be requested while the others are considered category attributes, or age can be treated as one of the categories for income statistics. However, in most cases the distinction can be made if the original purpose for collecting the data and its intended use is considered.

2.2. Sparse data

Consider an example database on trade activities between countries by year. The attributes involved are: (trade material, exporting country, importing country, year, quantity). The first four are category attributes, while the last is a summary attribute. If this database was stored as a "flat file" (or a matrix form), it will be quite sparse because most countries produce only a small number of trade materials, and trade only with a small number of countries. This a direct consequence of the cross product of all possible values of category attributes. There are two options of dealing with sparse data. The first is to leave the null values (or zeros, or any other designated constants) in the data base and then squeeze them out using compression techniques. The second option is to remove entries that have only null values from the data base. By doing so the row position can no longer be used to indicate a category, and therefore an additional column is required to describe the category values for each entry.

The second option is primarily used in publicly available data. One of the reasons for this is that data is physically represented as it would appear in a report or a two dimensional display. Some data bases are actually published in reference books, and the data that are distributed on a tape have the same format as that presented in the book. In the next section on physical organization, storage techniques for sparse data that are independent of a two dimensional layout are described.

2.3. Summary sets

When statistical data bases are very large, it is too expensive to work directly with the original data set. Users extract smaller data sets that are of interest to them, apply the usual selection functions to limit the number of entries in the data set (such as only the western states), apply projection functions to limit the summary data they are interested in, and join data from different data sets (although tools for joining are not always available). But in addition, a very common operation is to reduce the number of category attributes by summarizing over them. In the example shown in Figure 1, a user can request total consumption by oil type, by state, by year, so that the consumption values are totaled over the appropriate counties and months.

In an active data base, a large number of summary sets may be generated, causing management problems which will be discussed later.

2.4. Stability

Fortunately, a large proportion of statistical data bases are very stable. Initial corrections may be required but very little updating is necessary afterwards. This stems from the primary purpose of SDBs, which is to collect data for future reference and analysis. Once the data is collected, there usually is no reason to change it unless it is for the correction of identified errors. Even in data bases that are usually associated with a high degree of updating, such as inventories, the transactions are actually recorded over time, if further analysis is desired. Actually, most businesses, such as banks, retail stores, etc., record all transactions as verification that the transaction has taken place, along with the time and person performing the transaction.

The stability of SDBs is a benefit since many of the problems that arise in multiple updates to data bases can be avoided (such as concurrency control). It also simplifies the management of summary sets since it is not necessary to keep track of their dynamic updating.

There is another benefit to the stability of data bases which takes advantage of the trade-off between retrieval and update operations. If one assumes very little or no updating, it is possible to design more efficient retrieval algorithms on account of slow updating.

2.5. Proliferation of terms

This phenomenon is not unique to statistical data bases, but exists whenever a data base contains a large number of attributes. There are databases that contain thousands of terms. For example, in an energy database there may be hundreds of different types of oil, coal, and natural gas, hundreds of electric utilities, oil producers, refineries, and pipeline companies, and hundreds of measured (summary) attributes for combinations of these. How is one to remember the content of the data base, let alone the names and acronyms of the attributes or possible category values? When a data base has hundreds (or even a few tens) of attributes, it is necessary that some tools be provided for dealing with such complexity.

In order to formulate a query, a user must remember the following things in addition to the details of the query language: the names of data sets (or relations) needed, the names or acronyms of the attributes needed, the possible and legal values for these attributes, and the formats of the values (e.g. the format for age groups, or whether to use capitals in names of cities). In addition, the codes or abbreviations that were assigned to values (e.g. codes for states and counties) must be remembered. It is not surprising that such data bases require specialists to access them.

These difficulties are even more serious in SDBs, for two reasons. First, many data bases have categories that change their definitions over time. For example, counties may change their boundaries over time, but not their names. Also, the same terms are used with slightly different meanings. For example, the term "state" may include Guam and Puerto Rico in one data base, but not in another. The second reason stems from the summary sets. With every new summary set that is

created, new names are introduced, or perhaps old names with new meanings. It is necessary to control this proliferation of terms, and to keep track of what exists in the system.

The next sections organize the discussion of problems into research areas. Whenever appropriate, some solutions that have appeared in the literature are mentioned. This is not intended to be a comprehensive list of solutions, but rather to pick some representative solutions that we are familiar with as possible approaches to the problems.

3. PHYSICAL ORGANIZATION

Most of the problems discussed in this section stem from the need to compress the data in large SDBs, while permitting fast access. There are a large number of known compression techniques ranging from coding to intricate text compression. The purpose of this section is to highlight some representative techniques that are particularly applicable to SDBs.

3.1. Category attributes

Whenever several category attributes are used jointly to form a composite key, a large storage overhead results. This point was illustrated previously in Figure 1, where there is much repetition of values in the category attributes columns. Because the category attributes form a cross product, the storage requirements are multiplicative in nature.

One common technique to reduce this overhead, is to encode the category values, and to store only the codes with the data base. This can result in great savings, since some category values are descriptive text (for example, the oil types categories shown in Figure 1). Furthermore, the amount of storage needed for the category values depends on the number of distinct category values. Thus, only one bit is necessary to encode the two values of sex, and only four bits for the twelve values of months. Two example systems that were specifically designed to manage SDBs, use this technique: the RAPID system [Turner et al 79], and the ALDS system [Burnett & Thomas 81]. As was pointed out in [Gey 81] it is unfortunate that many systems which use encoding, do not provide software for the automatic translation

between the original values and the encoded values, but rather leave the burden on the user to determine which codes to use before querying the data base.

The previous technique still requires that the encoded values be stored repeatedly. Another approach is to use the logical extension of the matrix storage form. One can store the list of distinct category values of each attribute once (perhaps in a dictionary). Then, each category attribute can be used to form one dimension of a multi-dimensional matrix. For each combination of values from the category attributes, one can compute the appropriate position in the multi-dimensional matrix. There is a well-known algorithm for such a mapping (called "array linearization"); its use for category attributes is explained in [Eggers & Shoshani 80]. It is worth noting that the mapping is a simple computation, and therefore random access is essentially achieved.

3.2. Sparse data

As was discussed previously in the example on trade between countries, SDBs can be quite sparse. The greater the sparseness, the greater the chance that longer sequences of null values can be found in the data. But, in addition, experience suggests that in SDBs null values (or other designated constants) tend to cluster. To see the reason for this, refer back to figure 1. Suppose that a certain state does not consume a certain oil type. Then in the consumption column there would be zero (or null) values in consecutive positions for all the counties in that state, for all years, for all months. Of course, the order of the category attributes will change the length of the null sequences.

This brings up the following interesting problem: given a certain order of the category attributes and given the precise layout of the corresponding measured values, find an efficient algorithm for determining the best reordering of the category attributes such that the length of null sequences is maximized. We do not know of a (non-exhaustive) solution to this problem.

The length of sequences is very important since compression techniques can take advantage of them by essentially replacing a sequence with a count and a value. This technique (called run length encoding) can result in substantial reductions in the size of the data, depending on the sparseness of the data base. The main

problem with this technique is the need to access the data sequentially once it is compressed. The ability of random access according to the relative position is lost. In [Eggers & Shoshani 80] a technique was developed where logarithmic access can be achieved for data whose null sequences have been compressed. The technique, called "header compression", makes use of a header which contains the counts of both compressed and uncompressed sequences in the data stream. The counts are organized in such a way as to permit a logarithmic search over them. A B-tree is built on top of the header to achieve a high radix for the logarithmic access. In a later paper [Eggers et al 81] the technique was extended to sequences of multiple constant values.

This header compression technique is also used to compress sequences of values that vary in size requirements (i.e. one byte, two bytes, etc.). This can be useful in the case where the distribution of summary attribute values is skewed in such a way that the majority of the values are small. As an example, consider seismic activity measurements where most of the measurements consist of low level background noise.

3.3. Transposed files

The tendency for clustering of null values often occurs within a single column (representing a single summary attribute). This suggests that from a compression point of view, it is advantageous to transpose files, i.e. to store values by attribute, rather than as records or tuples. As discussed in [Teitel 77] and [Turner et al 79], there are other reasons to prefer transposed files (sometimes called "attribute partitioning" or "vertical partitioning") in SDBs. It is argued that in SDBs very few attributes are requested in a single query, and it is inefficient to access data organized as records, since it is necessary to read the data of the other attributes which are of no interest from secondary storage. Another approach is to cluster the attributes which are likely to be accessed together, but it is not a simple matter to determine the preferred clustering from a set of representative queries [Hammer & Niamir 79]. Fully transposed files (i.e. no clustering of attributes) are used in the RAPID and ALDS systems mentioned above, and in earlier systems such as IMPRESS [Meyers 69] and PICKLE [Baker 76].

3.4. Partial cross product

The problem of storing efficiently the cross product of category attributes was discussed above. However, there are situations where not every possible combination of the category attributes is valid, i.e. for the combinations that are not valid, the values for all the summary attributes are null. In such a case, the entire entry is missing from the data base. This situation is referred to as the "partial cross product".

The problem is to determine whether there is a way to compress partial cross products. Clearly, the method of value encoding still works, but is there a way to further compress the combinations of category attributes which are valid? In [Svensson 79] a technique which involves the use of a tree is suggested, but some redundancy of values is still left. In [Eggers et al 81] another solution is suggested. It combines the array linearization technique used for full cross product, and the header compression technique for null sequences. Imagine a vector of "ones" and "zeros" that corresponds to valid and invalid entries in the partial cross product, respectively. The partial cross product is treated as if it was a full cross product, and array linearization is used to map into this imaginary vector. Then, the header compression mapping is used to map from the imaginary vector into the actual positions of the valid entries. The outcome of this combination is that just a header is necessary to perform the entire mapping and to achieve a logarithmic access time.

4. LOGICAL MODELING

Can benefits be gained from modeling the semantics of statistical data bases? Is it worth adding to the complexity of the data model? There is a long standing controversy as to whether logical data models should be semantically simple (such as the relational model), or whether they should contain more semantics about the data structures (such as having generalization hierarchies or distinguishing between entities and relationships). In the case of SDBs, the question is whether to model data types such as "matrix" and "time series", and concepts such as category and summary attributes.

4.1. Representation of category and summary attributes

This section points out some of the work done in modeling of category and summary attributes, and the benefits achieved. It is worth noting that practitioners make a distinction between parameters (which correspond to category attributes) and variables (which correspond to summary attributes) because it provides a better understanding of the content of the data base and how it was established. For example, in a scientific experiment, the parameters that can be set by the experimentor are referred to as the "independent variables", and the measured data as the "dependent variables".

One of the main benefits of modeling the semantics of category and summary attributes is the capability of "automatic aggregation". It is the ability of the system to infer the subsets of values over which an aggregation (or statistical) function should be applied. For example, consider the following query when applied to the data base in Figure 1: "find heating oil consumption in Alabama during 1977". It is obvious that the result should be the total heating oil consumption over all counties in Alabama and over all months in 1977. Yet, without the explicit semantics of category and summary attributes the system would not be able to infer what is obvious to us. The benefit to the user is that it is not necessary to explicitly express which category attributes to summarize over. This can greatly simplify aggregation expressions in query languages.

An example of adding the above mentioned semantics to an existing model is described in [Johnson 81]. Using the framework of the Entity-Relationship model, an additional type of entity is allowed, called a summary set, which captures the semantics of category attributes. In addition, an attribute which is designated as a summary attribute, can have an aggregation function (e.g. sum, average) or any other desired function (defined as a program) associated with it.

4.2. Graph representation

Another possibility is to have these semantic concepts represented internally, so that they are invisible to the user. An example of a system that takes this approach is SUBJECT [Chan & Shoshani 81], in which these semantic concepts are represented as a graph. There are two kinds of nodes: a "cross product" node, and a "cluster node". The nodes can be connected by arcs to form a directed acyclic

graph.

Cluster nodes represent collections of items. Consider, for example a database on the employment levels of different industries by state. A cluster node labeled "industrial classes", would contain nodes such as "agriculture", "mining", "construction", etc. Each of these nodes can itself be a cluster node. For example, the node "mining" may represent the collection of iron ores, lead ores, zinc ores, etc. As can be seen, cluster nodes are used to represent a hierarchy of parameters. Cluster nodes are also used to represent the collection of summary attributes of the database.

Cross product nodes are used to represent composite keys of category attributes. Thus in the above example database, the node "state by industry", represent the cross product of the cluster of states with the cluster of industries. The semantics of this cross product node is such that each of its instances is made up of a pair of instances, one taken from the node "industry" and one from the node "states".

This graph structure is invisible to the user and is used to support a menu driven interface. The user does not need to know the types of nodes, but the system can make use of them to provide automatic aggregation. The graph can be either browsed by moving up and down the nodes, or can be searched directly with keywords. The sharing of nodes provides the capability to use the same clusters (e.g. state names) across data sets, and to avoid confusion of names. One of the main advantages of this representation is that the user can be shown the content of the data base by gradually revealing more details when requested. The possibility of viewing hierarchical menus of details alleviates the need to remember names and acronyms.

4.3. Summary sets

Summary sets are simply data base views that are generated by using aggregate functions. The main problem is one of managing a large number of sets. With each summary set new summary attributes are generated. Obviously, the newly computed values of the summary attributes have to be stored if recomputing them is to be avoided. But, is there a way to avoid duplicating the category attribute values? Similarly, new names are likely to be used for the new summary attributes (e.g. "total consumption" when we summarize over consumption). Is there a way of

using the same names of category attributes in the summary set?

This is another situation where distinguishing between the type of attributes can be beneficial. If the category attributes are organized as lists of category values (say, in a dictionary), then it is possible for the category attributes of the summary sets to "point" to these lists, and to share the same names. It is easy to visualize this point in terms of the SUBJECT graphs described above. If a category attribute is used in its entirety in the summary set, then a pointer to the corresponding node is all that is necessary. If a selection of a certain category is made, e.g. "Alabama", then the pointer points directly to the node representing "Alabama". If a selection of a subset of the category values was made (e.g. several states), then a new node is created whose members are the nodes belonging to that subset.

This idea is complementary to the technique described in the section on physical organization above, where the lists of category attribute values are stored only once, and array linearization is used to map between them and the appropriate positions of the summary attributes.

5. USER INTERFACE

One of the major problems for a user interfacing to large SDBs is to determine the content of the data base and the terms used for its attributes. Such problems are referred to as meta-data problems, since they deal with information about the data. Meta-data is much more complex than listing the record types (or relations) and the attribute names and types. It includes information such as missing data specification, data quality specification (to indicate how reliable the data could be considered), a history of data base creation and modifications, complex attribute structures (e.g. vectors to represent the boundaries of geographical regions), etc. In [McCarthy 82], a comprehensive list of requirements for meta-data is given , with special attention given to SDBs.

Whenever it is necessary to deal with the diversity and complexity of data bases, special techniques of classifying information may be needed. In fact, it was helpful to use a technique that is usually used in library systems, called "facet classification". Using this technique, summary attributes are described using facets. For example, some of the facets used for the energy data are: energy source (oil, coal, etc.), function (produced, shipped, etc.), units of measure, dates, etc.

Each facet is a hierarchy of terms that can be quite deep, as is the case with energy source. A combination of the terms from the facet hierarchies is then used to describe a summary attribute (for example, heating oil refined in mid-western states during 1977). This technique can avoid conflicts in definition of similar attributes since they have to be defined using terms from predefined facets.

It is interesting to note that the SUBJECT graphs described previously are powerful enough to describe facets, since cluster nodes can represent a facet hierarchy, and a cross product node can represent the combination of terms from the facets. Indeed, SUBJECT is used to describe meta-data in a hierarchical manner, so that a user can start at a high level (e.g. population data, energy data, etc.), and gradually narrow down to the data set needed.

The distinction between meta-data and data is not always obvious. Information that is stored in the data base can sometimes be thought of as meta-data and vice versa. This is particularly true of the values of category attributes. For example, if a user is inquiring about the content of the data base in Figure 1, it is as natural to ask what are the summary attributes (e.g. consumption) as it is to ask what years are covered or what are the oil types. Again, this argues for associating the list of values of category attributes with a dictionary rather than to store them with the data.

What about query languages? Are there any special problems associated with SDBs? Aggregation is a predominant function that needs to be supported. However, it is perhaps the most awkward function to express in many query languages. In addition to enriching data models to support automatic aggregation, work is being done to simplify the expression of aggregate functions. For example, [Klug 81] proposes an extension to query-by-example in order to support aggregate functions in SDBs. Perhaps a combination of menu driven techniques (such as those used in SUBJECT), graphics techniques (such as described in [Wong & Kuo 82]), and simple command languages can bring about more convenient user interfaces.

6. INTEGRATING STATISTICAL ANALYSIS AND DATA MANAGEMENT

In order to perform statistical analysis, an analyst needs both data management tools and statistical tools. Unfortunately, these tools are not usually

integrated into a single system. Data management systems support only a limited number of statistical functions, and statistical packages have limited data management capabilities.

There are three possible approaches to this problem. The first is to enrich statistical packages with more general data base structures and more powerful data management functions. Evidence of this approach can be found in new releases of statistical packages, where many systems now support some kind of hierarchical or network data structure, while past versions supported only "flat files". However, they still lack many functions, such as joining two tables, or supporting summary sets (or views).

The second approach is to enrich existing data management systems with tools useful to an analyst, such as taking random samples, and a library of statistical operators. An example of this approach is described in [Ikeda & Kobayashi 81], where statistical facilities are added to a commercial data management system, Model 204.

The third approach involves interfacing statistical packages to data management systems. There are three variations to this approach. the first is to tightly couple each pair of systems. Usually a pair is selected for an application and is expected to last a long time. One such experience is described in [Weeks et al 81]. The second variation involves defining a standard data format that all systems accept. In the long run this is a more effective method to implement since each new system added is only required to communicate with the standard format in order to communicate to all systems. However, this technique may be less efficient in terms of processing time since two translators are needed, unless changes can be made to the software of the systems involved. This approach was taken in the SEEDIS project mentioned above, where a fairly simple standard, called CODATA was quite successful in integrating several components of the system. An essential feature of such a standard is that it is self describing, i.e. that data bases carry their own data definition. The third variation involves a monitor that takes care of interfacing the systems, but presents the user with the impression of a single system. an example of this variation is described in [Hollabaugh & Reinwald 81].

An important point to note is that regardless of the approach taken, it is quite essential that statistical operations should produce self-describing data structures that contain meta-data as well as data. Analysts have been burdened by having to keep hand-written documentation of the meta-data as they perform the analysis. As the analysis process progresses, it becomes increasingly difficult to keep track of these meta-data descriptions.

It is not clear which of the above approaches is the most successful. Perhaps future systems can be designed from the start to accommodate both statistical analysis and data management needs. The system S [Becker & Chambers 80] was designed with this goal in mind. It also uses a certain form of self-describing data structures.

REFERENCES

[Baker 76] Baker, M., User's Guide to the Berkeley Transposed File Statistical System: PICKLE, Technical Report No.1, 2nd ed., University of California, Berkeley, Survey Research Center, 1976.

[Becker & Chambers 80] S: A Language and System for Data Analysis, Bell Laboratories, July 1980.

[Boral et al 82] Boral, H., DeWitt, D.J., Bates D., A Framework for Research in Database Management for Statistical Analysis, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, June 1982.

[Bragg 81] Data Manipulation Languages for Statistical Databases -- The Statistical Analysis System (SAS), *Proceedings of the First LBL Workshop on Statistical Database Management*, Dec. 1981, pp. 147-150.

[Burnett & Thomas 81] Burnett, R. A., and Thomas J. J., Data Management Support for Statistical Data Editing and Subset Selection, *Proceedings of the First LBL Workshop on Statistical Database Management*, Dec. 1981, pp. 88-102.

[Chan & Shoshani 81] Chan, P., Shoshani, A., Subject: A Directory driven System for Organizing and Accessing Large Statistical Databases, *Proceedings of the International Conference on Very Large Data Base (VLDB)*, 1980, pp. 553-563.

[Cohen & Hay 81] Why Are Commercial Database Management Systems Rarely Used for Research Data? *Proceedings of the First LBL Workshop on Statistical Database Management*, Dec. 1981, pp. 132-133.

[Eggers & Shoshani 80] Eggers, S. J., Shoshani, A. "Efficient Access of Compressed Data," *Proceedings of the International Conference on Very Large Databases*, 6, 1980, pp. 205-211.

[Eggers et al 81] Eggers, S., Olken, F., Shoshani, A., A Compression Technique for Large Statistical Databases, *Proceedings of the International Conference on Very Large Data Base (VLDB)*, 1981, pp. 424-434.

[Gey 81] Gey, F.G., Data Definition for Statistical Summary Data or Appearances Can Be Deceiving, *Proceedings of the First LBL Workshop on Statistical Database Management*, Dec. 1981, pp. 3-18.

[Hammer & Niamir 79] Hammer, M., Niamir, B. "A Heuristic Approach to Attribute Partitioning," *ACM SIGMOD Proceedings of the International Conference on Management of Data*, Boston, 1979, pp. 93-101.

[Hollabaugh & Reinwald 81] Hollabaugh L.A., Reinwald, L.T., GPI: A Statistical Package / Data base Interface, *Proceedings of the First LBL Workshop on Statistical Database Management*, Dec. 1981, pp. 78-87.

[Ikeda & Kobayashi 81] Ikeda, H., Kobayashi, Y., Additional Facilities of a Conventional DBMS to Support Interactive Statistical Analysis, *Proceedings of the First LBL Workshop on Statistical Database Management*, Dec. 1981, pp. 25-36.

[Johnson 81] Johnson, R.R., Modelling Summary Data, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1981, pp. 93-97.

[Klug 81] Klug, A., Abe -- A Query Language for Constructing Aggregates-by-example, *Proceedings of the First LBL Workshop on Statistical Database Management*, Dec. 1981, pp. 190-205.

[McCarthy 82] McCarthy J., Meta data Management for Large Statistical Databases, *Proceedings of the International Conference on Very Large Data Base (VLDB)*, 1982.

[McCarthy et al 82] McCarthy, J.L., Merrill, D.W., Marcus, A., Benson, W.H., Gey, F.C., Holmes, H., Quong, C., The SEEDIS Project: A Summary Overview of the Social, Economic, Environmental, Demographic Information System, Lawrence Berkeley Laboratory document PUB-424, April 1982.

[Meyers 69] Meyers, E.D. Jr., Project IMPRESS: Time Sharing in the Social Sciences, *AFIPS Conference Proceedings of the Spring Joint Computer Conference*, Vol. 34, 1969, pp. 673-680.

[Nie et al 75] Nie, N.H., et al, SPSS: Statistical Package for the Social Sciences, Second Edition, McGraw Hill, New York, 1975.

[SAS 79] SAS Institute, Inc., SAS User's Guide, 1979 Edition, Raleigh, North Carolina, 1979.

[Shoshani 82]
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.

[Svensson 79] Svensson, P. On Search Performance for Conjunctive Queries in Compressed, Fully Transposed Ordered Files, *Proceedings of the International Conference on Very Large Databases*, 5, 1979, pp. 155-163.

[Teitel 77] Teitel, R.F., Relational Database Models and Social Science Computing, *Proceedings of Computer Science and Statistics: Tenth Annual Symposium on the Interface*, Gaithersburg, MD, National Bureau of Standards, April 1977, pp. 165-177.

[Turner et al 79] Turner, M. J., Hammond, R. and Cotton, F. A DBMS for Large Statistical Databases, *Proceedings of the International Conference on Very Large Databases*, 5, 1979, pp. 319-327.

[Weeks et al 81] Weeks, P., Weiss, S., Stevens, P., Flexible Techniques for Storage and Analysis of Large Continuing Surveys, *Proceedings of the First LBL Workshop on Statistical Database Management*, Dec. 1981, pp. 310-311.

[Wong & Kuo 82] Wong, H.K.T., Kuo, I., A Graphical User Interface for Database Exploration, *Proceedings of the International Conference on Very Large Data Base (VLDB)*, 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.

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