Optimizing the Data Warehouse Design by Hierarchical Denormalizing

Morteza Zaker, Somnuk Phon-Amnuaisuk, Su-Cheng Haw
Faculty of Information Technology, Multimedia University, Malaysia

Abstract: Data normalization and denormalization processes are common in database design community as these processes have a great impact on the underlying performance. Current data warehouse queries involve a set of aggregations and joining operations. Thus, normalization process is not a good choice as many relations need to be merged in order to answer queries involving aggregation. On the other hand, denormalization process engages a lot of administrative task. This task takes into account the documentation structure of the denormalization assessments, data validation, schedule of migrating of data and so on. In this paper, we show that the mentioned justifications can not be convincible reasons, under certain circumstances, to ignore the effects of denormalization. Until now denormalization techniques have been introduced for various types of database design. One of the techniques is hierarchical denormalization. Our experimental results indicate that the query response time is significantly decreased when the schema is deployed by hierarchical denormalization on a large dataset with multi-billion records. Thus, we suggest that hierarchical denormalization could be considered as a fundamental method to enhance query processing performance.

Key–Words: Data warehouse, Normalization, Hierarchical denormalization,Query processing

1 Introduction

Data Warehouse (DW) is the foundation for Decision Support Systems (DSS) with large collection of information that can be accessed through an On-line Analytical Processing (OLAP) application. This large database stores current and historical data from several external data sources [1, 2] . The queries built on DW system are usually complex and include some join operations that incur high computational overhead. They generally consist of multi-dimensional grouping and aggregation operations. In other words, queries used in OLAP are much more complex than those used in traditional applications. The large size of DWs and the complexity of OLAP queries severely increase the execution cost of the queries and have a critical effect on the performance and productivity of DSS. [3, 4]

At present, the majority of database software solutions for real-world applications are based on a normalized logical data model. Normalization is easy to implement; however, it has some limitations in supporting business application requirements [5] .

Date [6] supports the fact that denormalization speeds up data retrieval, but one disadvantage of denormalization is low degree of support for potential frequently update. Indeed, data warehouses entail fairly less data updates and mostly data are retrieved only in most transactions [2] . In other words, applying denormalization strategies is best suited to a data warehouses system due to infrequent updating.

There are multiple ways to construct denormalization relationships for a database, such as Pre-Joined Tables, Report Tables, Mirror Tables, Split Tables, Combined Tables, Redundant Data, Repeating Groups, Derivable Data and Hierarchies[7, 8] .

The focus of the paper is on utilizing Hierarchical denormalization to optimise the performance in DW. Although, designing, representing and traversing hierarchies are complex as compared to the normalized relationship, the main approach to reduce the query response time is by integrating and summarizing the data[9, 10] . Hierarchical denormalization is particularly useful in dealing with the growth of star schemas that can be found in most data warehouse implementations [5, 10] .

The open issue at this point is, according to the conventional wisdom, all relational database designs should be based on a normalized logical data model [11] . Beyond this, even though the idea of improving the system by reducing table joins is not new, the decision for denormalization is hardly practical because it engages a lot of administrative devotion. This devotion takes the documentation structure of the denormalization assessments, data validation, schedules for migrating of data, and so on. On the contrary, there are some research studies which have indicated that denormalization may give better performance and a more flexible data structure for users[5, 10] . Altough normalization can organize data into well-balanced structure to optimize data accessability, there are still some deficiencies which decrease system performance[12, 13, 14] . It is also obvious that there are great diversities between the focus in database design of the academician and the IT practitioner. Certainly, denormalization is a process that has not attracted interest in the academic world but is a reasonable strategy in practical database community.

In this paper, we demonstrate that: (i) The data model and structure in a data warehouse which involves many joins operation, is strongly suggested to be considered as a candidate to be denormalized by hierarchical technique. (ii) Although data maintenance in a normalized structure is easier as compared to denormalized structure, query processing time in the structure provided by hierarchical denormalizing is extraordinarily less than normalized structure. (iii) Query processing performance is significantly affected by building Bitmap Index on columns which is involved by denormolized implementation.

In this paper, we give an overview of related works and background studies on normalization, denormalization and hierarchical denormalization in section 2. Section 3 defines a case study and performance methodology on a set of queries to compare the performances of normalized and denormalized table structures. Section 4 discusses the experimental results. Lastly, Section 5 concludes the paper.

2 Background

2.1 Normalization

Normalization is a technique first mentioned by Codd [15] and has been deployed by Date[6] for determining the optimal logical design to simplify the relational design of an integrated database, based on the groupings of entities to make better performance in storage operations.

While an entirely normalized database design decreases the inconsistencies, it can cause other kind of difficulties such as insufficient system response time for data storage and referential integrity problems due to the disintegration of natural data objects into relational tables [16] .

2.2 Denormalization

Some pertinent explanations exist for denormalizing of a relational design to improve its performance. Realizing of these explanations will assist to identify of systems and entities as denormalization candidates. These explanations need to be considered to help designers to reach the mentioned identification. These explanations are: (i) critical queries which involve more than one entity. (ii) Frequent times of queries which must to be processed in on-line environments. (iii) A lot of calculations which need to be applied to single or many columns upon queries can be efficiently answered; (v) entities which need to be extracted in different ways during the same time and (iiv) to be aware about relational database which can brings better performance and enhanced access options that may increase the possibility for denormalization.

The most OLAP processes within the data warehouse is to extract summarized and aggregated data, such as sums, averages, and trends to access aggregated and time series data with immediate display. The components which are best suited for denormalization in a data warehouse include: multidimensional analysis in a complex hierarchy, aggregation, and complicated calculations [5, 17, 18] .

2.3 Hierarchies

From a technical point of view, a parent-child relationship refers to a hierarchy where a child has only one parent. A hierarchy is a collection of levels which can be drill-down. Drill-down refers to traversing of a hierarchy from the top levels of aggregation to the lower levels of detail [9, 10, 18] .

To illustrate what is the structure of hierarchy, we show an example by [10] . *" Each member in a hierarchy is known as a "node." The topmost node in a hierarchy is called the "root node" while the bottommost nodes are known as "leaf nodes." A "parent node" is a node that has children, and a "child node" is a node which belongs to a parent. Apparent (except*

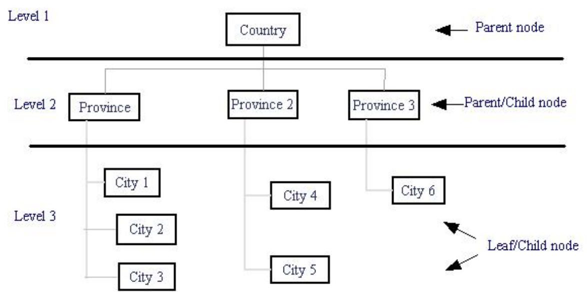

Figure 1: Hierarchy (balanced tree)

a root node) may also be a child, and a child (except a leaf node) may also be a parent." Figure 1 shows such a hierarchy.

Hierarchies are essential aspect of DWs. Thus, supporting different kind of hierarchies in the dimensional data, and allowing more flexibility in defining the hierarchies can enable a wider range of business scenarios to be modeled [19] . We outline several types of hierarchies in the data warehouse environment as follows [10] .

1. *" Balanced tree structure: In this structure, hierarchy has a consistent number of levels and each level can be named. Each child have one parent at the level immediately above it.*

2. *Variable depth tree structure: In this structure, the number of levels is inconsistent (such as a bill of materials) and each level cannot be named.*

3. *Ragged tree structure: This hierarchy has a maximum number of levels, each of which can be named and each child can have a parent at any level (not necessarily immediately above it).*

4. *"Complex tree structure: In this hierarchy, a child may have multiple parents.*

5. *"Multiple tree structures for the same leaf node [10] . "*

3 Related Works

Bock and Schrage [16] have indicated that a number of factors affecting system response time are related to ineffective use of database management system tuning, insufficient hardware platforms, poor application programming techniques, and poor conceptual and physical database design. In their studies they focused on the effects of multiple table joins on the system response time. In order to construct an object view that managers need to extract data for their trade activities, a business computer application may have to join several tables. Reducing the number of table joins will improve system response time.

Hanus [11] has outlined the advantages of normalization process, such as, easing the designs process and physical implementation, reduces data redundancy and protects data from update and delete anomalies. He also showed that entities, attributes, and relations can easily be modified without restructing the entire table. Moreover, since the tables are smaller with fewer numbers of bytes, less physical storage is required. Beyond this, however, a join operation must be accomplished. In such cases, it might be mandatory to denormalize that data. Nevertheless, he agreed that denormalization must be used with care by understanding of how the data will be used. He has also shown that denormalization can be used to improve system response time without redundant data storage or incurring difficulty of data maintenance.

Sanders and Shin [17] have presented a methodology for the assessment of denormalization effects, using relational algebra operations and query trees. They believed that denormalization is an intermediate step between logical and physical modeling, which is involved with analyzing the functionality of the design regarding to the applications requirements criteria. Nevertheless, their methodology is limited due to the lack of sufficient comparison of computation costs between the normalized and denormalized data structures.

Shin and Sanders [20] have discussed the effects of denormalization on relational database system performance with using denormalization strategies as a database design methodology. They have presented four common denormalization strategies and evaluate their effects with illustrating the conditions where which strategies are most effective. They have concluded that denormalization methods may receive positive effects for database performance such as Data warehouse.

4 Methodology

In order to compare efficiency of denormalization and normalization processes and analysis the performance of these data modeling, we build a series of queries on some columns for evaluation. In our dataset, there are 4 tables; Fact, D1, D2 and D1D2. Fact, D1 and D2 tables have approximately 1.7 billion of records and D1D2 table (a combination of D1 and D2 tables)

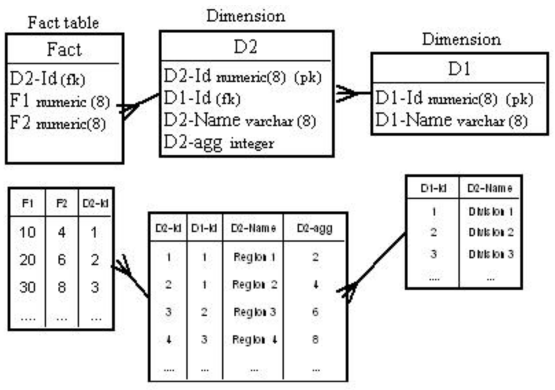

Figure 2: Schema 1 with normalized design

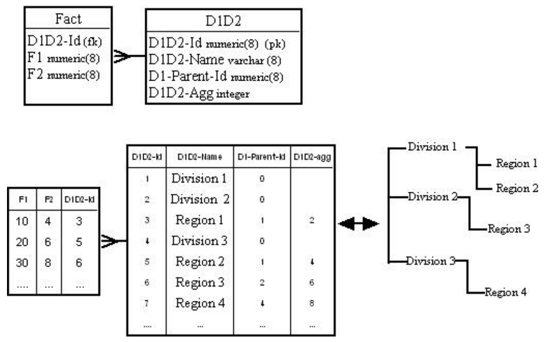

Figure 3: Schema 2 with denormalized design

has approximately 3.36 billion records. These records were randomly generated using PL/SQL Block by Oracle11G tools. These tables can be categorized into two database schemas, Schema 1 and Schema 2 which are portrayed in Figure 2 and Figure 3 respectively. In Schema 1, the tables were applied by normalization modeling where D1 table is connected to the D2 table by one-to-many relationship and similarly, D2 table is also connected to the Fact table by one-to-many relationship. In Schema 2, D1D2 table is directly connected to the Fact table by one to many relationship. The D1D2 table is implemented by hierarchical technique. All attributes, except the keys(PK), of the dimensions are associated by Bitmap index; Schema 1 are contained by 5 indexed columns while Schema 2 are contained by 3 indexed attributes.

Schema 1 has fact data which is chained with huge amount of data stored in D2 and D1 dimensions (shown in Figure 2) while Schema 2 contains the fact table and one dimension table implemented by hierarchy technique (shown in Figure 3).

4.1 Query Set

The Set Query Benchmark has been used for frequent-query application as much like as Star-Schema in the data warehouse design [22, 23] . The queries of the Set Query Benchmark have been designed on based business analysis missions. In order to evaluate the time required for answering different query types including range, aggregation and join queries; we implemented the three (out of six) queries adopted from Set Query Benchmark[22, 23] . Briefly, we describe all of our selected SQL queries used for our performance measurements as indicated in Table 1. Basically, for each query, we used suffix 'A' to represent query on Schema 1 and suffix 'B' to represent query on Schema 2.

Table 1: Descriptions for Set Queries

Query Description	Set-Query 1
Schema 1 One-Dimensional query which involve only one column at a time in the WHERE clause.[21]	Q1A: SELECT count (*) FROM D2 WHERE D2-Name = 'abcdefgh'
Schema 2	Q1B: SELECT count (*) FROM D1D2 WHERE D1D2-name = 'abcdefgh';

Query Description	Set-Query 2
Schema 1	Q2A: SELECT D2-name FROM D2 WHERE (D2-agg between 100000 and 1000000 or D2-agg between 1000000 and 10000000 or D2-agg between 10000000 and 30000000 or D2-agg between 30000000 and 60000000 or D2-agg between 60000000 and 100000000)
Schema 2	Q2B: SELECT D1D2-name FROM D1D2 WHERE (D2-agg between 100000 and 1000000 or D1D2-agg between 1000000 and 10000000 or D1D2-agg between 10000000 and 30000000 or D1D2-agg between 30000000 and 60000000 or D1D2-agg between 60000000 and 100000000)

Query Description	Set-Query 3
Schema 1	Q3A: SELECT D2-name, count (*) FROM D2 GROUP BY D2-name.
Schema 2	Q3B: SELECT D1D2-name, count (*) FROM D2 GROUP BY D1D2-name.

Query Description	Set-Query 4
Schema 1	Q4A: SELECT * FROM D1, D2 WHERE D1.D1-name='abcefgh'and D1.D1-id=D2.D1-id
Schema 2	Q4B: SELECT * FROM D1D2 WHERE D1D2-name='abcefgh'

Query Description	Set-Query 5
Schema 1	Q5A: SELECT D1.'D1.Name', sum(fact1.f1*D2.'D2-Agg) FROM D1,D2, fact WHERE D1.'D1.id'= D2.'D2-Id' and D2.'D2-Id'= Fact.'D2-Id' Group by D1.'D1-name'
Schema 2	Q5B: SELECT D1D2."D1D2-NAME",sum(T.jam) FROM(SELECT D1D2."D1D2-ID" Id,(D1D2."D1-PARENT-ID" pid,fact.f1 * D1D2."D1D2-AGG") as jam FROM D1D2, fact WHERE D1D2."D1D2-ID"= fact."D2-ID") T,D1D2 WHERE D1D2."D1D2-ID"= T.pid GROUP BY D1D2."D1D2-NAME"

Query Description	Set-Query 6
Schema 1	Q6A: SELECT D1."D1-NAME",D2."D2-NAME", D2."D2-AGG" FROM D1,D2 WHERE D1."D1-ID" = D2."D1-ID"
Schema 2	Q6B: SELECT D1D2."D1D2-NAME" D1, connect_by_root D1D2."D1D2-NAME", D1D2."D1D2-AGG" D2 FROM D1D2 WHERE level > connect by prior D1D2."D1D2-ID" = D1D2."D1-PARENT - ID"

4.2 Experimental Setup

We performed our tests on the Microsoft Windows Server 2003 machine with Oracle11G database systems. Table 2 shows some basic information about the test machines and the disk system. To make sure the full disk access time is accounted for we disable all unnecessary services in the system and keep the same condition for each query. To avoid inaccuracy, all queries are run 4 consecutive times to give an average elapsed time.

Table 2: Information about the test system

CPU	Pentium 4 (2.6 GHZ)
Disk	7200 RPM, 500 GB
Memory	1 GB
Database	Oracle11G

5 Results and Discussions

5.1 Query Response Time

In this section, we show and discuss on the time required to answer the queries. These timing measurements directly reflect the performance of normalizing or denormalizing methods. A summary of all the timing measurements on several kinds of queries is shown in Table 3.

Table 3: Query response time (per seconds)

		First Schema(QXA) (Normalized)	Second Schema(QXB) (Denormalized)
One-dimensional	Set-Query1	0.020	0.031
	Set-Query2	21.20	30.43
	Set-Query3	1646.98	2949.98
Multi-dimensional	Set-Query4	830.39	0.16
	Set-Query5	108000.34	10000.29
	Set-Query6	976.87	102.32

5.2 One-Dimension Queries

Firstly, we examine the performance on count queries (Set-Query 1). When the Schema is deployed by denormalizaion, it takes slightly more time to execute the queries. The time needed to answer higher amount of count queries is dominated by the time needed to answer the number of rows in the dimension. For example, Q1B applied on dimension with 3.6 billion records and Q1A applied on dimension with 1.7 billion records; we expect Q1B to take about twice as much time as Q1A. However from the results in Table 3, this estimation is not accurate, presumably because the initialization of the tablespace for Q1B and Q1A is easily associated with the Bitmap index techniques. This observation is accessed, because the average time used by normalized schema to read in the data blocks is nearly 0.020s (20 ms) and in denormalized schema is about 31 ms.

Next, we focus for Set-Query 2. We see that the time required by both of the schemas rises significantly. It is necessary to understand the retrieval time to compute how many pages or how many disk sectors are accessed for the retrieval operation. The query response time required to fetch data for Q2A and Q2B has the same doing as each other. The number of records by these queries that has to be selected are uniformly scattered among rows 100,000 and 100,000,000. Here, we also expect Q2B to take about twice as much time as Q2A. However, this estimation is not accurate. Consequently, the elapsed time of both schemas that is needed to answer the queries which are executed within a range of predicates is affected by the distribution of data; not by the normalization or denormalization conditions.

Another query that can be a main way to promise the indexing effects is Set-Query 3. In Q3A and Q3B, we see that the response time of Q3B is almost twice as much time as Q3A. On the other hand, the required time to answer these queries is extremely more than the other one-dimensional queries (Set-Query1). This is because to execute this type of query, the optimizer will not make use of any indexes. Rather, it will prefer to do a full table scan. Since there is an abnormal growth of data, table scan will be needed to increase physical disk reads to avoid insufficient memory allocation. As the result, this does not scale very well as data volumes increase. Even though, there is one implementation of Bitmap indexe (FastBit) which can support these queries directly [13, 14, 21], but Oracle 11G has not utilize this method of implementation.

In summary, Figure 4 shows the query elapse time for one-dimensional queries which were applied on the two schemas. This figure shows that although the query retrieval time on the Schema 1 (which has been designed by normalization method) is faster than Schema 2 (denormalized schema), query performance can be enormously enhanced by using of index techniques especially Bitmap index technique.

5.3 Multi-Dimension Queries

In Set-Query 4, the denormalized solution is impressive since no table joins are required. We see that the denormalized schema query remains unchanged, but the normalized schema structure results in a query that joins the D1 and D2 tables. This resulted in slower system response time and performance will degrade as the table size increases. This query shows that if the First Normal Form tables be transformed by denormalization method, the query response time

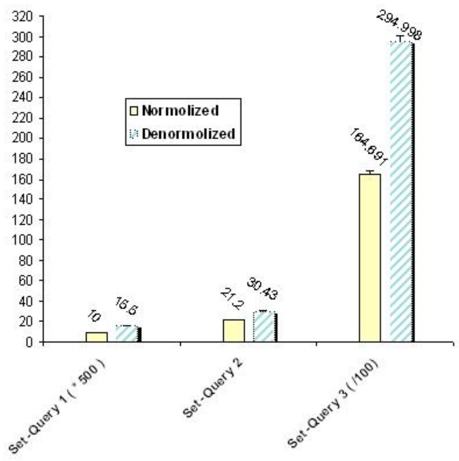

Figure 4: Query elapse times for one dimension queries

can be extraordinary decreased as the number of join operations are reduced.

Data aggregation is a frequent operation in data warehouse and OLAP environments. Data is either aggregated on the fly or pre-computed and stored as materialized views [18] . Since, data aggregation and data computation can really grow to be a complex process through time; having a good and well-define architecture to support dynamic business environments have more long term benefits with data aggregation and computation. In Set-Query 5, we see that the query response time of Q5B (denormolized schema) is excessively less than the Q5A. Thus, we claim that one of the component in a data warehouse that are good candidates for denormalization are aggregation, and complicated calculations.

A flexible architecture leads to potential growth and flexibility lean towards exponential growth. The database software which supports the flexible architecture such as hierarchical methods in data warehouses need to use in state-of-the-arts components to reply complex aggregation needs. It should also be able to support all kinds of reports by using modern operators. These operators should extend the flexibility of hierarchical queries. Oracle does show promises in hyper-computational abilities to process more complex structures by up-to-date operators which other database software might not be able to. To the best of our knowledge and experience there are many operators in oracle11g which can promise the high flexibility in query writing. Provably, we see in Set-Query6 that using oracle operators can enhance speed of data extraction from hierarchical table. Query elapsed time in Q6B has been enormously

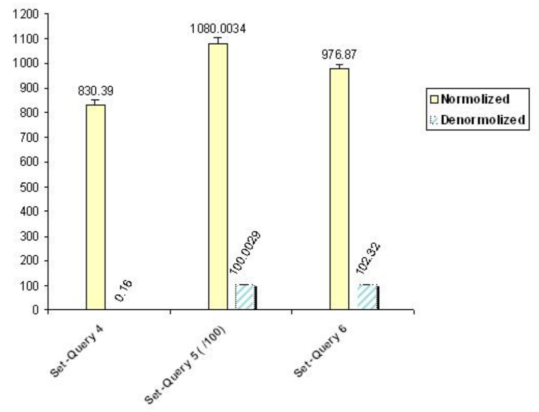

Figure 5: Query elapse times for multi dimension queries which are involved by join operations

decreased as comparing Q6A. Despite the complexity of query scripting, Oracle has long provided specific support for querying hierarchical data. This support comes in the form of the START WITH, CONNECT_BY_ISCYCLE pseudocolumn and so on which assist the designers to easily query hierarchical data.

Figure 5 shows the query elapse time for multidimensional hierarchical queries which have been applied on first and second schemas. This figure shows that using hierarchical denormalization method can improve system response time when the queries are unanticipated ad hoc queries.

6 Conclusion

In this paper, we have presented an experimental examination of denormalization with essential guidelines to be applied in a data warehouse design. We have also shown set queries for the measurement of denormalization effects using hierarchical implementation. In order to elucidate the effects of hierarchical denormalization in further detail, we have invested endeavors in preparing a real test environment for measuring query retrieval times, system performance, ease of utilization and bitmap index effects. These experiments compared aggregation and computation costs between normalized and hierarchical denormalized data structures. Most probably hierarchical denormalization can improve query performance by reducing the query response times when the data structure in Data warehouse is involved by many joins operations. Consequently, these experimental results should possibly be considered as a general guideline which can be applicable to the majority database designs. Moreover hierarchical denormalization is recommended as a fundamental phase in a data warehouse data modeling which is rarely dependant applications requirements since the data warehouse is infrequently updated.

Acknowledgements: Thanks God.

References:

[1] S. Chaudhuri and U. Dayal, *A*n Overview of Data Warehousing and OLAP Technology. ACM SIGMOD RECORD, 1997

[2] R. Kimball and L. Reeves and M. Ross, *T*he Data Warehouse Toolkit. John Wiley and Sons, NEW YORK, 2002

[3] W. H. Inmon, *B*uilding the Data Warehouse. John Wiley and Sons, 2005

[4] C. S. Park and M. H. Kim and Y. J. Lee , *R*ewriting olap queries using materialized views and dimension hierarchies in data warehouses. In Data Engineering, 2001. Proceedings. 17th International Conference on.

[5] S. K. Shin and G. L. Sanders, *D*enormalization strategies for data retrieval from data warehouses. Decis. Support Syst. Oct. 2006, pp. 267-282. DOI= http://dx.doi.org/10.1016/j.dss.2004.12.004

[6] C. J. Date, *A*n Introduction to Database Systems, Addison-Wesley Longman Publishing Co., Inc, 2003

[7] C. S. Mullins, *D*atabase Administration: The Complete Guide to Practices and Procedures. Addison-Wesley, Paperback, June 2002, 736 pages, ISBN 0201741296.

[8] C. Zaniolo and S. Ceri and C. Faloutsos and R. T. Snodgrass and V. S. Subrahmanian and R. Zicari,*A*dvanced Database Systems. Morgan Kaufmann Publishers Inc. 1997

[9] W. T. Joy Mundy, *T*he Microsoft Data Warehouse Toolkit: With SQL Server 2005 and the Microsoft Business Intelligence Toolset. John Wiley and Sons, NEW YORK, 2006.

[10] I. Claudia and N. Galemmo *M*astering Data Warehouse Design -Relational And Dimensional. John Wiley and Sons, 2003, ISBN: 978-0-471-32421-8.

[11] M. Hanus, *T*o normalize or denormalize, that is the. question. In Proceedings of Computer Measurement Group's 1993 International Conference, pp. 413-423.

[12] C. J. Date, *T*he normal is so...interesting. Database Programming and Design. 1997, pp.23-25

[13] J. C. Westland, *E*conomic incentives for database normalization. Inf. Process. Manage. Jan. 1992, pp. 647-662. DOI= http://dx.doi.org/10.1016/0306-4573(92)90034-W

[14] D. Menninger, *B*reaking all the rules: an insider's guide to practical normalization. Data Based Advis. (Jan. 1995), pp. 116-121

[15] E. F Codd, *T*he Relational Model for Database Management. In: R. Rustin (ed.): Database Systems, Prentice Hall and IBM Research Report RJ 987, 1972, pp. 65-98.

[16] D. B. Bock and J. F. Schrage, *D*enormalization guidelines for base and transaction tables. SIGCSE Bull.(Dec. 2002), pp. 129-133. DOI= http://doi.acm.org/10.1145/820127.820184

[17] G. Sanders and S. Shin, *D*enormalization Effects on Performance of RDBMS. In Proceedings of the 34th Annual Hawaii international Conference on System Sciences (Hicss-34)-Volume 3 - Volume 3 (January 03 - 06, 2001). HICSS. IEEE Computer Society, Washington, DC, 3013.

[18] C. Adamson, *M*astering Data Warehouse Aggregates: Solutions for Star Schema Performance. John Wiley and Sons, 2006, ISBN: 978-0-471-77709-0.

[19] R. Strohm.*O*racle Database Concepts 11g. Oracle, Redwood City,CA 94065. 2007

[20] S. K. Shin and G. L. Sanders, *D*enormalization strategies for data retrieval from data warehouses. Decis. Support Syst.(Oct. 2006), PP. 267-282. DOI= http://dx.doi.org/10.1016/j.dss.2004.12.004

[21] E. E-O'Neil and P. P-O'Neil, *B*itmap index design choices and their performance implications. Database Engineering and Applications Symposium. IDEAS 2007. 11th International, pp. 72-84.

[22] P. O'Neil, *T*he Set Query Benchmark. In The Benchmark Handbook For Database and Transaction Processing Benchmarks. Jim Gray, Editor, Morgan Kaufmann, 1993.

[23] P. ONeil and E. ONeil, *D*atabase Principles, Programming, and Performance. 2nd Ed. Morgan Kaufmann Publishers. 2001.