International Journal of Advances in Computer Science & Its Applications – IJCSIA
Volume 4: Issue 4 [ISSN 2250-3765]
Publication Date : 27 December,2014

Quality Oriented Logical Design for Data Warehouse Development

Munawar, Naomie Salim, Roliana Ibrahim

***Abstract*—To make an effective data warehouse logical design which is related to system performance, it is important to manage an appropriate methodology for data fragmentation, indexing and materialized view. Executing a query in a data warehouse may take hours or days to run without the proper optimization technique. This paper proposed a framework on integrating quality, deliverables in conceptual design as an input into the logical design, and the optimization technique to run query, to treat as continuous improvement from preceeding stage.**

***Keywords*—logical design, data warehouse, indexing, materialized view, fragmentation**

I. Introduction

A data warehouse (DW) integrates data from external sources and from internal OLTP (On Line Transactional Processing) to support analytical query processing. This analytical query processing can be used by an enterprise to achieve a great competitive advantage. OLAP (On-Line Analytical Processing) tools represent data in a multidimensional version, enabling business users to formulate queries and perform analysis.

Quality issues raised by DW are crucial. For an effective DW system, the quality aspects should be incorporated properly at the various levels of DW development including in logical design phase.

Logical design is the most attracted phase with strongly impacts to the system performance. It is aimed at deriving out of the conceptual schemata the data structure that will actually implement the data mart or DW by considering some sets of constraints (e.g., concerning disk space or query answering time [22]). During logical design the designer defines which structures will be used to store information and how their performance can be optimized.

An acceptable (or good) DW performance is one of the important features that must be guaranteed for DW users. For this reason, providing means for increasing the performance of a DW for analytical queries is one of the important research and technological areas.

Munawar
Esa Unggul University
Indonesia

Naomie Salim, Roliana Ibrahim
Universiti Teknology Malaysia
Malaysia

In this paper, we study logical design and practical issues related to the design of multidimensional modeling. The framework for investigation is **I-LogiQ** (**I**ntegrated **Logi**cal Design and **Q**uality), a logical model for OLAP systems that extends our earlier proposal [12; 13; 14]. This model includes a number of concepts that optimize the technique to run query using indexing, materialized view, and fragmentation, commonly used in multidimensional database.

The paper is structured as follows. Section 2 describes related research. Section 3 presents the proposed framework. Finally, conclusion is described in section 4.

II. Related Works

Due to the increasing complexity of DWs, continuous attention must be paid for evaluation of their quality throughout their design and development [6]. It is also very important to consider quality issues at various levels of models including logical models. Quality of the DW logical models has been assured by proposing several metrics to evaluate the quality of star schemas at logical level. These proposed metrics were validated theoretically and empirically [17, 16]. [17] have proposed metrics for measuring multidimensional schemas analyzability and simplicity. Nevertheless, the metrics proposed in these approaches have not been empirically validated and consequently, have not proven their practical applications [4]. Recently, [16] proposed a set of metrics for assessing the understandability of DW schemas using structural metrics and also validated theoretically and empirically through a family of experiments.

Our proposed frameworks mainly focus on integrating quality and deliverables in preceding stage (conceptual stage) as an input into the logical design in order to treat as a continuous improvement in the next phase. In a DW where data is processed in stages, and where the quality of data at one stage is dependent on the DQ measurements in preceding stages, DQ can be assessed and monitored continuously in order to guarantee high quality levels. As a result, DQ is not only an integral part of DW project, but will remain a sustained and ongoing activity [12].

III. Proposed Framework

The logical design can be used for many purposes [2]: (i) as an intermediate representation between the conceptual design and the physical design, providing an operational view of the DW without necessarily dealing with performance nor physical representation of data, (ii) as a reference schema from which the physical design starts and to which the benefit of the selected materialized views is balanced, (iii) as a support to control the DW evolution both at its client and source levels.

The logical design of the DW serves to define the structure to ensure an efficient access to information. It can be presented as relational or multidimensional structure that takes as input the conceptual schema representation, the information requirements, the source database and non-functional requirements [26]

In relational implementations, the so-called star, constellation, and snowflake schemata are widely accepted to manage data cubes and are supported by various vendors. Concerning multidimensional implementations, several efficient multidimensional data structures such as condensed cubes, dwarfs, and QC-Trees have been proposed to manage data cubes. Comparison between relational implementation and multidimensional implementation can be seen in Table I, Table II and Table III.

TABLE I. COMPARISON OF RELATIONAL IMPLEMENTATION FOR MULTIDIMENSIONAL MODELING [11]

	Star Schema	Fact Constellation Schema	Snowflake Schema
Efficiency	High	High	Moderate
Usability	High	Moderate	Moderate
Reusability	Low	Low	High
Flexibility	High	High	Moderate
Redundancy	High	High	Low
Complexity	Low	Moderate	Moderate

TABLE II. THE COMPARISON BETWEEN DW SCHEMAS IN TERM OF ADVANTAGES AND DRAWBACKS [31]

DW Schema	Advantages	Drawbacks
Star Schema	• It is the simplest structure [28] • It reduces the number of tables [27] • It reduces the number of relationships between the tables [27] • It reduces the number of joins required in user queries [27] • It speed up query performance	• It can be very inflexible [29] • For every gigabyte of row data a schema will require at least an additional gigabytes for aggregations [29] • The amount of development maintenance effort needed to manage schema oriented DW [29]
Fact Constellation Schema	• It reuses the dimension tables to save storage space [30]	• It may not be useful for small organization because of its complexity [36]
Snowflake Schema	• It shows explicitly the hierarchical structures of each dimension [29] • It is intuitive and easy to understand [31] • It can accommodate for aggregate data [31] • It is easily extensible by adding new attributes without inferring with existing database programs [31]	• It adds unnecessary complexity [29] • It reduces query performance [29]

TABLE III. COMPARISON OF DIMENSIONAL IMPLEMENTATION FOR MULTIDIMENSIONAL MODELING

	Condensed Cube [23, 5]	Dwarf [18, 19]	QC-Trees [10]
Size	Much smaller size of non-condensed cube	Highly compressed and clustered data cubes	Very compact data structure
Compression	• Fully pre-computed cube without compressio • Neither decompression or further aggregation is required when answering queries	• Complete architecture that support queries, updates and roll-up data. • A tunable granularity parameter that controls the amount of materialization performed	• It is elegant and lean in that the only information it keeps on classes are their upper bound and measure(s) • Better compression and construction time than Dwarf

Traditional database systems are inadequate for multidimensional analysis since they are optimized for on-line transaction processing (OLTP), which corresponds to large numbers of concurrent transactions, often involving very few records. Conversely, multidimensional database systems should be designed for the so-called *on-line analytical processing* (OLAP), which involves few complex queries over very large numbers of records. Current technology provides both OLAP data servers and client analysis tools. OLAP servers can be either relational systems (ROLAP) or proprietary multidimensional systems (MOLAP).

Unlike OLTP systems where the logical data schema is hidden underneath an application layer, the logical multidimensional (MD) schema of an OLAP system is directly used by the end user to formulate queries. Thus, the MD schema is crucial as it determines the type of queries the user can formulate. One of the main problems in MD data models occurs when the modeled OLAP scenarios become very large since the dimensionality increases significantly, and therefore, this leads to extremely sparse dimensions and data cubes [22]. The system architecture for DW logical designs can be seen in Figure 1.

The way data are actually stored gives rise to different types of OLAP: relational OLAP (ROLAP), multidimensional OLAP (MOLAP), and hybrid OLAP (HOLAP). The choice of ROLAP or MOLAP should depend on the query complexity and performance. For more complex queries and quicker response times, MOLAP should be used because it stores the data in multidimensional databases (cubes) that provide extensive OLAP capabilities. In ROLAP, on the other hand, the data are stored as relational tables and the ROLAP engine generates MD views on the fly. But the ROLAP model works fine when query complexity is not that high and response time demands are not that great.

International Journal of Advances in Computer Science & Its Applications – IJCSIA
Volume 4: Issue 4 [ISSN 2250-3765]
Publication Date : 27 December,2014

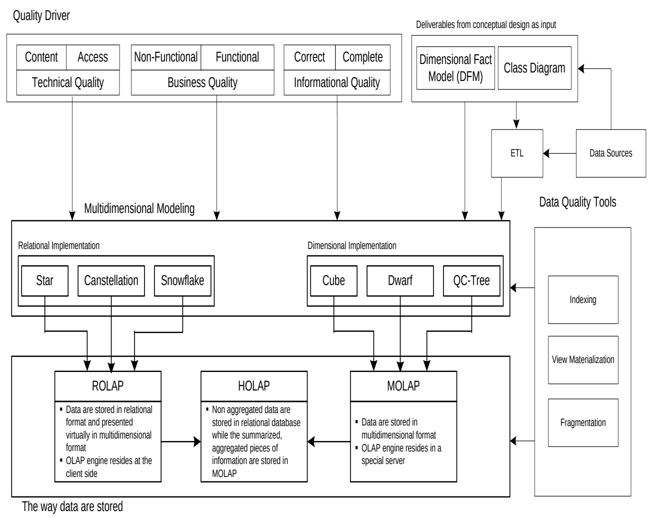

Figure 1. Proposed quality-based framework for logical design in DW development

DW consists of a huge fact table and multiple dimension tables. Queries executed on the top of this schema, typically perform aggregations on the fact table based on selections among the available dimension levels. The growth of the DW volume can degenerate OLAP query performance [20]. As a result, queries frequently submitted to the data warehousing environment no longer ensure satisfactory response times to users.

To reduce the cost of execution caused by decisional complex queries, implying a very voluminous fact table which is linked through a multitude of joins with the dimension tables, we can use structures and optimization techniques such as: indexes, materialized views and / or fragmentation of data [15]. Without optimization techniques, queries may take hours or days to run. This is due to the high complexity of queries.

Materialized views are physical structures that improve data access time by precomputing intermediary results. Then, user queries can be efficiently processed by using data stored within views and do not need to access the original data. Nevertheless, the use of materialized views is restricted by factors like space availability and query response time. Not all possible views for a multidimensional cube can be materialized because the number of views grows exponential to the number of cube dimensions. In general, a cube with n dimensions has $2n$ possible views [7].

A materialized view is much richer in structure than an index since a materialized view may be defined over multiple tables, and can have selections and GROUP BY over multiple columns. In fact, an index can logically be considered as a special case of a single-table, projection only materialized view [1]. This richness of structure of materialized views makes the problem of selecting materialized views significantly more complex than that of index selection.

A crucial problem related to view materialization is that of accurately estimating the actual cardinality of each view [49; 48]. Since the number of possible views which can be derived by aggregating a cube is exponential in the number of attributes, most approaches assume that a constraint on the total disk space occupied by materialization is posed, and attempt to find the subset of views which contemporarily satisfies this constraint and minimizes the workload cost [43; 44; 45]. If the DW has already been loaded, view cardinalities can be quite accurately estimated by using statistical techniques based, say, on histograms [47] or sampling [46]. However, such techniques can not be applied at all if the DW is still under development, and the estimation of view cardinalities is needed for design purposes.

Indexing a DW is very tricky [34]. If there are few indexes, the data loads up quickly but the query response time is slow. If there is too many indexes, the data loads slowly and need more storage space but the query response is good. So there is a trade-off between the number of indexes built and response time of queries Indexing in any database, transactional or warehouse, most often reduces the retrieval time of query results [35]. Different indexing techniques have been developed which are being used in for fast data retrieval in DW environments. Brief description of a few indexing techniques is given below.

TABLE IV. COMPARISON OF INDEXING TECHNIQUES [34]

Indexing Techniques	Advantages	Drawbacks
Bitmap indexing	• Widely used in DW environment [37] • Reduced response time for large classes of ad-hoc query • Reduced storage requirements as compared to other indexing techniques [37]. • Dramatic performance gains on hardware with a relatively small number of CPUs or a small amount of memory [37]. • Efficient maintenance	• It works pretty slow on high cardinality column data. • A modification to a bitmap index requires more work on behalf of the system. • The concurrency for modifications on bitmap indexes is outrageous.
Cluster indexing	• It can be formed dense and sparse to get optimized performance. • Good for range based queries but requires sorted data.	• If data is not sorted then cost of sorting is also added up. • Also Insertions often requires reordering of data, so it is costly operation in terms of time and resources in Data ware housing [38, 39].

Indexing Techniques	Advantages	Drawbacks
Hash-based indexing	• It reduces a large amount of data down to a reasonable number by transforming it through a hash function and indexing that number [40]. • It reduces the average lookup cost by a careful choice of the hash function, bucket table size, and internal data structures [41]. • It does not need the key to be sorted order. • Hash-based indexing technique is best for equality selections.	• Bad hash function leads to collusion. • Hash-based indexing technique cannot support range queries. • Static hashing can leads to long chains. • Reverse of hash is still impossible.

Although indexing can help in providing good access support at the physical level, the number of irrelevant data retrieved during the query processing can still be very high. The horizontal fragmentation aims to reduce irrelevant data accesses [42; 36]. Moreover, fragments can be allocated to data marts if the data in data marts are derived from the warehouse data (i.e. top-down approach). Another advantage of allowing partitioning of a warehouse data is that an OLAP query can be executed in a parallel fashion [36]

Executing an OLAP query in a DW can be very expensive, particularly on large warehouse data, if the data is not modeled properly. Moreover, if OLAP queries need only a portion of data, it is advisable to fragment data so that a set of queries can be executed on each fragment as far as possible, thus query response time can be minimized [36].

In the relational DW context, there are three fragmentation types [42]: vertical fragmentation, horizontal fragmentation and hybrid fragmentation (horizontal fragmentation followed by vertical, or vice versa). Vertical fragmentation splits a relation R into sub-relations that are projections of R with respect to a subset of attributes. It consists in grouping together attributes that are frequently accessed by queries. Vertical fragments are built by projection. The original relation is reconstructed by joining the fragments. Horizontal fragmentation divides a relation into subsets of rows using query predicates. It reduces query processing costs by minimizing the number of local accessed instances. Horizontal fragments are built by selection. The original relation is reconstructed by fragment union.

There are two versions of horizontal fragmentation [42]: *primary* and *derived*. Primary horizontal fragmentation of a relation is performed using attributes defined on that relation. This fragmentation may reduce query processing cost of selections. Derived horizontal fragmentation, on the other hand, is the fragmentation of a relation using attribute(s) defined on another relation(s). In other word, the derived horizontal fragmentation of a table is based on the fragmentation schema of another table(s).

IV. Conclusion

In this paper, we have proposed a framework based on logical model for integrating data quality dimensions and deliverables in preceding stage (conceptual stage: dimensional fact model and class diagram) as an input in order to treat as a contionus improvement in logical stage. In a DW where data is processed in stages, and where the quality of data at one stage is dependent on the DQ measurements in preceding stages, DQ can be assessed and monitored continuously in order to guarantee high quality levels. A key step in logical design is optimization technique to run query among data fragmentation, indexing and materialized view. This paper shows how to choose the proper technique to run query effectively

References

[1] Agrawal, S., Chaudhuri, S. and Narasayya, V. 2000. Automated Selection of Materialized Views and Indexes for Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, 2000

[2] Bouzeghoub, M. and Kedad, Z. 2000. A Logical Model for Data Warehouse Design and Evolution. In Y. Kambayashi, M. Mohania, and A M. Tjoa (Eds.): DaWaK 2000, LNCS 1874, pp. 178−188, 2000.

[3] S. Cohen, W. Nutt, and A. Serebrenik. Algorithms for rewriting aggregate queries using views. In *Proc. DMDW*, Heidelberg, Germany, 1999.

[4] Fenton, N., Pfleeger, S. 1997. Software Metrics: A Rigorous Approach (2nd ed.). London: Chapman & Hall.

[5] Feng, J., Fang, Q., and Ding. H. 2004. Prefixcube: Prefix-sharing condensed data cube. In *Proc. DOLAP*, pages 38–47.

[6] Inmon, W.H. 1997. Building the data warehouse (2nd ed.). John Wiley and Sons.

[7] Jamil, H. A. and Modica, G.A. 2001. A View Selection Tool for Multidimensional Databases in L. Monostori, J. V_ancza, and M. Ali (Eds.): IEA/AIE 2001, LNAI 2070, pp. 237{246, 2001.c Springer-Verlag Berlin Heidelberg 2001

[8] Jarke, M., Jeusfeld, M., Quix, C., Vassiliadis, P.: Architecture and quality in data warehouses: an extended repository approach. Information Systems, volume 24, number 3, 1999

[9] Gupta, Chauhan, Kumar & Taneja, (2011) "UREM-A UML-Based Requirement Engineering Model for a Data Warehouse", In Proceedings of the *5th National Conference; INDIACom-2011, Computing For Nation Development,* New Delhi, India. ISSN 0973-7529 ISBN 978-93-80544-00-7

[10] Lakshmanan, L. V. S., Pei, J. and Zhao, Y. 2003. QC-Trees: an efficient summary structure for semantic OLAP. In *Proc. ACM SIGMOD,* pages 64–75.

[11] Mishra, D., Yazici, A., and Basaran, B. P. 2008. A Case Study of Data Models in Data Warehousing. IEEE. 978-1-4244-2624-9/08

[12] Munawar, Naomie Salim, and Roliana Ibrahim. 2011. Toward Data Quality Integration into the Data Warehouse Development. Ninth IEEE International Conference on Dependable, Autonomic and Secure Computing. 978-0-7695-4612-4/11 © 2011 IEEE Computer Sociaty. DOI 10.1109/DASC.2011.194

[13] Munawar, Naomie Salim, and Roliana Ibrahim. 2011. Toward Data Warehouse Quality through Integrated Requirements Analysis. ICACSIS 2011. ISBN: 978-979-1421-11-9.

[14] Munawar, Naomie Salim, and Roliana Ibrahim. 2012. Comparative Study of Quality Dimensions for Data Warehouse Development : A Survey. A. Ell Hassanien et al. (Eds.): AMLTA 2012, CCIS 322, pp. 465–473, 2012. © Springer-Verlag Berlin Heidelberg 2012

[15] Aouiche, K. 2005. Automatic selection of indexes in data warehouses. Research report, Laboratory ERIC Lumière Lyon2University, Doctoral School in Cognitive Science, December 8, 2005.

[16] Serrano, M.A., Calero, C., Sahraoui, H.A., Piattini, M. 2008. Empirical studies to assess the understandability of data warehouse schemas using structural metrics.

[17] Si-Said, S., Prat, N. 2003. Multidimensional Schemas Quality: Assessing and Balancing Analyzability and Simplicity, ER 2003 Workshops, 140–151.

[18] Sismanis, Y., Deligiannakis, A., Kotidis, Y., and Roussopoulos, N. 2003. Hierarchical dwarfs for the rollup cube. In *Proc. DOLAP*, pages 17–24.

[19] Sismanis, Y. and Roussopoulos, N. 2004. The complexity of fully materialized coalesced cubes. In *Proc. VLDB*, pages 540–551.

[20] Cuzzocrea, A. 2006. Improving range-sum query evaluation on data cubes via polynomial approximation.Data & Knowledge Engineering, Vol. 56, No.2, p. 85-121.

[21] Theodoratos, D., and Sellis, T. Designing data Data warehouses. *DKE*, 31(3):279–301, 1999.

[22] Trujilo, J.C., Palomar, M. and Gomes, J. 2000. Applying Object-Oriented Conceptual Modeling Techniques to the design of Multidimensional Databases and OLAP Application in H. Lu and A. Zhou (Eds.): WAIM 2000, LNCS 1846, pp. 83–94, 2000._c Springer-Verlag Berlin Heidelberg 2000

[23] Wang, W., Lu, H., Feng, J., and Yu, J. X. 2002. Condensed cube: An efficient approach to reducing data cube size. In *Proc. ICDE*, pages 155–165.

[24] W.P. Yan and P. Larson. Eager and lazy aggregation. In *Proc. 21st VLDB*, pages 345–357, Zurich, Switzerland, 1995.

[25] Muralikrishna, M. and DeWitt, D.J. 1998. Equi-depth Histograms for Estimating Selectivity Factors for Multi-Dimensional Queries. In *Proc. ACM Sigmod Conf.*, pages 28–36, Chicago, IL.

[26] Peralta, V., Illarse, A., and Rugia, R. 2003. Towards the Automation of Data Warehouse Logical Design: a Rule-Based Approach

[27] B. P. Basaran, "A Comparison Of Data Warehouse Design Models", A MASTER'S THESIS in. Computer Engineering, 2005.

[28] D. L. Moody, M. A. R. Kortink, "From ER Models to Dimensional Models Part II: Advanced Design Issues", Journal of Business Intelligence, pp.1-12, 2008.

[29] F. Teklitz, "The Simplification of Data Warehouse Design", Sybase, 2000.

[30] M. Levene, G. Loizou, "Why is the Snowflake Schema a Good Data Warehouse Design?", In Source, Information Systems, pp. 225-240, 2003.

[31] N. Arfaoui, J. Akaichi. "Data Warehouse : Conceptual and Logical Schema – Survey. International Journal of Enterprise Computing and Business Systems. ISSN (Online) : 2230-8849. Vol. 2 Issue 1 January 2012

[32] Theodoratos, D. and Bouzeghoub, M.2000. A General Framework for the View Selection Problem for Data Warehouse Design and Evolution. In *Proc. DOLAP*, pages 1–8, Washington, DC.

[33] Vassiliadis, P. 2000. Gulliver in the land of data warehousing: practical experiences and observations of a researcher. In *Proc. DMDW*, pages 12/1–12/16, Stockholm, Sweden.

[34] Jamil, S., and Ibrahim, R. 2009. Performance Analysis of Indexing Techniques in DW. International Conference on Emerging Technologies. 978-1-4244-5632-1/09 ©2009 IEEE

[35] Poolet , M. A. 2008. Indexing the data warehouse. SQL server magazine August 2008.

[36] Bellatreche, L., Mohania., M., Karlapalem., K., and Schneider., M. 2000. What can Partitioning do for your Data Warehouses Or Data Marts. 0-7695-0789-1/00 ©2000 IEEE

[37] http://download-east.oracle.com/docs/html/A76994_01/indexes.htm

[38] S. B. Davidson, "Indexing, Sorting and Hashing", October 23, 2008, www.seas.upenn.edu/~cse330/docs/14-Indexing.ppt

[39] A. Aizawa, "A method of cluster-based indexing of textual data", In Proceedings of the 19thinternational conference on Computational linguistics - Volume 1, International Conference On Computational Linguistics, Taipei, Taiwan, pp. 1 – 7, 2002

[40] S. Delmarco, " Hash Indexes", January 27, 2006 http://www.fotia.co.uk/fotia/FA.03.Sql2KHashIndexes.01.aspx

[41] http://en.wikipedia.org/wiki/Hash_table

[42] Ozsu, M. T. & Valduriez, P. 1991. Principles of Distributed Database Systems. (pp. 657). Prentice-Hall, Inc. Upper Saddle River, NJ, USA

[43] Golfarelli, M., Maio, D., and Rizzi, S. 2000. Applying vertical fragmentation techniques in logical design of multidimensional databases. In *Proc. DaWaK*, pages 11–23.

[44] Gupta, H. 1997. Selection of Views to Materialize in a Data Warehouse. In *Proc. ICDT*, pages 98–112, Delphi, Greece.

[45] Harinarayan, V., Rajaraman, A., and Ullman, J. 1996. Implementing Data Cubes Efficiently. In *Proc. ACM Sigmod Conf.*, pages 205–216, Montreal, Canada.

[46] Hou, W. and O¨ zsoyoglu, G. 1991. Statistical Estimators for Aggregate Relational Algebra Queries. *ACM Transactions on Database Systems*, 16(4):600–654.