International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:1, 2007

A Conceptual Query-Driven Design Framework for Data Warehouse

Resmi Nair, Campbell Wilson, and Bala Srinivasan

***Abstract*—**Data warehouse is a dedicated database used for querying and reporting. Queries in this environment show special characteristics such as multidimensionality and aggregation. Exploiting the nature of queries, in this paper we propose a query driven design framework. The proposed framework is general and allows a designer to generate a schema based on a set of queries.

***Keywords*—**Conceptual schema, data warehouse, queries, requirements.

I. Introduction

A data warehouse is considered as a centralized repository which provides an integrated view of enterprise data. Analysts and decision makers query data warehouses to discover information about their business and this leads the decision making process. Data warehouse users like to view data in a multidimensional space which is close to the analysts way of thinking [1]. Likewise users are interested in trends rather than a single transaction which means data summarisation is required. Summarising data to different levels of details is called aggregation. Due to these properties data warehouse queries are generally known as multidimensional aggregate queries [9] [2]. To support this class of query, existing designs mainly adopted two types of approaches, namely; top-down and bottom-up. The top down designs define multidimensional space through facts and dimensions. A fact can be defined as an item or subject of interest for an enterprise and which is the subject of analysis in a multidimensional space. Fact is described through a set of attributes (typically numerical but not necessarily) called measures [17]. Dimensions represent the different points of view of data for the analysis or it can be defined as the context for analysing the facts. Aggregation is addressed through hierarchies. That is, dimensions are defined as hierarchies having different levels. Each level in a hierarchy represents a level of detail of data (granularity) required by desired analysis [21].

Bottom-up designs suggest methods to transform conceptual schemas of operational systems to a data warehouse model. In most cases it is assumed that the operational schema is ER and the transformation is suggested accordingly. Conceptualisation is not addressed in these approaches and also the designs are data driven.

In both top-down and bottom-up approaches real users queries are neglected during the design process in order to offer maximum flexibility for ad-hoc queries. We feel that this is a restrictive approach and queries need to be considered even though all the queries are not known at the beginning. A design should provide guidelines to streamline the model as per specific schema criteria. Another issue in existing designs is that each model has their own concepts and notations. So it is difficult to identify a general model from the existing ones.

When comparing data warehouse design with the traditional database design, the transformation from a model to a schema as well as schema refinement is not well addressed. In the light of this situation, a general framework, which is systematic as well query driven, is necessary. In this paper we propose such a framework.

The remaining part of this paper is organised as follows: Research related to data warehouse design is reviewed in Section II. Based on that discussion, the proposed framework is presented in Section III and each step associated with the framework is detailed in subsequent subsections. Finally the paper is concluded in Section IV.

II. Related Works

Design methodologies related to the system design could be found in [18] and [12]. These methodologies addressed conceptual to physical designs. In terms of conceptual design, there are approaches such as [17], [7], [3]. These top-down designs discussed requirements related to data representation and aggregation and suggested new models. The definition of modeling constructs as well as graphical notations could also be found here. The new models are general in terms of implementation. Nonetheless, guidelines on schema derivation as well as the involvement of queries in design are lacking.

Another set of works, such as [16], [8] and [15], provide methods to transform an ER model to a data warehouse model. The main focus in this case is identifying modeling constructs from ER model for a data warehouse model. These include schema refinement in certain cases.

Authors are with Monash University, Melbourne, Australia (e-mail: Resmi@beast.infotech.monash.edu.au).

There are query based approaches in [22] and [10]. Cube design based on queries is presented in [22]. An algorithm to select sub-cubes from a base cube is the main contribution of this work. The sub-cubes are selected with respect to queries. However the work insists queries are in MDX form, (MDX is a query language for multidimensional database designed by Microsoft) which is very restrictive. Requirement based data cube design is proposed in [10]. This approach identifies data cubes that need to be pre-computed in order to improve query performance. This work is an optimisation technique rather than a schema design method.

An XML data warehouse called X-Warehouse is proposed in [14] based on frequent query patterns. In this method, historical queries are transformed to query path transactions. From a set of query path transactions a database is created. Then, by applying a data mining technique, significant query patterns are discovered. The main focus of this work is the mining algorithm to find frequent query patterns instead of a complete schema design methodology. Based on the discussion presented here it is concluded that a design framework which is systematic, general, and query driven is necessary. The main focus of this paper is such a framework with the above mentioned properties.

III. The Proposed Framework

As in the traditional database design ([4]), our framework also starts with requirement specification. Since there are well- stated requirement analysis techniques like [5] and [20], we avoid a discussion on requirement collection. Instead assume, we have the necessary business related requirements and user queries. After collecting the requirements, business related needs are formalised in the form of a graph which is known as knowledge base (Refer Fig. 1). While the existing approaches use ER and OO concepts for the formalisation we choose a graph theory approach. This allows us to keep the framework as general as possible.

The knowledge base in the framework provides the initial knowledge required for a data warehouse schema. Since this is a query oriented approach, users may not be able to provide all the underlying relationships in a business. Normally user's business knowledge is restricted to specific business units. In the absence of a knowledge base, the user has to provide all the necessary relationships through the queries. This makes query construction difficult and formal representation of queries is required. The introduction of the knowledge base avoids such a scenario and users are allowed to present natural language queries.

Using the queries and the knowledge base, a query-oriented schema is generated in the second phase of the design. This schema is called the intermediate schema which is a graph representation similar to the knowledge base except to the fact that all the collected queries are clearly included. Intermediate schema acts as a common platform from where different data warehouse schemas can be derived. We have identified the importance of an information model in the design process from [13] and this leads to the addition of the intermediate schema in the framework. This schema provides the possibility of various schemas at the database level and a designer has the flexibility to choose a schema on demand. As the final step in the framework, the derivation of a data warehouse schema is presented which is further described as an optimisation at the conceptual level. In fact this could be seen as a conceptual reduction rather than a physical optimization. Each step involved in the framework is explained in detail in the following subsections.

A. Knowledge Base

As stated earlier, the knowledge base is a formalisation of business requirements which are:

Business measures: The business measures are defined as quantifiable standards used to evaluate the business performance [19]. That is, the strategic success of an organisation is measured through a set of standards. These standards determine whether the objectives are met and the strategies are implemented successfully. Examples of business measures from retail environments are annual sales, customers lost, gross margin, total assets, etc. We formalize business measures in terms of measure types. A measure type is defined as a class of business objects and each instance or a business object in a measure type is called a measure. The state of each business object in the class is of interest to the analyst and we assume it can be quantified. If a measure type is derived from other measure types then it is called a derived measure type and all others are basic. Every measure type is represented by a node in the graph.

Classification: Management typically structures their business in a hierarchical form. These business structures are accomplished by classifying the organisation, products and/ or services etc. Classification is defined as a grouping on objects based on certain criteria. A graph oriented formalisation of classification could be found in [6]. We follow that approach here as it suits our graph representation. Classification is further explained as: each group in a classification is called a class and any two classes are related by means of a relationship called classification relation. A class can be further described by means of attributes called descriptive attributes.

Classification is formally defined as a tuple (C; R) where C is a set of distinct classes and R is a subset equals $C \times C$ is a classification relation. Since business structures are generally hierarchical the classification relation is reflexive, transitive and antisymmetric. The classification concept is later used in the schema for aggregation. So the formalisation should address the conditions for correct aggregations, as discussed in [11]. To that extent, the definition of the classification schema is

specialized to a lattice structure called classification lattice. A classification lattice in a classification schema will have a greatest lower bound and least upper bound. The least upper bound is indicated by the node All_c and a greatest lower bound is the node where there is no incoming classification relation to it. (Readers are advised to refer [6] for a detailed discussion on classification)

By defining different relations between the classes in the classification and the measure types, these two concepts can be integrated to produce a graph called knowledge base graph. An example knowledge base graph from a retail case study is presented in Fig. 1. In this figure, Sales is the measure type and classification lattices based on day, store, product and customer are also shown along with the respective semantic relationships.

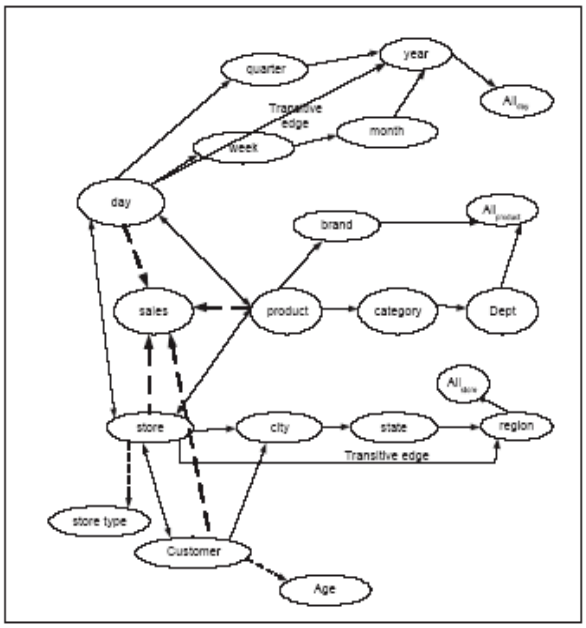

Fig. 1 The Knowledge Base Graph

Formally the knowledge base graph G_k is defined as: $G_k = (N_k, E_k)$ where N_k is a set of nodes of type measure types, classes and descriptive attributes. E_k is a set of edges represent various semantic relations that exists between the nodes. For example, in the figure, a thin directed arrow represents a member of classification relation whereas a thick dashed arrow stands for the relation between a class and a measure type. This graph is used as a starting point for the schema generation which also requires queries.

B. User Queries

The queries collected from the users are in natural language form. Including natural language queries as it is in the framework complicates the schema design, since processing natural language query is not easy. So we translate every natural language query to a formal representation which is known as a query tree.

A query tree, equivalent to a natural language query is defined as a tree with root being a measure type and successive nodes can be a class or a descriptive attribute from a classification. Additionally, a tree is allowed to have only one measure type. Then the leaf nodes in the tree are always nodes other than a measure type.

An edge in a query tree represents a function called requirement function. That is, a requirement function R_q can be written as:

R_q is a subset of $\{ f_1, f_2, \ldots, f_i\}$ where each f_i can be any function such as statistical functions like sum, average, minimum, maximum or functions that apply constraints. They may also represent combinations of more elementary functions. An example query tree for the query sales in quarter1, year 1999 is shown in Fig. 2. One query can be represented in more than one way and the framework does not offer any restriction on the representation. Any one of the possible representations can be taken for a query.

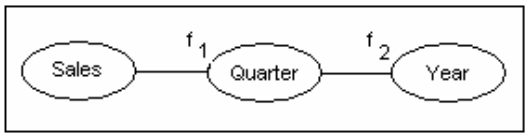

Fig. 2 An example query tree

Although certain queries are known initially, it can not be guaranteed that all queries are taken into account for the design process. To overcome this limitation, a query taxonomy which incorporates general data warehouse queries is suggested. For this classification of queries we use the semantic constructs and operations present in a query. As this is a conceptual approach, a set of operators are assumed to be part of each query type and for the classification only the query constructs are considered.

A measure type is present in every query; it can be basic or derived. Similarly the classes present in the query may or may not be from the same classification lattice. Taking these aspects into account queries can be classified as one of: basic measure type- single-classification, derived-measure type-single classification, basic measure type-multiple classification, and derived measure type- multiple classification. Queries belonging to these classes differ in their constructs. Then query trees also vary depending on the queries.

C. Intermediate Schema

Generation of the intermediate schema and the derivation of data warehouse schema constitute the next phase of our design. For the intermediate schema, the knowledge base

graph and the query trees are required. This schema carries the semantic richness of the knowledge base graph as well as the operational requirements from the queries. Moreover the intermediate schema clearly shows the queries that need to be answered.

An algorithm is developed to create the intermediate schema which takes the knowledge base graph and the query trees as inputs. The algorithm identifies a path equivalent to a query tree in the knowledge base so that the query can be supported. The path identification is also described as a mapping process in which a query tree is mapped to an existing graph.

The mapping starts with initialization of the graph, (G_k') as the knowledge base graph. This initialization allows us to check the validity of a query tree with respect to the business, and thus mapping is allowed. Since the query trees have different types of edges; (depending upon the query types that we have mentioned in the previous section), the type identification of queries is necessary. Then modification on the initial graph is suggested as per the query tree.

There are two graphs namely; the knowledge base graph and a query tree, involved in this proposed mapping. So it is important to check the similarity of nodes in the query tree to that of the nodes in the knowledge base graph. The similarity checking is achieved here by means of a special kind of function called Similarity Function. This function takes parameters from a maximum of two nodes. These nodes correspond to an edge in the query tree and connect the two nodes. The nodes are then compared against the nodes that exist in G_k. A simple example of such a comparison would be the domain intersection. That is, if the domain intersection of one of the nodes from query tree and a node from KB graph is not null, then those nodes are considered to be similar. If an edge corresponding to the input nodes exists in the knowledge base the function returns that edge. Otherwise, the function tries to identify more than one edge that connects the nodes. If there are no similar nodes in the knowledge base, the function returns null values and the process is terminated by accepting new query tree. Depending upon the function output, corresponding edges are selected in G_k and the query tree is mapped.

We explain the mapping process in more detail using the example query presented in section III-B. The query tree is processed from the measure type node and in this case it is sales. That means the edge (sales, quarter) is considered first and the similarity function takes parameters from these nodes. The function can be written as: SIM (sales; quarter) = {(day; sales), (day; quarter)}.

Since there is no direct edge between (sales, quarter) in the knowledge base the function returns two edges connecting the input nodes. The other edge (quarter, year) is processed next, and as per the function output the edge is selected in the knowledge base. Thus an equivalent path is identified. The final graph achieved after the mapping of all the queries, is called the intermediate schema. An indicated path in this schema represents a query. Hence we derive a schema tailored for the queries which is discussed in the next section.

D. Data Warehouse Schema

The intermediate schema, suggested in the previous section, should be able to answer queries. However, a complete design methodology should also include provision for schema refinement. Hence the intermediate schema has been used to derive a data warehouse schema that is optimized for the queries. When deriving a schema, schema properties need to be specified first. For example, we may consider the minimality property with a minimal schema, defined as a schema which is minimal with respect to the queries considered during the design. An example of minimality might be minimum path length criterion. That is, minimum path length for a query path is the reduction criterion and the schema is said optimal with respect to path length. As we mentioned earlier, the intermediate schema is a graph; so the derivation of the data warehouse schema is suggested as a graph reduction process.

A reduction algorithm is useful in this account and the algorithm selects only that part of the graph which is required for a query path and the rest of the intermediate schema is discarded. While selecting a path, the path with the minimum length is selected over other existing alternate paths. After the reduction process, the schema contains only the paths required for the queries. Each path can be used to support one or more queries. A reduced schema is shown in figure 3. Four types of queries considered for this reduction are yearly sales by region, Total sales in quarter1, year 1999, Sales of home brand products by age, and average age of customers who made transaction over $100. There are edges in this graph that are different from the knowledge base graph. These are the edges that are either modified or added in order to capture the query semantics in the intermediate schema. For example, the edge (year, region) indicates that the user likes to perform an operation between those two nodes and it is an operational edge. On the other hand, the edge (customer, age) is a semantic edge in the knowledge base and is modified in the intermediate schema to show that the attribute age is used for aggregation as per a query.

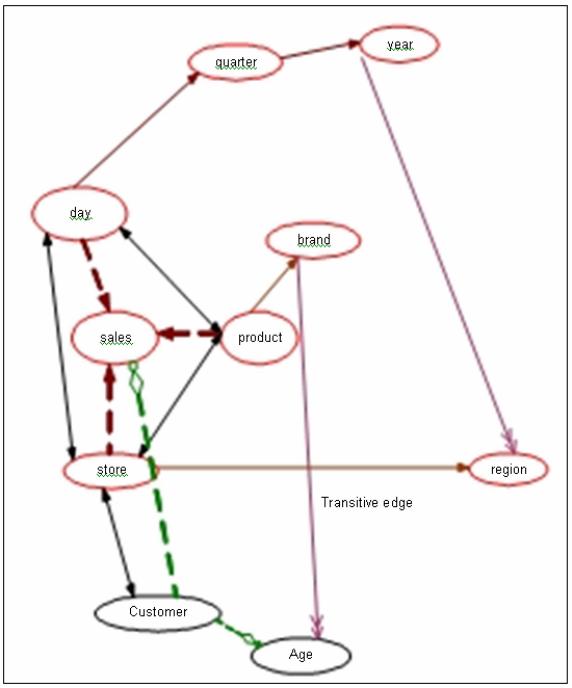

Fig. 3 A Data warehouse Schema

Number of operations, query response time and query frequency can be other reduction considerations. However in these cases, parameters related to implementation are required so we defer that discussion.

Another possibility with the intermediate schema is to use it as a platform for other implementation model like cube. For example, data cubes can be selected for implementation from the intermediate schema. In our design, cube selection is addressed with respect to queries. However the existing methods provide an ad-hoc design and the final schema is too general to implement. Selection of cubes for implementation is generally left out as post design issue and view materialization techniques are necessary to overcome this problem. In this proposed approach, rather than assuming queries, we have come up with query taxonomy and covered general data warehouse queries. These queries are then mapped to the intermediate schema. So it is straightforward to derive cubes from this schema which is suitable for the queries. This indicates the possible mapping of intermediate schema to cube model and shows the generality of our framework.

IV. Conclusion and Future Works

In this paper, we suggested a query based design framework for data warehouse. Compared with existing designs, our framework is systematic in the sense that it starts with requirements and finally identifies the schema constructs as per queries. The presented schema is general and can be transformed as cube or star. Other than the framework, another contribution of our work is the taxonomy and the conceptual representation of queries.

The mapping process, discussed in the context of intermediate schema, requires the similarity function. We have addressed this function only at the higher level. Implementing this function is an interesting issue for future work. We also like to extend our work towards further optimization by associating cost functions along the query paths.

References

[1] A. Abello, J. Samos, and F. Saltor. A framework for the classification and description of multidimensional data models. In International Conference on Database and Expert Systems Applications, pages 668-677, 2001.
[2] A .Gupta, V. Harinarayan, and D. Quass. Aggregate-query processing in data warehousing environment. In International Conference on Very Large Databases, pages 358-369, 1995.
[3] A. Tsois, N. Karayannidis, and T. Sellis. MAC: Conceptual data modeling for OLAP. In International Workshop on Design and Management of Data Warehouses, page 5, 2001.
[4] C. Batini, S. Ceri, and S. Navathe. Conceptual Database Design. Benjamin/Cummings, 1992.
[5] B. List, J. Schiefer, and A.M. Tjoa. Process-oriented requirements analysis supporting the data warehouse design process a use case driven approach. pages 593-603, 2000.
[6] C. Sapia. PROMISE: Modeling and Predicting User Behavior for Online Analytical Applications. PhD dissertation, Technische Universitt Mnchen, vorauss, 2001.
[7] C. Sapia, M. Blaschka, G. Hofling, and B. Dinter. Extending the E/R model for the multidimensional paradigm. In ER Workshops, pages 105-116, 1998.
[8] D. l. Moody. From enterprise model to dimensional models: A methodology for data warehouse and data mart design. In International Workshop on Design and Management of Data, page 5, 2000.
[9] D. Theodoratos. Exploiting hierarchical clustering in evaluating multidimensional aggregation queries. In International Work-shop on Data Warehousing and OLAP, pages 63-70, 2003.
[10] D.W. Cheung, B. Zhou, B. Kao, H. Lu, T.W. Lam, and F. Ting. Requirement -based data cube schema design. In Conference on Information and Knowledge Management, pages 162-169, 1999.
[11] H-J. Lenz and A. Shoshani. Summarizability in OLAP and statistical data bases. In Statistical and Scientific Database Management, pages 132-143, 1997.
[12] B. Husemann, J. Lechtenborger, and G. Vossen. Conceptual data warehouse design. In International Workshop on Design and Management of Data Warehouses, page 6, 2000.
[13] J.A. Bubenko Jr. On the role of understanding models in conceptual schema design. In International Conference on Very Large Databases, pages 129-139, 1979.
[14] J. Zhang, W. Wang, H. Liu, and S. Zhang. X- warehouse: Building query pattern-driven data. In International World Wide Web Conference, pages 896-897, 2005.
[15] L. Cabibbo and R. Torlone. A logical approach to multidimensional databases. In International Conference on Extending Database Technology, pages 183-197, 1998.
[16] M. Golfarelli, D. Maio, and S. Rizzi. Conceptual design of data warehouses from ER schemes. In International Conference on System Sciences, pages 334-343, 1998.
[17] M. Golfarelli, D..Maio, and S. Rizzi. The dimensional fact model: A conceptual model for data warehouses. International Journal of Cooperative Information Systems, 7(2-3):215-247, 1998.
[18] O.Herden. A design methodology for data warehouses. In International Baltic Workshop on Databases and Information systems, pages 292-293, 2000.
[19] P. R. Niven. Balanced Scorecard Step-by-step Maximizing Performance and Maintaining Results. John Wiley, Inc, 2002.

[20] R. Winter and B. S trauch. A method for demand-driven information requirements analysis in data warehousing projects. In International conference on Systems Sciences, pages 231-239, 2003.
[21] T. B. Pedrsen and C.S.Jensen. Multidimensional database technology. Computer, 34:40-46, 2001.
[22] T. Niemi, J. Nummenmaa, and P. Thanisch. Constructing OLAP cubes based on queries. In International Workshop on Data Warehousing and OLAP, pages 9-15, 2001.