OLAP and Statistical Databases: Similarities and Differences[1]

Arie Shoshani
Lawrence Berkeley National Laboratory
Berkeley, California 94720
shoshani@lbl.gov

Abstract

During the 1980's there was a lot of activity in the area of Statistical Databases, focusing mostly on socio-economic type applications, such as census data, national production and consumption patterns, etc. In the 1990's the area of On-Line-Analytic Processing (OLAP) was introduced for the analysis of transaction based business data, such as retail stores transactions. Both areas deal with the representation and support of data in a multi-dimensional space. Much of the OLAP literature does not refer to the Statistical Database literature, perhaps because the connection between analyzing business data and socio-economic data is not obvious. Furthermore, there are papers published in one area or the other whose results can be applied in both application areas.

In this paper, we compare the work done in these two areas. We discuss concepts used in the conceptual modeling of the data and operations over them, efficient physical organization and access methods, as well as privaccy issues. We point out the terminology used and the correspondence between terms. We identify which research aspects are emphasized in each of these areas and the reasons for that. We conclude by arguing for the support of a Statistical Object data type as one of the fundamental structures that object-oriented data models and systems should support.

1 Introduction

Both OLAP (On-Line Analytical Processing) and Statistical Databases (SDBs) deal with multidimensional datasets, and both are concerned with statistical summarizations over the dimensions of the data sets. Much of the work on SDBs took place in the 1980's but continues till today, while OLAP database work started mostly in the 1990's. Yet, despite the similarities in concepts the two communities do not, by and large, refer to each other's work.

We explore in this paper, the reasons for this state of affairs. Some can be traced to the applications that motivate each. Some can be explained by the different aspects emphasized in each. For example, much of the work in SDBs concentrated on conceptual modeling, while most of the work in OLAP concentrated on access performance.

We start, in section 2, with a couple of illustrative example of SDBs and OLAP databases, and expose the conceptual similarity between them. In section 3, we describe typical application areas in SDBs and OLAP, and discuss similarities as well as differences in such applications. In Section 4, we illustrate the various ways that OLAP and SDBs have represented their conceptual model structures, and in Section 5, we discuss conceptual operators. We also show the correspondence of different terms used in both areas. In section 6, we discuss the main ideas proposed in SDBs and OLAP for physical organization and indexing to achieve efficient storage and access. In section 7, we briefly discuss privacy issues and proposed techniques. We conclude with a summary and the reasons why a Statistical Data object should be supported as a specialized data type for systems that support statistical summarization.

We wish to point out that the references used in this papers are intended to be illustrative of the concepts discussed, rather than comprehensive. We apologize if we left out other relevant references.

2 Examples of SDBs and OLAP Databases

We describe in this section an example of the traditional 2-dimensional (2-D) representation of SDBs, followed by a data cube representation of OLAP databases, and show the correspondence in concepts.

2.1 The traditional representation of Statistical Databases

Consider the dataset represented in Figure 1 as a 2-D table, taken from [RS90]. It shows "Employment in California" by "sex" by "year" by "profession" (the numbers are fictitious). This form of representing multidimensional tables is very popular in the Statistics area, perhaps because in the early days it was only possible to represent information on paper and thus the 2-D restriction.

There are several items to note in this example:

(i) By necessity, more than one dimension must be represented by the rows and the columns if more than 2 dimensions exist in the dataset. This is accomplished by selecting an arbitrary order of the dimensions for the rows

[1] A shortened version of this paper is published in ACM TODS 1997

and the columns. In Figure 1, the rows represent the two dimensions "sex" and "year", which were arbitrarily ordered "sex" first, then "year".

Employment in California			Professional Class					
			Engineer		Secretary		Teacher	
			Profession		Profession		Profession	
			Chemical Engineer	Civil Engineer	Junior Secretary	Executive Secretary	Elementary Teacher	high school Teacher
Sex	Male	Year 91	197,700	241,100	534,300	154,100	212,943	123,740
		Year 92	209,900	278,000	542,100	169,800	213,521	145,766
		Year ..	.....	.....	.....	.....	.....	.....
	Female	Year 91	25,800	112,000	667,300	162,300	216,071	275,123
		Year 92	28,900	127,600	692,500	174,400	217,520	299,344
		Year ..	.....	.....	.....	.....	.....	.....

Figure 1: a 2-D representation of statistical data

(ii) The columns in this example do not represent 2 dimensions, although their layout looks exactly the same as for the rows. Rather, "professional class" and "profession" represent a hierarchical relationship between the instances of "professional class" (e.g. "engineer") and the instances of the "profession" (e.g. "civil engineer"). This structure is often referred to as a "classification hierarchy". This can be easily verified by realizing that the measure in a cell of this table refers to 3 dimensions only (sex, year, profession), not 4. For example, 287,000 refers to the number of male civil engineers employed in 1992.

(iii) The label "Employment in California" represents the summary measure for this dataset being "Employment", but it also says that this dataset has an additional dimension "state" where the instance value selected is a singleton "California". Indeed, dataset may be only one "page" of a collection of pages each representing another state.

(iv) There is a summary function implied with this data set for further summarization is done (such as over "sex" or "profession"). In this case the summary function is "sum". We note that while "sex" and "year" is a one level summarization, the summarization over "profession" can be done to the "professional class" level or over all professions and all professional classes. This is because of the classification hierarchy structure.

To summarize, this dataset has the following conceptual structure:

Summary measure: employment
Summary function: sum
Dimensions: sex, year, profession, state=California
Classification hierarchy: professional class --> profession

Note : we use the notation "-->" to mean one-to-many, as in: each "professional class" has many "professions".

2.2 An OLAP example using a data cube model

In Figure 2 we show a typical example of an OLAP database, represented as a multidimensional "cube". Obviously, this graphical representation can only be used for up to 3 dimensions. But, it is useful for illustrative purposed. This data cube example contains the "quantity sold" (in dollars) for a particular store chain, for each "product" by "store" by "day".

We note the following:

(i) The dimension "store location" has a natural hierarchy to it. "Store location" has two components: "city" and "store number". Since stores are organized according to the city they are located in, the hierarchy city --> store exists. The actual measures are for particular stores, and could be summarized to a city level if we so desire. However, if store numbers (or some other identifier) are not globally unique, then one needs to concatenate "city, store number" to make it unique. Following the terminology of ER models, one can say that there exists an "ID dependency" of store on city.

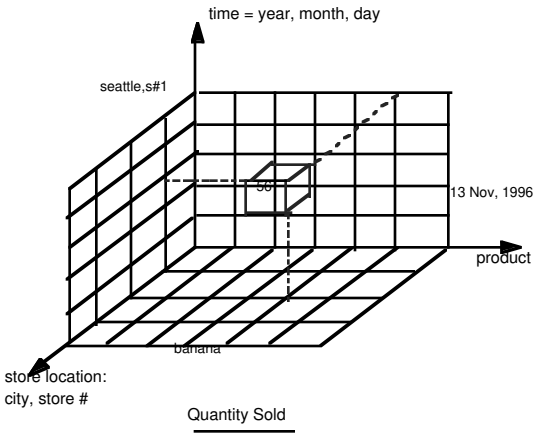

Figure 2: a "data cube" representation of OLAP data

(ii) The dimension "day" is another example of an "ID dependent" classification hierarchy. Given that day is identified with its month and year (e.g. 13 Nov., 1996), then it is ID dependent on the month (Nov., 1996) which in turn is dependent on the year (1996). Thus, it can be treated as a 3 level classification hierarchy for the purpose of summarization to the month or the year level.

(iii) The summary measure "quantity sold" has a unit associated with it: dollars. This is not unusual for SDBs as well, such as a measure for "average income" (in dollars)

over a population. The reason for lack of a unit, as was the case with "number of people employed" is the type of the summarization operation originally applied to the individual instances (i.e. people) of dataset: count.

To summarize, for this OLAP databases, we have the following conceptual structure:

Summary measure: quantity sold
Summary function: sum
Dimensions: product, store location, day
Classification hierarchy: city --> store
Classification hierarchy: year --> month --> day

As can be seen from a conceptual structure point of view, the SDB and OLAP example have exactly the same components: a summary measure, a summary function, one or more dimensions, zero or more classification hierarchies. It should be obvious that the OLAP example can be represented in the 2-D SDB representation, and the SDB example in the data cube form. We note here that it is easy to extend this model to have several summary measures associated with the same dataset over the same dimensions (such as "population and average income by state by year by sex by race"). In the SDB literature this is sometimes referred to as a "complex statistical object".

3 Application Areas in SDBs and OLAP

The lack of recognition of the similarity between the SDB and OLAP areas is probably due to the application areas that each originated from. Whereas the SDB area is mainly motivated by socio-economic databases, which are usually the domain of statisticians, the OLAP area is driven by business applications, and their analysis for the purpose of decision making. This is the reason that OLAP is considered a component of Data Warehousing capabilities. The ultimate purpose of a warehouse is to collect and analyze information to make decisions. Decision makers are not necessarily statisticians, but more typically business managers.

In spite of the origin of these two areas, there are many similarities in the problems they tackle. We explore some examples below, and identify various needs in each, to help identify similarities and differences in emphasis.

3.1 Example SDB application area.

(i) Census data

Census data are collected to assess trends of populations. One of the most crucial issues in such data is the representation of a multi-level geographic classification hierarchy. This hierarchy can be very voluminous. In the US, census data is collected on individuals, and summarized to the "block" level. There may be thousands of blocks per "county" and tens to hundreds counties per state. In addition, there are cities and towns that may cross county boundaries. To make things even worse, boundaries of blocks, counties, and cities are sometimes redefined. Thus, there is a temporal component to such information. It is easy to see that the management of such classification hierarchy structures require a sophisticated system. In reality, specialized systems have been developed for such purposes.

Census data can have discrepancies even in simple dimensions, such as "race" or "age group". For various reasons, some legislative, the classification of "races" and "age groups" can vary from state to state or county to county. For example, birth rates which are collected at the state level, may have different race classification. Thus, summarizing over many states poses special problems.

The census data cannot by law (in the US and most other countries) compromise the privacy of an individual. Thus, only summarized data is made available. The question is whether summarized data indeed maintains privacy. This is a major issue for the census bureau, which is the reason for "cell suppression" -- a term used to say that some cells in the multidimensional space will not be released in the summarized form, since information about individuals may be inferred.

In summary, the main problem areas are: complex classification hierarchies, discrepancies between dimension classifications, and privacy.

(ii) Economic data

This type of database contains statistics of sales and income of industries, or consumption and productions of various products such as crude oil.

Here again, regional classification hierarchies need to be supported, but the hierarchies are not likely to be as deep as in census data. They may go down to a county or city level, but not to a block level. On the other hand, the industry classifications may be many levels deep, and the definition of an industry or a product may vary over time. Privacy issues may exist here as well if one wants to protect certain companies as to their production or financial information.

(iii) Natural resources

This type of databases monitor such things as water levels in dams, logging in forests, floods and river flows. The most striking need in such databases is the support of

physical geography. This applies not only to the objects being monitored (e.g. where measurements are taken along a river) but also where the objects are relative to regional entities (e.g. how to represent rivers that cross or border states).

For natural resource databases, there is a strong connection to geographic databases. The multidimensional concept still needs to be represented, such as water level per month per measuring station of rivers, but the geographic dimension is where the complexity lies.

In this areas there may also be complex multi-level classification structures depending on what is measured. For example, botanical classifications for plant growth, or species classifications for endangered animal and birds can be quite complex. Such are usually contained in books or specialized databases.

3.2 Example OLAP application areas

(i) Retail store sales analysis

Sales analysis is performed mainly to discover trends of shopping habits or associations between sales of various items. Thus, each transaction has many attributes associated with it, such as the profile of the shopper (when known, such as credit card users), time of day, what items were bought in the same transaction, etc. It is typical that the dimensionality is high (10 dimensions is not uncommon), but the complexity of each dimension is fairly low, even for items such as location of store.

An exception is the classification of products. There may be thousands of items, classified into multi-level hierarchies. An interesting observation can be made here, in that products can be classified in many different ways, such as by type of the product, or by the price range of the product. We refer to this phenomenon as "multiple classifications over the same dimension".

(ii) Stock market databases

The most obvious feature of a stock market database is it temporal dimension. It is usually represented as a time series (i.e. regular interval of time, such as a day) of the days that the market is open (weekdays, excluding holidays). A classification hierarchy over time may exist such as for generating weekly or monthly averages, highs and lows. Another dimension that may have a complex structure is the stock classification, usually by type of industry (e.g. utilities). Here, too, multiple classifications over the stock may also be maintained, such as classification by rating (such as "AAA"), type of stock (e.g. whether it returns dividends), etc.

(iii) Health maintenance organizations (HMOs)

These databases track cost per patient per procedure or visit. They use multi-level disease classifications which are quite complex. Such classifications are maintained by standard medical committees, and the classifications can change over time. There is the additional complexity that the classification structure is not a strict hierarchy. For example, "lung cancer" belongs under the "cancer" disease category, as well as under the "respiratory" disease category. Similarly,
a physician may have multiple specialties.

Privacy of individuals and doctors information in such databases is essential. Typically summary sets are made public at a high level of summarization, so that privacy compromise (i.e. inference about an individual) is not likely. Nevertheless, privacy techniques need to be used and enforced.

3.3. Several points of difference

3.3.1 Differences in emphasis

From the examples above it is clear that SDBs and OLAP databases have very similar problems. However, each area has emphasized different aspects. The SDB literature has emphasized treatment of complex multi-level classification structures, while the OLAP literature assumes pretty simple structures. Some SDBs have a strong emphasis on regional, spatial and geographic dimensions, while in some OLAP databases the emphasis is on temporal dimensions, including time series data. The SDB literature has researched privacy issues extensively, while the OLAP literature ignored this issue. Another difference in emphasis is that most of the OLAP literature deals with efficiency of access of gigabytes datasets (to satisfy the "on-line" part of OLAP), while the SDB literature emphasized other issues as well, such as conceptual modeling.

While the emphasis is on different aspects, it should be clear from the examples above, that potentially the same problems exist in both areas. For example, OLAP research can benefit from work done in modeling of classification structures and privacy, while the SDB research can benefit from work done on efficient access of very large statistical data sets.

3.3.2 The summarizability property

Another area that was largely ignored so far by OLAP research is the problem of summarizability. One reason for this problem is that classification structures do not always make a strict hierarchy, and thus they cannot be readily

summarized. For example, if one adds physicians by specialty to produce a summary dataset, then summarization over multiple (or all) specialties of the number of physicians will produce erroneous results, since the physicians with multiple specialties will be counted multiple times. This problem is referred to as the "summarizability" property [RS90], and the conditions under which this property hold were investigated.

Other reasons for non-summarizability also exist. For example, suppose that we have a states--> cities classification structure. If we are given the populations of all cities in a state, it is clearly impossible to summarize that to the state level (to mean the population of a state), because of the missing villages, farms, etc. This is because this classification structure is incomplete relative to the measure "population". Another example is that is meaningless to add populations over time (say, over months), while it makes sense to add accident counts over time. This has to do with the type of summary measure used. A recent paper [LS97] contains a detailed exploration of the summarizability conditions. This is an extremely important issue that is largely ignored in the OLAP literature, while in the SDB literature it is a major issue.

3.3.3 Micro-data, macro-data, and metadata

There is another distinction between OLAP and SDBs that can be made. While SDBs are usually derived (summarized) from other base data, OLAP databases often represent directly the base data. For example, census databases are derived from the data about individuals, while a sale transactions database is the base data collected. The data about the individuals or original objects from which SDBs are derived is referred to in the SDB literature as the "micro-data", and the summarized dataset as the "macro-data". In addition, the data associated with the classification structures, is referred to as the "metadata". As was shown in the example applications areas above, the metadata can be quite extensive, and are often managed by specialized systems, or general purpose database systems, such as relational systems. The concept of the metadata is drawing interest in a variety of areas, such as scientific databases. However, in the context of SDBs it only refers to the classification hierarchies.

The reason that SDBs mostly represent the macro-data is either for privacy reasons, or because the original dataset is of no interest (i.e. only the summaries are needed for statistical analysis). In OLAP, summaries may obscure the phenomena we wish to discover, thus we start with the original dataset. These generalizations, of course, do not always hold, but by and large most examples bear this observation.

In the next 2 sections we will demonstrate that the conceptual structures and operations used in SDBs and OLAP are essentially the same, but different terminology is used in each. We will point out the correspondence between the terminology used with the hope that both areas will benefit from each other advances.

4 Conceptual Modeling -- Data Structures Representation

In this section, we will describe 3 popular models used for SDBs and OLAP: graph models, tabular models, and the data cube model. We compare concepts and show the equivalence in terminology used in each area.

4.1. Graph models

In the SDB area it was recognized early on that a conceptual model should avoid the limitations of the 2-D tabular representation of a statistical dataset [CS81]. The graph model, shown in Figure 3 for the 2-D table of Figure 1, shows the elements of the graph model. There are 3 kinds of nodes: the S-node for the "summary attribute" (which corresponds to "summary measure"), the X-node for the "cross product" representing the multidimensionality, and the C-node to represent the "category attribute" (which corresponds to "classification"). Note that a C-node above a collection of another C-node represents one level of the hierarchical classification structure, and thus is referred to as the "category hierarchy".

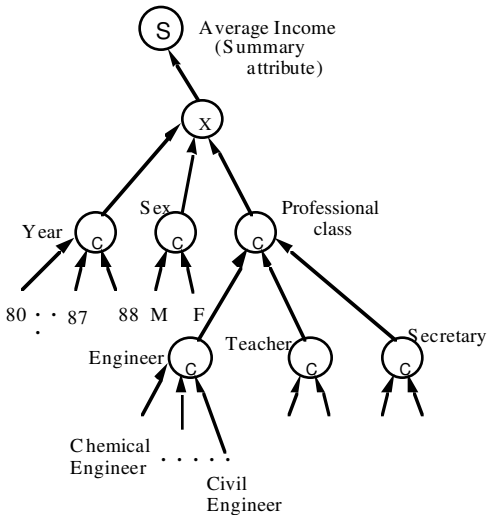

Figure 3: a graph model for statistical data

This representation has several advantages over the 2-D tabular representation. (i) The dimensions do not have to be partitioned into those that go into columns and those that go into rows. (ii) This representation is insensitive to

node permutation, i.e. there is no need to select an order for the dimensions assigned to rows or columns. (ii) The classification hierarchy is explicit, and thus a higher level category attribute (such as "professional class") cannot be confused with a dimension of the statistical dataset. (iv) It is possible to represent conveniently a moderate number of dimensions on a single screen (up to ten or so).

However, this representation has some problems in real applications in representing complex datasets. One of the problems is that in case that the number of categories (values) of a category attribute was large (e.g. 50 states), it was not possible to represent that on screens conveniently. But, a more fundamental problem is that intermediate nodes of a category hierarchy have two roles: to hold the category value of the node above it, and to represent the name of the category attribute for the nodes below it. For example, the node labeled "engineer" stands for the value of a "professional category", yet at the same time it represent "profession" whose values for engineer are shown below.

The reason for this is that a classification structure has a schema component and an "instances" (values) component. That is, the "category attributes" and their structure constitute the schema, and the category values constitute the instances (data) for this schema. In a later paper [RS90] this approach was taken. The definition of a Statistical Object was introduced, and the model was labeled STORM (Statistical Object Representation Model) to emphasize the independence of the model from 2-D layout semantics. The graph model for the schema of the example in Figure 1 is shown in Figure 4.

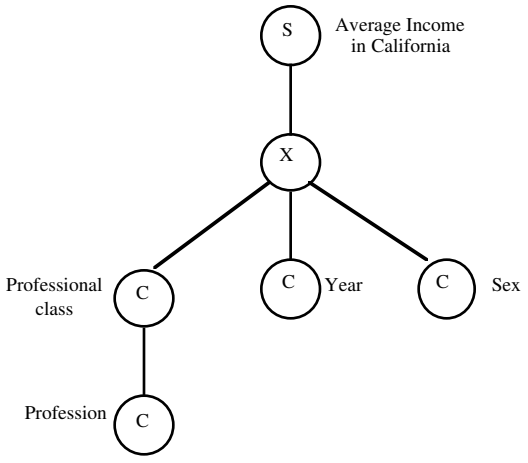

Figure 4: a schema graph model for statistical data

We note that the same advantages mentioned with the previous graph model hold for this representation as well. The previous disadvantages do not exist. In particular, there is no confusion of category values and category attribute labels, and there is no problem with a large number of category values. This representation is also naturally suitable for multi-window system. Selecting any category attributes opens an secondary window with the category values, which can be represented as a scrollable list.

This graph representation is also amenable to a better semantic organization of the multidimensional space. To illustrate this point, consider that we add 2 more dimensions to the previous example of Figure 4: "race" and "age". Figure 5 shows how multiple X-nodes can be used to organize the dimensions into semantic "subject" groups. The X-node labeled "socio-economic categories" is used as a place holder for the three socio-economic category attribute: "sex", "race" and "age".

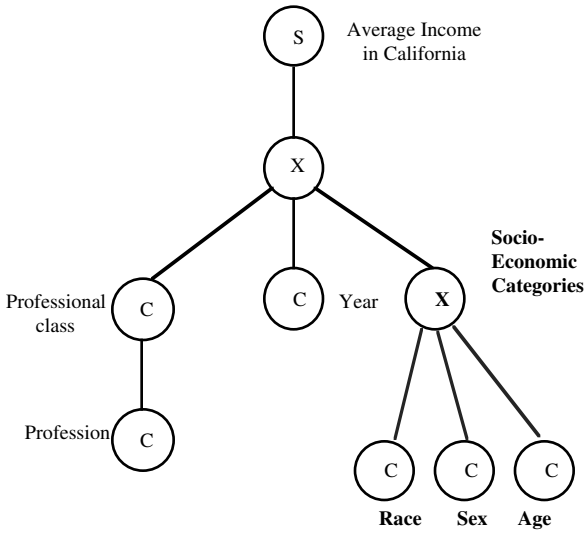

Figure 5: use of an X-node for semantic clarity

This "grouping of dimensions" can be applied iteratively to generate multiple levels. Mathematically, one can easily show that the two structures shown in Figure 6 are equivalent, and thus can be applied iteratively. The usefulness of this construct is the ability to partition the dimensions of a dataset with a large dimensionality into groups by content, thus making the schema and the user interfaces to it more comprehensible to users.

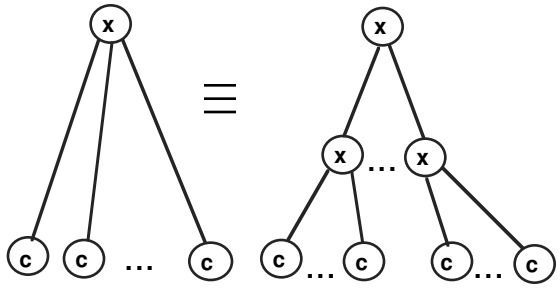

Figure 6: equivalence of dimension groups

It is interesting to note that 2-D tables can also use the same construct to represent the dimensions grouped into columns and rows" as shown in Figure 7. If we assign order to the node under rows and columns we can practically capture the physical layout of a 2-D statistical table. This can be useful in case that one needs to interface to legacy systems that store and access information according to the 2-D layout.

Before we discuss tabular models, we look briefly at the semantics and complexity of classification structures. The properties of classification structures need to be captured to make the semantics of the classification structure clear to the user. But in addition, it is necessary to capture these properties in order to determine summarizability as discussed in section 3.3.

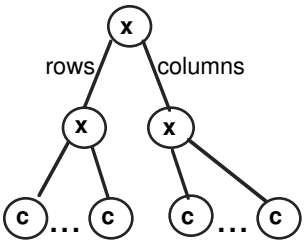

Figure 7: a schema graph representation of a 2-D table

4.2 Semantics of the classification hierarchy

Before we discuss tabular models, we look briefly at the semantics and complexity of classification structures. The properties of classification structures need to be captured to make the semantics of the classification structure clear to the user. But in addition, it is necessary to capture these properties in order to determine summarizability as discussed in section 3.3.

Figure 8 shows 3 types of classification structures. The first (from left) shows a regular "containment" hierarchy of cities into states. As mentioned above, it is important for the purpose of summarizability to know whether this hierarchy is "complete" relative to the summary attribute [LS97]. If for example, we want to summarize average income for the state, it is incorrect to leave out the parts of the state that are not considered cities. If however, we summarize the number of museums in states, and we know that all museums are only in cities, then it is OK to summarize that. This is a purely semantic condition that needs to be identified. The second condition is whether the hierarchy is a true hierarchy. If for example, a city sprawls over the boundary of 2 states (such as "Minneapolis-St. Paul), summarizability will not hold if the city appears as contained in these two states. Other conditions have been investigated in [LS97].

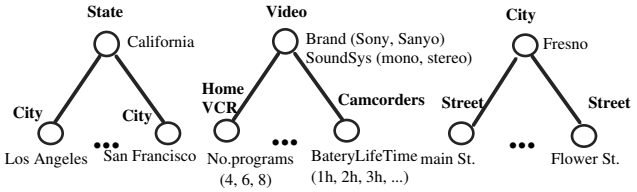

Figure 8: 3 types of classification structures

The middle example in Figure 8 illustrates the conceptual relationship of classification structures to ISA structures. In this example taken from [LRT96], the instances of the classification of the bottom category attribute ("home VCR", "Camcorders", etc.) inherit properties of the top category attribute ("video"). Associating properties with instances of the classification structure adds richness and complexity to the modeling of classification structures. The main advantage is that in querying the database, conditions can be applied to the properties of the classification instances (e.g. selecting only "Sanyo" products for summarization).

The rightmost example of Figure 8, shows the ID dependency property that was discussed in section 2.2. This property needs to be explicitly modeled as part of the classification structure in order to maintain and search by the complete object identity.

4.3 Tabular models

We will discuss in this section several tabular models that are used both in SDBs and OLAP.

We show in Figure 9 the same 2-D statistical table as in Figure 1, but we added a couple of columns that show summarization over the rows. As can be seen one such "total" column is associated with the professional class "teacher". Similar columns for the other professional classes also exist but are not shown. The second column "total" represents a summary over all the professional categories. These totals are often referred to by statisticians as "marginals" (because they appear on the "margins" of tables). The disadvantages of this tabular representation were discussed in Section 4.1 above in the context of using graph models.

Employment in California		Professional Category							
		Engineer		Secretary		Teacher			total
		Profession		Profession		Profession			
		Chemical Engineer	Civil Engineer	Junior Secretary	Executive Secretary	Elementary Teacher	high school Teacher	total	
Sex	Year								
Male	91	197,700	241,100	534,300	154,100	212,943	123,740	336,683	1,463,883
	92	209,900	278,000	542,100	169,800	213,521	145,766	.....	.....
	..	.....	.....	.....	.....	.....	.....	.....	.....
Female	91	25,800	112,000	667,300	162,,300	216,071	275,123	.....	.....
	92	28,900	127,600	692,500	174,400	217,520	299,344	.....	.....
	..	.....	.....	.....	.....	.....	.....	.....	.....
	99	30,800	134,100	811,500	213,600	221,057			

Figure 9: statistical table with "marginals"

The marginals are usually not included in the database if they can be derived. However, in situations where summarizability does not hold it is necessary to store these values in the database as well. It is generally not efficient to compute the marginals for very large datasets. This is one of the most important problems addressed by OLAP research as will be discussed in section 6.

Another tabular representation stems from the popularity of the relational model. The table in Figure 10 shows a relation with 8 columns for a statistical object. The advantage of this representation is its familiarity and the fact that it can be readily implemented on a relational system, as well as the ability to integrate it with other data in the same model.

State	county	year	race	sex	age-group	population	avg. income
Alabama	Autauga	1990	white	male	1-10	11,763	0
Alabama	Autauga	1990	white	male	11-20	9,763	3,342
Alabama	Autauga	1990	white	male	21-30	15,763	34,342
...	...	...	...	...	31-40	14,566	37,444
...	...	...	...	...	...	11,372	38,776
...	...	...	...	male	ALL	89,483	36,775
...	...	...	...	Female	1-10	8,457	0

Figure 10: a relational representation of a statistical object

However, there are several problems with this representation. (i) There are no semantics associated with a relational table to distinguish between category and summary attributes. The last two columns, "population" and "avg. income" are artificially separated from the previous 6 to illustrate this point. (ii) There is no distinction between the attributes associated with the classification hierarchy and the dimensions. This table represents a 5-dimensional dataset (not 6) since state is not one of the dimensions (only county is). We note that a normalization process would have identified this situation and state would have been removed into another relation with "state", "county" as attributes. (iii) The repetition of values in the first six columns make it difficult to see the categories of each of these category attributes. Furthermore, if the relation is implemented in a straight forward way, this layout is very wasteful of space since it stores the entire cross product. Consequently, summarization and query processing is slowed down accordingly. In section 6 we discuss other alternatives to the physical representation of the cross product space.

Another issue for using this model is how to represent marginals. In a recent paper [GB+96] the authors suggest to use the reserved keyword value "ALL" to achieve this purpose. This is illustrated in Figure 10 in the column "age-group". Thus, the row that contains "ALL" has a summary value over all age-groups in the columns "population" and "avg. income". However, one would have to know which of the columns represent the dimension and which the summary attribute, since "ALL" cannot be applied to summary attribute. In addition, this solution does not deal with classification structures which exist in other relations if normalization is applied. With this construct, the authors have defined a "data cube" operator, which we will discuss in the next section.

A relational representation that partially overcomes the above problems is used by an OLAP company [MicroStrategy]. An example of this representation is shown in Figure 11 for a hospital and patient procedures database. They call it the "star model", since it has one relation that represents the multidimensional space in the center and the other relations that represent dimensions around it .

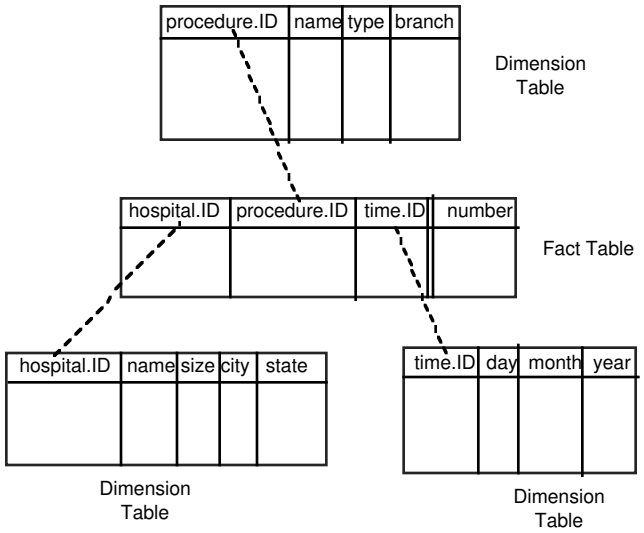

Figure 11: an example of a STAR schema

As can be seen from Figure 11, this is an attempt to introduce the semantics of a statistical object in the context of the relational model. By labeling relations as the "fact

table" and the "dimension tables", they make a distinction between the summary data and the data associated with the dimensions. By placing the "fact table" visually at the center of the "star", they convey the multidimensional nature of this database. Furthermore, each dimension table can hold the category attributes of the classification structure for that dimension. For example, the "hospital" dimension table has attributes "city" and "state" that make up the classification structure.

This representation is more helpful in capturing the semantics of a statistical object, but many of the difficulties described previously still exist. Specifically, (i) the "fact table" still requires the storage of the entire cross product; (ii) it is not possible to know whether an attribute in the dimension tables is a "category attribute" (such as "type" in the "procedure" table) or a regular descriptive attribute (such as "name" in the "procedure" table); (iii) it is not possible to structurally assess the hierarchical relationship of the attributes that make up the classification structure. For example, there is no structural information on whether "procedures" group into "types", and "type" into "branches", or whether "procedures" group into "types", and also "procedures" group into "branches" independently of "types".

4.4 The data cube model

An example of the data cube model was given in Figure 2. This graphical representation is used in OLAP papers (e.g. [GB+96]) and multidimensional OLAP products (e.g. [ArborSoft]) since it is very useful in presenting the concept of the multidimensional space. Obviously, this representation is intended to be used only to convey concepts. It cannot be used for real databases for display of the schema or for display of the data. For real databases, one needs to resort to other representations such as the graphical or tabular representation described above.

While the multidimensional space is clearly represented in this data cube model, the structure of the dimensions is not well presented. Only through labels such as "hospital, city, state" associated with the dimensions one can guess that the dimensions has structure and data of it own.

In spite of the above deficiencies, the data cube concept is useful in visualizing certain operations, such as "slice" and "dice". A "slice", for example, refers to a "cut" though one of the dimensions for a fixed value (such as "race=black"). It is sometimes used to describe the summarization over all values of one of the dimensions (such as summarize over all "races"). We discuss the terminology used for operators over data cubes further in section 5.

4.5 Terminology correspondence

The table in Figure 12 shows the correspondence of terms used in the SDB and OLAP areas. As can be seen the SDB area was influenced by the fact that dimensions usually have a discrete set of values, called "categories" and by the use of the term "attribute" and "object" in the database literature. The OLAP area uses the mathematical terms of "dimensions" but was also influenced from relational terms, such as "table" and "column", as well as the visual term of a "data cube" for the multidimensional space.

Statistical DB	OLAP
Category Attribute	Dimension
Category Hierarchy	Dimension Hierarchy (Table)
Category Value	Dimension Value
Summary Attribute	Measures (fact column)
Statistical Object	Data Cube (fact table)
Cross Product	Multidimensionality
Summary Table	Table / Data Cube

Figure 12: correspondence of terms in SDBs and OLAP

5 Conceptual Modeling -- operations

In this section we discuss operations that were found useful for SDBs and OLAP databases. We first introduce the concept of "automatic aggregation", then examine the statistical version of relational algebra operators. This is followed by a description of OLAP operators and their correspondence to SDB operators. The issue of completeness of SDB operations is discussed next. We conclude with a short discussion of more complex statistical operators, such as sampling. We also discuss briefly the need to automatically support mappings between related classifications.

5.1 Automatic aggregation

Given that the semantics of the data structures of a statistical object are well defined, it is possible to express a minimum number of conditions in a query and infer the rest. This permit the use of very concise query languages. We refer to the capability as "automatic aggregation" [S82], because of the ability to automatically infer the conditions for applying the aggregation operation. We illustrate this with an example, shown in Figure 13. It shows a graph model of a statistical object "average income of professionals by sex by year by profession".

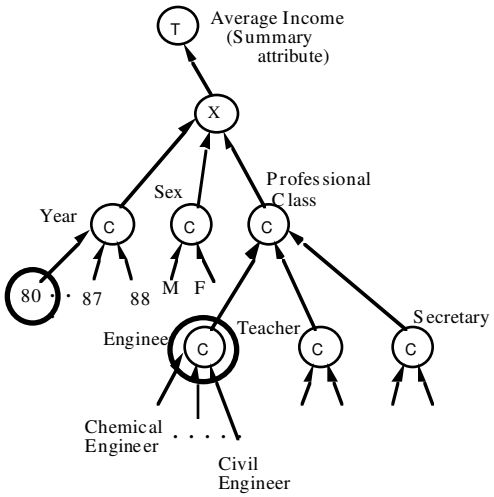

Figure 13: illustration of the "automatic aggregation" concept

There are two circles shown around elements of Figure 13, one around the value 80 for the node "year", and one around the node "engineer". This selection is equivalent to expressing the query "find the average income of engineers in 1980". The semantics of the statistical object imply that: (i) the summarization will be done over all "engineers" (i.e. "chemical engineer", "civil engineer", etc.), because the node "engineer" is a non-leaf node; (ii) the summarization will be done over all sexes, because the "sex" node is a dimension node (attached to the X-node). In fact leaving out any selection on all the nodes in the tree under a dimension implies summarization over all elements of that dimension; (iii) the requested summary of "average income" was inferred from the "summary attribute"; (iv) the requested operation "average" was inferred from the association of the "average" function with the statistical object (note: to perform "average" it is assumed that the "sum" and "count" of each cell are maintained).

As can be seen, it is very useful to take advantage of the known semantics of a statistical object for expressing queries, especially for graphical query user interfaces. If this database used a relational database paradigm devoid of these semantics, the SQL query necessary to express an equivalent query will require a complex aggregation expression.

5.2 Statistical operators

In the SDB literature there were many approaches to define operators which reflect the structural model of the statistical object. This work was dependent on the model used, such as the graphical or tabular models discussed in the previous section. We describe a couple of examples below.

One approach was to use the relational framework, but to enrich the structural model with semantics relating to the statistical object properties, specifically: multidimensionality, and the classification hierarchy. In [MRS92] operators that correspond to the relational algebra operators "select", "project", and "union", as well as "aggregate" were defined. They have been labeled "S-select", S-project" etc. (S stands for "statistical operator"). They have the following semantics in the context of a statistical object:

S-select: selects a subset of category values of a category attribute. This does not reduces the cardinality of the multidimensional space, except if a single value is selected.

S-project: summarizes over all values of a dimension. This reduces the cardinality of the multidimensional space by one.

S-aggregation: summarizes over the values of the classification hierarchy. One can specify a summarization over one or more levels. This also does not reduce the cardinality of the multidimensional space.

S-union: is used to combine multiple statistical objects which have overlapping (or partially overlapping) category values.

We will show that similar operators have been identified in the OLAP area.

Another example which illustrates the dependency of the operators suggested on the conceptual structure is the operators developed in [OOM85] . In this case, the model used was the 2-D representation of statistical objects, being the most commonly used by statisticians. The operators permit the summarization over the multidimensional space and classification hierarchy, but in addition there are operators, such as "attribute split" and "attribute merge" which permit users to specify how the category attributes are organized on rows and columns, or in multiple tables.

5.3 OLAP operators and their correspondence to SDB operators

The operators defined in the OLAP areas use the graphical image of the data cube. Thus, a "slice" is a cut through the multidimensional space, a "dice" is the selection of some ranges over the dimensions, "roll up" (also called "consolidation") is the summarization over one level of the classification hierarchy, and "drill down" is the opposite operation. Since these terms are descriptive rather than formal, they can have multiple interpretations. For example, "slice" is sometimes used to mean the summarization over all the values of a dimensions, rather then selecting a single value of that dimension. In reality

OLAP systems often use existing languages such as SQL to express the summarization conditions, and are mostly concerned with the efficient processing of the queries.

As a side point, we note that "Drill down" is indeed useful in data analysis, where one may start at a high level of summarization and wishes to see more details by going into more specific breakdowns of the categories of a dimension. For example, one may start with "disease types", then see an interesting phenomenon in the disease type "cancer", and "drill down" to the breakdown of the various cancer diseases. This was also recognized as a useful operation in the SDB area and is referred to as "disaggregation" [S82]. Statisticians have also used the concept of disaggregation by proxy" to estimate lower level breakdown of a classification. For example, if the "population" is only known at the state level, but the area of each county is known, one can use the area of the counties as a "proxy" to estimate the population at the county level.

Figure 14 summarizes the (informal) correspondence in concepts between SDB and OLAP operators. Our purpose is only to demonstrate that the SDB and OLAP areas are concerned with similar operators.

Statistical DB	OLAP
S-projection	Slice
S-selection	Dice
S-aggregation	Roll up (consolidation)
S-disaggregation	Drill down
S-union	- - -

Figure 14: correspondence of SDB and OLAP operators

5.4 SQL extensions for OLAP

In a recent paper [GB+96], the authors have demonstrated how awkward and verbose are SQL statements for expressing summarization over the multidimensional space. In some cases, one needs to express a "group by" statement for each category value of a dimension, and then perform a "union" over them. Their proposed solution is to introduce as special "cube" operator, that will conceptually produce all possible summarizations over the dimensions of the multidimensional space. To represent this in a relational table, it was necessary to introduce the keyword value "ALL" as discussed in Section 4.3. This is demonstrated in Figure 15. Note that "ALL" in all the columns representing category attributes produces the "grand total" for the entire multidimensional space.

Using this concept, the authors proposed various extensions to SQL, such as "GROUP BY CUBE (state, year, sex)" to express more concisely OLAP summarization operations. The advantage of this approach is that it makes it possible to express SDB operations within the relational database model. However, as mentioned in Section 4.3 this relational (tabular) structure is devoid of the semantics of statistical objects, and thus the SQL expressions are still quite complex.

Using this concept, the authors proposed various extensions to SQL, such as "GROUP BY CUBE (state, year, sex)" to express more concisely OLAP summarization operations. The advantage of this approach is that it makes it possible to express SDB operations within the relational database model. However, as mentioned in Section 4.3 this relational (tabular) structure is devoid of the semantics of statistical objects, and thus the SQL expressions are still quite complex.

State	county	year	race	sex	age-group	pop	avg. income
Alabama	Autauga	1990	white	male	1-10	11,763	0
Alabama	Autauga	1990	white	male	11-20	9,763	3,342
Alabama	Autauga	1990	white	male	21-30	15,763	34,342
…	…	…	…	…	31-40	14,566	37,444
…	…	ALL	ALL	ALL	ALL	11,372	38,776
ALL	ALL	ALL	ALL	ALL	ALL	89,483	36,775

Figure 15: The effect of the "cube" operator on the relation of Fig. 11

5.5 Completeness of SDB and OLAP operators

A natural question to ask is: "how do we know that the operators provided by a system are sufficiently powerful, or complete in some sense". This issue is largely ignored in both the SDB and OLAP areas.

One approach explored in the SDB area [MRS92] was to find an analogy to relational completeness. The technique used is a "proof by homomorphism" shown schematically in Figure 16.

The main idea is as follows. Start with the "micro-data" and the summarized "micro-data" derived from it by some "summarization function", as represented in the top horizontal arrow. consider a "result micro-data" set that was generated by some relational algebra operations, represented by the left vertical arrow. Now, apply the same summarization function to the "result micro-data" to generate the "result macro-data", represented by the bottom arrow. The question is whether there are "statistical algebra" operations that can generate the same "result

macro-data" from the "macro-data" set originally produced, represented by the right arrow. If one can prove that for all relational algebra operations, there is a statistical algebra operation (or a combination of operations) that produce the same result, we say that the statistical algebra operations are "complete" with respect to the relational algebra operations.

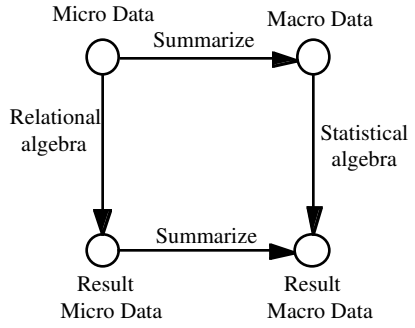

Figure 16: A technique for showing "completeness" of statistical algebra

Another paper that explores the appropriate operators to a multidimensional dataset is reported in a recent paper [AGS97] .

5.6 Higher level statistical operations

Since the main purpose of SDBs and OLAP databases is statistical analysis, where do statistical packages (such as SPSS or SAS) fit in the picture. Database systems provide only simple aggregation functions (usually only count, sum, average, maximum, minimum). For more complex operations, such as standard deviation, percentiles, trimmed means, sampling, etc. it is necessary to use an external "statistical package".

The issue of how to merge the two technologies was never answered. There were some attempts to interface the two in a seamless fashion. For example, SAS has a product that interfaces to an ORACLE relational database system, where a relational table is used to store a 2-D statistical object. SAS manages the "metadata" about the dimensions, and ORACLE manages the storage of the relations and provides concurrency control.

There is also a debate as to whether more statistical functions should be added to database systems, or what data management capabilities should be added to statistical packages. There are no clear answers. The only compelling reason is efficiency. Otherwise, one can develop a higher level interface that can perform more complex functions (such as "find the trimmed means over a sample of the data") by calling the data management system, and passing the results to the statistical package for further processing.

One example where efficiency is at issue is sampling. It is very inefficient to extract large collections of data from the database system, only to sample the collection outside the system by a statistical package. It is clearly more efficient to add the sampling function to the database management system that support statistical (and OLAP) data. There was quite a bit of work in this area, that include random sampling from relational databases, from hash files, from B+trees, from spatial databases etc. The authors of many of these papers provided a good summary of this work in [OR95] .

It seems that except for special operations where efficiency can improve greatly, data management systems and statistical packages will continue their independent existence. Therefore, clean interfaces between them is the key to future integration of these technologies.

5.7 Support for classification matching

We conclude this section with an extremely important issue in SDB applications, and most likely in OLAP applications as well. It concerns the merging of statistical results from different sources, and the mechanisms needed to support them. A typical example is the summarization of production of goods as reported by farms, factories and businesses. Such data is often collected locally at the level of a city or a county. When such data is aggregated to the next level of a classification hierarchy, discrepancies between the sources of the data exist. Most often this occurs because the categories selected by one source are not compatible with another source.

Consider the categories shown in Figure 17. At the top, there are two age group classifications which may have been used by two regions that need to be summarized. One can use interpolation methods to come up with a combined age-group classification. This is usually done by analysts in a way that it not documented. Thus, the dataset may be altered by calculations that are done outside the system.

The bottom example of Figure 17 shows temporal variations of a classification. In this case, a new industry (internet) was added in 1991 to the industry classification. Variations over time are common, such as change in items that a store stocks over time, changes in the definition of what is included when GNP is calculated, changes in the number of stores a store chain has, changes in the boundaries of regional entities, etc.

Database 1	Database 2
Age Group	Age Group
0-5	0-1
6-10	2-10
11-15	11-20
16-20	21-30
non-overlapping granularities:	
industries in 1990	industries in 1991
agriculture	agriculture
automobiles	automobiles
…	…
-- --	Internet
Time varying categories	

Figure 17: Examples of non-matching classifications

No system today supports an orderly management of such variations in classifications. We claim that any SDB (or OLAP) system that permits summarization from multiple sources needs to have the facilities to support various extrapolation functions, or functions provided by the database designer. The "metadata" of the methods used to perform integrated summaries need to be maintained as part of the database. Furthermore, such facilities are needed for time varying classifications even if the data comes from a single source.

6 Physical Organization

In this section we describe the main issues and techniques explored for achieving data storage and access efficiency.

6.1 Transposed files

Early work in the SDB area was already concerned with access efficiency. For example, a system developed by Statistics Canada in the late 1970's [THC79] tackled the problem of efficient access of SDBs stored in a relational tabular form, such as the example shown in Figure 10. Based on the observation that statistical queries usually involve a few category attributes (dimensions), and usually only one summary attribute, their system took the approach of "transposing" the table. "Transposition" is used in this context to say that each column of the table is stored separately, as shown in Figure 18. It is sometimes referred to as a "vertical partitioning" of a table.

State	county	year	race	sex	age-group	population	avg.income
Alabama	Autauga	1990	white	male	1-10	11,763	0
Alabama	Autauga	1990	white	male	11-20	9,763	23,342
Alabama	Autauga	1990	white	male	21-30	15,763	34,342
...	...	...	...	...	31-40	14,566	37,444
...	...	...	...	...	...	11,372	38,776
...	...	...	...	male	ALL	89,483	36,775
...	...	...	...	Female	1-10	8,457	0

Figure 18: a transposed table

Clearly, the main advantage of this approach is that for summary operations and statistical query (such as cross tabulations) only the relevant columns need to be retrieved. This improved greatly the access efficiency of statistical queries.

The above techniques did not reduce the size of storage for the cross product space. Another paper [WL+85] also used transposed tables, but also explored the benefit of encoding the category values in each column. Since many of the category attributes have very few category values (such as sex, race, state, etc.) one could take advantage of that to encode the values in a small number of bits. An example of such encoding is shown in Figure 19.

In addition, it was observed that the values in the least rapidly varying columns have the same value repeated for many rows. They proposed to use run length encoding to compress the repeating sub-sequences. Simulation results showed that the reduction in space was quite dramatic, and with it the access time improved. Finally, this paper considered the idea of transposing the table to the extreme. The idea is transpose the entire table into files that contain single bit columns. For example, the race column in Figure 19, will be stored as 3 bit-transposed files. It was shown in simulation results that this extreme transposition further increases the compression and performance.

race	sex	age-group
white	male	1-10
white	male	11-20
white	male	21-30
…	…	31-40
…	…	…
…	male	ALL
…	Female	1-10

encode

race	sex	age-group
000	0	0000
000	0	0001
000	0	0010
…	…	0011
…	…	…
…	0	1010
…	1	0000

Figure 19: encoding of columns category values

The penalty paid for using transposed files comes about when a few entire rows need to be retrieved. in such cases, there is the overhead of reading pieces of each row from multiple files. This is a trade off that needs to be taken into account when considering the query mix of an application. For applications dominated by summary operations, the benefits of transposed files have been demonstrated.

6.2 Techniques for compressing the multidimensional space

The most obvious idea for compressing a multidimensional space is "array linearization". This term refers to a fairly simple well-known calculation of the position of a cell in a multidimensional array, based on the number of possible values in each dimension (in the 2-D case it is the numbers of rows and columns). Thus, instead of storing all the columns of the tabular representation, we need to store the distinct values of each dimensions only once. An example is shown in Figure 20. This is the main idea behind Multi-dimensional OLAP (MOLAP) products (e.g. Essbase [ArborSoft]).

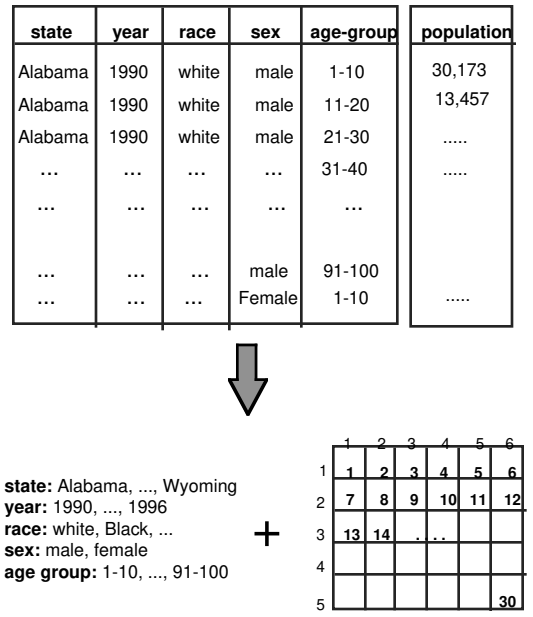

Figure 20: Array linearization for table structure

Using array linearization works well when the multidimensional space is dense, i.e. that each cell has a measured value associated with it. However, in many cases many of the cells have nulls or zeros. For example, in a production database of products by month by county, all the counties of the states that do not produce oil will have nulls or zeros (depending on the choice of encoding) for all the reported months. In reality very few counties, if any, produce all the products monitored by economists.

In such situations it is useful to consider some compression method. One such method was explored in [EOA81] . This method is based on the observation that nulls (or zeros) tend to cluster in the sequence of values of the summary attribute column. Thus, run length encoding was used. This is illustrated in Figure 21. As can be seen, all the nulls are compressed out, and only the non-null values are stored. In addition, a run length encoded sequence of the number of elements of each sub-sequence is generated.; i.e. no. of values, no. of nulls, etc.

Now, in order for the run-length sequence to be searched efficiently, it is accumulated, so that a monotonically increasing sequence is formed. This permits the use of a b-tree for efficient search. This sequence is called a "header" and the compression technique a "header compression". It was shown that this technique can be used for the inverse mapping as well, i.e. given the position of a given value, one can use the b-tree to find its array position.

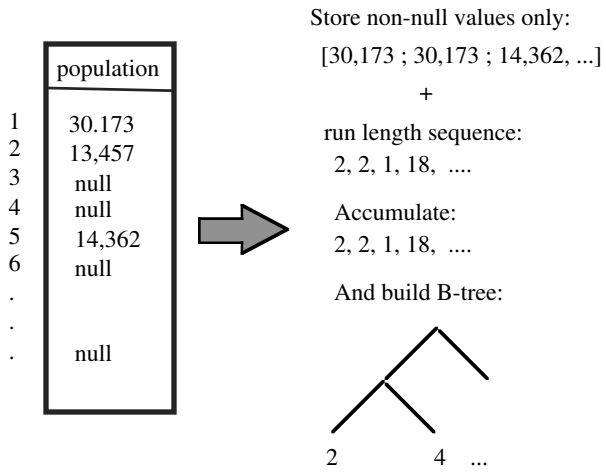

Figure 21: header compression

Other compression methods can be used as well, such as the well known LZW method. The most effective method depends on the distribution of nulls.

6.3 View materialization methods

For very large datasets (in the gigabyte range) it is expensive (and therefore slow) to calculate summarizations, since too much data has to be retrieved from secondary storage. Since SDBs and OLAP databases are often stable, i.e. have very few updates if any, pre-generating and storing summarization views is a worthwhile possibility to consider. This is a problem of space-time tradeoff. Given a limited extra space, which summarizations should be pre-

calculated for the maximum access benefit. A recent paper [HUR96] considered this question given that no information is known about access patterns. It was assumed that all summarization queries are equally likely. The paper considered which of the 2^n -1 summarizations should be materialized. This problem is shown in Figure 22 for a dataset with 3 dimensions (taken from [HUR96]). The lines between the items show derivation possibilities. For example, the "location" summarization can be derived from either the "location, day" dataset of from the "product, location" dataset.

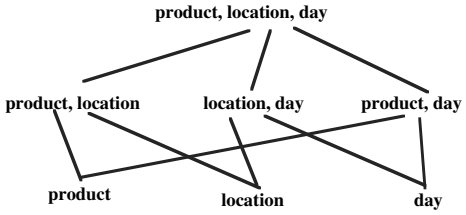

Figure 22: the lattice of materialization possibilities

This problem is non-symmetric, because the number of category values (cardinality) of each dimension is, in general, different, and therefore each materialized summarization size is different. An elegant analysis has yielded an optimal solution for this problem using a greedy algorithm.

6.4 Pre-partitioning the data cube

Another idea for improving access efficiency is takes advantage of "range" type queries. Realizing that many of the queries involve "slicing" and "dicing", it makes sense to pre-partition the data cube into subcubes as shown in Figure 23. The advantage of this approach is that only the subcubes that contain data relevant to the query or summarization request need to be read from secondary storage. For a given query that spans multiple subcubes, each subcube may contain part of the relevant data. Consequently, the access software needs to manage multiple subcubes and assemble the required data from them.

If there is no information about the access patterns to the data cube, a symmetric partitioning is appropriate; that is, all the subcubes are of equal sub-dimensions. The only parameter to be determined is the size of a subcube. This can be determined by estimating the average size of a query. An example of taking this approach of symmetric partitioning was reported in [SS94] in the context of dealing with multidimensional earth science databases.

When knowledge exists on the access patterns to the data cube (typical queries), a non-symmetric partitioning approach can further improve performance. The analysis of which subcubes to form can be quite complex, and an index of the dimensions of the various subcubes has to be created and maintained. In [CD+95] such an approach was used for climate modeling spatio-temporal data. In this case the data set was only 4 dimensional, yet the analysis of which subcubes to form was shown to be NP-complete. Heuristic algorithms were used to achieve close to optimal solutions, and the improvement to the access time was significant.

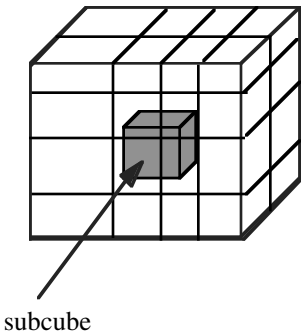

Figure 23: Partitioning a data cube into subcubes

6.5 Data cube update techniques

It is usually assumed that no updates are made to the data cube. Indeed the concept of a "data warehouse" assumes that all the data relevant to the analysis is moved to the warehouse and do not change frequently. In reality, it is very common to append to the data cube over time. The frequency of appends depends on the application, but it is not uncommon to have daily appends.

This problem was addressed in a recent paper [RZ86]. Rather than restructuring the data cube for each append, the approach taken is to perform incremental appends. The proposed structure is called an "extendible array" as shown in Figure 24. A tree-based index structure is used to keep track of the multidimensional increments. The paper also describes access methods that use this structure to support "range queries" efficiently.

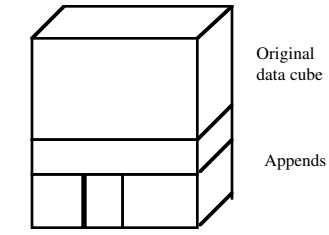

Figure 24: Incremental updates to a data cube

We note that this approach uses non-symmetric incremental sub-cubes. Thus, this indexing method could be used to manage a non-symmetric partitioning of a data cube. Of course, this assumes that the process of determining which

subcubes to generate is known. This is an active area of current research. Another example of dealing with the updates to a data cube is a recent paper [RKR97] which uses a packed R-tree structure to support bulk updates.

6.6 The debate of ROLAP vs. MOLAP

ROLAP stands for "Relational OLAP" and MOLAP for "Multidimensional OLAP". There are several vendors on the market that advocate one approach or the other. Two such examples can be found in Web pages. See [MicroStrategy] for ROLAP, and [ArborSoft] for MOLAP.

We discuss this issue in this section because much of the debate is based on physical organization of the data cube, and the efficiency of access. Below are the main arguments made by each.

MOLAP proponents claim that:

(i) Relational tables are unnatural for multidimensional data.
(ii) Multidimensional arrays provide efficiency in storage and operations.
(iii) There is a mismatch between multidimensional operations and SQL.
(iv) For ROLAP to achieve efficiency, it has to perform outside current relational systems, which is the same as what MOLAP does.

ROLAP proponents claim that:

(i) ROLAP integrates naturally with existing technology and standards.
(ii) MOLAP does not support ad hoc queries effectively, because it is optimized for multidimensional operations.
(iii) Since data has to be downloaded into MOLAP systems, update is difficult.
(iv) Efficiency of ROLAP can be achieved by using techniques such as encoding and compression.
(v) ROLAP can readily take advantage of parallel relational technology.

The claim that MOLAP performs better that ROLAP is intuitively believable. In a recent paper this was also substantiated by tests [ZDN97]. However, the debate will continue as new compression and encoding methods are applied to ROLAP databases. Both sides make good points, many of which were discussed in this paper. It is unclear which approach will prevail, but efficiency and integration with existing technology are the key issues.

7 Privacy

It is usually permissible to make public summarized data, but not the individual data used for generating the summary. As discussed in Section 2, Census data is an obvious example, but other domains, such as criminals records, or financial data about individual companies, need to be protected by agreement or by law. The problem is whether privacy protection can be guaranteed. This problem is referred to as the "statistical inference" problem.

There was a lot of work done in this area in the 1980; many references can be found in [S82]. However, it is worth mentioning here an important negative result described in [DS80]. In this paper it was shown that it is always possible to compromise a database by using a combination of queries. In fact, the authors have developed a procedure (called a "tracker") for finding the collection of queries to infer a desired fact.

One possible solution seems to restrict the number of individuals included in the response to a statistical summary query, called a "query set". For example, we may choose to restrict the response to "average salary", only if the average is for 5 people or more. We say that the query set is restricted to be greater than 5. Restricting query set size if not enough, since intersecting query sets can compromised information. Thus, it is necessary to monitor the size of query set intersections as well.

It was also shown that even if we restrict the size of a query set some compromise is still possible. To illustrate this point, suppose that we know that there is only a single employee whose age is 65, an no employee is older than 65. To find the salary of this employee, which is restricted information, one need only to issue a query for the "average salary and count of all employees", and then find the "average salary of all employees under 65".

Consequently, many other approaches were proposed. We mention the main ideas next. (i) Limiting the query set intersection. This requires keeping track of all query sets, and making sure that a new query set do not intersect with previous one to produce a query set size below the permitted size. This can be used for small databases, but eventually it reaches the point that no new queries can be answered. (ii) Random sample from a query set. This is useful for very large datasets, when the typical query set is large. (iii) pre-partition the dataset into cells, and give responses that involve whole cell only. This approach is used fairly effectively with the Census data in the US, but requires "cell suppression" (i.e. some cells that contain too few individuals are cannot be reported). (iv) Input data perturbation -- stored statistically correct, but perturbed data for general consumption. (v) Output data perturbation –

perturb results given to users within some statistical limits.

As can be seen there are no easy solutions to the privacy problem. All the solutions proposed have some disadvantages. However, given the importance of privacy, an imperfect solution is better than none. It is interesting to note that although privacy needs to be supported in many OLAP applications, they are currently ignored in OLAP systems and literature. The interest in privacy continues in the SDB community, such as the Conference on Data Protection [DS80].

8. Conclusions

We have presented the properties that characterize Statistical Objects. We have examined examples form the Statistical Data Base (SDB) area and the OLAP areas, and pointed out that they are very similar in concepts, but differ somewhat in the emphasis of the problems they address. We discussed the conceptual modeling structures and operators used in SDBs and OLAP and showed the correspondence of term between them. We also discussed physical organization and access method techniques used in these area, and concluded with a discussion on privacy in summary databases.

We observed that the overlap between these area is great, and that the work done in one area can greatly benefit the other. The work in the SDB area, much of it reported in the conference series on Statistical and Scientific Data Base Management (SSDBM), has emphasized conceptual modeling, while the OLAP area has emphasized physical organization and efficient access. However, both areas have investigated both aspects, and should refer to each other's work.

We believe that a Statistical Object data type should be considered seriously for support by extensible systems designed to incorporate new data types, such as Object-Relational or Object-Oriented systems. Some of these systems already support temporal, spatial, and other data types to make them more appealing to application that have such needs. The Statistical Object data type is useful for applications that need to model, query, and perform efficient statistical summarization on a multidimensional dataset with classification structures. Many such applications exist as was discussed in Section 2.

We note that a Statistical Object data type is fairly complex. It needs to support not only the semantics, operations, and physical structures of the multidimensional space, but also of the classification structures. Supporting the classification structures implies the storage and management of all the "metadata" of the category values, and their hierarchical associations. The methods to be supported should provide automatic aggregations (discussed in Section 5.1), advanced statistical operators (such as "sampling"), and mechanisms to deal with time varying and incompatible classifications. Perhaps this is too much to expect to be incorporated into a general purpose database system, and specialized systems (such as OLAP systems) will continue to assume this role.

Acknowledgement

This work is supported by the of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.

References

[AGS97] R. Agrawal, A. Gupta, S. Sarawagi , Modeling Multidimensional Databases, ICDE 1997.

[ArborSoft] Relational OLAP: Expectations & Reality, An Arbor Software White Paper, http://www.arborsoft.com/papers/rolapTOC.html (Also see: The Role of the Multidimensional Database in a Data Warehousing Solution)

[CS81] Chan, P., Shoshani, A., Subject: A Directory driven System for Organizing and Accessing Large Statistical Databases, VLDB 1981, pp. 553-563.

[CD+95] Ling Tony Chen, R. Drach, M. Keating, S. Louis, Doron Rotem, Arie Shoshani, Efficient organization and access of multi-dimensional datasets on tertiary storage systems, Information Systems 20(2), pp. 155-183 (1995).

[DP96] Conference on Data Protection, organized by Eurostat, Bled, Slovenia, October, 1996.

[DS80] Dorothy E. Denning, Jan Schlorer, A Fast Procedure for Finding a Tracker in a Statistical Database, TODS 5(1), pp. 88-102 (1980).

[EOA81] Susan J. Eggers, Frank Olken, Arie Shoshani, A Compression Technique for Large Statistical Data-Bases, VLDB 1981, pp. 424-434.

[GB+96] Jim Gray, Adam Bosworth, Andrew Layman, Hamid Pirahesh, Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total, ICDE 1996.

[HUR96] Venky Harinarayan, Jeffrey D. Ullman, Anand Rajaraman: Implementing Data Cubes Efficiently, SIGMOD 1996.

[LRT96] W. Lehner, T. Ruf, M. Teschke, CROSS-DB: A Feature-Extended Multidimensional Data Model for Statistical and Scientific Databases, CKIM conference, 1996.

[LS97] Hans J. Lenz, Arie Shoshani, Summarizability in OLAP and Statistical Data Bases, 9th International Conference on Statistical and Scientific Database Management (SSDBM) 1997.

[MRS92] Leonardo Meo-Evoli, Fabrizio L. Ricci, Arie Shoshani, On the Semantic Completeness of Macro-Data Operators for Statistical Aggregation, 6th International Conference on Statistical and Scientific Database Management (SSDBM) 1992, pp. 239-258.

[MicroStrategy] The Case For Relational OLAP, A White Paper Prepared by MicroStrategy, Incorporated, "http://www.strategy.com/dwf/wp_b_a1.htm"

[OR95] Olken, F. and Rotem, D., ``Random Sampling from Databases - A Survey,'' (invited paper), Statistics & Computing, March 1995, vol.5, no.1, pages 25-42. (many more reference in http://www.lbl.gov/~olken/sampling.html)

[OOM85] Gultekin Ozsoyoglu, Z. Meral Ozsoyoglu, Francisco Malta, A Language and a Physical Organization Technique for Summary Tables. SIGMOD 1985: pp. 3-16.

[RS90] Rafanelli, M., and Shoshani, A., STORM: A statistical Object Representation Model, 5th International Conference on Statistical and Scientific Database Management (SSDBM) 1990, pp. 14-29.

[RZ86] Doron Rotem, J. Leon Zhao, Extendible Arrays for Statistical Databases and OLAP Applications, 8th International Conference on Statistical and Scientific Database Management (SSDBM) 1996, pp. 108-117.

[RKR97] Nick Roussopoulos, Yannis Kotidis, Mema Roussopoulos, Cubetree: Organization of and Bulk Updates on the Data Cube, SIGMOD 1997.

[SS94] Sunita Sarawagi, Michael Stonebraker, Efficient Organization of Large Multidimensional Arrays, ICDE 1994, pp. 328-336.

[S82] Arie Shoshani, Statistical Databases: Characteristics, Problems, and some Solutions, VLDB 1982, pp. 208-222.

[THC79] M. J. Turner, R. Hammond, P. Cotton, A DBMS for Large Statistical Databases, VLDB 1979, pp. 319-327.

[WL+85] Harry K. T. Wong, Hsiu-Fen Liu, Frank Olken, Doron Rotem, Linda Wong, Bit Transposed Files, VLDB 1985, pp. 448-457. (An expanded version appeared in Algorithmica (1986) 1, pp. 289-309.)

[ZDN97] Yihong Zhao, Prasad Deshpande, and Jeffrey F. Naughton, An Array-Based Algorithm for Simultaneous Multidimensional Aggregates, SIGMOD 1997.