A REALIZATION STRATEGY FOR A STATISTICAL DATA MODEL WITH HIERARCHY OF STATISTICAL OBJECTS

A dissertation submitted in partial fulfilment of the requirements for the **M.Tech. (Computer Science)** degree of Indian Statistical Institute

by

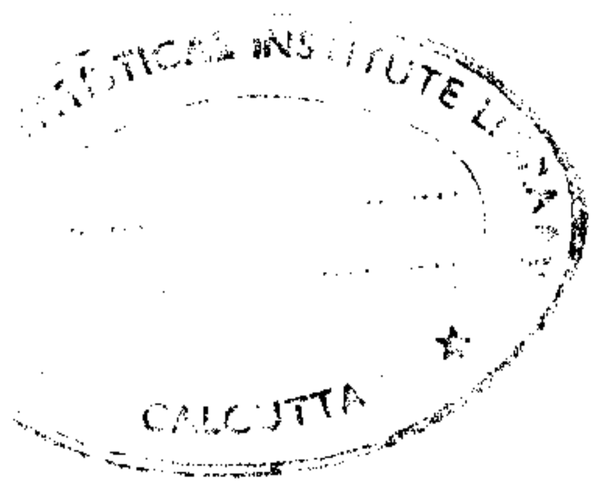
Kaushik Dutta

under the supervision of

Aditya Bagchi

Computer and Statistical Services Center
INDIAN STATISTICAL INSTITUTE
203, Barrackpore Trunk Road
Calcutta-700035

1996



Certificate of Approval

This is to certify that the thesis titled **"A Realization Strategy for a Statistical Data Model with Hierarchy of Statistical Objects"** submitted by Kaushik Dutta, in partial fulfilment of the requirements for **M.Tech. in Computer Science** degree of the Indian Statistical Institute, Calcutta is an acceptable work for the award of the degree.

Date : July 26, 1996

(Supervisor)

(External Examiner)

Acknowledgement

My sincerest gratitude goes to *Prof. Aditya Bagchi* for his guidance, advice, enthusiasm and support throughout the course of the dissertation.

I would also like to take this opportunity to thank *Prof. Bhargab B. Bhattacharya, Mr. Dipti Mukherjee, Prof. B. P. Sinha* and *Dr. Subhash Nandi* for the excellent courses they have offered in M.Tech. which helped me in this work.

Thanks to *Mr. Subhash Kundu* and *Dr. Partha Pramanik* who encouraged the research by giving proper technical advice whenever required.

Mr. Pranab Chakraborty, Mr. Subhomoy Maitra, Mr. Pinakpani Pal, Mr. Sounak Mishra and *Mr. Paramartha Dutta* played very crucial role in contributing numerous suggestions throughout the project. I thank them all and also my other classmates who had made my two years stay at ISI enjoyable.

I express my heartfelt thanks to the members of the M.Tech. Dissertation Committee.

Contents

Chapter 1

Introduction

1.1 Motivation

Commercially successful database packages are usuallly based on Codd's relational data model[1]. The conceptual model of the Codd's system uses two dimensional table. It provides operators which can logically relate two tables or take a subset of an existing table. Facilities are also available to create new tables and to alter existing tables. Relational model, however provides limited facilities for numerical and statistical computation. The data produced by such operations are also not stored in the database.

However there are certain application areas like the census of a country, large socio-economic surveys etc. where extensive statistical analysis is done on the data. The users are also not interested in the raw data, rather queries are made on the summary data only. It is not very advantageous to create such an environment under relational framework. A new data model is needed.

1.2 Statistical Database

Statistical database handles applications where the users prefer to do statistical computation over the stored raw data to generate summary data. Storage, updation, retrieval etc. are done on the summary data instead of the raw data. For example, in case of a country wide census, the users are mainly interested in the summary data like average population per state, per capita income etc. and not about data related to individual citizen. For this

purpose statistical functions like Total, Count, Mean, Standard Deviation etc. are applied on the stored raw data (also called micro data) to generate the required summary data (also called macro data or statistical object).

1.3 Salient Features of SDB

1.3.1 Multi-dimensionality

While statistical computations are done over one or more variates (called attributes in relational system), other features or attributes (called category attributes) associated with the variates define the universe of such computation. The result of the computations creates a new attribute called summary attribute. For example a country wide census collects data about each citizen of the country. Now if we want to know the total number of citizen against each unique combination of values of religion, state and profession recorded during the census, then the new attribute population generated by the computations is the summary attribute and religion, state and profession are the category attributes. These category attributes associated with a summary attribute thus define a multi-dimensional space which cannot be represented effectively in the two dimensional structure of the relational tables.

1.3.2 Classification Hierarchy

Similar to generalization/specialization hierarchy present among the attributes and relations, "is_a" relationship may also be present among category attributes. The extensions of relational model proposed so far for modelling SDB could not take care of this property. However "Graph Oriented Model" of SDB can effectively handle multi-dimensionality and classification hierarchy.

1.3.3 Intermediate or Meta-Data

In the process of computing a summary attribute, one may come across certain intermediate functions which cause additional summary attributes to be created. These intermediate functions may be stored for ease of future computations.

For example, $SUM(X)$, $SUM\text{-}OF\text{-}SQUARE(X)$ and $VAR(X)$ are created and stored when $VAR(X)$ (variance of X) is computed on variate X. Later

Figure 1.1:

Population		Profession		
		P1	P2	P3
State	Religion			
S1	R1			
	R2			
	R3			
S2	R1			
	R2			
	R3			

they may be used for future computations of other summary attributes. The set of category attributes for such computations should, however, remain unaltered.

1.4 Previous Work

Substantial amount of work has been carried out in the field of statistical database during the last one decade. Different data models have been proposed to understand the fundamentals of statistical database management systems. The data models proposed so far follow either of the two following methodologies.

1.4.1 Two dimensional tabular representation

This work [2] was done under the influence of the relational model. Here the summary data is represented by two dimensional tables again. One possible 2-D table representation of population against State, Religion and profession is shown in the fig 1.1. State has two values S1 and S2, Religion has three values R1, R2, R3 and Profession has three values P1, P2 and P3.

The above representation of statistical database has the following limitations,

- *The concept of multi-dimensionality is distorted*
  Because of 2-D representation, the multi-dimensional space is squeezed

Figure 1.2:

Average Income		Professional Category					
		Engineer		Secretary		Teacher	
		Profession		Profession		Profession	
Sex	Year	Chemical	Civil	Junior	Executive	School	College
Male	1980						
	81						
	82						
	...						
	88						
Female	1980						
	81						
	82						
	...						
	88						

into two dimensions. This is typically done by choosing several of category attributes as rows and columns. Since the category attributes around a summary attribute define the multidimensional space, a 2-D representation with the loss of multidimensionality makes query processing difficult. In the table shown in fig-1.1, a query on Religion within a State can be processed more easily than a query on State within a Religion. In general, the 2-D representation of a multi-dimensional statistical object forces an arbitary choice of rows and columns. But the proper model should retain the concept of multi-dimensionality and represent it explicitly. The table of fig-1.1 uses a very simple function like "Count". If more complicated functions are used under this 2-D representation, some queries may even need run-time computation.

- *The classification relationship is lost*
  In the 2-D representation, classification hierarchies are represented in the same manner as the multi-dimensional categories. Consider, for example, that "Professions" are classified into "Professional Categories" as shown in the 2-D table of fig-1.2.

  As can be seen from the figure, there is no difference in the representation of "Sex" and "Year" and the representation of "Profession" and "Professional Category". However it is obvious from this example

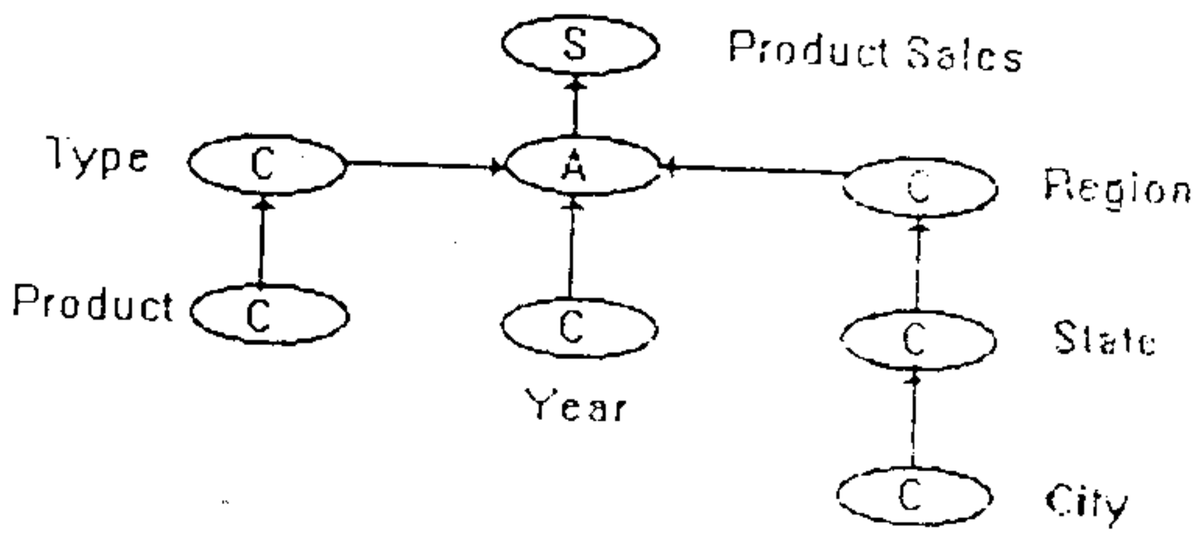
that the values of average income are given for specific combination of "Sex", "Year" and "Profession" only. Thus "Professional Category" is not part of the multi-dimensional space of this statistical object. As can be seen from the above example, there is fundamental difference between category classification relationship and multidimensionality. Only the lowest level sub-class in the classification relationship participates in the multi-dimensional space. This fundamental difference should be explicitly represented in a semantically correct statistical data model.

- *Lack of meta-data level*
  A 2-D represtation requires that the category names as well as the category instances be represented together. There is no separate description of what the statistical database is about (metadata). For example, the meta-data for the above table consists only the "Average Income" by "Sex", "Year" and "Profession", and professions are classified in "Professional Category". However it is represented together with the values of the category attributes. Consequently, the 2-D representation becomes very large for tables with high dimensionality, or when the categories have a large number of instances. Such representation cannot comfortably fit on a page or screen. In such cases, the representation spreads into multiple pages or screens. For example if we add another dimension, "State" to the above table we may need to represent each state on a separate page. This confuses the understanding of the statistical object. It is therefore desirable to separate the representation of the categories and the category instances in order to achieve compactness of the semantic description of the database.

1.4.2 Graph Oriented Representation Model

Graph Oriented Statistical Object Representation Model (STORM) as proposed by Rafanelli & Shoshani[3] arranges the statistical objects in a directed tree. The summary attribute and each of the category attributes are represented as nodes of type S and C respectively. The root of the tree is always the node S. In addition, another node type is used, denoted as A node, to represent an aggregation of nodes pointing to it. In most cases the nodes pointing to a node will be C nodes, but it is possible that an A node may point to another A node. An example of a STORM representation of the SO (statistical object) "product sales" is given in figure-1.3.

Figure 1.3:

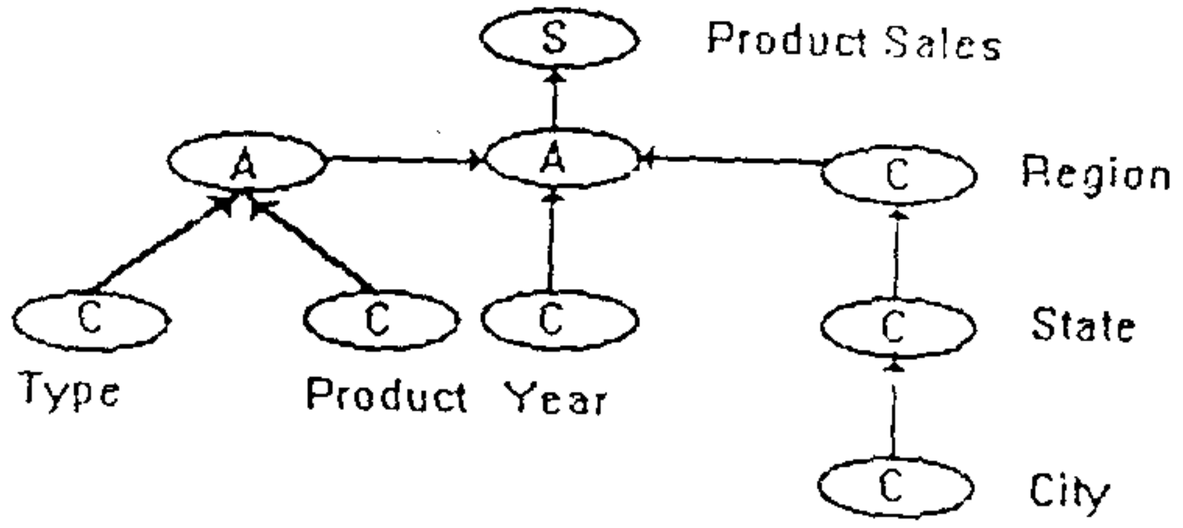


Another possible representation of the same example is shown in the fig-1.4, which illustrates the possibility of an A node pointing to another A node.

The two representation of the Statistical Object "product sales" given in the above two figures have radically different meanings. In the first figure the implication is that the sales amounts are given for each product (e.g. chair, table,...) and that products are grouped into types (e.g. metal, wood, ...). In this example product may belong to more than one type. On the other hand in the second figure the implication is that the sales amounts are given for each type-product combination. Thus, the sales figures are given for "metal chairs", "plastic chairs" etc. Some of these figures may be zero or "non-existing".

However the major disadvantage of the above model is that there is no way to determine which representation is the desired one from the original description of the Statistical Object.

Figure 1.4:


S
Product Sales
A
A
C
Region
C
C
C
C
State
Type
Product
Year
C
City

Chapter 2

HISTO Model

2.1 HISTO - A Statistical Database System

The Graph-Oriented STORM model can effectively handle multidimensionality and classification hierarchy. Statistical Relational Tables proposed by S.P.Ghosh [4] can incorporate meta-data. But on one side none of the Graph Oriented Model can exploit meta-data and on the other hand 2-D model can not incorporate multi-dimensionality.

In order to exploit the facilities of both types of representations, a new SDB model HISTO has been proposed [5]. In this section the HISTO model hass been discussed. Next section would deal with the implementation.

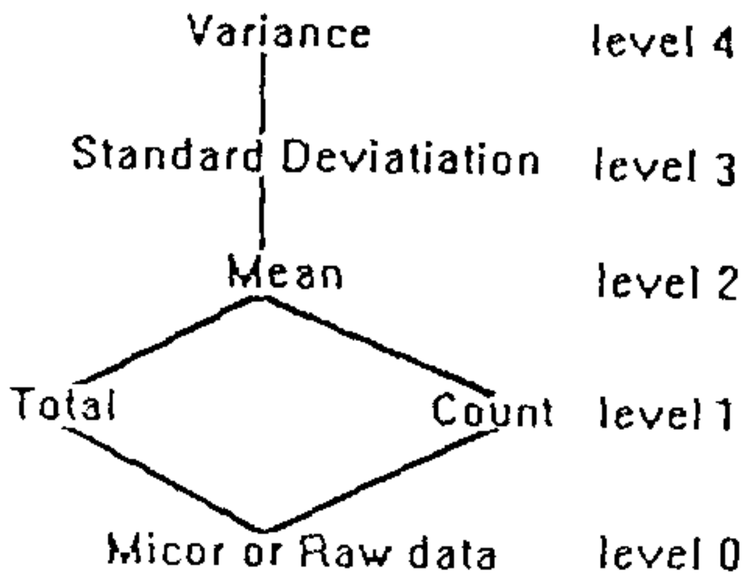
The earlier statistical data models have considered a very small set of simple statistical functions like Total, Count, Mean, Median etc. However in practice, when statistical analysis is done on any survey data, more complex functions are used. These statistical functions can be placed in a hierarchy called the Functional Hierarchy as shown in the fig-2.1.

So a function f when placed in the Functional Hierarchy, is assigned with a level value and is specified as a tuple

$< function\_name, level\_value >$

As shown in the figure, functions Total and Count are required to generate the Mean which in turn generates Standard-Deviation. Variance is again created from Standard Deviation. So, the summary attributes generated by these functions on the same variate with the same set of category attributes should also maintain the same hierarchy.

Figure 2.1: Functional Hierarchy with function levels



In HISTO (Hierarchy of Statistical Objects) model a statistical object type S_i is defined as,

$S_i =< N, C, V, S_j, f_i >$ where

S_i : a statistical object type.
N : name or id of S_i.
C : set of category attributes associated with S_i.
V : the variate on which f_i has been applied.
f_i : the statistical function applied on V to create S_i.
S_j : set of statistical object types used to generate S_i.

Here i in f_i signifies the level at which f_i has been placed in the functional hierarchy.

If $s_j \in S_j$ and s_j has been generated by the function f_j, then $j < i$. That means, the function f_j is placed at a level lower than f_i in the functional hierarchy.

Let us consider that a company database offers the following attributes for

the employees :

$$\{Employee - no., Name, Age, Salary, Dept - Name\}$$

The employee have been classified into four age groups in years,

$$([20 - 30], [31 - 40], [41 - 50], [Above50]).$$

There are three departments,

$$(D1, D2, D3)$$

in the company.
A query demands,
Mean *Salary* of employees against *Age* and $Dept - Name$.
So here,

S_i = Mean *Salary*
C = $\{Age, Dept - Name\}$
V = *Salary*
f_i = Mean
S_j = Total *Salary*, Count of Employees

In the above example, the function 'Mean' uses the function 'Total' and 'Count' to create the statistical object type 'Mean_Salary'. 'Total' and 'Count' on the other hand create the statistical object types under S_j. In other words, the statistical object types under S_j are accessed to generate S_i. So the hierarchy among the statistical functions automatically maps onto a corresponding hierarchy of the statistical object types. So according to this hierarchy S_i is a supertype of all members of S_j.

Each statistical object type has a number of instances. Each set of value combination of the category attributes alongwith the corresponding value of the summary attribute provides an instance. So in the above example, since there are 4 Age_Group and 3 Dept-Name, S_i or Mean_Salary will have 12 instances.

Now in the definition of the statistical object type S_i, the subtype set S_j may be null. That means, S_i is getting created from the raw data. Category attribute set C can also be null signifying that S_i is accessing the Root of the Category Hierarchy. Now, if the statistical function f_i is also null, then the statistical object type may have only the variate V. This would be equivalent to the creation of a query on raw data similar to a query made

in relational or any such standard data model.

Besides the functional hierarchy, the statistical object types can also be placed in another hierarchy according to the set of category attributes associated with them. We call this the Category Hierarchy.

Again from the above example we get,

Mean *Salary* against $\{Age, Dept - Name\}$

but we should also be able to retrieve the marginals like,

Mean *Salary* against $\{Age\}$ irrespective of $Dept - Name$

and

Mean *Salary* against $\{Dept - Name\}$ irrespective of *Age*

We should also be able to retrieve the Mean *Salary* for the total set of employees irrespective of any category attribute.

Let's consider that a function f is applied on variate V. The environment has four category attributes $\{C1, C2, C3, C4\}$. Two different statistical object types $S1$ and $S2$ have been created such that,

$$S1 =< N1, \{C1, C2, C3\}, V, f >$$ and

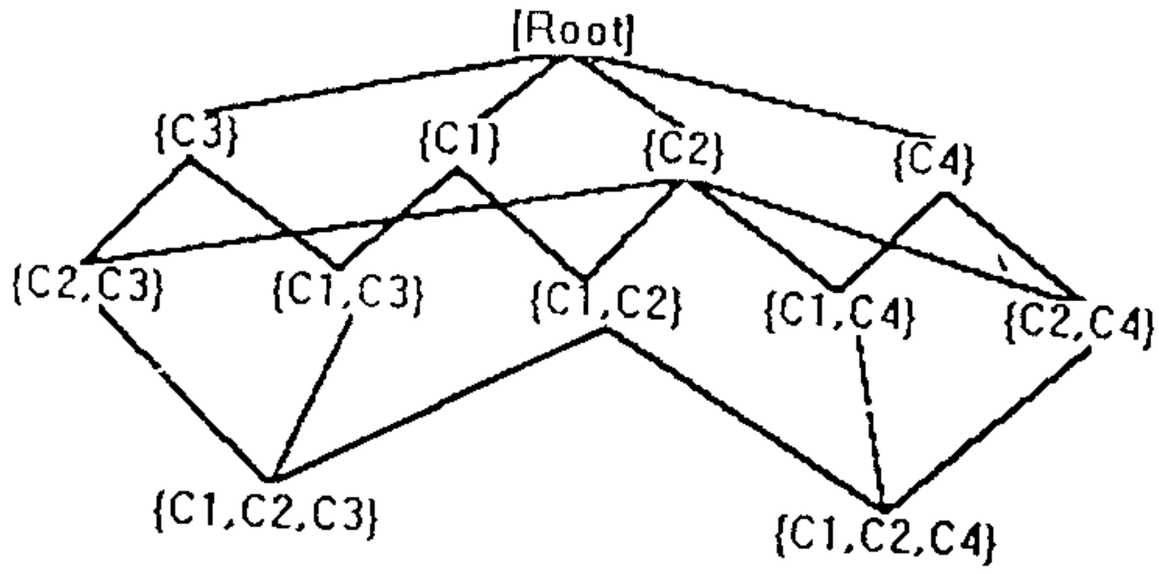
$$S2 =< N2, \{C1, C2, C4\}, V, f >$$

For the same function f and variate V, if all the marginal types for the above two statistical object types are created and stored in the database, we get the hierarchical structure shown in fig-2.2. This hierarchy is called *Category Hierarchy.*

The hierarchical structure shows that two different statistical object types may have common marginal object types. Object types involving category attributes $\{C1, C2, C3\}$ and $\{C1, C2, C4\}$, for same f and V, have the common marginal object type with category attributes $\{C1, C2\}$.

The two main problems of 2-D representation as discussed previously can be solved with the help of the above two hierarchies as explained later.

Figure 2.2: The Category Hierarchy



Chapter 3

Implementation of HISTO model

In this dissertation a methodology for implementation of statistical database have been developed following the HISTO model.

3.1 Salient Features

The SDB has been developed as a shell above the relational database. The relational environment provides the management of raw data. It consists of only one table, with all the attributes of the system, such a relation is called **Universal Table**. The raw data can be accessed by the usual SQL language. It has been assumed that the interface between the SDB shell and the relational database is developed using PRO*C (Oracle 7.0) or similar languages. Development of such interface is permissible in any standard RDBMS available in the market (e.g. Oracle, VAX-RDB, INFORMIX etc.). Standard SELECT statement has been used to access the raw data in the Universal Table. These commands are available in any RDB having ANSI SQL or its extension as its query language. So the proposed methodology for development of SDB shell on RDB is totally system independent. The major assumptions made in the development of SDB are discussed below.

3.1.1 Assumptions

1. No change in micro datatypes and instances.

2. Instances of a statistical object would always be numeric.

3.1.2 Properties

1. Functional hierarchy for the creation of Statistical Object (SO) is uniquely defined, and runtime function definition is not permissible.

2. SOs may be generated dynamically during query processing.

3. All the SOs generated during processing of a query may not be stored permanently.

4. Any SO instance returning zero should not be used for further computation in a hierarchy.

5. Intermediate SOs used to create a new SO must have same set of category attribute instances.

6. For creating a new SO, any other SO lower in the hierarchy may appear more than once during computation.

3.1.3 Condition for Object Instance Creation

1. An SO is created if and only if lower order SOs in the hierarchy are also created.

2. Any SO accessing any other SO in the hierarchy is only concerned with the instance of such SO and not with its structure.

3. An SO can, therefore, be created as an Abstract Datatype.

3.2 Data-Structure

3.2.1 Attribute Dictionary

The information about the attributes of the SDB will be stored here. The attribute information that should be stored in the attribute dictionary for running the SDB shell are as follows,

Attribute Name The attribute name may be anything. It is used only for external user interface.

Attribute Type The attribute type may be *Category* (e.g. hobby, sex, ...), *Variate* (e.g. salary, number of dependants, ...) and *Both* (e.g. age may be age group meaning category and also age may be a variate).

Variate Number This property of an attribute is valid for the type variate and both only. Each variate in the SDBMS is mapped to an unique integer internally. For all the internal processing a variate is identified by its *Variate Number*. This can be any natural number starting from 1. For category the value is stored as 0, signifying that it is not to be considered for computations.

Category Number This property of an attribute is valid if the attribute is of the type category or both only. Each category in the SDBMS is mapped to an unique integer internally. For all the internal processing a category is identified by its *Category Number*. This can be any natural number starting from 1. For variate the value is stored as 0.

3.2.2 Category Instance Dictionary

For each category its instances are stored for use in the SDB. The information stored for each attribute of type category or both are as follows,

Category Number This is the same number as stored in the *Attribute Dictionary*.

Instance Number The instance number of a category or both type attribute gives the number of instance of the attribute in the SDB.

Instance Type The instance type may be numeric or alpha-neumeric. Age attribute has numeric instances. Numeric category instances are basically numeric ranges. So Numeric instances has two part,

- Lower Limit
- Upper Limit

Instances For Numeric Category Instances the instances are set {*lower_limit*, *Upper_limit*}. For Alpha-numeric Category Instances, the instances are stored as strings.

3.2.3 Function Dictionary

For each function defined in the SDB, the following information are stored,

Function Name Function name is the name of a function defined in the SDB, this is the name by which a user will access a function.

Function Number Each function defined in the SDB is mapped to a natural number. All internal reference to the function is done through this number. This number will start from 1 and it is generated internally.

Security Level This is the level in the functional hierarchy in which the function belongs. This security level will be used for implementing the Authorization Model as discussed later.

Set Of Subfunctions This will contain the *Function Number* of functions on which the current function depends directly in the functional hierarchy. Clearly the subfunctions should be entered in the dictionary before.

3.2.4 Object ID

Every object type in SDB has an unique **OBJECT ID**. This OID will be later used to get information about the object types and its instances. Any query on the SDB will basically access the instances of such an object. So any user given query will be internally represented as the Object Id whose instances are to be accessed. We will discuss about the algorithm for creating a unique object id.

3.3 Algorithm

Three data are globally available by all the algorithms, they are

1. *no_of_func* This gives total number of statistical functions available in the SDB

2. *no_of_cat* This gives total number of category or both type attributes in the SDB

3. *no_of_var* This gives total number of variates or both type attributes in the SDB

In addition, all the algorithms can access the database dictionary described above.

3.3.1 Creation of Object ID

Input

- Set of Category C
- *Variate_Name*
- *Function_Name*

Output

- Object ID *oid*

BEGIN CREATE_OBJECT_ID

$func\_id \leftarrow 2^{Function\_Name \rightarrow Function\_No-1}$
$var\_id \leftarrow 2^{Variate\_Name \rightarrow Variate\_No-1}$
$cat\_id \leftarrow 0$
for each $c \in C$
begin

$cat\_id \leftarrow cat\_id \vee 2^{c \rightarrow Category\_No-1}$

end
$id \leftarrow func\_id \vee 2^{no\_of\_func} \times cat\_id$
$\vee\, 2^{(no\_of\_func+no\_of\_cat)} \times var\_id$
return id

END CREATE_OBJECT_ID

3.3.2 Find Status of a Object Type

The Object type whose instances have been already created and stored in the SDB need not be created again. So we keep a table of OID of object types, already created. The table is indexed first by the variate_number (*var_id*) then by the category set (*cat_id*) and lastly by the function_number (*func_id*) The table will only contain the OID of the object types which have been created already. OID of an object whose instances are to be created will not be in the table.

Input

- *oid*, Object ID of the Object referred by the user's query

Output

- Status of the Object Type

  Return 1 if object instances have already been created

  Return 2 if variate and category combination have been accessed previously, but not the function. So object instances have not been created

  Return 3 if variate has been accessed previously but not the category combination.

  Return 4 if variate hasn't been accessed

  For 2, 3 and 4 object instances have to be created.

BEGIN CHECK_STATUS_OF_OBJECT

```
var_id ← retrieve from oid
search OID table
if (!FOUND)
begin
    return 4
end
cat_id ← retrieve from oid
search OID table
if (!FOUND)
begin
    return 3
end
func_id ← retrieve from oid
search OID table
if (!FOUND)
begin
    return 2
end
return 1
```

END CHECK_STATUS_OF_OBJECT

Figure 3.1:

Summary Data of **SALARY** by Hobby, Sex & Age					
HOBBY	SEX	AGE	TOTAL	COUNT	MEAN
reading	male	10-20	...	...	...
reading	male	20-30	...	...	...
...	...	...	...	...	...
gossiping	female	90-100	...	...	...

3.3.3 Storage Structure of Summary Data

The summary data for each variate and unique category combination are stored in separate flat file. The filename is generated internally by concanating variate name and *cat_id*. All the summary data with same variate and category combination are stored in the same table as shown in fig-3.1. This makes the system efficient during development of functional hierarchy. Values of the lower level functions required to calculate some higher level function are available at the same row.

In fig-3.1 we give example of such a table for variate *salary*, category combination *hobby*, *sex*, *age* and functions *mean*, *total* and *count*.

3.3.4 Object Instance Creation

For each query made by the user to the SDB the following things happen,

1. The query is initially transformed into a object ID *oid*, as described above.

2. Search is made for the *oid* in the object ID table by the module *CHECK_STATUS_OF_OBJECT*.

3. If the return value by step 2 is 1, just query processing is done on the corresponding table

4. If the return value by step 2 is 2, the required summary data is calculated and the corresponding summary column is added to the existing table.

5. If the return value by step 2 is 3 or 4, the summary table is created.

Figure 3.2:

Summary Data of **SALARY** by Hobby, Sex & Age			
HOBBY	SEX	AGE	TOTAL
reading	male	10-20	...
reading	male	20-30	...
...	...	...	...
gossiping	female	90-100	...

For new object creation two modules are defined,

- *CREATE_TABLE* If summary table for a variate v does not exist for category set $\{C\}$ then this module is called. For the functional hierarchy this module will be called during creation of level 1 object, such as $< \{C\}, v, total >$, $< \{C\}, v, count >$.

  Typical structure of the summary table after creation is given in fig-3.2.

- *APPEND_SUMMARY_COLUMN* For creation of objects with - higher level functions, this module will be called. This will add corresponding summary data as a new column to the appropriate summary table.

The *CREATE_TABLE* module require to generate the combinations of the instances of $c \in \{C\}$ (the query category set). Say, $\{C\} = \{c_1, c_2\}$, the c_1 has 4 instances and the c_2 has 5 instances, then the *CREATE_TABLE* is required to generate 20 combination for the category columns of the summary table. For this purpose we use the following *CREATE_COMBINATIONS* sub-module.

Input

- The array *cat_instance*[i], itself an array of instances of the i^{th} category, terminated by NULL. Typical structure of the array is given in fig-3.3.

- *query_cat_no* The number of members in the query category set $\{C\}$.

Figure 3.3:

Category_no	Instances				
1	gossiping	stamp_collection	reading	singing	NULL
2	male	female	NULL		
3	0-30	31-60	61-90	NULL	

Output

- The combinations of the category instances.

```
BEGIN CREATE_COMBINATIONS

    for i = 1 to query_cat_no step 1
    begin

        m[i] ← 1

    end
    CF ← query_cat_no
    while( !(CF == 1 && cat_instance[CF][m[CF]] ==NULL) )
    begin

        if( CF == query_cat_no )
        begin

            /* output one possible combination */
            for j = 1 to query_cat_no step 1
            begin

                print cat_instance[j][m[j]]


            end


        end
        m[CF] ← m[CF] + 1
        if( cat_instance[CF][m[CF]] == NULL && CF !=
        1)
        begin

            m[CF] ← 1
            CF ← CF − 1
```

end
else
begin
CF ← *query_cat_no*
end

end

END CREATE_COMBINATIONS

3.3.5 Development of Category Hierarchy

We develop an algorithm to create all the lower level objects in the category hierarchy before creation of the object type mentioned in the query. This module recursively checks whether all the lower level objects exist or not. If does not exist then the object is created. If an intermediate object exist then all the sub-objects under it also exist, so the recursion stops at that level. The detail algorithm is given below.

Input

- *oid*, OID of the object, to be created if it does not exist.

BEGIN CREATE_CAT_HIERARCHY

var_bit ← *retrieve from oid*
func_bit ← *retrieve from oid*
cat_bit ← *retrieve from oid*
status ← *CHECK_STATUS_OF_OBJECT(oid)*
if(status == 1*)*
begin

return /* *object exists, no creation necessary* */
end /* *object does not exist, check lower level objects* */
for i = 1 *to no_of_cat step 1*
begin

if((2^{i} ∧ *cat_bit) !*= 0*)*
begin

```
temp ← 0
for j = 1 to no_of_cat step 1
begin
    if( i != j)
    begin
        temp ← temp ∨ 2^j
    end
end
t_cat_bit ← temp ∧ cat_bit
sub_oid ← func_bit ∨ 2^(no_of_func) × t_cat_bit ∨
2^(no_of_func+no_of_cat) × var_bit
CREATE_CAT_HIERARCHY(sub_oid)
    end
end
create the object oid /* all its sub-ordinates exist */
END CREATE_CAT_HIERARCHY
```

3.3.6 Development of Functional Hierarchy

We develop an algorithm such that if an object type is created, then all the object types for intermediate functions with same variate and category set are also created. The algorithm explicitly uses the **Function Dictionary**. The Function Dictionary for the functional hierarchy of fig 3.4 is shown in fig 3.5. In this section we describe an algorithm, such that if a request is made to create an object type, then before creation of the object, check is made whether all the objects created by the subfunctions exist or not. Those objects must be created before the creation of the required object.

Input

- *oid*, OID of the object, to be created if it does not exist.

```
BEGIN CREATE_FUNCTIONAL_HIERARCHY
    cat_bit ← retrieve from oid
    var_bit ← retrieve from oid
    func_no ← retrieve from oid
```

Figure 3.4:

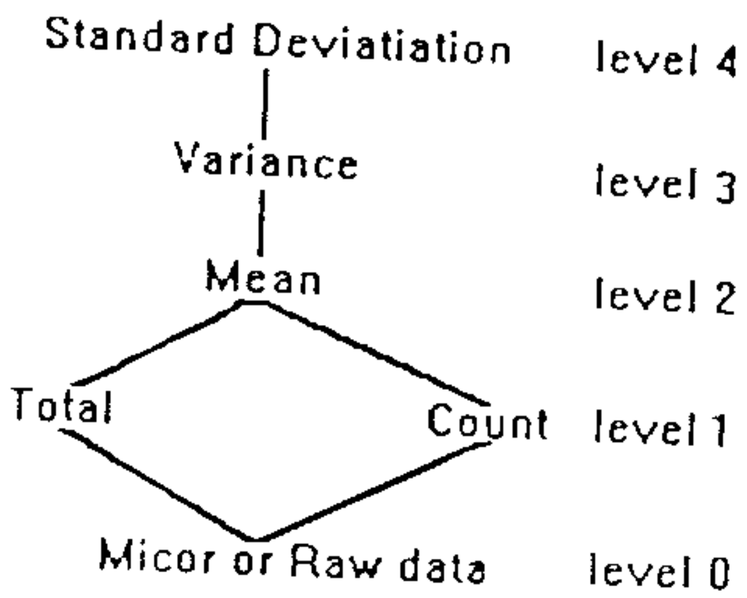


Figure 3.5:

Function Name	Function Number	Security Level	Set of Subfunctions (F)
TOTAL	1	1	ϕ
COUNT	2	1	ϕ
MEAN	3	2	{1,2}
VAR	4	3	{2,3}
SD	5	4	{4}

```
∀f ∈ func_no → F
/* F is set of subfunctions */
begin
    sub_oid ← Create from cat_bit, var_bit & f
    status = CHECK_STATUS_OF_OBJECT(oid)
    if( status != 1 )
    begin
        Create the object type sub_oid
    end
end
END CREATE_FUNCTIONAL_HIERARCHY
```

Chapter 4

Authorization Model for HISTO Model

In this section we describe the security aspect of the HISTO model as proposed by Kundu & Bagchi [6]. This autorization model controls the access rights of a user at the attribute level. Access rights to statistical object can also be restricted. We will show how the hierarchical arrangements of statistical objects provides a better prevention against database compromise.

4.1 Basic Assumptions

1. Different users may have different types of authorization.

   In a hospital a doctor may have access to the data related to individual patients, whereas a research scholar or social worker will get access to summary data only.

2. Same user may have different types of access right for different attributes.

   A user who can make queries about the Hobby of an employee, may not have access to the Salary attribute.

3. In HISTO, an attribute can be of three types - Category, Variate and Both as mentioned earlier. If a user is authorized to use an attribute as a variate, he can also access it as a category for a set of pre-specified

value ranges. However, if a user can access an attribute as a category, he may not be authorized to access it as variate as well.

4.2 Basic Definition

4.2.1 Authorization

An authorization in HISTO model is defined as a 4-tuple (u, a, fl, t) where

$$u \in U$$

is the set of users in a system. For a particular authorization 'u' may represent a single or a group of users.

$$a \in A$$

is the set of attribute in a system. For a particular authorization 'a' may represent a single attribute, a set of attributes or all the attributes in the database. 'fl' is the function level. For a particular authorization the concerned users can access all the functions at that level in the Functional hierarchy.

If 'fl' is 0, then the concerned users are allowed to access the raw data stored in the database for the attributes specified in the authorization.

't' is the attribute type which can have one of the three values, Category Variate or Both. If an authorization specifies 't' as Category, all the attributes involved in that authorization can be accessed as Category only even if some of them are originally defined as Both in the system.

Starting from a single user and a single attribute, one authorization specification can cover all the users and the attributes associated with a system.

An authorization specified as (u, a, fl, t) can have two values True or False.

4.2.2 Authorization assignment

Only DBA assigns authorization to any user or group of users. A user can not pass his authorization to any other user.

So a user or a group of users 'u' gets authorization on an attribute set 'a' with function level 'fl' and the attribute type 't' by the 4-tuple (u, a, fl, t) by the DBA.

DBA can 'Grant' authorization to or 'Revoke' authorization from any user / users against some attributes and functions.

4.2.3 Implicit Authorization

It is an authorization that can be derived from other authorizations. Some implicit authorization rules can be derived directly from the structures of Functional and Category hierarchies.

Rule-1 When DBA grants an authorization (u, a, fl, t),
'u' will have authorization on 'a' with same 't' for any functions at any levels $> fl$.
'u' will not have authorization on 'a' with same 't' for any function at levels $< fl$

So if a user has explicit authorization to access 'MEAN', he will have implicit authorization to access 'SD' or 'VAR' but he can not access 'TOTAL' or 'COUNT'.

Rule-2 When DBA grants an authorization (u, a, fl, t) for same 't' and 'fl', 'u' will have authorization for any a', where $a' \subseteq a$.

So, if a user has explicit authorization for attribute combination {C1, C2, C3}, he will have implicit authorization for the subsets {C1, C2}, {C2, C3} and {C1, C3} etc. moving upwards along the Category hierarchy till the root.

Rule-3 Let's consider that an attribute 'a' is defined as type 'Both' in the system. Now,

- If an explicit authorization has t = Both, the concerned user 'u' can access 'a' as Category or Variate.
- If an explicit authorization has t = Variate, the concerned user 'u' can access 'a' both as Category and Variate in the same object type S.

- If an explicit authorization has t = Category, the concerned user 'u' can access 'a' only as Category for a set of pre-specified value ranges even if 'a' is of type Both.

Rule-4 If an attribute 'a' is defined either as Category or Variate in the system, an authorization on 'a' will be valid if it assigns 't' with the same value as defined in the system. So, if the explicit authorization conflicts with the attribute type defined in the system, an error is flaged and the authorization is ignored. However, if 'a' is defined as Both in the system, 't' can either be Category or Variate or Both.

Rule-5 If en explicit authorization conflicts with an implicit authorization, explicit authorization will override the implicit one.

Rule-6 Let AN be the total set of authorization explicitly issued by DBA for user or a group of users in a system. The authorization of the user for a particular statistical object type will now be computed from the members of an AN and the implicit authorization derived from them.

If a user has certain authorization to access a statistical object type, he will have the same authorization to access all the instances of that type.

Rule-7 If there is an explicit authorization

$$(u, a, fl, t) \in AN$$

and there is an implicit authorization $(u, a1, fll, t)$ such that

$$(u, a, fl, t) \rightarrow (u, a1, fll, t)$$

then

$$(u, a1, fll, t) \in AN$$

This is the Nonredundancy property of authorization. So if an authorization can be implied from an explicit authorization, it can not be specified explicitly any more.

Chapter 5

Conclusion

HISTO, is a recently proposed statistical data model. Breaking the boundary of 2-D representation, this new model places a statistical object into two hierarchies, Functional and Category. These two hierarchies remove the limitations of 2-D representation and make the query processing simpler. These hierarchical structures also help in storing statistical objects created by complex statistical functions. The future works would include on development of suitable query language for accessing the SDB. In this work, we have assumed raw data to be static one. Dynamic data should also be considered. In that case the stored summary data have to be updated accordingly with the updation of raw data. Classification and aggregation hierarchies should also be considered. Implementation of authorization model is to be considered in detail. In the authorization model instance level access control can not be done, future work can be extended in this direction as well. The HISTO model needs to be extended for set and tuple type attributes.

Bibliography

[1] E.F.Codd, "A relational model of data for large shared data banks," *Communication of ACM*, vol. 13, no. 6, 1970.

[2] R. L.Lakhal and S.Miranda, "Complex statistical table structure and operators for macro statistical databases," in *Lecture notes in Computer Science, vol 367*, pp. 422–438, Springer, 1989.

[3] M.Rafanelli and A.Shoshani, "Storm : a statistical object representation model," in *Lecture notes in Computer Science, vol 420*, pp. 14–29, Springer, 1990.

[4] S.P.Ghosh, "Statistical relational tables for statistical database management," *IEEE Trans. Software Engg.*, vol. SE-12, no. 12, pp. 1106–1116, DEC-1986.

[5] S.P.Ranjan and A.Bagchi, "A statistical data model with hiearchy of statistical objects," *Proc IEEE ACE'92*, pp. 57–60, 1992.

[6] A.K.Kundu and A.Bagchi, "Authorization model for histo, a statistical database system," *Proc ACM CCS-3*, 1996.