Conceptual Level Design of Object Oriented Data Warehouse: Graph Semantic Based Model

ANIRBAN SARKAR[1]
SANKHAYAN CHOUDHURY[2]
NABENDU CHAKI[2]
SWAPAN BHATTACHARYA[1]

[1] National Institute of Technology, Durgapur, West Bengal,India
[2] Department of Computer Science, University of Calcutta, Kolkata, India
[1]sarkar.anirban@gmail.com
[2]sankhayan@gmail.com
[2]nabendu@ieee.org
[1]swapan.bhattacharya@nitdgp.ac.in

Abstract. This paper proposes a Graph semantic based Object Oriented Multidimensional Data Model (GOOMD). The model defines a set of graph based formal constructs that are used to specify the conceptual level design of data warehouses (DW). Besides offering a novel graph based semantic for multidimensional data representation, the proposed GOOMD model renders a unified realization of different logical structures of DW like star, snowflake, constellation, etc. The proposed simple but powerful algebra for GOOMD provides an efficient operational model for Online Analytical Processing (OLAP) to realize different OLAP operations like roll-up, drill-down, slicing, dicing, drill-across and drill-through. The proposed GOOMD model also supports a formal representation of advanced concepts like parallel hierarchies, many-to-many associations, additivity constraints etc.

Keywords: Data warehouse Conceptual modeling, Multidimensional data model, Graph data model, Object Oriented Data Model.

(Received June 15, 2009 / Accepted September 23, 2009)

1 Introduction

A data warehouse (DW) is typically used to facilitate complex, sophisticated, online and multidimensional analysis of data by fetching just-in-time information from multidimensional databases. It is a subject oriented, integrated, consolidated, non-volatile, historical collection of data in support of management decision making process. The analysis performed is referred to On Line Analytical Processing (OLAP). For the purpose of data analysis form DW, the notion of the Cube has been widely accepted as the underlying logical structure of DW or multidimensional databases.

DW design framework should span in three perspective namely, the *conceptual* one, dealing with the high level representation of the world in order to capture the user ideas using rich set of semantic constructs, the *physical* one, dealing with the details of the representation of the information or data storage techniques in the specific database management system, and the *logical* one, which acts as an intermediate between the two aforementioned extremes, trying to balance a storage independent paradigm and a natural representation of the information in terms of computer-oriented concepts.

This paper has proposed Graph Object Oriented Multidimensional Data Model (GOOMD), a novel graph based semantic with simple but powerful algebra for multidimensional data model to conceptualize the

multidimensional database structure and operational model for OLAP. The model is revealed a set of concepts to the conceptual level design phase of DW, which are understandable to the users, independent of implementation issues and provide a set of constructs along with a set of operations to facilitate the designers of DW. Beside the basic concepts, the proposed GOOMD model also offers a formal representation for advanced concepts of DW like parallel hierarchies, many-to-many associations, additivity constraints etc. The proposed model also renders a unified representation to apprehend different logical schemas of data warehouse like star, snowflake, constellation schema etc. Finally, the GOOMD model also provides the formal specification of OLAP algebra that will realize different OLAP operations like roll-up, drill-down, slicing, dicing and drill-across operation.

This work is motivated from Graph Data Model (GDM) [3]. Since, GDM is an effort to design and develop a next generation platform for Online Transaction Processing (OLTP) system, with the aim of extracting the positive features of both Object and Relational data models besides providing the structured graphic representation for interacting to the designer. The preliminary version of this work is published in [17], which has been now enriched and completed with comprehensive formalization of different concept of multidimensional data model.

2 Related Work

Nevertheless, there exists a scope towards developing a formal multidimensional data model at conceptual level. In [22] a detail survey shows that upto late 90's more effort has been given on developing the logical models of OLAP using relational and cube oriented approach.

Published models or design methods, very often, encompass only some phases of data warehouse design. In these previous research works, the design level abstractions covered by the method namely the conceptual, logical and physical phases. Some approaches are dedicated to the conceptual phase only [4, 21, 8, 12, 10, 9, 20, 2], others draw a linkage between the conceptual and logical phases and others concentrate on a unique conceptual-logical-physical phase for data warehouse design [7, 14, 6]. In the majority of research the underlying conceptual model is mapped to relational model for logical design phase. The relational model has serious deficiencies in many aspects [19, 5].

Some approaches [1, 11, 18, 16, 13, 23] has been used the Object Oriented Paradigm based on the concept of Unified Modeling Language (UML) to describe the design phases of data warehouse. Some of the proposals within these has described the conceptual design phase of data warehouse using multidimensional meta-model. The proposals based on a combination of multidimensional meta-models with ER or UML concepts are often associated by the poor semantics for the constructs of the basic multidimensional model.

Few approaches [16, 23] has been used the concept of Model Driven Architecture (MDA) and Object Constraint Language (OCL) to describe the different concepts in the design phases of data warehouse. The proposals mainly have been focused on the structure of multidimensional data. The Common Warehouse Metamodel (CWM) [15] is a recent endeavor to standardize data warehousing and business intelligence applications based on UML. It provides a framework for representing metadata about data sources, data targets, transformations, analysis, the processes and operations for the purpose of creating and managing warehouse data. However, CWM is oriented towards generalization of logical specification and not conceived as conceptual model.

A majority of the approach has not handled the dynamic aspect of the data warehouse design as well as semantic enriched detailed operational model for OLAP. Both of these are extremely important for exhaustive data analysis and multidimensional data visualization.

3 Proposed GOOMD Model

The GOOMD model is the core of the comprehensive object oriented model of a DW containing all the details those are necessary to specify a data cube, a description of the dimensions, the classification hierarchies, a description of the context of analysis i.e. fact and the quantified attributes of facts i.e. measures. The model also renders the formal concept of Cube to conceptualize the multidimensional data visualization and operational model for OLAP.

In this context, a *fact* represents the subject of analysis in some business context, consisting of measures and other contextual data element e.g. Total Sales. *Dimensions* are the parameters over which the fact will be analyzed using OLAP. For example, to analyze all sales of products from DW, common dimensions could be: Time, Location/region, Customers, Sales person etc. One can arrange the members of a dimension into one or more hierarchies, called *dimension hierarchies*. Each hierarchy can also have multiple hierarchical levels. Hierarchy in dimensions also exhibits the granularity in data

elements. *Granularity* of data in the DW is concerned with the level of detail available in the data elements of dimension. The more detail data is available through the lower level of data granularity. A *measure* is an attribute of a fact, representing the performance or behavior of the business relative to the dimensions. For example, measures are the sales in money, the quantity supplied, and so forth. Facts are used to analyze based on the measure attributes, particularly through aggregation function (summation, counting average etc.).

The proposed data model allows the entire multidimensional database to be viewed as a Graph (V, E) in layered organization. At the lowest layer, each vertex represents an occurrence of an attribute or a data item, e.g. product name, day, customer address etc. Each such basic attribute is to be represented as separate vertex. A set of vertices semantically related is grouped together to construct an Elementary Semantic Group (ESG). So an ESG is a set of all possible instances for a particular attribute or data item. On next, several related ESGs are group together to form a Contextual Semantic Group (CSG) - the constructs of related data items or attributes to represent one business item. The edges within CSG are to represent the association between different ESG in the said CSG. The most inner layer of CSG is the construct of highest level of granularity in the business data in multidimensional database formation. This layered structure may be further organized by combination of one or more CSGs as well as ESGs to represent next upper level layers and to achieve further lower level granularity of business data. From the topmost layer the entire database appears to be a graph with CSGs as vertices and edges between CSGs as the association amongst them.

3.1 Component of GOOMD Model

Since from the topmost layer, a set of vertices V is decided on the basis of data granularity whereas the set of edges E is decided on basis of the relation between different semantic groups. The basic components for the model are as follows,

(i) A set of *t* distinct *attributes* $A = \{a_1, a_2, \cdots, a_t\}$ where, each a_i is an attribute or a data item semantically distinct.

(ii) A set of *k* measures $M = \{m_1, m_2, \cdots, m_k\}$ where, each m_k is a measure name related to some fact.

(iii) *Elementary Semantic Group (ESG):* An elementary semantic group is an encapsulation of all possible instances or occurrences of an attribute or measure, can be expressed as graph ESG (V, E), where the set of edges *E* is a *null set* $\emptyset$ and the set of vertices *V* represent the set of all possible instances of an attribute or a measures $x_i \epsilon A \cup M$. Henceforth, there will be set of *t+k* ESGs and can be represented as $E_G = \{ESG_1, ESG_2, \cdots, ESG_{t+k}\}$. The graphical notation for the any ESG is *circle*.

(iv) *Contextual Semantic Group (CSG):* A contextual semantic group is an encapsulation of two or more related ESGs to represent one elementary context of business analysis. A CSG is the construct for different level of granularity of the contextual data and use to exhibit respective level of details in Multidimensional database formation. Let, the set of *n* CSGs can be represented as $C_G = \{CSG_1, CSG_2, \cdots, CSG_n\}$. Then any $CSG_i \subseteq C_G$ can be represented as a graph (VC_i, EC_i) where vertices $VC_i \subseteq E_G$ and the set of edges EC_i represents the association amongst the vertices. The graphical notation for any CSG is *square*.

(v) *Dimensional Semantic Group (DSG):* A dimensional semantic group is a specialized form of CSG. A DSG is an inheritance or composition of one or more DSGs of adjacent inner layer along with encapsulation of one or more related ESGs. The lowest layer DSG in the dimension hierarchy will exhibit the high level of granularity in multidimensional database and will be constituent of ESGs only. The higher layer DSG in the dimension hierarchy is the combination of one or more ESGs and DSGs of adjacent inner layer. So a DSG is a graph (V_D, E_D) where $V_D \subseteq E_t^+ \cup D_G^*$, D_G^* is the set of DSGs of adjacent inner layer with zero or more occurrences and E_t^+ is the set of ESGs defined on attributes with one or more occurrences. For any DSG, it is also possible that, determinant vertex may determine an unordered set of instances of one or more encapsulated ESGs or DSGs. The graphical notation for the former type of association is *solid undirected edges* and *dashed undirected edges* for later type of association.

(vi) *Fact Semantic Group (FSG):* A fact semantic group is another specialized form of CSG. A FSG is an inheritance of all related topmost layer DSGs as well as encapsulation of a set of ESGs defined on measures. One FSG can be represented as a graph (V_F, E_F), where $V_F \subseteq DET(D_G) \cup$

$ESG(M)$ and E_F is the association amongst the vertices. $DET(D_G)$ is the union of determinant ESGs of all related topmost layer DSGs and ESG(M) is the elementary semantic group defined on all related measures. The FSG is used to represent the root of the graph semantic based GOOMD model schema. Also a GOOMD model schema may have multiple root FSGs with shared topmost layer DSGs.

(vii) *Cube:* In order to materialize the cube, one must ascribe values to various measures along all dimensions and can be created from FSG. It facilitates multi-dimensional data visualization, navigation and analysis. A cube can be represented using a graph (V_{CI}, E_{CI}), where the set of edges E_{CI} is a null set $\emptyset$ and a set of vertices $V_{CI} = ESG(m_1) \times ESG(m_2) \times \cdots \times ESG(m_j)$. A cube can be materialized from multiple FSGs with shared top level DSGs. For schema containing multiple FSGs with shared DSGs, the DSG set $\{D_1, D_2, \cdots, D_p\}$ are the common set of DSGs for all FSGs in the schema.

(viii) *Association and Links:* A set of undirected edges are declared between determinant ESG and encapsulated ESGs as well as encapsulated CSG to address the association between member vertices of one CSG. These associations within any CSG can be of two type, (i) solid undirected edges to represent bijective mapping between determinant vertex and any other member vertex, and (ii) dashed undirected edges to represent one to many mapping between determinant vertex any other member vertex. A set of directed edges are declared between adjacent inner layer parent DSGs and inherited DSG, and also between topmost layer DSGs and a FSG to address the relationships between the constructs for dimensions and fact. These edges are called links. Links between topmost layer DSGs and FSG can be of two type, (i) directed link from DSGs to FSG to represent the one to many associations and (ii) bi-directed link from between DSGs to FSG to represent the many to many associations.

3.2 An Example

This section will provide an example to clarify the component definitions of GOOMD. The example is based on Sales Applications with Amount as measure and with four dimensions - Customer, Model, Time and Location. Model, Time and Location dimensions have upper level hierarchies as Product, QTR and Region respectively. Also, PDESC is a complex structure, which is a constituent CSG encapsulated in Product DSG. So, there will be four DSGs $D_{Sales} = \{D_{Customer}, D_{Model}, D_{Location}, D_{Time}\}$ with hierarchy. The Product DSG $D_{Product}$ is comprised of ESGs like PID, PNAME and PDESC and will be represented as the inner layer of the graph. The Model DSG D_{Model} is inherited from $D_{Product}$. The $D_{Product}$ and D_{Model} DSG graphically can be represented as Figure 1.

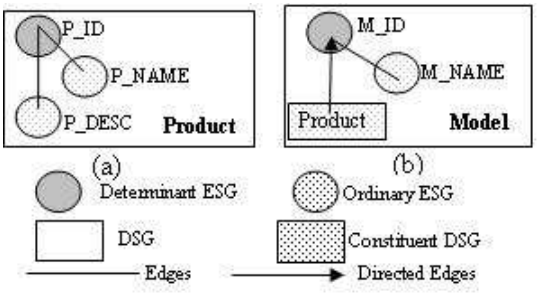

Figure 1: (a) Lower Level DSG, (b) Higher Level DSG

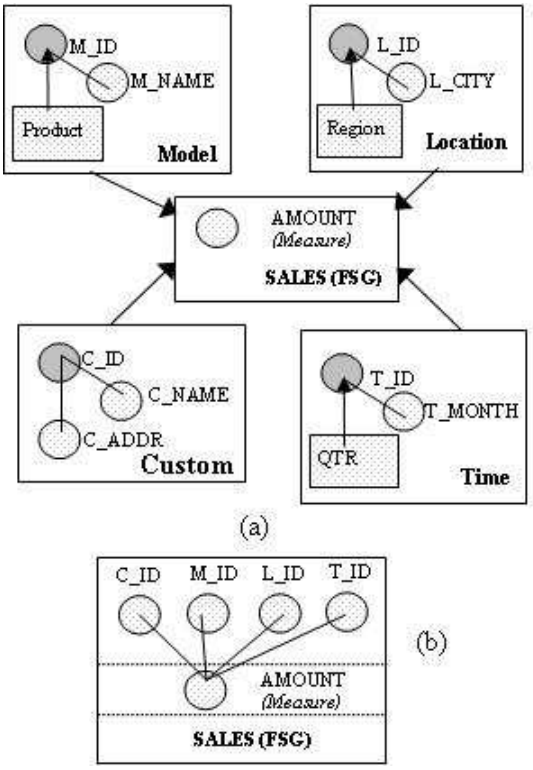

Figure 2: (a) Schema for Sales Application in GOOMD Model (b) SALES FSG construct after inheritance

The FSG for the DW can be described as $F_{Sales} = \{DET(D_{Customer}), DET(D_{Model}), DET(D_{Location}), DET(D_{Time}), E_{AMOUNT}\}$. Where E_{AMOUNT} is the ESGs defined on the measure Amount and $DET(D_{Customer}), DET(D_{Model}), DET(D_{Location}), DET(D_{Time})$ are the determinant vertices, inherited from topmost layer of associated DSGs. The schema is shown in Figure 2. The base cube C or the cube with low granularity on all dimension can be described as $C = \{E_{AMOUNT}, \emptyset\}$ with functional constraint $f : D_{Customer} \times D_{Model} \times D_{Location} \times D_{Time} \rightarrow E_{AMOUNT}$.

3.3 The Hierarchical View of GOOMD Model

The hierarchical views of the above described Sales Application in GOOMD notation has shown in Figure 3. The Sales application in hierarchical view consists of three layers. Lowest layer is *ESG Layer*, which is collection of all ESGs Defined on attributes or measures. Constituent DSGs are placed in intermediate layer i.e. *DSG Layer*. DSGs with different granularity level for any Dimension Level may be placed in different layer and so there may exists multiple DSG layers. The *Top Most Layer* consists of DSGs with lowest level granularity and FSGs. For the Sales Application, there are four DSGs and one FSG.

Different *Dimension Levels* in the Figure 3 shows the hierarchy of different dimensions, each consists of DGSs and their corresponding constituent DSGs. For the Sales Application, there are four *Dimension Levels*. ESGs outside of any Dimension Levels are the ESGs defined on different measures. In Figure 3 only one such ESG exist, that is defined on measure amount. As discussed earlier, the Cube with lowest level granularity or highest detail of data can be materialized from the top most layer of the GOOMD model.

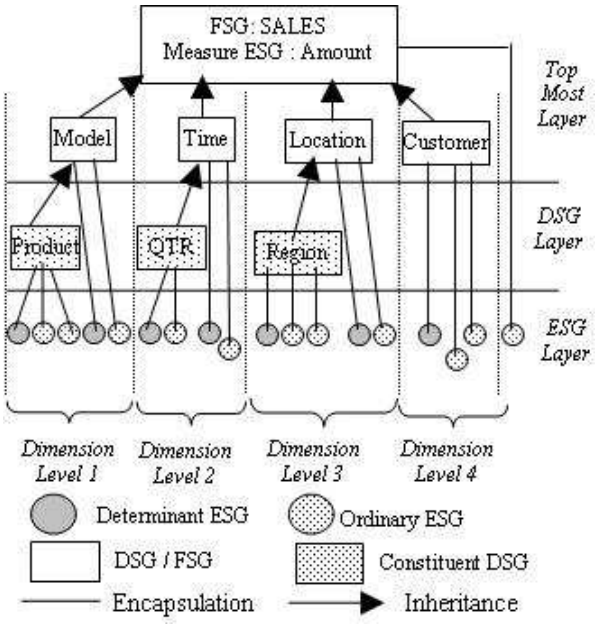

Figure 3: Hierarchical View of GOOMD Model

4 Proposed OLAP Algebra for GOOMD Model

In this section we will define the OLAP algebra for GOOMD model that will operate on Cube concept and/or several constructs described in the model. The algebra consists of a set of operators and few of them also can be used with the constructs like ESG, DSG and FSG. The running example of *Sales Application* will be used to illustrate the functionalities of operators.

4.1 Proposed GOOMD Operators for OLAP

(i) *dSelect* (π): The operator will extract vertices from some dimension semantic group or some elementary semantic group defined on some measure, depending on some Predicate P. P can be an atomic prediction, denoted as p or it can be a composite predicate, denoted as $p_1 < op > p_2 < op > \cdots < op > p_n$, where <op> is a logical operator (e.g. AND, OR). The algebraic notation of the operator is $\pi_{\emptyset}(D) = D_O$, where D is the original DSG or ESG on measure and the D_O is the output DSG or ESG on measure after the restriction using predicate P. Also for null prediction the operator will return the original semantic group. Hence $\pi_{\emptyset}(D) = D$.

(ii) *Retrieve* (σ): The Retrieve operator extracts vertices from the cube C using some constraint *CON* over one or more DSGs or ESGs defined on measures. The output for the operation will be another resultant cube C_O. The constraint *CON* will be in the form, *CON* = ($\pi_1(D_1)$<op>$\pi_2(D_2)$ <op> $\cdots$ <op> $\pi_j(D_j)$) *AND* ($\pi_{j+1}(E_{m1})$<op> $\pi_{j+2}(E_{m2})$<op> $\cdots$ <op> $\pi_{j+k}(E_{mk})$) where, any $\pi_i (1 \leq i \leq k)$ is a logical expression or a predicate over the DSG D_i or a predicate over the ESG on some measure m_i and <op> is a logical operator (e.g. AND, OR). The algebraic notation of the operator is $\sigma_{CON}(C) = C_O$.

(iii) *Aggregation(α and $+\alpha$)*: The Aggregation operators perform aggregation on one or more DSG vertices and will operate on the measure ESG of the base cube *C*. The output of the operators will be another cube C_O. The aggregation

will be based on the relational aggregation function like Sum, Average, Count etc. and the *Additivity_Constraint* specification of the measure ESGs. The algebraic notations of the operators are $\alpha_{F,m,DS}(C) = C_O$ and $+\alpha_{F,m,DS}(C) = C_O$, where F is the relational aggregation function which will operate over measure m and the set of lowest layer DSGs DS.

Normally, the operator will perform the aggregation function *F* for measure *m* on each outer layer DSGs of the specified set of DSGs. Whereas $+\alpha$ operator will perform the aggregation function *F* for measure *m* on each outer layer DSGs including the specified set of DSGs and also persist the output of the intermediate operations. Since these realize another important OLAP operation *Roll - Up*. Also the output of corresponding operation related to the $+\alpha$ operation is useful to perform the level wise expansion of Cube for further lower level granularity.

(iv) *Zoom-in Operator (β)*: The Zoom-in operator performs cube expansion on one or more DSG vertices and will operate on measure ESGs of the apex cube Ca generated using Aggregation operator $\alpha_{F,m,DS}(C) = C_O$, where *C* is the base cube. Since, $+\alpha$ operator will perform the aggregation function *F* for measure m on each outer layer DSGs including the specified set of DSGs and also persist the output of the intermediate operations i.e. cubes with lower level granularities. Now, *Zoom-in* operator will traverse down by one level through the specified dimension hierarchies to achieve the cube with next lower level granularity. The algebraic notations of the operator is $\beta_{DS}(C_a) = C_O$, where C_a can be achieved from $+\alpha_{F,m,DS}(C) = C_O$ and *DS* is the set of lowest layer DSGs over which the operators will operate. This will realize the important OLAP operation *Drill-Down*.

(v) *Union, Intersection and Difference Operator ($\cup$, $\cap$ and -)*: The operators will find Union, Intersection and Difference of two cubes. The algebraic notation for the operators is $C_1 \oplus C_2 = C_O$, where C_O is the output cube and the symbol $\oplus$ should be replaced by $\cup$ for Union operation, $\cap$ for Intersection operation and - should be replaced for Difference operation. The operations can be performed iff both the cube C_1 and C_2 are related with identical set of DSGs and Measures.

Let two cubes are C_1 with a set of DSG D_1, a set of Measures M_1 and a set of Vertices V_1 and C_2 with a set of DSG D_2, a set of Measures M_2 and a set of Vertices V_2. Then the output cube C_O will be comprised of set of vertices $V_O = V_1 \oplus V_2$, set of DSGs $D_O = D_1 = D_2$ and set of measures $M_O = M_1 = M_2$.

(vi) *Cartesian Product ($\times$)*: It is a binary operator to relate any two cubes. The algebraic notation of the operator is $C_1 \times C_2 = C_O$, where C_O is the output cube. Let two cubes are C_1 and C_2. Also let, a set of DSG $D1 = \{D_{11}, D_{12}, \cdots, D_{1P}\}$ and a set of Measures $M_1 = \{M_{11}, M_{12}, \cdots, M_{1Q}\}$ is related with cube C_1 and a set of DSG $D_2 = \{D_{21}, D_{22}, \cdots, D_{2R}\}$ and a set of Measures $M_1 = \{M_{21}, M_{22}, \cdots, M_{2S}\}$ is related with cube C_2. Then the output cube C_O will be comprised of a set of vertices $V_O = ESG(m_{11}) \times ESG(m_{12}) \times \cdots \times ESG(m_{1Q}) \times ESG(m_{21}) \times ESG(m_{22}) \times \cdots \times ESG(m_{2S})$. The cube C_O will also obey the functional restriction $f : D_{11} \times D_{12} \times \cdots \times D_{1P} \times D_{21} \times D_{22} \times \cdots \times D_{2Q} \rightarrow V_O$.

(vii) *Join ($\bowtie$)*: The Join operator is a special case of Cartesian Product operator. It is also a binary operator to relate two cubes where one or more identical DSGs in common. Since two DSGs are identical iff the constituent CSGs or ESGs are identical for both the DSGs. The algebraic notation of the operator is $C_1 \bowtie C_2 = C_O$, where C_O is the output cube. Let, two cubes are C_1 and C_2. Also let, a set of DSG $D_1 = \{D_{11}, D_{12}, \cdots, D_{1P}\}$ is related with cube C_1 and a set of DSG $D_2 = \{D_{21}, D_{22}, \cdots, D_{2R}\}$ is related with cube C_2. Then the Join operation between C_1 and C_2 is possible iff $D_1 \cap D_2 \neq \emptyset$. Since, $\cap$ is the *Intersection* operator used in the context of DSGs, which will extract vertices from two set of DSGs $D_A(V_A, E_A)$ and $D_B(V_B, E_B)$ to form an another set of DSGs $D_C(V_C, E_C)$, such that these vertices belong to both D_A and D_B i.e. $D_A \cap D_B = C$ where, $V_C = \{x : x \epsilon V_A \text{ AND } x \epsilon V_B\}$. Now let $D_1 \cap D_2 = \{D_X, D_Y, D_Z\}$ then the *Join* operator can be expressed as, $C_1 \bowtie C_2 = \sigma_{CON}(C_1 \times C_2)$, where, *CON* is equal to $((C_1.D_X = C_2.D_X)$ AND $(C_1.D_Y = C_2.D_Y)$ AND $(C_1.D_Z = C_2.D_Z))$

Further, the operators like *Union, Intersection, Difference* and *Cartesian Product* can operate on any ESG also. In that case the operators will extract the set of vertices from the ESGs on which it is operating and will form another related ESG, without changing the meaning of the operator as defined above.

4.2 Realization of Advanced Operations of OLAP

(i) *Drill-Across Operation*: Since, in GOOMD model concept, a Cube can be materialized from multiple FSGs with shared DSGs where multiple measures attributes can be taken from different FSG. Further using separate Retrieve operations on the same Cube but different CON specification with separate ESGs on measure, one can change the cell value in multidimensional space without changing the associated top level dimension classes. This will realize the *Drill-Across* operation of OLAP, which is implicit in the Cube concept of GOOMD Model. Also there is no restriction to define a derived FSG by inheriting the existing base FSG with new set of ESGs defined on measures. By materializing a Cube on new FSG, one can perform the *Drill-Across* operation between new set and existing set of measure attributes.

(ii) *Drill-Through Operation*: This is a specific operation of OLAP, whereby the user views the detail data pertaining to a high level construct from a given dimension hierarchy. For example, in Sales Application, for the Cube defined FSG *Sales*, user may intend to view the detail of *Products*, involved in each transaction of some specific Qtr. Since, the GOOMD Model concept is based in Object Oriented Paradigm, where inner layer DSGs are either inherited to or encapsulated to upper layer DSGs. Hence DSG of any inner layer is accessible from its correspondence top most layer DSG as well as Cube. Further dSelect operator defined in GOOMD model can also operate on DSG Construct and can be used to view the detail of some specific DSG. An illustration of *Drill-through* operation has been described in example *Query 2* of sub-section 4.4.

4.3 Properties of Proposed Set of Operators

Several operators for GOOMD model have been proposed in previous section. Those operators are used to operate on different constructs defined for GOOMD model like ESG, DSG, FSG etc. Further several operators are used to operate on cube concept as defined in the model, for the purpose of OLAP. To realize the OLAP operations both basic and advanced, *four* major operators are defined in GOOMD model, namely, *dSelect(π), Retrieve(σ), Aggregation(α) and Zoom-in(β)*. The summary of different OLAP operations and corresponding GOOMD Model operators are given below in Table 1.

There are several interesting properties can be defined for the proposed OLAP algebra for GOOMD model based on Table 1 and the definition of dSelect, Retrieve, Aggregation and Zoom-in operators.

Lemma 1: The Set of GOOMD Operators $\{\pi, \sigma, \alpha, \beta\}$ *are complete.*

Proof: In Table 1, a mapping between the GOOMD operators and OLAP operations has been demonstrated. We find that several the standard OLAP operations including Drill-Across and Drill-Through can be performed only by using *dSelect (π), Retrieve (σ), Aggregation (α) and Zoom-in(β)* operators defined insection 4.1. Thus it may be inferred that the set of GOOMD operators $\{\pi, \sigma, \alpha, \beta\}$ are complete with respect to OLAP operations as mentioned in Table 1.

Lemma 2: The Set of GOOMD Operators $\{\pi, \sigma, \alpha, \beta\}$ *are independent.*

Proof: It is evident from the Section 4.1 and Table 1, that none of the operator can be expressed in terms of others. The proposed *dSelect (π), Retrieve (σ), Aggregation (α) and Zoom-in(β)* operators are defined independently for different purpose to perform analytical tasks on OLAP specific contents. The $+\alpha$ operator could be considered as the special form of *Aggregation (α)* operator with the option to persist the output of the all successive level aggregation operations. The significance of $+\alpha$ *operator* has been discussed in section 4.1

Lemma 3: The Set of GOOMD Operators $\{\pi, \sigma, \alpha, \beta\}$ *are minimal.*

Proof: The minimal property of selected GOOMD operators may be proved by using the method of contradiction. We would assume one operator at a time as redundant and would attempt to validate the assumption.

Firstly, say *dSelect (π)* operator is redundant and may be dropped from the set of operators without affecting the OLAP operations. Now from Table 1, it is evident that Drill-through operation of OLAP can not be performed if we drop dSelect (π) operator. Henceforth it is contradictory to the assumption that dSelect (π) operator is redundant.

In the same way if we assume one at a time that Retrieve (σ), Aggregation (α) and Zoom-in (β) operators are redundant then we cannot perform the OLAP operations Slice, Dice and Drill-Across, Roll-Up and Drill-Down respectively.

Since $+\alpha$ operator is a special form of Aggregation (α) operator, used aggregate the data on the dimension hierarchies with the options to persist the output of the all successive level aggregation operations. Thus is also useful to perform the level wise expansion of

Table 1: OLAP Operations and Corresponding GOOMD Model Operators

$OLAPOperations$	$GOOMDModelOperators$	$Notation$	$Syntax$
Slice	Retrieve	σ	$\sigma_{CON}(C) = C_O$,Where $CON = \pi_{\emptyset}(D_1)$ AND $\pi_{\emptyset}(D_2)$
Dice	Retrieve	σ	$\sigma_{CON}(C) = C_O$, Where $CON = \pi_{p1}(D_1)$ AND $\pi_{p2}(D_2)$ AND $\cdots$ AND $\pi_{pk}(D_k)$ and $P_i \neq \emptyset$ for $1 \leq i \leq k$ where $k \leq n$
Rollup	Aggregation	α	$\alpha_{F,m,DS}(C) = C_O$, Where *F* is relational aggregation function *SUM*
Drill-Down	Zoom-in	β	$\beta_{DS}(C_a) = C_O$, where C_a can be achieved from $+\alpha_{F,m,DS}(C)$
Drill-Across	Retrieve	σ	$\sigma_{CON}(C) = C_O$, Where separate *CON* specification with separate ESGs on measure related to separate FSGs will yield separate cell value corresponding to specific FSGs
Drill-Through	dSelect	π	$\pi_P(D) = D_O$, where *P* should be equated to the corresponding Determinant ESG.

Cube for further lower level granularity. The Zoom-in (β) operator can only be operable after the operation of $+\alpha$ aggregation operator as it stores all the intermediate resultant cubes. So dropping the $+\alpha$ aggregation operator will affect the *Zoom-in(β)*.

Henceforth we may conclude that the set of GOOMD Operators $\{\pi, \sigma, \alpha, \beta\}$ including the special aggregation operator $+\alpha$ is minimal.

Lemma 4: The GOOMD Operators $\{\pi, \sigma, \alpha, \beta\}$ are closed.

Proof: As given the definitions of proposed operators in section 4.1 and Table 1, each operator operate on some valid construct or on cube as defined in GOMMD model concept. Further those operators yield the same type of constructs or cube respectively as the result. Thus it may be inferred that the set of GOOMD operators $\{\pi, \sigma, \alpha, \beta\}$ are closed.

5 Features of the GOOMD Model

The proposed GOOMD model is a comprehensive object oriented model of a DW viewed as a Graph (V, E) in layered organization and contains all the details those are necessary to specify the artifacts of DW system as well as it is enriched with simple but powerful algebra for multidimensional data model and operational model for OLAP. Apart from these, one of the major advantages of the model is that it defines each level of structural and operational detail on the constructs which are independent of the OLAP context. Also the graph structure maintains the referential integrity inherently. The features of the proposed model are as follows,

(a) *Explicit Separation of structure and Content*: The model provides a unique design framework for DW with the detail definition of different level (from elementary to composite) of data structure using graph. The model reveals a set of structures like ESG, CSG, DSG and FSG along with a set of relationships like association, inheritance, and encapsulation between the structures, which are not instance based or value based. So, the nature of contents that corresponded with the instances and the functional constraint on the instances has been separated from the DW structural description. The topmost layer view will hide the detail structural complexity from the users. Such a representation is highly flexible for the user to understand the basic structure of data warehouse and to formulate the necessary queries. The referential integrity between the different constructs in the structure is inherent for the model.

(b) *Measures Set and Dimensions Set*: The model has defined the measures and dimensions as different elementary constructs of the DW. Also, the model supports the features of derived measure or derived dimension by using the *dSelect* operator. Thus, the model provides a *symmetric* treatment of dimensions and measures.

(c) *Association and Links*: An association is a struc-

tural relationship that specifies how the elements of one CSG are related between themselves and also how encapsulated CSGs are related with parent CSG. A link is structural relationship that specifies how different CSGs are related between themselves. Graphically a link is a directed edge to exhibit the inheritance as structural relationship between inner layer DSG to upper layer DSG, and also between DSGs and FSG. The links can normally be viewed at the topmost layer of graph.

(d) *Inheritance*: Inheritance from a CSG reflects specialization of the parent semantic group in the model. Any DSG exhibit the higher granularity is normally inherited from its inner layer DSG. The former one reflects the specialization and the later reflects generalization. Since the topmost layer DSG is representing most specialized dimension.

(e) *Parallel Hierarchies*: Two different constituent DSGs in same layer connected through link with the parent DSG will be considerer as alternative path through the DSG level or dimension hierarchy.

(f) *Aggregation*: An aggregation reflects a *whole/part* relationship. From the structural point of view of the model any CSG i.e. DSG or FSG represent a larger thing (the *whole*), which consists of smaller things (the *parts*), i.e. ESGs and / or constituent CSGs. Graphically, this aggregation is a square region marked for the CSG consisting of ESGs or constituent CSGs. The *aggregation* operators described above will use the aggregated structure of DSGs for the important OLAP operation *Roll-Up*.

(g) *Many-to-many relationships between Fact and Dimension*: There is no constraint forbidding this in GOOMD as there is a possibility of bi-directed link between topmost layers DSGs to FSG, to represent the many - to - many associations. Since, the relationship between DSGs and FSG always should be one-to-many but there is a possibility of having many-to-many association between DSGs and FSG for a particular analysis need. In this case, a new DSG need to be introduced with a new attribute, which will also encapsulate the original DSG that is exhibiting many-to-many relation. The new attribute in new DSG will act as determinant vertex and will determine an unordered set of instances of encapsulated DSG uniquely. The new DSG will exhibit one-to-many relation with FSG.

(h) *Additivity Constraints*: In DW design, the additivity is a constraint specification over the summarization function, which is associated with measure. To represent the additivity constraints in GOOMD model, each measure ESG is tagged with *Additivity Constraint* property in the construct of relevant FSG and which will affect the functionalities of Aggregation Operators (α and $+\alpha$). Further, GOOMD model provides a set of certain semantic behavior of *Additivity Constraint* to realize the non-additive, semi-additive and fully-additive nature of measures in DW.

(i) *Unified Representation of Logical Structures*: In GOOMD model concept there is no binding on existence of multiple FSGs and shared DSGs in one schema. Further, FSG are used to represent the root of the graph semantic based GOOMD model schema and a schema with multiple roots is valid structure in the model. The difference between *Star Structure* and *Snowflake Structure* will appear with top level view and hierarchical view representation respectively of the conceptual multidimensional schema. Also the *Constellation Structure* can be realized using multi rooted GOOMD model schema with shared topmost layer DSGs.

6 Conclusion

In this paper, an approach has been introduced for the data warehouse conceptual design phase using graph based semantics. This is a comprehensive object oriented conceptual model of a data warehouse and viewed the entire multidimensional database as a Graph (V, E) in layered organization. It also contain all the detail constructs those are necessary to specify the facets of DW system as well as it is enriched with simple but powerful algebra for OLAP to conceptualize the multidimensional data visualization and operational model for DW system. We have also demonstrated the features and the capabilities of the proposed model by providing examples of typical OLAP queries expressed by our algebra. The proposed model also facilitates a unified realization of different logical structure of data warehouse like star, snowflake, constellation etc. Further the layered organization of the model facilitates to view the data warehouse structure from different level of abstraction.

The graph based semantics in GOOMD model extracts the positive features of both Object and Relational data models and also it maintains the referential integrity inherently as in Graph Data Model (GDM) [3]. The GOOMD model proposed in this paper, extent the concepts of GDM for dual implications, which would formally realize both OLTP system and OLAP system. Any schema non-rooted graph

semantic consisted with the ESG and CSG constructs only along with the proposed set of association types can apprehend the schema suitable for OLTP system. Also several proposed operators like *dSelect, Retrieve, Union, Intersection, Difference, Cartesian Product and Join* can work on the schema suitable for OLTP system and without any further modification.

Finally, the set of constructs along with a set of operators for the GOOMD model provides better understandability to the users, independency from the implementation issues and high flexibility to the designers for creation and / or modification of DW structure at conceptual level.

References

[1] Abelló, A., Samos, J., and Saltor, F. Yam2: A multidimensional conceptual model extending uml. *Information Systems*, 31(6):541–567, 2006.

[2] Binh, N. T. and Tjoa, A. M. Conceptual multidimensional data model based on object-oriented metacube. *ACM Symposium on Applied Computing*, pages 295–300, 2001.

[3] Choudhury, S., Chaki, N., and Bhattacharya, S. Gdm: A new graph based data model using functional abstraction. *Journal of Computer Science and Technology*, 21(3):430–438, 2006.

[4] Datta, A. and Thomas, H. The cube data model: A conceptual model and algebra for on-line analytical processing in data warehouses. *Decision Support Systems*, 27:289–301, 1999.

[5] E., S. M. Information systems: Records, relations, set, entities and things. *Information Systems*, 1(1):3–13, 1975.

[6] Franconi, E. and Kamble, A. A data warehouse conceptual data model. *16th Int. Conference on Scientific and Statistical Database Management*, 2004.

[7] Franconi, E. and Kamble, A. The gmd data model and algebra for multidimensional information. *16th International Conference on Advanced Information Systems Engineering (CAiSE-04)*, 2004.

[8] Franconi, E. and Sattler, U. A data warehouse conceptual data model for multidimensional aggregation: a preliminary report. *Italian Association for Artificial Intelligence*, 1:9–21, 1999.

[9] Golfarelli, M., Maio, D., and Rizzi, S. The dimensional fact model: A conceptual model for data warehouses. *Cooperative Information Systems*, 7:215–247, 1998.

[10] Hahn, K., Sapia, C., and Blaschka, M. Automatically generating olap schemata from conceptual graphical models. *3rd ACM Int. Workshop on Data warehousing and OLAP*, pages 9–16, 2000.

[11] Hüsemann, B., Lechtenbörger, J., and Vossen, G. Conceptual data warehouse design. *2nd Int. Workshop on Design and Management of Data Warehouses*, 2000.

[12] Lechtenbörger, J. and Vossen, G. Multidimensional normal forms for data warehouse design. *Information Systems*, 28:415–434, 2003.

[13] Luján-Mora, S., Trujillo, J., and Song, I.-Y. A uml profile for multidimensional modeling in data warehouses. *Data and Knowledge Engineering*, 59(3):725–769, 2006.

[14] Malinowski, E. and Zimányi, E. Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data and Knowledge Engineering*, 59(2):348–377, 2006.

[15] OMG. Modeling and metadata specifications. *Available at Object Management Group webSite*.

[16] Prat, N., Akoka, J., and Comyn-Wattiau, I. A uml-based data warehouse design method. *Decision Support Systems*, 42(3):1449–1473, 2006.

[17] Sarkar, A. and Bhattacharya, S. The graph object oriented multidimensional data model: A conceptual perspective. *16th International Conference on Software Engineering and Data Engineering*, pages 165–170, 2007.

[18] Schneider, M. A general model for the design of data warehouses. *Int. Journal of Production Economics*, 112(1):309–325, 2008.

[19] S.Kunii, H. Graph data model and its data language. *SpringerVerlag*, 1990.

[20] Tryfona, N., Busborg, F., and Christiansen, J. G. B. starer: a conceptual model for data warehouse design. *2nd ACM Int. workshop on Data warehousing and OLAP*, pages 3–8, 1999.

[21] Tsois, A., Karayannidis, N., and Sellis, T. K. Mac: Conceptual data modeling for olap. *Book Title: "Design and Management of Data Warehouses"*, 2001.

[22] Vassiliadis, P. and Sellis, T. A survey of logical models for olap databases. *SIGMOD Record*, 28(4):64–69, 1999.

[23] Zepeda, L. and Celma, M. A model driven approach for data warehouse conceptual design. *7th Int. Baltic Conference on Databases and Information Systems*, pages 114–121, 2006.