Design of an Authorization Model for a Data Cube

A dissertation submitted in partial fulfillment
of the requirements of M.Tech.(Computer Science)
degree of Indian Statistical Institute, Kolkata
by

Pravin Kumar Singh

under the supervision of

Prof. Aditya Bagchi
CSSC

Indian Statistical Institute
Kolkata-700 108.

July 2003

Indian Statistical Institute

203. Barrackpore Trunk Road,

Kolkata-700 108.

Certificate of Approval

This is to certify that the thesis entitled **"Design of an Authorization Model for a Data Cube"** submitted by **Pravin Kumar Singh** towards partial fulfillment of the requirements for the degree of M.Tech. in Computer Science at Indian Statistical Institute, Kolkata is an acceptable work for the award of the degree.

24/7/03

Aditya Bagchi
Computer and Stastical Services Centre
Indian Statistical Institute
Kolkata-700 108.

A Bandyopadhyay
(External Examiner)

Acknowledgments

I take pleasure in thanking Prof. Aditya Bagchi for his friendly guidance throughout the dissertation period. His pleasant and encouraging words have always kept my spirits up.

I take the opportunity to thank all my classmates for all their help and suggestions. Finally I express my gratitude to my family for their encouragement to finish this work.

Pravin Kumar Singh

Pravin kumar Singh

Abstract

With the use of data warehousing and online analytical processing (OLAP) for decision support applications, new security issues arise. Traditionally data warehouses were queried by high level users (executive managers, business analysts) only. As the range of potential users with data warehouse access is steadily growing, this assumption is no longer appropriate and the neccesity of proper acess control arises. The basic aim of this work is to implement a strategy for controlling access to different aggregated views (*cubes and sub-cubes*) for different users based on their privileges for an OLAP application built over a relational system. To achieve this objective, a mechanism to specify privileges for different users on different category attributes and variates of a particular raw data table and a naming convention for aggregated views has been used.

Contents

Chapter 1

Introduction to OLAP

1.1 Data Warehouse and OLAP

Data warehousing and on-line analytical processing are essential elements of decision support which has increasingly become the focus of database industry. Decision suport places some rather different requirements on database technology compared to traditional on-line transaction processing (OLTP).

A data warehouse is a subject-oriented. integrated. time-varying, non-volatile collection of data that is primarily used in organizational decision making. Online Analytical Processing(OLAP) is a software that uses data warehouse to extract information, and to generate aggregates for trend analysis and other decision support related querying. Typically, the data warehouse is maintained separately from the organization's operational databases. There are many reasons for doing this. The data warehouse supports OLAP. the functional and performance requirments of which are quite different from those of the OLTP applications supported by the operational databases.

1.2 OLAP vs. OLTP

OLTP applications typically automate clerical data processing tasks such as order entry and banking transactions that are the bread-and-butter day-to-day operations of an organization. These tasks are structured and repetitive. and consists of short, atomic, isolated transactions. The transactions require detailed, up-to-date data. and read or update a few(tens of) records accessed typically on their primary keys. Operational databases tend to be hundreds of megabytes to gigabytes in size. Consistency and recoverability of the database are critical, and maximizing transaction throughput is the key performance metric. Data warehouse, in contrast, are targeted for decision support. Historical. summarized and consolidated data is more important than detailed, individual records. Since data warehouses contain consolidated data. perhaps from several operational databases . over potentially long periods of time, they tend to be orders of magnitude larger than operational databases(hundreds of gigabytes to terabytes). The workloads are query intensive with mostly ad hoc. complex queries that can access millions of records and perform a lot of scans, joins and aggregates. Query throughput and response times

are more important.

1.3 Multidimensional Data Model

Olap applications require viewing the data from many different business perspectives (*dimensions*). ***Data Cube*** [1] is a multidimensional view of a database where a critical value, e.g., sales, is organized by several dimensions, for example sales of automobiles organized by model, color, day of sale and so on. The metric of interest is called the *measure attribute (or variate)*, which is sales in the example. It is generally accepted that OLAP systems need to present such a multidimensinal view of the data to users, where each *cell* of the data cube corresponds to a unique set of values. As mentioned in [1] , the domain of each dimension is augmented with the special value "ALL". In order to present this multidimensional view, the data is stored in the form of "summary tables" corresponding to the subcubes of data cube. Often the dimensions are hierarchical; day of sale may be organized as a day-week-month-quarter-year hierarchy, model may be organzed as a model-category(small car segment, sports utilty vehicles etc.) hierarchy.

1.4 OLAP Operations.

Olap operations include *rollup* (increasing the level of aggregation i.e , say, moving from a subcube with aggregated data for different quarters of year to year-wise aggregated data) and *drill-down* (decreasing the level of aggregation or increasing detail) along one or more dimension hierarchies, *slice and dice* (selection and projection), and *pivot* (re-orienting the multidimensional view of data).

1.5 OLAP Security

With the increasing emphasis by business community on using historical data for decision support, OLAP applications are being used by growing number of users within an organizarion from top level managers to low level employees. The information locked in the historical and consolidated data in the data warehouse is usually valuable and sensitive. So, on one hand, the goal is to make all necessary data accessible as easily as possible to all those who legitimately seek them. On the other hand, the data must be made available to legitimate users only and only to the extent their privileges allow them. The nature of security requirements in an OLAP application is dealt in the following chapter. In chapter 3, a security mechanism developed as part of this work is explained.

Chapter 2

OLAP Security

2.1 General Security Considerations

Obviously a lot of communication takes place in a data warehouse system, creating a need for proper ***communication security*** measures. The data load process(from operational databases to the data warehouse) defines new requirements for a network infrastructure. Independent (possibly distributed) source databases have to be consolidated over a network. As the data may be highly sensitive it is essential to protect it. For the communication between front-end applications and the OLAP server usually a client/server connection will be utilized, possibly to remote sites. Even though information on this channel is most likely aggregated and less complete, it may be highly security critical. The use of the internet or other possibly insecure networks for the above mentioned connections makes suitable security measures necessary. ***Authentication*** and ***audit*** are other security measures that have to be installed in a data warehouse environment. An unauthorized user should not be allowed and a genuine user should not be denied. Corresponding user identification and authentication mechanisms check the authenticity of the pretended identity, either inside the front-end tools or by making use of the authentication mechanisms provided by modern operating systems or tools allowing a "single sign-on" strategy. ***Access control*** is another important aspect of security. According to the requirements and privilege of an user. the amount of aggregated information or the micro data (raw data) that the user can access should be controlled. Every legal user should be given only what he is entitled for.

2.2 OLAP Security Requirements and Policies

Different aplications lead to very different requirements (i.e. possible policies) for OLAP access control. The result of the reqirements analysis is high level guidelines that can be translated in a rule format, suitable for formalization and subsequent design phases. However, a proper foundation for multidimensional conceptual security modelling is not available.

Figure 2.1. defines the access control requirements.Hiding whole sub-cube is a straight forward requirement. In the context of sales of automobiles data a particular user may not be entitled to access any summarized data. Hiding certain measures (count, sum, average, standard de-

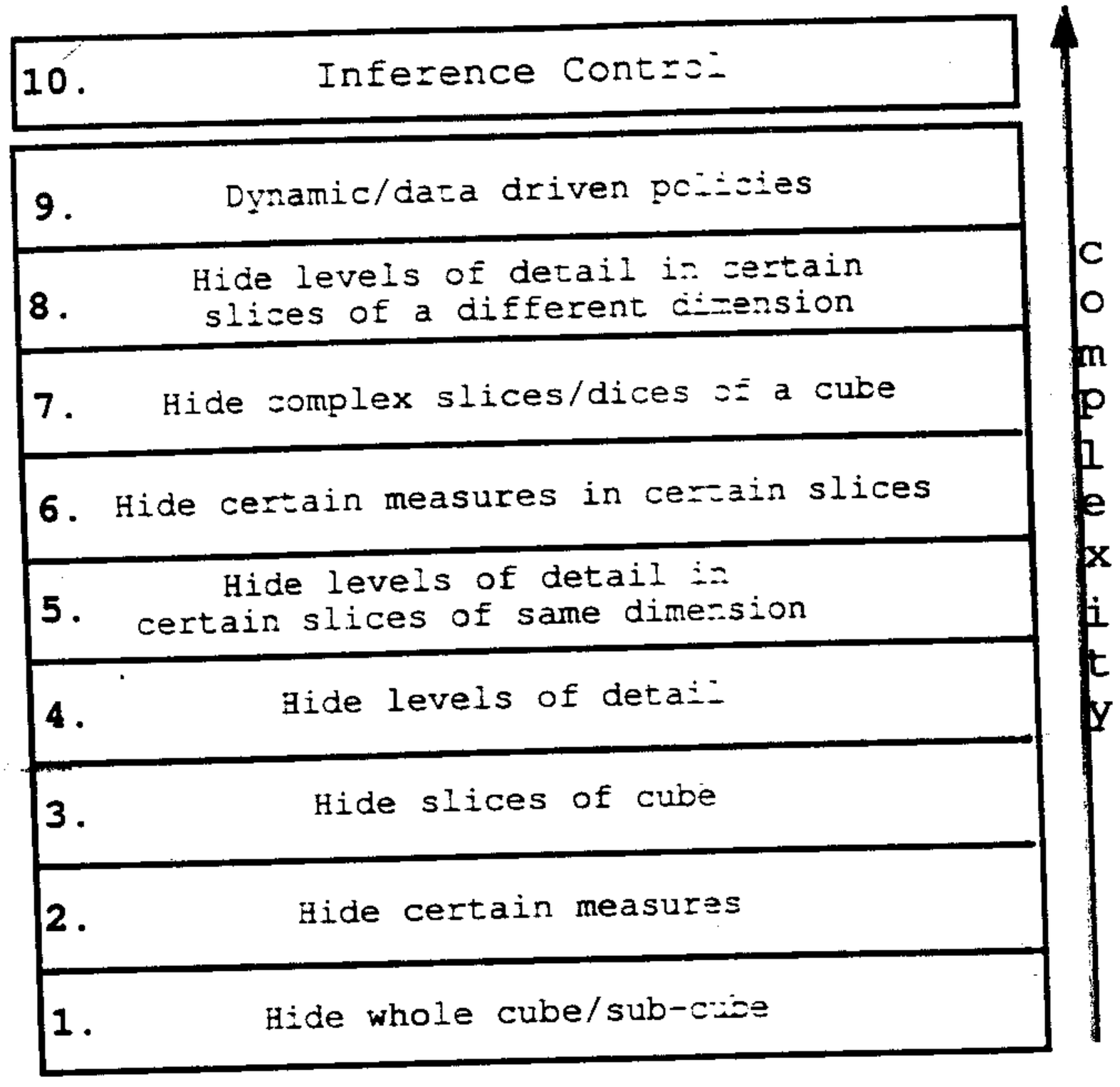

Figure 2.1: Different OLAP Access Requirements

viation, variance) would mean that a user can see only some of these aggregated measures as required by his job. Hiding slices of a cube pertaining to a particular measure means that a user can see aggregated data only for some of the category attributes. for example if a user has to analyse only variations in total sale of all vehicles year-wise. he may not be allowed to access the category attributes 'model' and 'color'. Detailed data is often considered more sensitive than summarized data. Thus it may be necessary to restrict access to detailed data below a particular dimension detail level (hiding levels of detail). For example, for the user doing analysis of year-wise sales of all vehicles. may not be allowed to access quarterly, monthly, weekly or daily sales data i.e. data may not be accessed on finer granularity than *year* on the *Time* dimension. Level 5 onwards in figure 2.1 can be considered advance requirements as they are difficult to implement. Suppose the sales of automobile example has another attribute "region" which captures the region where the actual sale was made. Then a manager of a particular region may have access to all levels of details of automobile sales data pertaining to sale made in his region but may not enjoy the same privilege for data related to other regions(hide levels of detail in certain slices of same dimension). Similarly,a regional manager may be allowed to access all

type of measures (count, sum, etc) for data related to his region but he may be restricted to see only the total sales (sum as measure) data for data of other regions. Constraints of specific application may require restricting some slices (dices) of the cube to only selected users. In dynamic or data driven policies, access rules are not defined on certain structure elements of the multidimensional data model. Access permissions depend on the data itself. For example there may be certain high ranking executives who are allowed to see details of showroom where the number of vehicles sold exceeds 5000 (assuming that the automobile sales data warehouse has information of showrooms also) . It can be done to provide special benefits to such showrooms with secrecy. As this can change with every data load, the policy is called dynamic. OLAP systems are particularly subject to inference problems as they rely on summary or aggregate data. Due to the access on such aggregate data, smart (so called tracker) queries might reveal data that is not accessible directly. Obviously the policy is that- "you cannot get indirectly what you cannot get directly". For example, if certain OLAP query retrieves a single record implying thereby access to micro or raw data, it must not be allowed even though the query was perfectly legal one.

Available security mechanisms can be distinguished on the basis that whether they use *view* or *rule* based approach. A *rule* based approach does not present (structurally) different cubes/subcubes to the users. All users know the existence of the entire cube (with all dimensions and measures), however they are not allowed to view all of it. Access rules (usually in the form of Boolean expressions) define what a user may see and what he may not see. In *view* based approach different users (or user groups/roles) see different cube/sub-cubes which are all built from the same raw or micro data. A query executed by an user is allowed or disallowed based on the fact that the view resulting from the query is accessible to the user or not.

In the present work a security mechanism for OLAP application has been implemented which has a *view* based approach. The model implemented here is able to achieve access control requirements 1, 2, 3, 6 in figure 2.1 directly. With some extension of the incorporated concepts access control requirement 4 and 5 of figure 2.1 can also be implemented.

Chapter 3

A View Based Security Model

This chapter explains the implementation of a *view* based security mechanism.

The data in the data warehouse is the raw data or the micro data. For example let us consider the automobile sales data (basically table) which captures the number of vehicle of a particular model and color sold :

CarSales(model(VARCHAR), dateOfSale(DATE), color(VARCHAR), sales(NUMBER))

At this point only, the granularity of the data is defined. For example whether each record should pertain to individual sale or it should be consolidated sales value for entire month. If the records pertain to total sales of a vehicle of particular color and model in a month, then the granularity of Time dimension is *month*. Suppose, the records here pertain to total number of vehicle sold in an year.
Consider another table :

Table2(cat1(VARCHAR), cat2(VARCHAR), cat3(VARCHAR), var1(NUMBER), cat4(NUMBER))

No specific relevance of data in this table are attached to some real world situation, but the example will help in clarifying certain concepts. Subsequently CarSales table may be referenced as "table 1" and Table2 as "table 2".

The attributes associated with the data warehouse can be either a *category attribute*, *variate*, or *both*. Category attributes basically define a multi-dimensional space and are used in the group-by clause to generate different summarized views. Variate, also called as *summary attribute*, are the attributes on which different functions (count(), sum(), average(), stddev(), var()) are applied. If an attribute is declared as both, then it can be used as a category attribute as well as a variate. Associated with every attribute is a field - *type* whose value is stored in the database in a table or in some data structure defined in the front-end application and whose value can be interpreted as:
1. A value of 0 means category attribute.
2. A value of 1 means variate.
3. A value of 2 means the attribute is both a category attribute as well as a variate .

In the examples above. cat4 is of type 2. sales and var1 are of type 1 and rest all are of type 0.

The functions that are applied to variates can be placed in a functional hierarchy (figure 3.1). The functions are numbered as : count = 1. sum = 2. average = 3, standard deviation = 4, variance = 5.

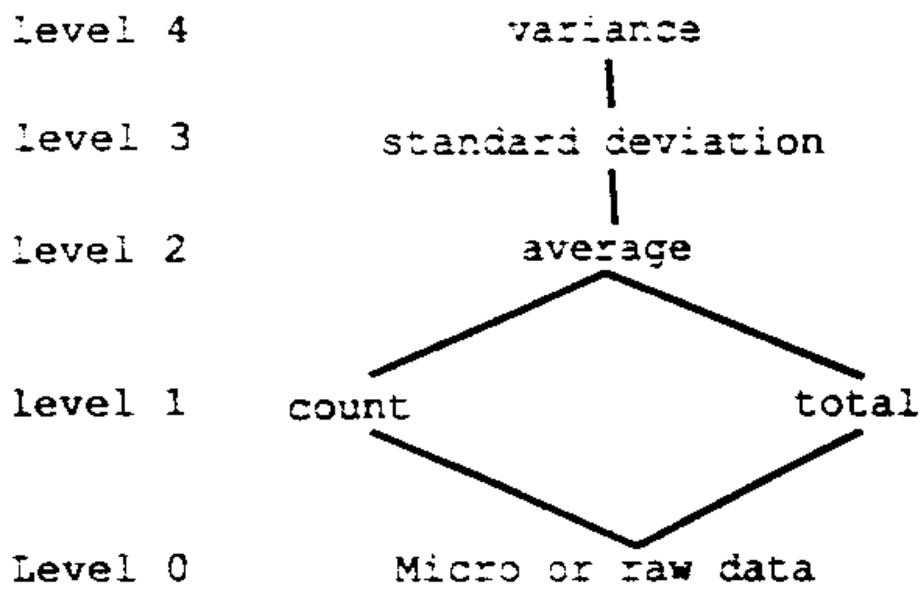

Figure 3.1: function hierarchy with function levels.

Consider that the function used is avg(), then the cube generated would be as in figure 3.2 5 such cubes will be generated corresponding to 5 different functions. For a cube corresponding to function avg(), each cell of the cube would be an average value and for a cube corresponding to the function stddev() then each cell of the cube would be a standard deviation value and so on. All the 8 Views in figure 3.2 are subcubes and the union af all these is the cube. So,if we consider the cubes generated by all the 5 functions, then we will have 8*5=40 views corresponding to CarSales table. For table Table2 if attribute cat4 is considered as a category attribute, then we will have 16*5=80 views. If attribute cat4 is considered as a variate , then we will get 8*5=40 more views. A mechanism has to be built to control access to these views!.

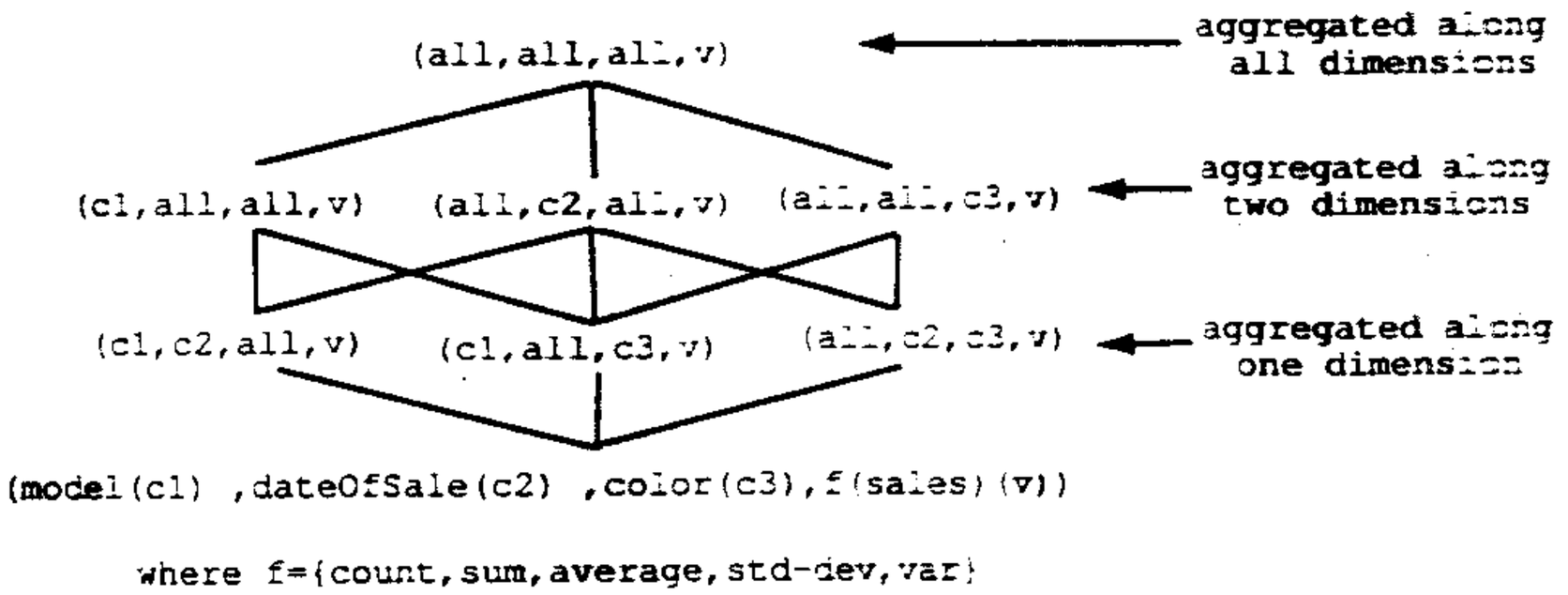

Figure 3.2: Cube (also the category hierarchy).

To facilitate the implementation of the proposed security mechanism, a naming convention for these views are adopted (figure 3.3). The name of the view obtained from table 1 is a character array of size 7 with character at each index having specific meaning as depicted in figure 3.3. An example for view obtained from table 2 is also shown in which case the name of the view is a character array of size 8. The character entry at positions pertaining to attributes position 4 to 7 in table 1 and position 4 to 8 in table 2) has the following meaning:

1. '0' means that attribute is not present.
2. '1' means that attrubute is present as a:
   (1) category attribute if the attribute type is 0
   (2) variate if attribute type is 1
   (3) category attribute if the attribute type is 2.
3. '2' means that the attribute is present as a variate (applicable only for attribute of type 2).

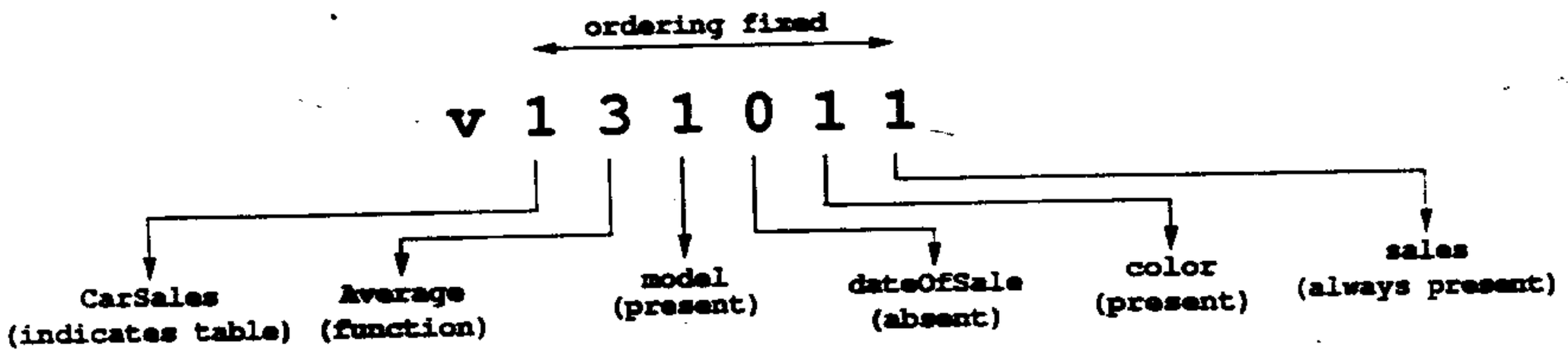

TABLE 1

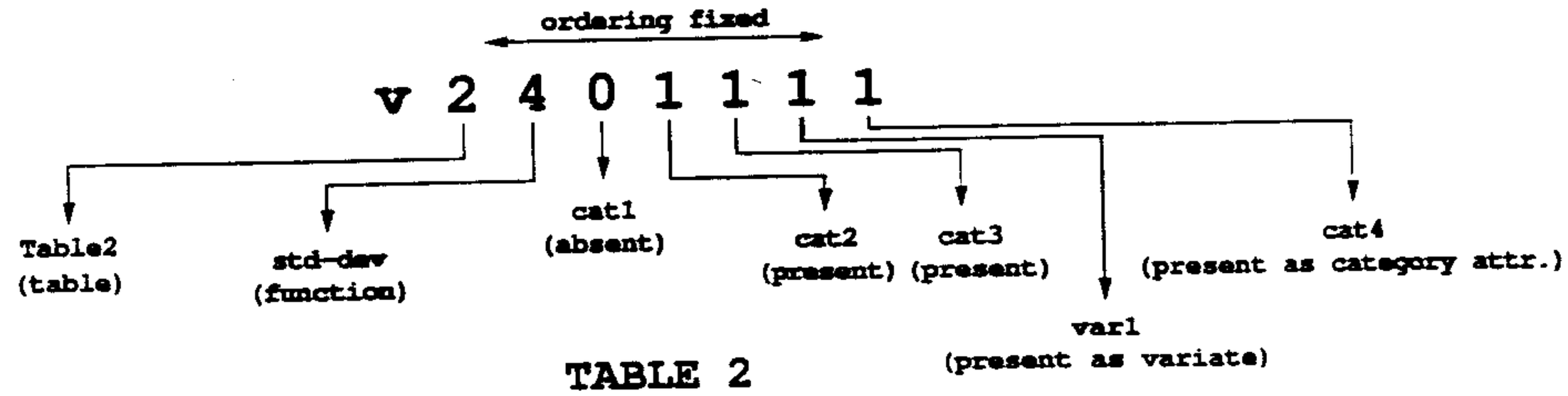

TABLE 2

Figure 3.3: View nomenclature.

The privilege(s) for each user is specified table wise. figure 3.4 is self suggestive.

After getting acquainted to view nomenclature and the way to define user privilege(s), the ***rules*** laid down for security can be looked at:

1. An user can use an attribute only if he has explicit privilege to use it. He can use the attribute only as specified by the corresponding privilege.
2. If an user has fuction level 3 as his privilege, then he can access measures obtained using function of level 3 or higher. He cannot access function level below his specified privilege.

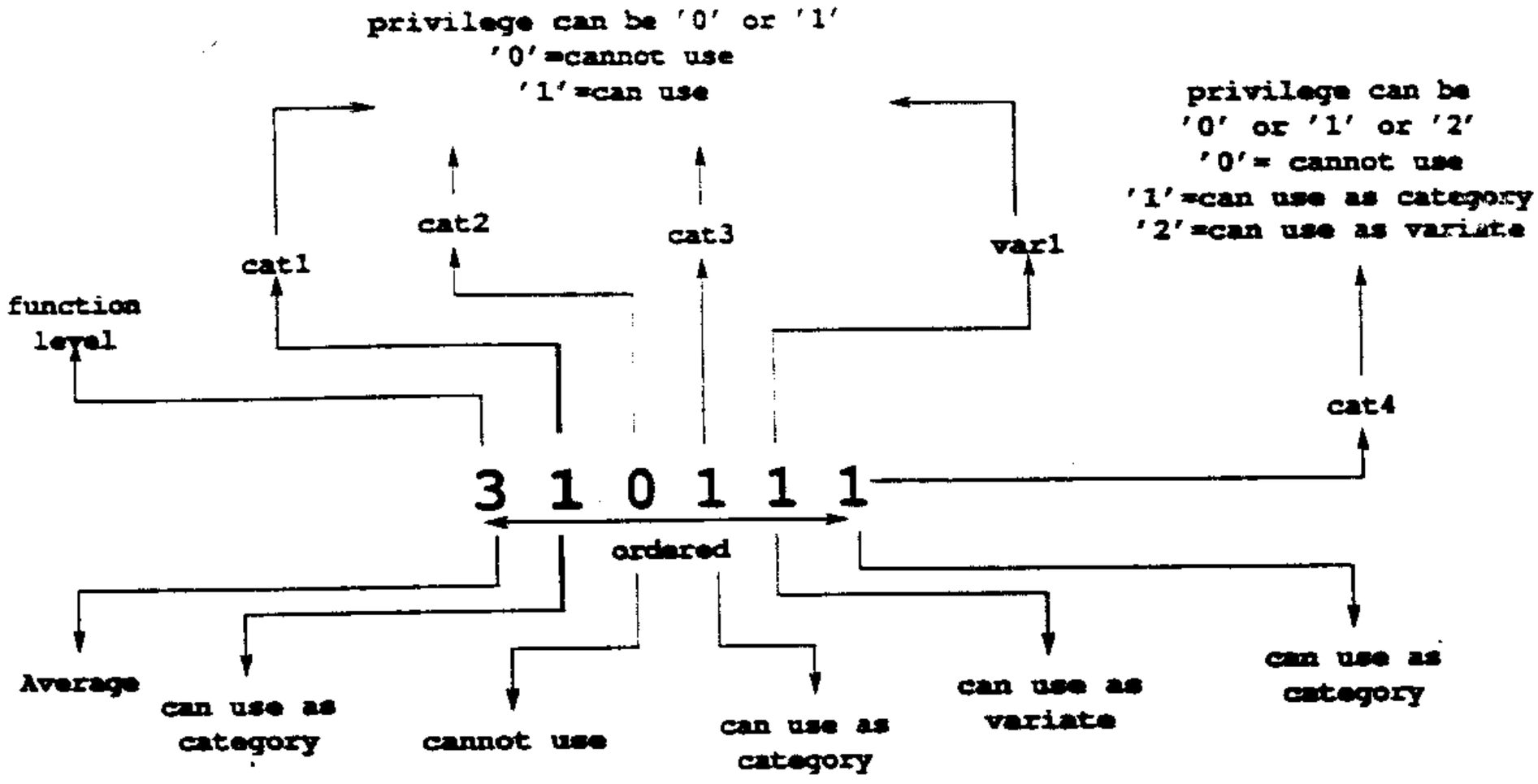

Figure 3.4: User privilege for table 2

We can say that measure obtained using function level 4 and 5 are implicit authorization.

3. An user can see a view with a subset of category attributes that he is entitled to use.
4. If an user cannot use any variate, then he cannot see any of the views.
5. Suppose a user has for table 1 (i.e. CarSales) a privilege of "40001" (i.e. cannot use any category attribute, but can use the variate), then he can see the two views "v140001" (which has only one record - (all,all,all,std-dev(sales))) and "v150001" (which has also only one record-(all,all,all,var(sales))).
6. If a user doesn't have the permision to use any variate, then he cannot access any view.

The logic for allowing an user to access the function of higher levels is that the aggregates using higher level of function can be calculated from those that the user is entitled for. For example, if the user knows the average, he can calculate the standard deviation (This assumes that the user knows sum and count when he calculates average). However this may not be the case if the application uses direct functions sum(), count(), avg(), stddev() and var() available with almost every database product. This can lead to even stricter control that the user can access only the views that has as its measure the value calculated using the function to which he is entitled. Instead of checking that the function used in view is either of the same level or higher than the user function privilege level, now it will be checked whether the two are equal.

Implementation

A OLAP application was made with C and Oracle9i to test the security system. Whenever a new user is added, his/her privilege(s) corresponding to each attributes are specified table-wise and he/she is alloted a user-id and a password. This information can be either stored in a table created in the database or in the front-end. Every request for seeing a view passes through the

application build in C. No one is allowed to directly enter the database except the DBA. When a user logs in. the application checks his/her user-id and password. If the user is a genuine one, then he/she is allowed to make query. It should be noted that at this stage the privilege(s) of the user "logged in" is known (has been retrieved based on user-id). The user is now prompted for selecting a particular table, say. either CarSales or Table2 . After this. the user is prompted to select the category attributes from the list displayed. Since the table of interest is known, the category attributes of that table are displayed. After this the user is prompted to select a variate followed by function. Suppose the user selects table 1, selects as category attribute model and color, selects sales as variate and var() as function. Then the view that he is looking for is "v151011" Now the application checks whether as per the rules specified above, he/she can access this view or not. The view is retrieved only if the user's privilege(s) allows it. Here it can be observed that view "v151011" is a subset of the view "v151111" in the sense that "v151011" can be obtained from "v151111" through proper *select* clause. So, if the view "v151111" is already present, then the view asked for is retrieved from it, or else the asked view is first created and then retrieved.

The important task is to generate the view asked for after proper verification dynamically. Adopting a naming convention helps in identifying the view easily. Using the naming convention, finding that a particular view can be obtained from another view through some proper *select* statement also becomes easier. But the difficult part is to dynamically generate the view(s). Appendix A lists down the sql statements that need to be run to generate different views.

The model presented in this report can also be extended to hide levels of details along one dimension hierarchy. To explain this. let us consider the example of *Time* dimension and assume it to be organized in a day-week-month-quarter-year hierarchical fashion. The view names depicting these levels of details must be identified. To do so. instead of the view name having only the characters '0' and '1' at the position corresponding to *Time* dimension (*Time* is a category attribute), it can now have other values with following meanings:

'0' means *Time* dimension not present.
'1' means *Time* dimension present with details upto day level.
'2' means *Time* dimension present with details upto week level.
'3' means *Time* dimension present with details upto month level.
'4' means *Time* dimension present with details upto quarter level.
'5' means *Time* dimension present with details upto year level.
Obviously the correspondence between these numbers and the hierarchy level should be stored in a table like:

Day	Week	Month	Quarter	Year
1	2	3	4	5

Similar entries for each category attribute organized in a hierarchical fashion should be made. Then with the help of this table, the view asked for can be easily inferred.

Appendix A

Generating Views

For view **"v141011"** the sql command would be:

```
SELECT      DECODE(GROUPING(model),1,'all',model) as model_name,
            DECODE(GROUPING(color),1,'all',color) as color_of_vehicle,
            STDDEV(sales) as total_sales
FROM        CarSales
GROUP BY    CUBE(model,dateOfSale,color)
ORDER BY    model,dateOfSale,color;
```

For view **"v130111"** the sql command would be:

```
SELECT      DECODE(GROUPING(dateOfSale),1,'all',dateOfSale) as date_of_sale
            DECODE(GROUPING(color),1,'all',color) as color_of_vehicle,
            AVG(sales) as total_sales
FROM        CarSales
GROUP BY    CUBE(model,dateOfSale,color)
ORDER BY    model,dateOfSale,color;
```

For view **"v2111111"** the sql command would be:

```
SELECT      DECODE(GROUPING(cat1),1,'all',cat1) as category1,
            DECODE(GROUPING(cat2),1,'all',cat2) as category2,
            DECODE(GROUPING(cat3),1,'all',cat3) as category3,
            DECODE(GROUPING(cat4),1,'all',cat4) as category4,
            SUM(var1) as total_sales
FROM        Table2
GROUP BY    CUBE(cat1,cat2,cat3,cat4)
ORDER BY    cat1,cat2,cat3,cat4;
```

Oracle9i directly provides **cube()** operator which was introduced in [1]. Similarly, Oracle9i also provides **rollup()** function. Using these the views can be created easily.

Bibliography

[1] Jim Gray, Adam Bosworth, Andrew Layman, Hamid Pirahesh
Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab and Sub-Totals,Research Report. IBM Almaden Research Center, San Jose, California, 1995.

[2] Surajit Chaudhuri and Umeswar Dayal
An Overview of Data Warehousing and OLAP Technology, ACM SIGMOD Record 26(1), March 1997.

[3] Rakesh Agrawal, Ashish Gupta, Sunita Sarawagi
Modelling Multidimensional Databases,Research Report, IBM Almaden Research Center, San Jose, California, 1995.

[4] Aloke K. Kundu and Aditya Bagchi
Authorization Model for HISTO, a Statistical Database System, Invited paper to ACM Computer Communication Security Conference, 1996.

[5] Sunita Sarawagi, Rakesh Agrawal, Ashish Gupta
On Computing the Data Cube, Research Report 10026, IBM Almaden Research Center, San Jose, California, 1996.

[6] Venky Harinarayan, Anand Rajaraman. Jeffrey D. Ullman
Implementing Data Cubes Efficiently, Proc. ACM SIGMOD '96, 205-216.

[7] Torsten Priebe and Gunther Pernul
Towards OLAP Security Design - Survey and Research Issues, In Proc. DOLAP'00.