This paper appeared in the 6th International Conference on Management of Data, Bangalore, India, December 19–21, 1994

A Tuple Model for Summary Data Management*

Wee K. Ng Chinya V. Ravishankar

Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, MI 48109-2122
E-mail: {wkn,ravi}@eecs.umich.edu

Abstract

The use of precomputed results, known as *summary data*, is a common approach to efficient statistical query processing. The technique relies upon two factors: (1) the selection of a representative set of summary data so that most queries may be satisfied, and (2) the efficient storage and management of the summary data set. The latter is the focal issue of this paper. We propose a simple, intuitive and efficient method for the organization and managment of summary data, which also permits the easy maintenance of a dynamic set of summary data. We also examine the issue of data derivability pertaining to the use of the method. The results that are derived can be used to form the basis of a query processor.

Key words: Data modeling, summary statistics, physical organization, statistical database, query processing.

1 Introduction

Efficient statistical query processing is difficult to achieve in practice because the queries are by nature aggregational, and require accessing a large portion of the database. This incurs a lot of I/O, and consequently the responses to the queries are slow. The use of precomputed results is a common approach in the processing of statistical queries [Ghos87, IkKo81, Sato81]. The principle is to use what was already computed to help answer a new query. Much work has been done on the operational aspects of this method; examples include extending the relational algebra to handle *summary data* [Ghos86], defining a new language and algebra for summary data [FRRS86, OOM85], and investigating properties of summary data [Malv88, MaMo90]. While the summary data approach may not be universally applicable, there are cases where it is clearly useful, and so the approach has been discussed in the literature. However, little attention has been given to the efficient storage and management of summary data [ChMc89, OOM85]. In this paper, we address the following issue: *How does one store and manage the selected set of summary data so that the access, insertion, deletion and modification of summary data can be efficiently performed?*

A *summary datum* consists of a *category instance* and a computed statistic [ChMc89]. A category instance is a set of attributes. To each category instance there may correspond

*This work was supported in part by the Consortium for International Earth Science Information Networking.

No.	C_1	C_2	C_3	C_4	S_1	S_2
1	male	operator	white	English	4	0
2	male	operator	white	French	7	2
3	male	operator	hispanic	English	6	5
4	male	operator	hispanic	Spanish	5	2
5	male	operator	hispanic	French	1	3
6	male	teacher	white	English	17	9
7	male	teacher	white	Spanish	16	6
8	male	teacher	hispanic	English	19	0
9	male	teacher	black	English	10	1
10	male	teacher	black	Spanish	17	2
11	male	teacher	black	French	9	9
12	male	technician	white	English	15	7
13	male	technician	white	Spanish	1	1
14	male	technician	hispanic	English	1	5
15	male	technician	black	English	15	3
16	male	technician	black	Spanish	14	7
17	male	technician	black	French	1	0
18	male	secretary	white	Spanish	11	3

No.	C_1	C_2	C_3	C_4	S_1	S_2
19	male	secretary	black	English	11	9
20	male	secretary	black	Spanish	4	9
21	female	operator	white	English	13	2
22	female	operator	hispanic	English	3	4
23	female	operator	hispanic	French	4	1
24	female	operator	black	English	15	9
25	female	operator	black	French	16	1
26	female	teacher	white	Spanish	19	6
27	female	teacher	white	French	16	0
28	female	teacher	black	English	10	7
29	female	technician	white	English	13	5
30	female	technician	white	Spanish	15	8
31	female	technician	hispanic	French	2	0
32	female	technician	black	English	3	5
33	female	technician	black	Spanish	5	8
34	female	secretary	white	Spanish	8	1
35	female	secretary	hispanic	English	19	0
36	female	secretary	hispanic	French	14	3

Table 1: A statistical relation R with four category domains: sex (C_1), occupation (C_2), race (C_3) and languages spoken (C_4), and two data domains: income in thousands (S_1) and number of hours worked (S_2). The category domains are of sizes $2, 4, 3, 3$ respectively. The relation is divided into two tables for convenience only.

many statistical functions and statistics. For simplicity, we assume that a summary datum contains only a single computed statistic. An example is: $\langle\{male, engineer\}, mean\_age\rangle$. The category instance is the set of male engineers and the statistic is mean age. Here, the statistical function is the mean function.

Given a relation, a large number of summary data may be derived depending on users' querying needs. The problem to be addressed is their organization and storage in a way that does not sacrifice operational performance, but allows the efficient access and modification (including insertion and deletion) of summary data. This is non-trivial as summary data are non-uniform, and differ not only in the number but also in the types of category instances and statistics involved. Section 2 surveys the conventional approaches and highlights their weaknesses.

As the relational model and its algebra are used throughout this paper, a brief outline is given here. A *relation* is a table of tuples. Each tuple is uniquely identified by an *access key*, which is usually a component of a tuple. A relation is an *instance* of a *relation scheme* that defines the sets of attributes assumed by the tuples. The *relational algebra* defines a set of operations for evaluating queries addressed to a relational database. The most commonly used ones are the *projection* operator π, and the *selection* operator σ. π selects columns from a relation, while σ selects a subset of tuples of a relation, in each case based on a selection criteria called the *predicate*.

The organization of the paper is as follows: The next section surveys current approaches of summary data management. The two subsequent sections cover our approach in detail. Section 3 describes the structure of a *summary relation*. In order to accomodate the variety of summaries, a comprehensive relation scheme must be defined. The scheme must also decide the kinds of summary attributes to store. Section 5 describes the physical implementation of the conceptual summary relation for efficient access. In Section 6, we address the issue of query answerability with respect to the summary relations. We show that certain queries

$$
\begin{tikzcd}
 & \{C_1, C_2, C_3\} \arrow[dl] \arrow[dr] & \\
\{C_1, C_2\} \arrow[d] & \{C_2, C_3\} & \{C_1, C_3\} \arrow[d] \\
\{C_1\} & \{C_2\} & \{C_3\}
\end{tikzcd}
$$

Figure 1: A hierarchy category with three category domains where a category in one level is set-contained in some categories of the previous level.

can be satisfied directly from the summary relation, while others may be satisfied indirectly. The last section concludes the paper.

2 Current Approaches

Ozsoyoglu *et al.* [OOM85] proposed storing each summary datum as a *summary table* with operators for their manipulation. Each summary table is a two dimensional representation of the summary data, and is separately stored. The number of summary tables derivable from a relation is potentially large. Without an efficient means for the collective organization and management of these tables, it is difficult to (1) distinguish the relationships among the tables, (2) to identify the table for satisfying a query, and (3) to differentiate among tables derived from other relations.

The concept of a category hierachy to organize summary data is about the only approach available for the overall organization of summary data. It has been proposed simultaneously and independently by Chen and McNamee [ChMc89] and Abad-Mota [Abad92]. The hierachy is defined according to the number of category domains contained in the summary data. An example is illustrated in Figure 1. The figure shows three category domains. The hierachy may not necessarily be complete; some summary data with certain combinations of category domains may be absent.

It is difficult to construct a small and efficient index for accessing a category hierarchy randomly. Chen and McNamee build a Logical Summary Data tree index structure (LSD tree) directly on the set of categories. Each LSD tree node is identified by a parent category and some child-categories that are set-contained within the parent. The child-categories point either to other nodes where they are the parent categories or directly to the statistical data. Thus, all the category information of the hierarchy is contained the internal nodes, while the statistical data are found in the leaves.

This approach suffers from two weaknesses: (1) The LSD tree is quite unlike other index structures, such as the B-tree. The entire tree mirrors the category hierarchy which it is trying to index. The height of the tree is $O(n)$, where n is the number of category domains. Therefore, searching the tree to locate a category takes on average $O(n)$ time. (2) The space requirements of the tree are potentially as large as the category hierachy itself because it captures all the category information of the hierarchy. Since category information forms the bulk of a summary datum, the tree is not at all the type of metadata found in a typical index, such as a B-tree. Since searching the tree is on average $O(n)$, accessing the tree generates a lot of disk I/O because most parts of the tree reside on secondary storage.

The hierarchy concept becomes complicated when there are many pre-computed statistics, some of which share a common category. Should a hierarchy contain all the different statistics that pertain to the same category or should a separate hierarchy be constructed for each statistic? In the former case, the concept of summary data is complicated because there are multiple statistics contained within a summary datum. In the latter case, there is excessive duplication because the same category information is replicated in multiple hierarchies.

In this paper, we offer a simple and practical method for the organization and storage of summaries that also provides efficient access and modification. By treating all summary data with a common category as a single tuple (referred to as a *summary tuple*, as opposed to a tuple of an ordinary relation), the collection of summary data may be organized into a relation, called the *summary relation*. With a suitable ordering rule for the tuples, an access mechanism such as the B^+ tree index may be constructed to provide random access to the tuples. Hence, the access and modification of summaries reduces to the simple process of tuple access and modification. By applying the well-established *relational model* of data to the representation of summary data, one achieves simplicity and efficiency in the management of summary data.

3 Summary Relation Model

We group all attributes of a statistical relation into two classes: *category* and *data* [Ghos86]. Category attributes are generally descriptive and are used as access keys to retrieve the data attributes, which are mostly numerical. The category attributes usually have discrete values known in advance, while the data attributes are usually the observed or measured values of some experiment or survey from which summary data are derived. Data attributes are also known as *summary attributes* in the literature. We have chosen to call them data attributes so as to distinguish them from summary statistics derived from the raw data attributes.

Example 3.1 Table 1 shows a fictitious census relation with four category domains and two data domains. The relation is displayed as two tables for convenience only. ∎

A brief formalization of a statistical relation is given. Let $\mathcal{C} = C_1 \times C_2 \times \cdots \times C_n$ and $\mathcal{S} = S_1 \times S_2 \times \cdots \times S_m$ be the set of all category and data *instances* respectively, where the C_i's and S_j's are the category and data attribute domains[1]. They are denoted as $\mathcal{C} = \langle C_1, C_2, \ldots, C_n \rangle$ and $\mathcal{S} = \langle S_1, S_2, \ldots, S_m \rangle$. A statistical relation scheme $\mathcal{R} = C_1 \times \cdots \times C_n \times S_1 \times \cdots \times S_m$ is the *tuple space* containing the set of all possible tuples. It is denoted as $\mathcal{R} = \langle\!\langle \mathcal{C}, \mathcal{S} \rangle\!\rangle$. A statistical relation instance (or simply a relation) R, is a subset of tuples from the tuple space, i.e., $R \subseteq \mathcal{R}$.

As a notational sidenote, we shall use, throughout the paper, the notation $\langle A_1, A_2, \ldots, A_n \rangle$, where the domains A_i's are in upper case, as the canonical representation for their cartesian product $A_1 \times A_2 \times \cdots \times A_n$. It is not to be confused with $\langle a_1, a_2, \ldots, a_n \rangle$, which is an element of the cartesian product.

A distinction is observed between a *category* and a *category instance*. A category $\mathcal{C}_q = \langle C_{k_1}, C_{k_2}, \ldots, C_{k_q} \rangle$ is the *set of all tuples* whose constituent domains $\{C_{k_1}, C_{k_2}, \ldots, C_{k_q}\}$, form a subset of $\{C_1, C_2, \ldots, C_n\}$. Throughout the rest of the paper, we shall denote such a

[1] The phrases *category attribute domain* and *data attribute domain* are henceforth shortened to *category domain* and *data domain* respectively for brevity.

relationship as $\mathcal{C}_q \subseteq \mathcal{C}$ although strictly speaking, $\mathcal{C}_q$ is not a subset of $\mathcal{C}$. On the other hand, a category instance $\langle c_{k_1}, c_{k_2}, \ldots, c_{k_q} \rangle$ is a single tuple such that $c_{k_i} \in C_{k_i}$ for $i = 1, 2, \ldots, q$.

To each statistical relation scheme, our approach defines a corresponding *summary relation scheme*, $\mathcal{R}_s = \langle\langle \hat{\mathcal{C}}, \mathcal{F} \rangle\rangle$, where $\hat{\mathcal{C}}$ and $\mathcal{F}$ are the collections of *augmented* category and *statistical* domains respectively. An augmented category domain $\hat{C}_i$ is an ordinary category domain with the additional attribute value *null*, denoted as "$\star$". That is, $\hat{C}_i = C_i \cup \{\star\}$ for $i = 1, 2, \ldots, n$. Its purpose is simple: As we propose treating groups of summary data as tuples, a comprehensive scheme must include at least all the category domains. Since a summary datum involves only a subset of these domains, null attribute values are used to represent the excluded domains.

The augmented category attributes cover the descriptive part of a summary tuple. The *statistical* part is covered by the *statistical domains*. Each statistic from a statistical domain is the result of applying a *statistical function* on the data attributes in the original relation R. Consider the *sum* function f applied to some data domain $S_j \in \mathcal{S}$. Given a subset of tuples from R, f totals the data attributes of S_j that occur in the subset into a statistic f_j. Other functions such as count, maximum, minimum, etc. may similarly be applied to the same subset to derive different statistics. All statistics derived under the same category instance form a single summary tuple. The collection of these tuples forms the summary relation $R_s \subseteq \mathcal{R}_s$, which stores all the statistics derived from R as tuples.

Example 3.2 A summary relation R_s that is derived from relation R in Table 1 is shown in Table 2. This example shall be used throughout the paper. Notice the $\star$'s in the summary tuples. Tuples represent *concepts*. For instance summary tuple 38 expresses the concept of a *black, male teacher*. Demographers may be interested in the average number of hours worked, the average income, and any correlation between the hours worked and income, etc., of teachers who are black and male. Any summaries of interest related to this concept may then be retrieved by consulting R_s.

Each summary tuple is a combination of five elementary statistics, all sharing the same category. f_0 is the *cardinality* function which counts the number of tuples in R corresponding to each concept. f_1 and f_2 are defined on S_1 and they compute the *sum* and *sum square* respectively. f_3 is the *sum* function defined on S_2 and f_4 is the *product sum* function defined on S_1 and S_2. To see how the statistics are computed, consider summary tuple 34: $\langle male, operator, hispanic, \star, 3, 12, 62, 10, 38 \rangle$. Tuples 3, 4, 5 from the census relation R satisfy the concept expressed in the summary tuple. Since there are three tuples, $f_0 = 3$ in the summary tuple. The S_1 sum is $6 + 5 + 1 = 12$, the S_1 square-sum is $6^2 + 5^2 + 1^2 = 36 + 25 + 1 = 62$. The S_2 sum is computed similarly. The product sum for S_1, S_2 is $6 \times 5 + 5 \times 2 + 1 \times 3 = 30 + 10 + 3 = 43$. ∎

Notice that there are 68 summary tuples in R_s, while there are only 36 tuples in R. The relative sizes of these two relations should not be taken as indicative of the actual relation sizes. Two points are worthy of note: (1) Since each summary tuple is a set of elementary statistics described by a concept, there will still be 68 tuples in R_s even when R is 100 times larger. We have shown R with 36 tuples simply because of space limitations. (2) Although R_s may get larger as more concepts get stored, it is important to note that only a small fraction of it is accessed during query processing because it contains precomputed summaries. This amount accessed is significantly less than that of accessing the entire R relation.

No.	$\hat{C}_1$	$\hat{C}_2$	$\hat{C}_3$	$\hat{C}_4$	f_0	f_1	f_2	f_3	f_4
1	male	$\star$	$\star$	$\star$	20	184	2410	83	853
2	female	$\star$	$\star$	$\star$	16	175	2445	60	667
3	$\star$	operator	$\star$	$\star$	10	74	802	29	250
4	$\star$	teacher	$\star$	$\star$	9	133	2093	40	558
5	$\star$	technician	$\star$	$\star$	11	85	1081	49	494
6	$\star$	secretary	$\star$	$\star$	6	67	879	25	218
7	$\star$	$\star$	white	$\star$	13	155	2201	50	735
8	$\star$	$\star$	hispanic	$\star$	10	74	1010	23	106
9	$\star$	$\star$	black	$\star$	13	130	1644	70	679
10	$\star$	$\star$	$\star$	English	16	174	2416	71	770
11	$\star$	$\star$	$\star$	French	11	115	1579	53	590
12	$\star$	$\star$	$\star$	Spanish	9	70	860	19	160
13	male	operator	$\star$	$\star$	5	23	127	12	57
14	male	teacher	$\star$	$\star$	6	88	1376	27	374
15	male	technician	$\star$	$\star$	6	47	649	23	254
16	male	secretary	$\star$	$\star$	3	26	258	21	168
17	female	operator	$\star$	$\star$	5	51	675	17	193
18	female	teacher	$\star$	$\star$	3	45	717	13	184
19	female	technician	$\star$	$\star$	5	38	432	26	240
20	female	secretary	$\star$	$\star$	3	41	621	4	50
21	male	$\star$	white	$\star$	7	71	957	28	402
22	male	$\star$	hispanic	$\star$	5	32	424	15	48
23	male	$\star$	black	$\star$	8	81	1029	40	403
24	female	$\star$	white	$\star$	6	84	1244	22	333
25	female	$\star$	hispanic	$\star$	5	42	586	8	58
26	female	$\star$	black	$\star$	5	49	615	30	276
27	male	$\star$	$\star$	English	9	98	1374	39	447
28	male	$\star$	$\star$	French	7	68	904	30	308
29	male	$\star$	$\star$	Spanish	4	18	132	14	98
30	female	$\star$	$\star$	English	7	76	1042	32	323
31	female	$\star$	$\star$	French	4	47	675	23	282
32	female	$\star$	$\star$	Spanish	5	52	728	5	62
33	male	operator	white	$\star$	2	11	65	2	14
34	male	operator	hispanic	$\star$	3	12	62	10	43
35	male	operator	black	$\star$	0	0	0	0	0
36	male	teacher	white	$\star$	2	33	545	15	249
37	male	teacher	hispanic	$\star$	1	19	361	0	0
38	male	teacher	black	$\star$	3	36	470	12	125
39	male	technician	white	$\star$	2	16	226	8	106
40	male	technician	hispanic	$\star$	1	1	1	5	5
41	male	technician	black	$\star$	3	30	422	10	143
42	male	secretary	white	$\star$	1	11	121	3	33
43	male	secretary	hispanic	$\star$	0	0	0	0	0
44	male	secretary	black	$\star$	2	15	137	18	135
45	female	operator	white	$\star$	1	13	169	2	26
46	female	operator	hispanic	$\star$	2	7	25	5	16
47	female	operator	black	$\star$	2	31	481	10	151
48	female	teacher	white	$\star$	2	35	617	6	114
49	female	teacher	hispanic	$\star$	0	0	0	0	0
50	female	teacher	black	$\star$	1	10	100	7	70
51	female	technician	white	$\star$	2	28	394	13	185
52	female	technician	hispanic	$\star$	1	2	4	0	0
53	female	technician	black	$\star$	2	8	34	13	55
54	female	secretary	white	$\star$	1	8	64	1	8
55	female	secretary	hispanic	$\star$	2	33	557	3	42
56	female	secretary	black	$\star$	0	0	0	0	0
57	male	operator	$\star$	English	2	10	52	5	30
58	male	operator	$\star$	French	1	5	25	2	10
59	male	operator	$\star$	Spanish	2	8	50	5	17
60	male	teacher	$\star$	English	3	46	750	10	163
61	male	teacher	$\star$	French	2	33	545	8	130
62	male	teacher	$\star$	Spanish	1	9	81	9	81
63	male	technician	$\star$	English	3	31	451	15	155
64	male	technician	$\star$	French	2	15	197	8	99
65	male	technician	$\star$	Spanish	1	1	1	0	0
66	male	secretary	$\star$	English	1	11	121	9	99
67	male	secretary	$\star$	French	2	15	137	12	69
68	male	secretary	$\star$	Spanish	0	0	0	0	0

Table 2: A summary relation R_s constructed from relation R (Table 2.1). The stars indicate *null* values. Multiple elementary statistics are computed within each tuple, which is referred to as a *summary tuple*.

Type	M_1	M_2	M_3	M_4
mean	$\times$	$\times$		
variance	$\times$	$\times$	$\times$	
covariance	$\times$	$\times$		$\times$
correlation	$\times$	$\times$		$\times$
linear regression	$\times$	$\times$	$\times$	$\times$
multiple regression	$\times$	$\times$	$\times$	$\times$
multivariate analysis	$\times$	$\times$	$\times$	$\times$

Table 3: Statistical metadata required by various statistical analyses. M_1, M_2, M_3 and M_4 denote the count, sum, sum of squares and sum of products functions.

4 Statistical Metadata

An important issue in the definition of a summary relation scheme is the set of candidate statistics to compute, i.e., what the precomputed statistical functions should be. This issue is crucial and determines the utility of the summary relation scheme concept. The answer depends on the types of statistical analyses the users perform, such as regression analysis, multivariate analysis, etc.

It is impractical to enumerate the results of all such analyses and store them in the summary relation. Not only are they numerous, their value is also limited as they cannot be used to compute other statistics. However, they can be synthesized from the *elementary statistics* used in their computation. These statistics, such as sums, sum squares and sum of products, are called *statistical metadata* [Ghos87]. A statistical metadatum is a numerical function of some observed data that can be used to compute other statistics. Table 3 illustrates the types of statistical metadata needed in various types of statistical analyses. Therefore, the summary relation should store the statistical metadata.

The utility of storing statistical metadata derives from the fact that they satisfy the *additive* property [Ghos87]. Given a data attribute S_j, suppose we store its sum $\sum S_j$ with respect to several category instances: $\langle \mathcal{C}_i = 0, \mathcal{C}_j = 0 \rangle$, $\langle \mathcal{C}_i = 0, \mathcal{C}_j = 1 \rangle$, and $\langle \mathcal{C}_i = 0, \mathcal{C}_j = 2 \rangle$. If category domain $\mathcal{C}_j$ has only three attribute values $\{0, 1, 2\}$, then the sum of S_j for the category $\langle \mathcal{C}_i = 0 \rangle$ is equal to the cumulative sums of the three categories. This property extends the usefulness of the statistical metadata. For metadata that preserve the multiplicative property, the multiplicative operations can be transformed into additive ones by logarithmic transformations. Our approach is hence to store an appropriate set of elementary statistics in the summary relation R_s.

Having explained the transformation of data in R into summary tuples in R_s, we turn to how R_s may be used to answer statistical queries. Two issues are involved here: (1) providing tuple access to R_s, (2) and using summary tuples for query processing. These issues are addressed in the next two sections.

5 Storage and Access

The issues in this section are examined in greater depth in [NgRa94, NgRa95]. Here, we elaborate these ideas in the context of summary data management. The goal in this section is to examine how one may be able to retrieve any summary tuple as desired. In a relational database, every tuple is uniquely defined by a *primary key*, which is usually a component of the tuple. For R_s, we propose using all the category domains together as the primary key. For this, we need an ordering rule for tuples so that they may be compared. We shall see how this can be done. An advantage of our method is that it also achieves compression without sacrificing efficiency.

5.1 Tuple ordering scheme

We first map every category domains *numerically*, i.e., we map each category attribute value to a numeric value. The mapping of category domains is easy as they are usually discrete and of finite size with all possible attribute values known in advance. For every attribute $c \in C_i$, we map c into its *ordinal* position within the domain. Only the category domains are mapped.

Example 5.1 Table 4(a) is the numerically mapped version of Table 2. We use the domain size as the range of the integer map. For example, $C_3 = \{white, hispanic, black\}$ is mapped into $C_3 = \{0, 1, 2\}$. ∎

Every summary tuple has two portions, corresponding to the category and statistical attributes, and referred to respectively as the *category sub-tuple* and *statistical sub-tuple*. Since every category sub-tuple is now a tuple of numbers, we may treat each sub-tuple as a mixed-radix integer, and use the value of this integer as the access key. The summary tuples may then be ordered by ordering the category sub-tuples lexicographically.

Example 5.2 Continuing with our previous example, we note that the sizes of C_1, C_2, C_3, C_4 are $2, 4, 3, 3$ respectively. By adopting these sizes as the radices, the last category sub-tuple becomes: $1 \times (4 \times 3 \times 3) + 3 \times (3 \times 3) + 1 \times (3) + 2 = 36 + 9 + 3 + 2 = 50$. Thus, tuples may be compared and sorted by comparing their numerical values. Table 4(b) shows the new ordering of the summary tuples in Table 4(a). This is the final form in which tuples are physically stored. We re-iterate that the mixed-radix integral concept is applied to the category half of a summary tuple only. The statistical portion tags along unchanged. ∎

Below we give a formal definition of the mixed-radix idea. Define a function $\varphi : \hat{\mathcal{C}} \rightarrow \mathcal{N}_{\hat{\mathcal{C}}}$, where $\hat{\mathcal{C}} = \{\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_n\}$ is the set of augmented category domains, $\mathcal{N}_{\hat{\mathcal{C}}} = \{0, 1, \ldots, ||\hat{\mathcal{C}}|| - 1\}$ is the *mixed-radix integer space*, $||\hat{\mathcal{C}}|| = \prod_{j=1}^{n} |\hat{C}_j|$, as a bijection:

$$\varphi(c_1, c_2, \ldots, c_n) = \sum_{i=1}^{n} \left(c_i \prod_{j=i+1}^{n} |\hat{C}_j| \right) \tag{1}$$

for all $\langle c_1, c_2, \ldots, c_n \rangle$[2] $\in \hat{\mathcal{C}}$. The inverse of φ is defined as:

$$\varphi^{-1}(e) = \langle c'_1, c'_2, \ldots, c'_n \rangle \tag{2}$$

[2] A tuple is generally enclosed in angle brackets. When used as an argument of a function, the angle brackets are omitted when no confusion arises.

No.	$\hat{C}_1$	$\hat{C}_2$	$\hat{C}_3$	$\hat{C}_4$	f_0	f_1	f_2	f_3	f_4
1	0	4	3	3	20	184	2410	83	853
2	1	4	3	3	16	175	2445	60	667
3	2	0	3	3	10	74	802	29	250
4	2	1	3	3	9	133	2093	40	558
5	2	2	3	3	11	85	1081	49	494
6	2	3	3	3	6	67	879	25	218
7	2	4	0	3	13	155	2201	50	735
8	2	4	1	3	10	74	1010	23	106
9	2	4	2	3	13	130	1644	70	679
10	2	4	3	0	16	174	2416	71	770
11	2	4	3	1	11	115	1579	53	590
12	2	4	3	2	9	70	860	19	160
13	0	0	3	3	5	23	127	12	57
14	0	1	3	3	6	88	1376	27	374
15	0	2	3	3	6	47	649	23	254
16	0	3	3	3	3	26	258	21	168
17	1	0	3	3	5	51	675	17	193
18	1	1	3	3	3	45	717	13	184
19	1	2	3	3	5	38	432	26	240
20	1	3	3	3	3	41	621	4	50
21	0	4	0	3	7	71	957	28	402
22	0	4	1	3	5	32	424	15	48
23	0	4	2	3	8	81	1029	40	403
24	1	4	0	3	6	84	1244	22	333
25	1	4	1	3	5	42	586	8	58
26	1	4	2	3	5	49	615	30	276
27	0	4	3	0	9	98	1374	39	447
28	0	4	3	1	7	68	904	30	308
29	0	4	3	2	4	18	132	14	98
30	1	4	3	0	7	76	1042	32	323
31	1	4	3	1	4	47	675	23	282
32	1	4	3	2	5	52	728	5	62
33	0	0	0	3	2	11	65	2	14
34	0	0	1	3	3	12	62	10	43
35	0	0	2	3	0	0	0	0	0
36	0	1	0	3	2	33	545	15	249
37	0	1	1	3	1	19	361	0	0
38	0	1	2	3	3	36	470	12	125
39	0	2	0	3	2	16	226	8	106
40	0	2	1	3	1	1	1	5	5
41	0	2	2	3	3	30	422	10	143
42	0	3	0	3	1	11	121	3	33
43	0	3	1	3	0	0	0	0	0
44	0	3	2	3	2	15	137	18	135
45	1	0	0	3	1	13	169	2	26
46	1	0	1	3	2	7	25	5	16
47	1	0	2	3	2	31	481	10	151
48	1	1	0	3	2	35	617	6	114
49	1	1	1	3	0	0	0	0	0
50	1	1	2	3	1	10	100	7	70
51	1	2	0	3	2	28	394	13	185
52	1	2	1	3	1	2	4	0	0
53	1	2	2	3	2	8	34	13	55
54	1	3	0	3	1	8	64	1	8
55	1	3	1	3	2	33	557	3	42
56	1	3	2	3	0	0	0	0	0
57	0	0	3	0	2	10	52	5	30
58	0	0	3	1	1	5	25	2	10
59	0	0	3	2	2	8	50	5	17
60	0	1	3	0	3	46	750	10	163
61	0	1	3	1	2	33	545	8	130
62	0	1	3	2	1	9	81	9	81
63	0	2	3	0	3	31	451	15	155
64	0	2	3	1	2	15	197	8	99
65	0	2	3	2	1	1	1	0	0
66	0	3	3	0	1	11	121	9	99
67	0	3	3	1	2	15	137	12	69
68	0	3	3	2	0	0	0	0	0

Table (a)

No.	$\hat{C}_1$	$\hat{C}_2$	$\hat{C}_3$	$\hat{C}_4$	f_0	f_1	f_2	f_3	f_4
33	0	0	0	3	2	11	65	2	14
34	0	0	1	3	3	12	62	10	43
35	0	0	2	3	0	0	0	0	0
13	0	0	3	3	5	23	127	12	57
57	0	0	3	0	2	10	52	5	30
58	0	0	3	1	1	5	25	2	10
59	0	0	3	2	2	8	50	5	17
36	0	1	0	3	2	33	545	15	249
37	0	1	1	3	1	19	361	0	0
38	0	1	2	3	3	36	470	12	125
60	0	1	3	0	3	46	750	10	163
61	0	1	3	1	2	33	545	8	130
62	0	1	3	2	1	9	81	9	81
14	0	1	3	3	6	88	1376	27	374
39	0	2	0	3	2	16	226	8	106
40	0	2	1	3	1	1	1	5	5
41	0	2	2	3	3	30	422	10	143
63	0	2	3	0	3	31	451	15	155
64	0	2	3	1	2	15	197	8	99
65	0	2	3	2	1	1	1	0	0
15	0	2	3	3	6	47	649	23	254
42	0	3	0	3	1	11	121	3	33
43	0	3	1	3	0	0	0	0	0
44	0	3	2	3	2	15	137	18	135
66	0	3	3	0	1	11	121	9	99
67	0	3	3	1	2	15	137	12	69
68	0	3	3	2	0	0	0	0	0
16	0	3	3	3	3	26	258	21	168
21	0	4	0	3	7	71	957	28	402
22	0	4	1	3	5	32	424	15	48
23	0	4	2	3	8	81	1029	40	403
27	0	4	3	0	9	98	1374	39	447
28	0	4	3	1	7	68	904	30	308
29	0	4	3	2	4	18	132	14	98
1	0	4	3	3	20	184	2410	83	853
45	1	0	0	3	1	13	169	2	26
46	1	0	1	3	2	7	25	5	16
47	1	0	2	3	2	31	481	10	151
17	1	0	3	3	5	51	675	17	193
48	1	1	0	3	2	35	617	6	114
49	1	1	1	3	0	0	0	0	0
50	1	1	2	3	1	10	100	7	70
18	1	1	3	3	3	45	717	13	184
51	1	2	0	3	2	28	394	13	185
52	1	2	1	3	1	2	4	0	0
53	1	2	2	3	2	8	34	13	55
19	1	2	3	3	5	38	432	26	240
54	1	3	0	3	1	8	64	1	8
55	1	3	1	3	2	33	557	3	42
56	1	3	2	3	0	0	0	0	0
20	1	3	3	3	3	41	621	4	50
24	1	4	0	3	6	84	1244	22	333
25	1	4	1	3	5	42	586	8	58
26	1	4	2	3	5	49	615	30	276
30	1	4	3	0	7	76	1042	32	323
31	1	4	3	1	4	47	675	23	282
32	1	4	3	2	5	52	728	5	62
2	1	4	3	3	16	175	2445	60	667
3	2	0	3	3	10	74	802	29	250
4	2	1	3	3	9	133	2093	40	558
5	2	2	3	3	11	85	1081	49	494
6	2	3	3	3	6	67	879	25	218
7	2	4	0	3	13	155	2201	50	735
8	2	4	1	3	10	74	1010	23	106
9	2	4	2	3	13	130	1644	70	679
10	2	4	3	0	16	174	2416	71	770
11	2	4	3	1	11	115	1579	53	590
12	2	4	3	2	9	70	860	19	160

Table (b)

Table 4: Category domain mapping and tuple re-ordering. Table (a) is the numerically mapped version of Table 3.2. Table (b) is the re-ordering of tuples in Table (a) via the mixed-radix integral order applied to the category sub-tuples only. Notice that the null values have been replaced by the size of the respective domains. For instance, $\star$ is replaced by 2 under column $\hat{C}_1$.

for all $e \in \mathcal{N}_{\hat{\mathcal{C}}}$ and $i = 1, 2, \ldots, n-1$,

$$c'_i = \left\lfloor \frac{c^r_{i-1}}{\prod_{j=i+1}^{n} |\hat{C}_j|} \right\rfloor \tag{3}$$

$$c^r_i = c^r_{i-1} - c'_i \prod_{j=i+1}^{n} |\hat{C}_j| \tag{4}$$

where $c^r_0 = e$ and $c'_n = c^r_{n-1}$. Let t_c be the category portion of a summary tuple t, i.e., $t_c = \pi_{\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_n}(t)$. Given $t_c \in \hat{\mathcal{C}}$, φ converts it to its mixed-radix integer representation $\varphi(t_c)$. The uniqueness of mixed-radix representation is shown in [Knut73].

5.2 Tuple access

As summary tuples are now clustered lexicographically (applied to the category sub-tuples only), a primary index may be constructed that uses category sub-tuples as search keys. Figure 2 shows an order-3 primary B^+ tree index constructed for relation R_s. Notice the placement of tuples into disk blocks.

The primary index is useful only when the category portion of a tuple is completely available as the search key. When only some category attributes are known, secondary indices are needed. A secondary index requires a level of indirection between the attribute values and the data blocks where they might be found. For instance, the following provides the indirection for domain C_2: $(\mathit{operator} \mid 1, 5, 7), (\mathit{teacher} \mid 1, 2, 5, 6, 7), (\mathit{technician} \mid 2, 3, 6, 8), (\mathit{secretary} \mid 3, 4, 6, 8)$, which says that tuples whose $C_2 = \mathit{operator}$ are located in block $1, 5, 7$. As the example relation R is too small, we are unable to construct a full-scale secondary index for any of the category attributes. Nevertheless, we may conclude that with the help of the primary and secondary indices, summary tuple access is as usual.

5.3 Tuple insertion and deletion

Suppose we wish to insert tuple $t = \langle \star, \star, 2, 2, 03, 26, 338, 10, 97 \rangle$. The primary index provides the means to locate the block which contains tuples that are physically ordered in the neighourhood of t. With this index, data block 8 is found to be the candidate block for inserting t. For tuple deletion, the primary index is similarly used to locate the data block and changes made within the block.

5.4 Tuple modification

In conventional database, any attributes of a tuple may be modified. For summary relation, modification is limited to the statistical domains only. The category attributes are never modified because modification changes the concept described by the attributes. If new a concept is needed, a summary tuple is inserted. Although tuples in the summary relation are ordered differently from ordinary relations, tuple modification is still perform *in situ*, as in conventional database.

In brief, the concept of managing summary data as tuples of a relation allows us to access them through well-defined standard database operations. This simplicity makes the summary relation model a practical model for the efficient management and organization of summary data.

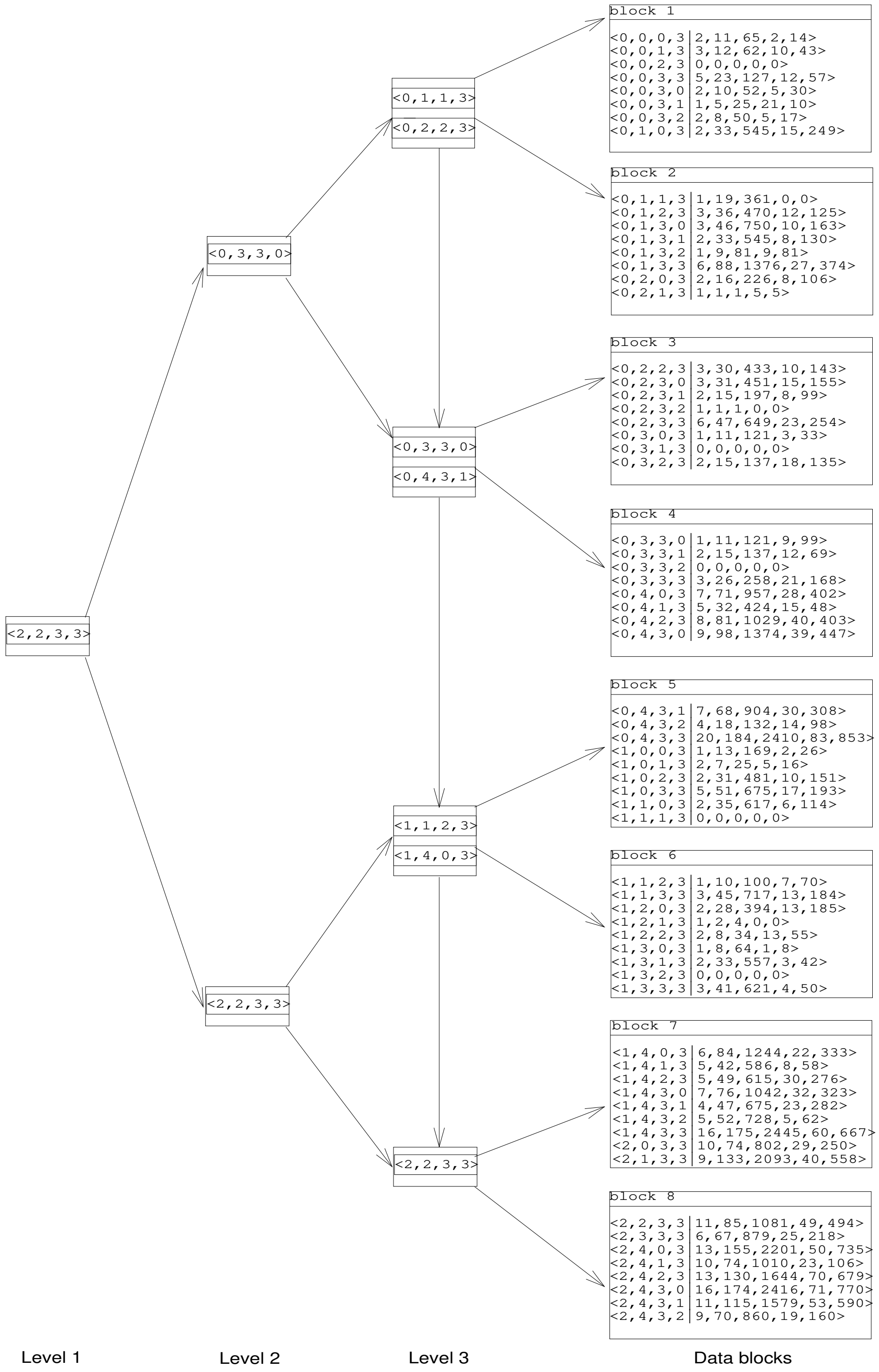

Figure 2: Primary index. The data blocks contain difference tuples in φ order. The statistical attributes are separated from the category attributes by a bar.

average income			C_1 *male*	*female*
C_2 *operator*	C_3	*white*	5.5	13
		hispanic	4	3.5
		black	0	15.5
teacher	C_3	*white*	16.5	17.5
		hispanic	19	0
		black	12	10
technician	C_3	*white*	8	14
		hispanic	1	2
		black	10	4
secretary	C_3	*white*	11	8
		hispanic	0	16.5
		black	7.5	0

(a)

average income			C_1 *male*	*female*
C_2 {*operator, teacher*}	C_3	*white*	11	16
		hispanic	32.8	3.5
		black	12	13.7
{*technician, secretary*}	C_3	*white*	9	12
		hispanic	1	11.7
		black	9	4

(b)

Figure 3: Cross tabulation of average income (S_1) with respect to sex (C_1), occupation (C_2) and race (C_3). Summary tuples numbered 33 to 56 in R_s (Table 3.2) are used to construct Tables (a) and (b). Observe that attribute values of C_2 are grouped in Table (b).

6 Query Processing and Evaluability

This section addresses the essence of the summary relation model: How does the concept of summary tuples support efficient statistical query processing? We first have to ask what are the typical statistical queries. In the next subsections, we examine some commonly encountered statistical queries.

6.1 Cross tabulation

Suppose a query desires the average income (S_1, in thousands) of a population categorized by their sex (C_1), occupation (C_2) and race (C_3), as expressed by the following SQL query:

```
SELECT     AVG(income)
FROM       census
GROUP BY   sex, occupation, race
```

The SQL query translates briefly into the following sequence of operations:

(i) The following attributes are projected from R_s: sex (C_1), occupation (C_2), race (C_3), cardinality (f_0) and sums for income (f_1).

(ii) Those tuples whose C_1, C_2, C_3 attributes are non-null are selected, the rest are discarded. The final tuples correspond to summary tuples numbered from 33 to 56 in R_s.

(iii) The average income statistic (i.e., f_1/f_0) is computed for each summary tuple. Each statistic is an entry in the final tabulation shown in Table (a) of Figure 3. For instance, tuple 33 $\langle 0, 0, 0, 2, 11\rangle$ gives a value of $11/2 = 5.5$ which forms the top-leftmost entry in Table (a).

This example demonstrate how a desired statistic may be obtained by retrieving the relevant summmary tuples, and applying a simple additional operation to the elementary statistics in them.

6.2 Attribute filtering

A statistician is usually not interested in all the attribute values of a domain. Attribute filtering is often performed to restrict the values and to refine the statistics. For instance, instead of tabulating the average income by age, he may be interested in age groups (e.g., $10 \sim 20, 20 \sim 30, 30 \sim 40$) or ages in the range of $20 \sim 30$ only. The process of answering such queries is identical to the cross tabulation type query described above except that the final statistics undergo an additional *filtering* operation to derive the desired statistics.

Example 6.1 Suppose the above query is refined to finding the average income of a population categorized by sex, race and occupation groups: $\{operator, teacher\}$, $\{technician, secretary\}$. The tabulation appears in Figure 3(b). The top-leftmost entry is derived from the relevant fields of summary tuples 33 and 37:$\langle 0,0,0,2,11\rangle, \langle 0,1,0,2,33\rangle$. These are fields from tuples 33 and 36 obtained after the projection operation in Step (i) of the previous example. The average income is: $(11+33)/(2+2) = 11$. ∎

Filtering is a form of data derivability operation where the statistics of several summary tuples are *condensed* into a few results. This is a feature which the summary relation model supports with simplicity. We will discuss data derivability in Section 6.4.

6.3 Correlation analysis

Suppose a demographer is interested in the correlation, if any, between the income and the number of hours worked for white males teachers. From tuple 36 in Table 2, one obtains f_1, f_3 and f_4, which are figures for the sum of income ($\sum S_1$), sum of hours worked ($\sum S_2$), and product sum of income and hours worked ($\sum S_1 S_2$). The covariance of S_1 and S_2 is:

$$\begin{aligned} cov(S_1, S_2) &= E(S_1 - E(S_1))(S_2 - E(S_2)) \\ &= E(S_1)E(S_2) - E(S_1 S_2) \\ &= \frac{f_1}{f_0}\frac{f_3}{f_0} - \frac{f_4}{f_0} \\ &= \frac{33}{2}\frac{15}{2} - \frac{249}{2} \\ &= 123 \end{aligned}$$

Therefore, there is positive correlation between the income and number of hours worked for all white male teachers.

6.4 Query derivability

The queries we have illustrated are immediately satisfiable by retrieving the relevant tuples from the summary relation and performing simple arithmetic to obtain the results. There may be cases where the tuples are not available. Suppose we have the query:

```
SELECT     AVG(income)
FROM       census
GROUP BY   sex, occupation
```

The query is immediately satisfiable by consulting tuples 13 to 20 in Table 2. Suppose these tuples were not in R_s. Then one can still satisfy the query by accessing indirectly relevant tuples.

Observe that the C_1, C_2, C_3 attribute domains are fully enumerated in tuples 33 to 56, i.e., all possible combinations of attribute values appear among the tuples. This set of tuples may be used to answer the above query by condensing the statistics of those tuples whose C_1 (sex) and C_2 (occupation) attributes are identical. For instance, combining tuples 33 to 35 gives $\langle 0, 1, \star, \star, 5, 23, 127, 12, 57 \rangle$. Such condensation produces a set of $|C_1| \times |C_2| = 2 \times 4 = 8$ summary tuples which fully enumerates the C_1, C_2 attributes. The statistics in these tuples may then be tabulated to produce the results.

In all the above examples, the benefits of summary statistics are clear. Queries are satisfied by accessing only a subset of tuples, directly or indirectly, from the summary relation R_s. The set of tuples accessed is only a small fraction of the large original relation from which the summary relation is derived. Thus, a lot of I/Os has been eliminated. Moreover, the summary relation concept simplifies the access of relevant summary statistics.

7 Management of Summary Relation

What are the summary tuples to put into R_s, and how many should there be? The effectiveness of the summary approach depends on the contents of R_s so that most queries are satisfiable. However, the size of R_s is potentially large. Therefore, a balance must be maintained between the size of R_s and the response time of queries relying on R_s.

We propose an adaptive approach, whereby summary tuples are added to R_s as they are needed and obsolete tuples are deleted. Over a sufficiently long period of time, a state of stability is reached whereby the contents of R_s reflect the average users' needs in the sense that most queries are satisfiable. The formalization and adequacy of this stability is part of our future work.

8 Conclusions

We have proposed a simple, intuitive and practical method for the organization and management of large numbers of summary data derived from a relation. The idea is that summary data derived from an ordinary relation may themselves be organized collectively into a relation (or *meta*-relation) called the summary relation. This has several advantages that distinguish it from previous approaches [Abad92, ChMc89, CMM88, OOM85]:

- The model is conceptually simple because a summary tuple represents a set of *elementary* summary data all described by the same category. In addition, the category attribute values are atomic. In most previous approaches, the category of a summary datum is unrestricted in complexity, making the collective management and physical storage of summary data difficult.

- The elementary summary data contained in the summary tuples permit the derivation of complex summaries for various types of statistical analyses such as regression analysis, multivariate analysis and variance analysis. This is a more flexible approach than previous ones where complete statistics such as averages and variances are stored.

- The approach is more space-efficient as several statistics described by a common category are stored in a single summary tuple. In previous approaches, a separate *summary table* is constructed for each of the statistics, duplicating the category in each table.
- The tuple approach benefits from the well-established field of relational databases. It not only provides an easy organization for summary tuples, but permits easy access and modification.

Our method is particularly suitable for statistical databases where the volume of data is large and retention is high. Examples of which are databases constructed from social, economic, technological, inventory, environmental, medical or demographic surveys. Data seldom, if ever, changes. Thus, a summary relation requires no maintenance with respect to the contents of the computed statistics. However, summary tuples may be easily inserted and deleted as users' requirements vary over time.

We are currently working with the Center of Population Studies at the University of Michigan, Ann Arbor, to construct summary relations for the Public User Microdata Samples (PUMS) from the 1990 U.S. Census of Bureau [PUMS92]. The PUMS contain records representing 5% or 1% samples of the housing units in the U.S. and of persons residing in them. The 1% samples contain 2.3 million records and occupy 800 megabytes, while the 5% samples contain 13 million records and occupy 4.4 gigabytes. Accessing the entire database per statistical query is prohibitive. The availability of summary tuples decreases the response time of statistical queries.

References

[Abad92] S. Abad-Mota. Approximate Query Processing with Summary Tables in Statistical Databases. *Lecture Notes in Computer Science*, Vol. 580, pp. 499-515, 1992.

[ChMc89] M. C. Chen, L. P. McNamee. On the Data Model and Access Method of Summary Data Management. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, No. 4, pp. 519–528, Dec. 1989.

[CMM88] M. C. Chen, L. P. McNamee, M. Melkanoff. A Model of Summary Data and its Applications in Statistical Databases. *Proceedings of the 4th International Working Conference on Statistical and Scientific Database Management*, 1988.

[FRRS86] E. Fortunato, M. Rafanelli, F. Ricci, A. Sebastio. An Algebra for Statistical Data. *Proceedings of the 3rd International Workshop on Statistical Databases*, 1986.

[Ghos84] S. P. Ghosh. An Application of Statistical Databases in Manufacturing Testing. *Proceedings of COMPDEC*, Vol. 1, pp. 96–103, 1984. Also published as *IBM Research Report RJ4055*.

[Ghos86] S. P. Ghosh. Statistical Relational Tables for Statistical Database Management. *IEEE Transactions on Software Engineering*, Vol. 12, No. 12, pp. 1106–1116, Dec. 1986. Also published as *IBM Research Report RJ4394*.

[Ghos87] S. P. Ghosh. Statistical Metadata: Linear Regression Analysis. *Foundations of Data Organization*. S. Ghosh, Y. Kambayashi, K. Tanaka, Ed. Plenum, New York, 1987.

[Hebr86] G. Hebrail. A Model of Summaries for Very Large Database. *Proceedings of the 3rd International Workshop on Statistical Databases*, 1986.

[IkKo81] H. Ikeda, Y. Kobayashi. Additional Facilities of a Conventional DBMS to Support Interactive Statistical Analysis. *Proceedings of the 1st International Workshop on Statistical Database*, 1981.

[Knut73] D. E. Knuth. *The Art of Computer Programming*, Vol. 2, Addison-Wesley Publishing Company, Reading, Massachusetts, 1973.

[Malv88] F. M. Malvestuto. The Derivation Problem for Summary Data. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 82–89, Jun. 1988.

[MaMo90] F. M. Malvestuto, M. Moscarini. Query Evaluability in Statistical Databases. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2, No. 4, pp. 425–430, Dec. 1990.

[NgRa94] W. K. Ng, C. V. Ravishankar. A Physical Storage Model for Efficient Statistical Query Processing. *Proceedings of the 7th International Working Conference on Statistical and Scientific Databases*, pp. 97–106, Charlottesville, Virginia, September 1994.

[NgRa95] W. K. Ng, C. V. Ravishankar. Relational Database Compression Using Lossless Vector Quantization. *Proceedings of the 11th International Conference on Data Engineering*, Taipei, Taiwan, March 1995.

[OOM85] G. Ozsoyoglu, Z. M. Ozsoyoglu, F. Mata. A Language and a Physical Organization Technique for Summary Tables. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 3–16, 1985.

[PUMS92] Census of Population and Housing, 1990: Public Use Microdata Samples U.S. (machine readable data files). Prepared by the Bureau of the Census. Washington: The Bureau, 1992.

[Rame89] E. Ramez. *Fundamentals of Database Systems*. Benjamin/Cummings, Redwood City, California, 1989.

[Sato81] H. Sato. Handling Summary Information in a Database: Derivability. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1981.

[Shos82] A. Shoshani. Statistical Databases: Characteristics, Problems, and Some Solutions. *Proceedings of the International Conference on Very Large Data Bases*, pp. 208–222, Sep. 1982.