A FRAMEWORK FOR RESEARCH IN DATABASE MANAGEMENT FOR STATISTICAL ANALYSIS

by

Haran Boral
David J. DeWitt
Doug Bates

Computer Sciences Technical Report #465
Feburary 1982

A Framework for Research in Database Management for Statistical Analysis

or

A Primer on Statistical Database Management Problems for Computer Scientists

Haran Boral[*]
David J. DeWitt[*]
Doug Bates[+]

[*]Computer Sciences Department
[+]Statistics Department
University of Wisconsin
Madison, Wisconsin

This research was partially supported by the Department of Energy under contract #DE-AC02-81ER10920 and National Science Foundation under grant MCS78-01721.

ABSTRACT

This paper is intended to introduce those familiar with database management issues to the problems of managing large statistical databases. We begin with a characterization of statistical databases based on the structure and use of the data in the database. Several data management problems are then described. In particular, we discuss the problem of repetitive computations on large segments of the database during the lifetime of a statistical analysis. The organization of a data management system which avoids this problem by caching previously computed results and automatically maintaining their integrity is presented. We conclude with a list of problems that this organization raises and a discussion of related work.

1. Introduction

Researchers in the database area have lavished attention on providing support for access to what we term "corporate" databases - databases of the supplier-parts flavor. Research projects have culminated in very high-level, user-friendly query languages, sophisticated access path mechanisms, storage structures, and database machine support.

Statistical databases exhibit different characteristics than corporate databases, both in terms of the data (its structure and size) and its use (types of queries). Thus, existing database management systems cannot provide efficient access to statistical databases. A number of software packages for analyzing statistical data exist. It is our contention that despite the fact that these packages were designed for the specific purpose of analyzing statistical data they lack suitable data management capabilities. We believe that this is the consequence of the fact that most database research has concentrated on problems associated with "corporate" databases. Thus the developers of the statistical packages had no body of knowledge upon which to draw when designing their systems.

The paper does not contain the answers to the problems of statistical database management. Rather, it is intended to serve as a framework for future research by ourselves and other computer scientists. We believe that this is an important area of research that deserves considerably more attention than it has received in the past. We hope that this paper will serve to spark additional interest in the problems associated with statistical database management.

In Section 2 we characterize statistical databases in terms of their structure and the operations performed on them. We then describe some of the problems associated with managing such databases. In Section 3, we outline the structure of a database management system that facilitates access to, and management of very large statistical databases. Section 4 contains a discussion of the merits of this architecture and outlines areas that require further investigation. Our conclusions, a survey of related work, and a summary of the paper are presented in Section 5.

2. Data Management for Statistical Analysis

In this section we attempt to answer such questions as: What is a statistical database and how is it used? Next, we list data management functions required for statistical analysis and then critique existing data management software in statistical as well as commercial database management packages. We conclude with an overview of recent research in data management software for statistical analysis.

2.1. What is a Statistical Database?

There are a large number of software packages for the statistical analysis of data sets. Some of the better known packages are SAS [SAS79], S [BECK78], Minitab [RYAN81], SPSS [NIE75], BMDP [DIXO79], and P-STAT [BUHL79]. Each package generally supports simple summary-statistics operations such as min, max, mean, median, and standard-deviation in addition to complex operations such as cross tabulations, multivariate correlations, regression analysis, and other multivariate analysis techniques. Some of the packages provide more sophisticated services such as data editing and operations on files.

Although the functionality of the packages varies widely, almost all packages provide the user with a "flat-file" view of each data set that, much like a relation, consists of attributes (columns) and records (rows). Like relations, the number of attributes in a data set varies from one data set to another and can be very large. Those attributes that together uniquely identify each record in a data set are referred to as category attributes. That is, they form a composite key. Values in non-category attributes quantify the composite value of the category attributes with which they are associated.

The example data set shown in Figure 1 illustrates several important points about statistical data sets. First, the number of records (observations) in the statistical data set can equal the cross product of the ranges of the category attributes values (e.g. data gathered by observing the results of all combinations of the control parameters in an experiment). Second, for those data sets with a large number of category attributes, each of which can assume a wide range of values, the cost of just storing the category attribute values can become prohibitive. In order to reduce storage space, data values, such as age in Figure 1, are frequently encoded. Thus, a table such as that found in Figure 2 must be used to interpret the values of the AGE_GROUP

SEX	RACE	AGE_GROUP	POPULATION	AVE_SALARY
M	W	1	12,300,347	33,122
M	W	2	21,342,193	25,883
M	W	3	18,989,987	42,919
M	W	4	9,342,193	15,110
F	W	1	15,821,497	31,762
F	W	2	33,422,988	29,933
F	W	3	29,734,121	28,218
F	W	4	20,812,211	17,498
M	B	1	2,143,924	29,402
		. . .		

Figure 1
An Example Data Set

CATEGORY	VALUE
1	0 to 20
2	21 to 40
3	41 to 60
4	over 60

Figure 2
AGE_GROUP Data Set

attribute.[1] Furthermore, this encoding can generate inconsistencies when different code values are used, for example in the 1970 and 1980 census. Finally, the values of a number of attributes in a data set may have been derived by aggregating over other data values in the process of forming or analyzing the data set. For example, the values of the AVE-SALARY attribute of the data set shown in Figure 1 were derived by computing the average salary of all individuals in each corresponding SEX, RACE, AGE_GROUP partition.

2.2. How are Statistical Databases Used?

After the initial loading of the database, the statistician spends a lengthy period of time "exploring" the data during a process known as exploratory data analysis [TUKE77]. This initial exploration of the data begins with checking for invalid values. For example, in the case of the data set shown in Figure 1, the data checker would ensure that all income values were within some reasonable range. A value outside this range must be marked as suspicious and then investigated in order to determine whether it should be invalidated or not. Generally, data checking is a lengthy procedure. The data checker must scan all the values in each attribute of each row of each data set. This is typically done using histograms or range checking programs. In addition, for those cases in which a known relationship exists between pairs of values, the data checker must also examine all

[1] For the 1970 Census public-use-sample database, the book describing the encoded values for each attribute type is over 200 pages of fine print [CENS72].

pairs of values to insure that they indeed behave according to the relationship. Again this involves plotting the data, although other, computational, techniques exist.

A second function of the initial data exploration phase is getting a "feel" for the structure of the data by posing such questions as: Do the data values in a given attribute conform to a particular distribution? Is there a relationship between the values of two attributes? During this process data may be aggregated to create a new, or modified, data set, when, for example, the analyst discovers that the level of detail in a data set is too fine. For example, the analyst may discover that he no longer wants to differentiate between M and F entries for a given age group in the data set shown in Figure 1. The new data set would be created by, for each RACE/AGE_GROUP partition, adding the M and F populations and forming a weighted average of the two AVE_SALARY fields.

During this preliminary stage of data analysis it may be the case that, for some operations, there is no need for the statistician to use the entire data set. Instead, in order to enhance responsiveness, the statistician may base this preliminary analysis on a set of sample records drawn at random from the data set. Forming an impression of the structure of the data based on a small sampling is sufficient.

The second phase of the data analysis process is known as *confirmatory data analysis*. In this phase, the statistician applies several additional tests to the initial as well as other, perhaps enlarged, samples, and finally the entire data set in order to confirm his initial impressions of the structure of the data. The tests used in this phase are considerably more complicated and specialized than the ones used in the exploratory phase. For example, a goodness-of-fit test may be applied to see if a particular attribute does indeed follow a hypothesized distribution or a chi-squared test may be applied to a cross-tabulation of data according to two attributes to see if the attributes depend on each other (e.g. is the proportion of people who live past 40 dependent on race?). In fact, most statistical theory and research in statistics is devoted to building up such tests and their interpretations. Thus there are a large number of different operations that can be applied during this phase.

The analysis of a data set will usually involve several iterations of exploratory and confirmatory data analyses. At the end of each such iteration the statistician may wish to update the data set to incorporate what he has learned about it. The update operations may add a new attribute to the data set to capture the results of a time-consuming calculation that are to be used later; temporarily mark a particular record (or set of records) as invalid; or, form a new data set, perhaps by either joining, projecting, etc. existing data sets into a new one, or by extracting new data from the raw database.

Except when a sample is used, execution of a statistical operation on a data set almost always requires access to a few columns of every row in the data set (e.g. computing the median of the AVG_SALARY values). Furthermore, because data analysis is basically an ad-hoc process, access to any column is equally likely.

2.3. What are the Required Data Management Services for Statistical Analysis?

Operations for Materializing Views

Unlike corporate databases where views can be, and often are, virtual, each analyst must have his own private concrete view for a number of reasons. First, because of its enormous size, the raw database will almost always reside on slow secondary storage devices such as tapes. A typical analysis will require access to a small portion of the database, which for reasons of efficiency, must be migrated to disk storage while in use. Using concrete views requires some additional tape storage but avoids the generation of the view from tape storage each time it is used. Thus, the cost of materializing the view is amortized over its period of use. Second, during the lifetime of an analysis the statistician may access the data in the view according to certain patterns that can either be communicated to the DBMS or perhaps gleaned by the DBMS from the use of the data. This information can then be used, for example, to create auxiliary storage structures such as indices or to transpose the data in some manner to facilitate efficient access to frequently used data.

The operations required for materializing views are the traditional relational operations which create and transform tables. In addition, the DBMS must provide operations for reformatting of the data, creation of auxiliary data structures, and control over the layout of the data on mass storage. Another, very important, set of operators are aggregates, in particular aggregate functions.

View Management

The existence of multiple views of the same raw database introduces a number of important problems that must be resolved. First, since an analyst may update a view in the process of the exploratory data analysis phase, it should be possible for him to "undo" recent changes to the view if he discovers, through subsequent analysis, that the changes made to the view were incorrect. Second, should views ever be shared? Suppose two analysts are examining the same underlying data. One wishes to analyze the effects of pollution on population by race and another wishes to study the effect of pollution on population by age group. Third, a mechanism is needed to insure that an analyst does not recreate (from the raw database) a view that is either identical to one that has already been created by another analyst or which can be formed by a limited number of operations on an existing view. Finally, there should be a means by which the results of an analyst´s data editing can be made public.

Management of the Meta-Data

Since a large statistical database may consist of several thousand tables, each with a large number of attributes, just understanding the logical structure of the database is a non-trivial task. In fact, one can view the meta-data as residing in a separate database with its own "data model", data structures, access mechanisms, etc. The SUBJECT system [CHAN81] has made some important first steps in the problem of navigating through the meta-data. A user views the meta-data as a graph in which nodes represent attributes. Additional, "higher-level", nodes represent generalizations of lower-level nodes. A user enters the system at a fairly high "level", navigating his way through the meta-database down to the level of desired detail. SUBJECT keeps track of the path followed by the user and at the end of the session can generate requests to the DBMS for the view described by his path through the meta-database structure. SUBJECT also has primitive operations that enable management of the graph (i.e., means for updating the graph).

Support for Repetitive Transactions

During the lifetime of an analysis, the statistician may execute an operation, such as median, repeatedly on the same data set. (Note that lifetime here can mean a lengthy period of time - as long as a few months!) The data management system should attempt to save the result of several such computations for future reference. This is particularly important for those cases in which computing the value involves accessing every row in a data set. Since updates on the view are allowed, care must be taken to insure that "cached" values are correct.

2.4. Limitations of Statistical Packages

The database management software found in current statistical software packages is very limited or non-existent. Most packages support only sequential files with limited or no support for secondary indices. In addition, although data sets are basically identical to relations, very few of the commonly used packages support the relational join operation. Thus, instead of simply being able to join the table in Figure 2 with the table in Figure 1 to decode AGE_GROUP values, the statistical package user is generally forced to manually "look up" the encoded values in a code book.

Statistical packages are not adequate for analyzing large data sets (like the census database) for a number of reasons. Some packages, like Minitab [RYAN81] and S [BECK78], require a data set under analysis to fit in the computer's virtual address space. An important side-effect is that memory is managed according to some scheme which is not necessarily suited to the access patterns exhibited for statistical databases. The performance of other packages, such as SAS and SPSS, degrades significantly when required to process very large data sets because of excessive I/O activity. Finally, no support for repetitive execution of operations is provided by any of the packages.

2.5. What about Conventional DBMS?

In the statistical data management literature, conventional database management systems are cited as being inappropriate for managing statistical data. A number of reasons are usually given [BRAG81]. These include (1) cost, (2) unfriendly user interfaces, (3) inaccessibility of database software to statisticians, (4) lack of appropriate statistical functions and procedures, (5) difficulty of interfacing database software to statistical software (recall, almost all statistical packages need to view the data as a flat file), and (6) poor performance. Although the first five objections are perhaps applicable to network and hierarchical database systems, it is not apparent that these objections hold for relational systems.

Performance is, however, an important problem when a conventional database system is used as the basis for statistical data management. First, many statistical applications require only sequential access to all rows in a data set, not the need to model and process complex relationships. Furthermore, because of this sequential access to arbitrary columns, normal techniques to enhance performance such as indices and sorting will not work. Finally, because each analyst generally works with a private data set that is infrequently updated, the overhead of sophisticated concurrency control and recovery mechanisms cannot be justified.

2.6. Recent Work

In the past few years several projects have been initiated to enhance analysis of very large data sets. These efforts have taken a variety of approaches to the problem. One line of research has concentrated on compression techniques [EGGE80, EGGE81] to reduce the storage space occupied by the data set and hence the I/O time necessary to access and process it. Another approach, as exemplified by SEEDIS (Socio-Economic-Environmental Demographic Information System) [GEY80], has relied on the development of customized software to enhance access to one particular database.

A more general purpose approach to the problem has been taken by the ALDS/SDB [BURN81] and RAPID [TURN79] systems. Both of these systems rely on the use of transposed files to minimize access time to a column of a data set. Transposed files facilitate the processing of statistical operations primarily for two reasons. First, since the patterns of access shown by exploratory and confirmatory data analysis operations are generally oriented towards a few columns of the data set and every row, a transposed file organization will minimize the number of I/O operations needed to retrieve all entries in a column. The second advantage of transposed files is that run-length compression techniques [EGGE80, EGGE81] are more likely to improve storage efficiency when they are applied down a column rather than across a row.

The major disadvantage of transposed files is that they provide poor performance on "informational" queries such as "find the average salary and population of all white males in the 21-40 age group". Performance of transposed files for informational queries is further degraded when compression techniques leave attribute values from the same row at different positions in their respective files. Despite this drawback, the transposed file structure appears to be the best all-around storage structure for statistical data sets since the higher-level software can continue to view the data set as a flat file, while drastically reducing the number I/O operations required to process a statistical (as opposed to informational) operation.

2.7. Summary

In summary we feel that the proper software architecture for the development of statistical database software requires that individual views of the data be materialized for each analyst. Implementation of such a system requires that a number of problems be investigated and solved. First, the operation of materializing a view will need the appropriate tools (software/hardware) for specifying exactly what view is to be materialized and for providing reasonably efficient access to the very large raw databases from which the view will be formed. Once materialized, software should be provided for "managing" the view. Included should be tools that permit an analyst to "undo" recent changes to the view and "intelligent" access methods that interpret reference patterns to the view and dynamically reorganize the storage structures used to maintain the view. Finally, the view software should facilitate the efficient execution of repetitive operations on the view. In the following section, we will present an overview of the techniques we are investigating for achieving the final objective.

3. A Preliminary Architecture for a Statistical DBMS

The types and sizes of data sets as well as the complexity of the structure of the data determine, to a large extent, the amount of time required to analyze a data set. For our research we have assumed that data sets will require lengthy analyses - perhaps spanning several man-months. In this section we briefly discuss how the use of an ordinary statistical package to analyze a flat file may result in unnecessary computation. We then propose an architecture for the data management component of a statistical package that would eliminate the need for the unnecessary computation, particularly in the exploratory data analysis phase.

3.1. The Problem of Repetitive Computations

As indicated in Section 2, much of the data analyst's time is spent exploring the data. At several points, outliers may be encountered. These must be investigated to ascertain whether they indeed reflect an abnormal measurement (as in the case of a 5 digit salary in Beverly Hills) or an incorrect measurement (a person's age recorded as 1,000). In the latter case the value must be marked as invalid -- "missing value" in the statistics vernacular.

Several values computed during the exploratory data analysis phase are used later. For example, the min and max values of an attribute are required for axis labeling in plots as well as for generation of histograms. Another example is the use of measures of central tendency and dispersion such as mean (M) and standard deviation (SD). At some early point during the analysis the analyst may wish calculate these values for a given attribute. Later on, he may wish to count the number of (possibly unique) values that lie outside the range defined by the M $\pm$ k*SD, for some k. Yet another example is the use of quantiles. Initially, the analyst may be interested in finding out the 5th and 95th quantiles. Later, the analyst may ask for the trimmed mean (the mean of all the values in a given range) bounded by the 5th and 95th quantile values of the same attribute.

One approach is for the analyst to record (on a piece of paper or in a database) the results of the various computations that he has performed. Alternatively, he can simply rerun the computations whenever their results are required for further analysis. Clearly, for the analysis of a large data set, the first choice is much more appealing. Therefore, the data management component of a statistical package should provide the users with the capability of storing, and easily retrieving, results of previous computations. In some cases these results should be saved for the duration of the entire analysis (the median of most columns, for example) whereas in other cases it should be stored for a short period of time (less general order statistics, such as the 10th largest value, can usually be disposed of early on in the analysis).

3.2. *Proposed Organization of the DBMS*

In Figure 3 we show the overall organization of the data management component of a statistical database management system that we are currently investigating. We envision several concrete views over a single raw database. Each view is private to a single user (or a group of users). Associated with each view is a *Summary Database*. The main purpose of the Summary Database is to reduce the number of accesses to the database (actually the view) by acting as a cache for frequently used data.

A second component of this organization is the *Management*

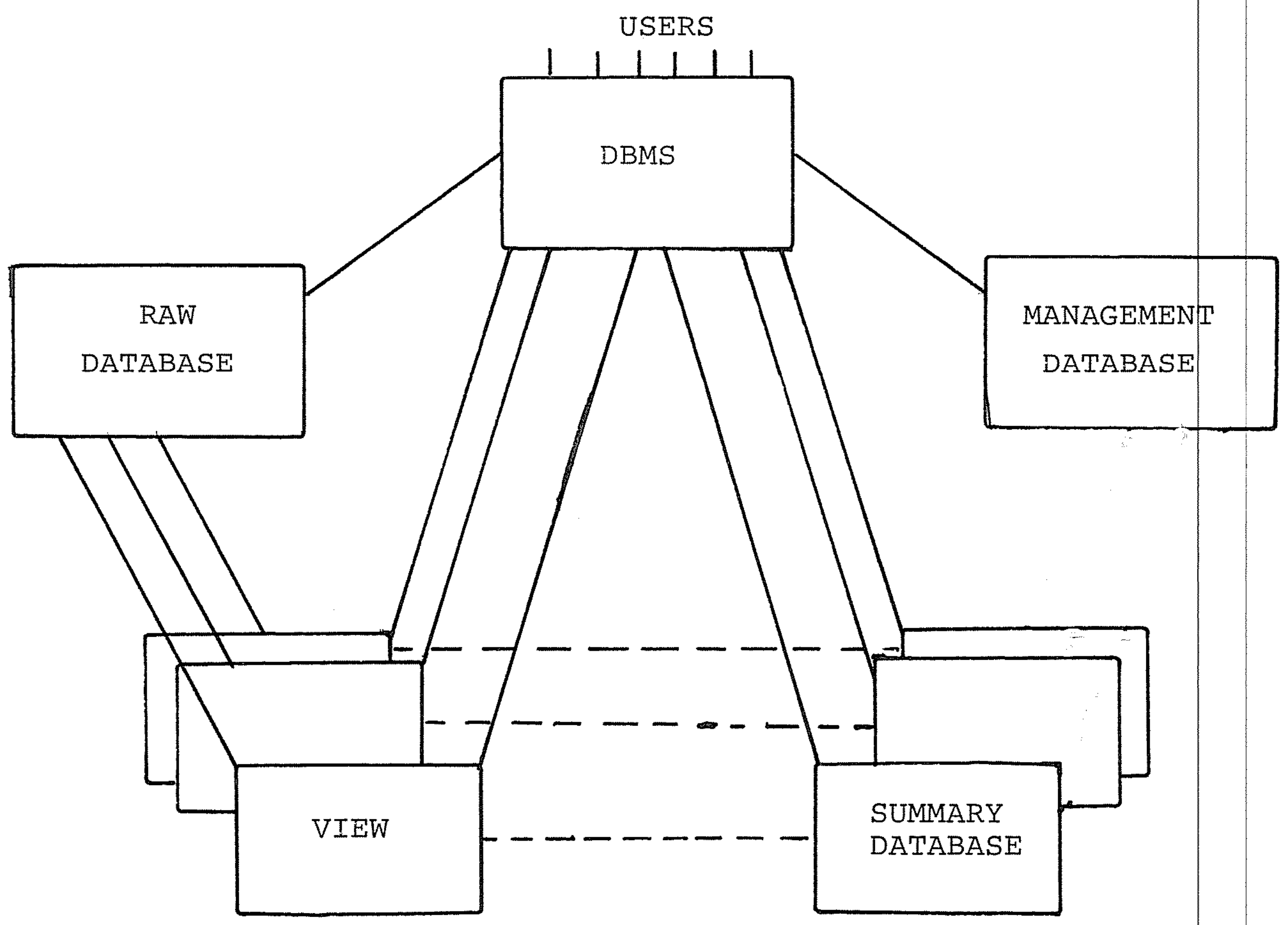
USERS
DBMS
RAW DATABASE
MANAGEMENT DATABASE
VIEW
SUMMARY DATABASE

Figure 3

Database. One Management Database is associated with the DBMS. The purpose of the Management Database is to serve as a repository for information that describes the organization of the data, the functions that are applied to it, rules for manipulating information in the Summary Databases, view definitions, update histories of the views, and other control information. Below, we describe the organization of both databases and the manner in which the DBMS would use them.

The Summary Database

Each Summary Database serves as a cache for the user view. Rather than storing frequently used data in the Summary Database we choose to store results of query (or function) executions. This leads to a savings in execution time each time a function whose result is already in the cache is invoked. In addition, the size of the cache is much smaller, reflecting the relationship between the sizes of the results of and inputs to most functions. As is the case with all caches there is some overhead for managing the cache. In this case it involves detection of inconsistencies caused by updating the view. However, the relatively static nature of statistical databases indicates that this overhead will be more than offset by the gains due to the time and space savings of this approach.

A second function of the Summary Database is as a repository of certain descriptive, or summary, information about the data in the view. Included in the summary information are mode, mean, median, quartiles, the ranges of values in each column (min &

max), the number of unique values, and some measure of frequency of values. Additional summary information to be stored in the Summary Database, perhaps under user control, might include histograms and verbal descriptions of the data set (for example, a statement of how far analysis has proceeded, what difficulties have been encountered, etc.). Note, however, that computing the median (or any summary values) of the AGE_GROUP attribute in Figure 1 does not make sense. Thus, the system will have to rely on meta-data to decide for which attributes summary information should be computed.

A Summary Database will contain results of significantly different types. For example, the mean of a column (attribute) will be stored as an integer (or a floating point), whereas a histogram will be stored as two vectors (one for specifying the ranges and the other for the number of values that fall in each range). As shown in Figure 4, the Summary Database can be logically viewed as a 2-dimensional array with three columns. The first entry in a row contains a description of the function. The third entry in a row contains the result of applying the function specified in the first column to the attribute(s) identified in the second column. Note that implicit here is the fact that the values in the third column will be of varying length.

Searching a Summary Database will require using a function name-attribute name(s) pair as the search argument. If the desired pair is found, the corresponding result will be returned. Otherwise, after the function has been applied to the attribute(s) specified, the new information will be inserted into

FUNCTION_NAME	ATTRIBUTE_NAME	RESULT
Min	POPULATION	2,143,924
Max	POPULATION	33,422,988
Median	AVE_SALARY	29,933

Figure 4
Example Summary Database
for Data Set in Figure 1

the Summary Database. To enhance access to the Summary Database (which may itself become relatively large), we envision the use of a secondary index on function name-attribute name. Data will most likely be clustered on attribute name to facilitate efficient access to all results on a given column.

As stated above, one problem with maintaining the summary information is that inconsistencies in the database may arise due to updates to the view. In some instances this may not present any problems to the user -- a change of one or two values has very little effect on the value of the median. However, there will be other cases when the user will require the values in the Summary Database to accurately reflect the state of the view. The user should have the capability of communicating his wishes regarding the desired accuracy for answers to his questions to the system (see discussion in Section 5). Whether or not a value in the Summary Database must be precise at all times, the DBMS must be able to periodically bring it up to date. We believe that this should be done automatically (given the user's initial wishes regarding the frequency of the updates).

One possibility for obtaining the updated value is to recompute the function using the updated data as input. A more attractive alternative is to incrementally recompute the result using the old function value, changes made to the data, and perhaps some auxiliary information, without having to access all of the data used for the initial function computation. Clearly, code to incrementally recompute some of the functions used by analysts can be generated manually. However, since new statistical methods are evolving it would be desirable to have some means for automatically generating an incrementally recomputable algorithm for a function given the function definition in some high-level form. In Section 4 we discuss some work that has been done towards achieving that goal and some issues that must be addressed.

The Management Database

As indicated above the Management Database serves as a repository for control information. Of particular importance is the information regarding the formation and maintenance of all the views on the database. Keeping a history of updates for each view will enable the DBMS to roll a view back to a previous state should such an action be desired by the analyst. The update history of a view may also be used by other analysts who wish to use some of the data in the view. Rather than repeating the mundane and time consuming data checking operations they can examine what actions were taken by their predecessors and use the "clean" data for their needs.

Our discussion of the use of incrementally recomputable algorithms as a means for efficiently updating values in the Summary Database implied that the incrementally recomputable code for each function must be stored someplace. One could view, and possibly store, the code as a collection of rules for updating the values. These rules are stored in the Management Database. In addition to rules defining how a function is to be recomputed we propose to store rules that describe how derived data is to be updated when the data upon which they are based are changed. To illustrate consider the following example.

Since the residuals of a model may be required for several "goodness of fit" tests they are typically stored as a new attribute in a data set (i.e., they are incorporated into the view). Updating even a single value in the attribute upon which the residuals depend requires regeneration of the entire vector (since the model may change). Thus, the rule stored in the Management Database for this case would specify regeneration of the entire vector (or simply marking it as out of date).

As an another example consider storing in a new column the result of a different function, say the sum of three attributes, or the logarithm of some attribute. For both of these functions the derived value is dependent only on values in the same row. Thus, the rule stored in the Management Database would indicate that the effect of the update to the input attribute is "local", i.e., it will require the computation of only one value.

To summarize, our proposed statistical DBMS organization consists of several Summary Databases and a single Management Database. Each Summary Database acts as a cache for results of previous function executions as well as a storing house for commonly accepted and used summary values describing the structure of the data for a specific concrete view over the database. The Management Database contains rules to be used by the DBMS when accessing the Summary Database and when updating the user views. In other words, the Management Database contains control information that drives the operation of the DBMS on the concrete views as well as their respective Summary Databases.

4. Discussion

In Section 3 we motivated and outlined the overall organization of the data management component of the statistical database system that we are investigating. In this section we discuss several issues that arise in the proposed environment that must be resolved before we can proceed with an implementation. In some cases we shall describe a potential solution to the problem we raise. In others, we will indicate the line of thought we are pursuing.

4.1. Handling Updates

We envision that the analyst will specify an update to the data set by using a predicate in a similar manner to what is currently done in relational systems. Thus, the operation specifies the attributes affected and the nature of the update. We propose to cluster values stored in the Summary Database on the attribute name. Thus, given an attribute name we can retrieve all the values associated with that attribute, along with their respective function names, stored in the Summary Database. For each function we must retrieve from the Management Database the list of rules that specify the actions to be applied in order to obtain the new value.

4.2. Specification of the Update Rules

Given the description of a function, in some suitable high-level form, how do we obtain rules that specify computation of the new value without accessing data in the view? This problem, described in other terms, has received some attention in the literature. Of particular interest is the work on finite differencing [PAIG80]. The purpose of finite differencing is to transform a program into another, more efficient, equivalent program by taking the "derivative" of the original code segment. The origin of finite differencing is in Cocke´s reduction of operator strength -- a method for program optimization. The basic idea is outlined below.

We have a code segment that computes the function f on arguments x1,x2,...,xn, that is f(x1,x2,...,xn). Assume that we have computed f for specific values of the argument list. Now, suppose we need to recompute f several additional times for the same argument values, but with the value of some argument, say x2, changing each time. We illustrate this in Figure 5. What finite differencing attempts to do is to generate a new version of the function f, say f´, that will take advantage of the fact that the values of most of its arguments are to remain constant. In fact, f´ may only have one argument (x2). Description of the method used to accomplish this task is beyond the scope of this paper. We do wish to mention, though, that Koenig and Paige [KOEN81] discuss the application of finite differencing to the generation

```
{initialize x1,x2,...,xn}
for i := 1 to n do
   x2 := g(i);  {g is some function}
   result[i] := f(x1,x2,...,xn);
end;
```

Figure 5

of the incrementally recomputable code for several commonly used aggregate operators. In particular, they consider totals and averages.

It is not clear to us, at this point, whether finite differencing can be applied to more complicated functions such as median. The problem lies in the fact that some functions reflect an ordering on the input data. Updating the data will change the ordering and, possibly, the result of the function. (Most updates to the data set will not affect the min or max values; medians, on the other hand, are more susceptible to changes in the data). Unlike the total or average operators, there are no methods for describing the ordering of the data in some concise manner which can be manipulated to yield a description of the new ordering of the data after the update operation.

An alternative to the use of finite differencing, or any other automatic technique, for the "difficult" functions is to deal with them manually. Thus, for the median and quartiles (and other commonly used order statistics) one can use the following approach (described for the median). Initially, we must compute the median. Rather than saving a single value as the result of this computation, we will store, in the Summary Database, a histogram of some number, say 100, of values around the median. Associated with the histogram will be a pointer which will initially be set to the median. As updates are made to the original data set the pointer can be moved up and down the list reflecting the changes. When the pointer runs off the list a new histogram will have to be generated.

An important observation to be made is that generation of the new histogram will require only a single pass over the data. We will know what the approximate range of values for the new histogram will be since updates to the list cause the value of the median to change only slightly. Thus, using a simple hashing scheme that has 101[2] buckets we can generate the new histogram passing over the entire data set only once (the 101st bucket is used for all the values other than the 100 desired values)[3].

4.3. Database Machine Support

Our interest in management of statistical databases arose because such databases seem to be a natural candidate for database machine support. Statistical database are very large; update operations are relatively infrequent (in most types of statistical databases); and, operations access large amounts of data in a regular manner. We began our work on this project in the Summer of 1981 hoping to select the data management component of some statistical package and use it as a front-end for a database machine. We found, to our chagrin, that the state of data management software in statistical packages in relatively poor. Thus, we felt that we had to begin by characterizing the data management functions of statistical packages before examining how a database machine could be used to improve the performance of

---

[2] Note that we may wish to have more than 100 buckets if we are not sure what the density of the values is around the expected new median.

[3] Floating point values present some difficulties for this scheme.

such packages.

It is still too premature to indicate exactly how a database machine could be used in the environment that we have proposed. There are, however, four obvious areas to examine. First, since no auxiliary information exists about the raw database, a database machine might be used to materialize views by executing the various relational operators (selection, join, projection, aggregate). However, it is not clear to us that whether the use of a database machine for view materialization is viable if the raw database cannot be kept "on-line".

A database machine could also be used to manage the Summary Databases. Operations on the Summary Databases are primarily searches whose result sets are small. A pseudo-associative disk [SLOT70] of some type seems to be reasonable database machine organization for this purpose.

It may be the case that the overhead associated with managing the Summary Database may be too high; or that generating rules for incrementally recomputing functions is a research topic still unripe for implementation. In that event a Summary Database could still be maintained. However, after each update operation all the values associated with the updated attribute will be marked as invalid. When required they will be regenerated using the original algorithm. A database machine may be used to perform the function computation if the concrete view can maintained on a "fast" mass storage device, such as disk. Note that unlike the previous proposed use for a database machine, the machine organization would have to be considerably more sophisticated than an associative disk because of the nature of the operations performed.

Another possible use for a database machine is for execution of operations whose values will not be stored in the Summary Database. Examples are operations whose results are vectors which are added to the data set stored in the concrete view (e.g., the residuals of a linear regression).

5. Conclusions and Future Plans

5.1. Summary

In this paper we have presented a description of how statistical databases are structured and accessed. In particular, we have demonstrated that they exhibit significantly different characteristics from "corporate" databases, both in terms of the data they contain (its structure and size) and its use (types of queries). It is our contention that neither existing database management software nor statistical software provide users of scientific and statistical databases with efficient and easily used means for accessing the data.

We have also presented a preliminary organization for a database system to facilitate the statistical analysis of very large data sets. The key components of the proposed system include concrete views of the raw database for each analyst, a Summary Database for each view that automatically maintains the results of a number of system and analyst defined functions while the view is being edited, and a Management Database that is used for maintaining definitions of new functions and the history of changes that have been made to each view (including a specification of the operations that were utilized to materialize the view). This organization is intended to serve as a framework for future research by ourselves and other computer scientists interested in statistical data management problems.

We feel the proposed design is a viable approach for statistical database management systems as it appears to have the potential of providing a flexible but very responsive environment for statistical data management. Two papers at a recent workshop on Statistical Database Management, describe research efforts similar to ours.

A group of researchers from the Universities of Tsukuba and Hiroshima described enhancements to an existing relational system [IKED81]. The data dictionary of the DBMS is expanded to include definitions of statistical functions, means for specifying codings (as in Figure 2), and certain summary statistics such as mean and standard deviation. In addition the DBMS provides the user with the capability of creating and querying summary tables which are essentially cross tabulations. Finally, sophisticated means for displaying the data (particularly complex cross tabulations) are provided. The main differences between this approach and ours are:

(1) Reliance on a conventional DBMS and the storage structures and access mechanisms it provides.

(2) Updates to the raw database are not propagated to the summary data.

(3) No means for examining the update history to a view and/or undoing updates are provided.

Neil Rowe of Stanford University [ROWE81] proposed using a Database Abstract in which some precomputed values of statistical functions will be stored. A set of inference rules will be used to calculate the results of other functions, based on the values stored in the Database Abstract.

All three research projects are "close" in the sense that they all try to reduce the number of accesses to the database (in our case the view) by storing summary descriptions of the data. The Database Abstract differs from the other two projects in that it attempts to provide the users with estimates as the results of queries. All three projects need to deal with queries whose results are non-scalar.

5.2. Future Plans

After further investigation of the problems outlined in Section 4, we intend to design and implement the statistical database system that we have proposed. The basis for this system will most likely be the Wisconsin Storage System (WiSS). WiSS is package of storage structures and access methods that is currently being implemented at Wisconsin by David DeWitt, Randy Katz, and a group of students. As the basis of the statistical software components of the proposed system, we intend to investigate using the S [BECK78] system from Bell Labs.

After implementation we intend to analyze the performance of the resulting system to locate bottlenecks. Then, we intend to design and implement a database machine to eliminate these bottlenecks.

REFERENCES

[BECK78] Becker, R.A. and J.M. Chambers, "Design and Implementation of the 'S' System for Interactive Data Analysis," Proceedings of the 1978 IEEE COMPSAC Conference, pp 626-629.

[BRAG81] Bragg, A. W., "Data Manipulation Languages for Statistical Databases -- The Statistical Analysis System (SAS)", Proceedings of the Workshop on Statistical Database Management, Menlo Park, CA., December 1981.

[BUHL79] Buhler, S. and R. Buhler, "P-STAT 78 User's Manual" P-STAT Inc., P.O. Box 285, Princeton, N.J.

[BURN81] Burnett, R.A. and J.J. Thomas, "Data Management Support for Statistical Data Editing and Subset Selection," Proceedings of the Workshop on Statistical Database Management, Menlo Park, CA., December 1981.

[CENS72] "Public Use Sample: Description and Technical Documentation," U.S. Bureau of the Census, Washington, D.C., 1972.

[CHAN81] Chan, P. and A. Shoshani, "SUBJECT: A Directory Driven System for Organizing and Accessing Large Statistical Databases," Proceedings of the 7th International Conference on Very Large Data Bases, France, 1981, pp. 553-563.

[DIXO79] Dixon, W.J. and M.B. Brown, "BMDP-79 Biomedical Computer Programs P-Series" University of California Press, Berkeley.

[EGGE80] Eggers, S.J. and A. Shoshani, "Efficient Access of Compressed Data," Proceedings of the 6th International Conference on Very Large Data Bases, Montreal, 1980, pp. 205-211.

[EGGE81] Eggers, S.J., Olken, F., and A. Shoshani, "A Compression Technique for Large Statistical Databases," Proceedings of the 7th International Conference on Very Large Data Bases, France, 1981, pp. 424-434.

[GEY80] Gey, F. and H. Holmes, "An Overview of the LBL Socio-Economic Environmenta Demographic Information System," Computer Science and Mathematics Department, Lawrence Berkeley Laboratory, 1980.

[IKED81] Ikeda, H. and Y. Kobayashi, "Additional Facilities of a Conventional DBMS to Support Interactive Statistical Analysis," Proceedings of the Workshop on Statistical Database Management, Menlo Park, CA., December 1981.

[KOEN80] Koenig S. and R. Paige, "A Transformational Framework for the Automatic Control of Derived Data," Proceedings of the 7th International Conference on Very Large Data Bases, France, 1981, pp. 306-318.

[NIE75] Nie, N.H., C.H. Hull, J.G. Jenkins, K. Steinbrenner, and D.H. Bent, "SPSS Statistical Package for the Social Sciences (second edition)" McGraw Hill, New York.

[PAIG80] Paige, R., "An Efficient Implementation of Automatic Finite Differencing," Department of Computer Science, Rutgers University, August 1980.

[ROWE81] Rowe, N.C., "Rule-Based Statistical Calculations on a Database Abstract," Proceedings of the Workshop on Statistical Database Management, Menlo Park, CA., December 1981.

[RYAN81] Ryan, T.P., B.L. Joiner, and B.F. Ryan, "Minitab Reference Manual", Minitab Project, Statistics Department, Penn. State Univ., University Park, PA.

[SAS79] "SAS User´s Guide 1979 Edition", SAS Institute Inc., P.O. Box 10066, Raleigh, North Carolina.

[SLOT70] Slotnik, D.L., "Logic per Track Devices" in "Advances in Computers", Vol. 10., Frantz Alt, Ed., Academic Press, New York, 1970, pp 291 - 296.

[TUKE77] Tukey, J.W., "Exploratory Data Analysis", Addison-Wesley, Reading, Massachusetts.

[TURN79] Turner, M.J., Hammond, R., and P. Cotton, "A DBMS for Large Statistical Databases," Proceedings of the 5th International Conference on Very Large Data Bases, Brazil, 1979, pp. 319-327.