INTERNATIONAL JOURNAL OF COMPUTERS
Issue 1, Volume 3, 2009

Hierarchical Denormalizing: A Possibility to Optimize the Data Warehouse Design

Morteza Zaker, Somnuk Phon-Amnuaisuk, Su-Cheng Haw

***Abstract*—Two of the most common processes in database design community include data normalization and denormalization which play pivotal roles in the underlying performance. Today data warehouse queries comprise a group of aggregations and joining operations. As a result, normalization process does not seem to be an adequate option since several relations must combine to provide answers for queries that involve aggregation. Further, denormalization process engages a wealth of administrative tasks, which include the documentation structure of the denormalization assessments, data validation, and data migration schedule, among others. It is the objective of the present paper to investigate the possibility that, under certain circumstances, the above-mentioned justifications cannot provide justifiable reasons to ignore the effects of denormalization. To date, denormalization techniques have been applied in several database designs one of which is hierarchical denormalization. The findings provide empirical data that show the query response time is remarkably minimized once the schema is deployed by hierarchical denormalization on a large dataset with multi-billion records. It is, thus, recommended that hierarchical denormalization be considered a more preferable method to improve query processing performance.**

***Index Terms*—Data warehouse, Normalization, Hierarchical denormalization, Query processing**

I. Introduction

DATA Warehouse (DW) can be regarded as the foundation for Decision Support Systems (DSS) with their huge collection of information available in On-line Analytical Processing (OLAP) application. Current and previous data from numerous external data sources can be stored in this large database [1]–[3] . The queries that are created on DW system commonly have a complicated nature that comprises a number of join operations incurring high computational overhead. Usually, they include multi-dimensional grouping and aggregation operations. By contrast, queries applied in OLAP are relatively more sophisticated than those that are employed in traditional applications. Due to the massive volume of DWs and the complicated quality OLAP queries, for one thing, the execution cost of the queries have to be raised and the performance, and for another, the productivity of DSS are influenced dramatically. [4]–[6]

A large portion of database software solutions for real-world applications today depend on normalized logical data models. Even though normalization follows a simpler implementation process, it imposes several limitations in supporting business application requirements [7] .

Previous research [8] provides proof for data retrieval accelerating properties of denormalization; nonetheless, a demerit of denormalization includes its weak support of potentially frequent updates. Indeed, data warehouses entail relatively fewer data updates and data are usually retrieved only in most transactions [2]. That is to say, an application of denormalization strategies is most appropriate for data warehouses systems as long as they experience infrequent updating.

It is possible to create denormalization relationships for a database through a number of methods, some of which include Pre-Joined Tables, Report Tables, Mirror Tables, Split Tables, Combined Tables, Redundant Data, Repeating Groups, Derivable Data and Hierarchies [9], [10] .

The study emphasizes the use of Hierarchical denormalization to raise the efficiency of the performance in DW. Inasmuch as designing, representing and traversing hierarchies have complicated processes compared to the normalized relationship, integrating and summarizing the data remains as the most commonly utilized approach to minimize the query response time [11], [12] . Hierarchical denormalization can especially be advisable when dealing with the growth of star schemas found in a majority of data warehouse implementations [7], [12] .

At this point, what remains a problem is that, following the conventional wisdom, all designs that include a relational database should be supported by normalized logical data models [13] . Beyond this, even though the system can be enhanced by decreasing table joins is not new, denormalization cannot be advised since it is a highly demanding task of administrative devotion involving the documentation structure of the denormalization assessments, data validation, and schedules for migrating of data, to name but a few.

By contrast, a considerable number of studies in the related literature contend that denormalization can optimize performance by creating a more flexible data structure for users [7], [12] . Data can be arranged into a well-balanced structure through normalization in order to optimize data accessability, yet such a procedure entails certain deficiencies that result in turn in a lower system performance [14], [16], [17] . The evident point is that IT academicians and

practitioners share great diversities concerning their view towards database design. Certainly, denormalization is a process that has not attracted interest in the academic world but is a reasonable strategy in practical database community.

The paper has the following contentions to make: 1. Hierarchical technique is strongly recommended to denormalize the data model and structure in a data warehouse involving several join operations. 2. Query processing time in the structure provided by hierarchical denormalizing is significantly shorter than normalized structure even though data maintenance in a normalized structure is easier in comparison to a denormalized structure. 3. Query processing performance is influenced considerably by building Bitmap Index on columns involved by denormolized implementation.

The paper includes five sections beginning with a brief review of the related studies followed by an overview of previous research conducted on normalization, denormalization and hierarchical denormalization in section 2. A case study and performance methodology on a set of queries are proposed in the next section in order to compare the performances of normalized and denormalized table structures. The discussion of the results of the study comes in the next part preceding the conclusion, or Section 5.

II. Background

A. Normalization

Normalization is a technique, first mentioned by Codd [18], and has been deployed by Date [8] for determining the optimal logical design to simplify the relational design of an integrated database, based on the groupings of entities to improve performance in storage operations.

While an entirely normalized database design decreases the inconsistencies, it can cause other 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 [19] .

B. Denormalization

Some pertinent explanations exist for denormalizing of a relational design to improve its performance. An awareness of these explanations will prove helpful to identify 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 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 taken care of; (iv) entities which need to be extracted in different ways during the same time and (v) to be aware about relational database which can brings better performance and enhanced access options that may increase the possibility for denormalization.

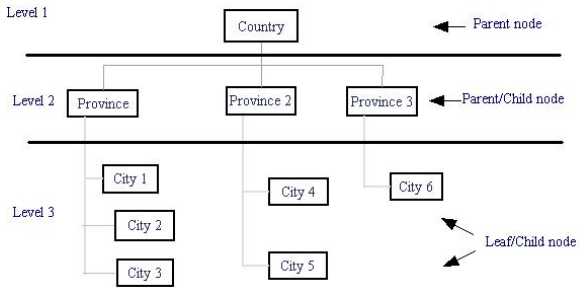

Fig. 1. Hierarchy (balanced tree)

Most OLAP processes within the data warehouse 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 [7], [15], [20], [21] .

C. 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 [11], [12], [21] .

To illustrate what the structure of hierarchy is, we show an example by [12] . *" 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. A parent (except a root node) may be a child, and a child (except a leaf node) may also be a parent."* Fig 1 shows such a hierarchy.

Hierarchies are essential aspects of DWs. Thus, supporting different kinds 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 [22] . We outline several types of hierarchies in the data warehouse environment as follows [12] .

1) *" Balanced tree structure: In this structure, hierarchy has a consistent number of levels and each level can be named. Each child has 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).*

TABLE I
BASIC BITMAP INDEX ADOPTED BY[24]

RowId	C	B0	B1	B2	B3
0	2	0	0	1	0
1	1	0	1	0	0
2	3	0	0	0	1
3	0	1	0	0	0
4	3	0	0	0	1
5	1	0	1	0	0
6	0	1	0	0	0
7	0	1	0	0	0
8	2	0	0	1	0

4) *"Complex tree structure: In this hierarchy, a child may have multiple parents.*
5) *"Multiple tree structures for the same leaf node [12] . "*

D. Bitmap index

Bitmap index is built to enhance the performance on various query types including range, aggregation and join queries. It is used to index the values of a single column in a table. Bitmap index is derived from a sequence of the key values which depict the number of distinct values of a column. Each row in Bitmap index is sequentially numbered starting from integer 0. If the bit is set to "1", it indicates that the row with the corresponding RowId contains the key value; otherwise the bit is set to "0".

To illustrate how Bitmap indexes work, we show an example which is based on the example illustrated by E.E-O'Neil and P.P-O'Neil [25] . " *Table I shows a basic Bitmap index on a table containing 9 rows, where Bitmap index is to be created on column* C *with integer values ranging from 0 to 3. We say that the column cardinality of* C *is 4 because it has 4 distinct values. Bitmap index for* C *contains 4 bitmaps, shown as* B0*,* B1*,* B2 *and* B3 *corresponding to the value represented. For instance, in the first row where* RowId =0*, column* C *has the value 2. Hence, in column B2, the bit is set to* "1"*, while the rest of bitmaps bits are set to* "0" *. Similarly, for the second row, bit of* B1 *is* "1" *because the second row of C has the value 1, while the corresponding bits of* B0*,* B2 *and* B3 *are all* "0" *. This process repeats for the rest of the rows [25].*"

III. RELATED WORKS

Bock and Schrage [19] 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 [13] has outlined the advantages of normalization process, such as, easing the designs process and physical implementation, reducing data redundancy and protects data from update and deleting anomalies. He/she also has shown 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 agrees that denormalization must be used with care by understanding of how the data will be used. He has also confirmed that denormalization can be used to improve system response time without redundant data storage or incurring difficulty of data maintenance.

Sanders and Shin [20] 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 was limited due to the lack of sufficient comparison of computation costs between the normalized and denormalized data structures.

Shin and Sanders [23] 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 evaluated their effects by illustrating the conditions which strategies are most effective. They have concluded that denormalization methods may receive positive effects for database performance such as Data warehouse.

Morteza Zaker and others [24] have discussed that although creating indexes on database is usually regarded as a common issue, it plays a key role in the query performance, particularly in the case of huge databases like a Data Warehouse where the queries are of complicated and ad hoc nature. Should an appropriate index structure be selected, the time required for query response will decrease extensively. Their empirical results have indicated that how the Bitmap index can be more expeditious than B-tree index on a large dataset with multi-billion records.

IV. METHODOLOGY

In order to compare efficiency of denormalization and normalization processes and analysis the performance of these data models, 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) has approximately 3.36 billion records. These records are randomly generated using PL/SQL Block by Oracle11G tools. These tables can be categorized into

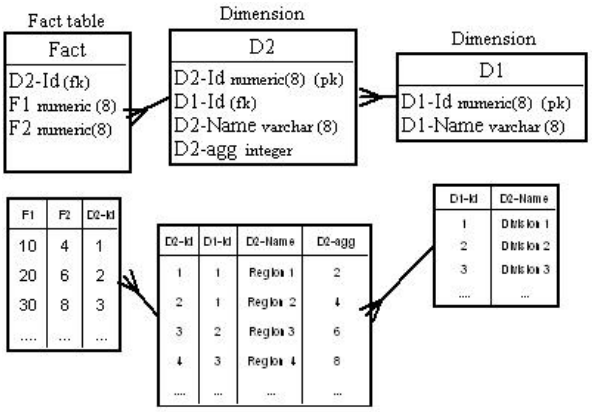

Fig. 2. Schema 1 with normalized design

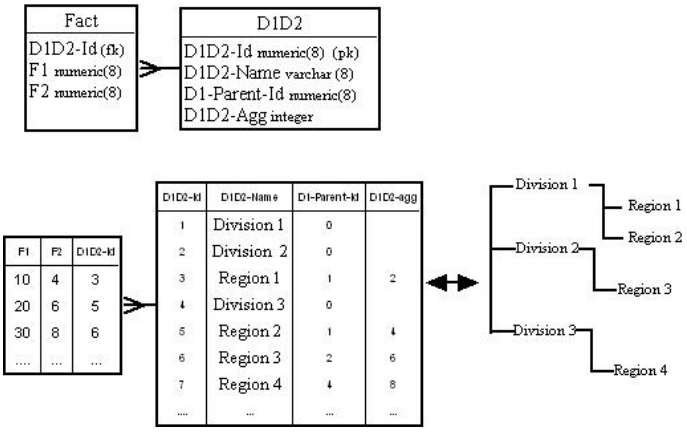

Fig. 3. Schema 2 with denormalized design

two database schemas, Schema 1 and Schema 2 which are portrayed in Fig 2 and Fig 3 respectively. In Schema 1, the tables are 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 contains 5 indexed columns while Schema 2 includes 3 indexed attributes.

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

TABLE II
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. [25]	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"

A. Query Set

The Set Query Benchmark has been used for frequent-query application much like a Star-Schema in the data warehouse design [26], [27] . 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 [26], [27] . Briefly, we describe all of our selected SQL queries used for our performance measurements as indicated in Table II. Basically, for each query, we used suffix 'A' to represent query on Schema 1 and suffix 'B' to represent query on Schema 2.

B. Experimental Setup

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

TABLE III
INFORMATION ABOUT THE TEST SYSTEM

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

V. RESULTS AND DISCUSSIONS

A. Query Response Time

In this section, we show and discuss 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 IV.

TABLE IV
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

B. 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 was applied on dimension with 3.6 billion records and Q1A was 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 IV, 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 on Set-Query 2. As it can be observed, 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 is 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 a result, this does not scale very well as data volumes increase. Even though, there is one implementation of Bitmap index (FastBit) which can support these queries directly [16], [17], [25], but Oracle 11G has not utilized this method of implementation.

In summary, Fig 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 index techniques especially Bitmap index technique.

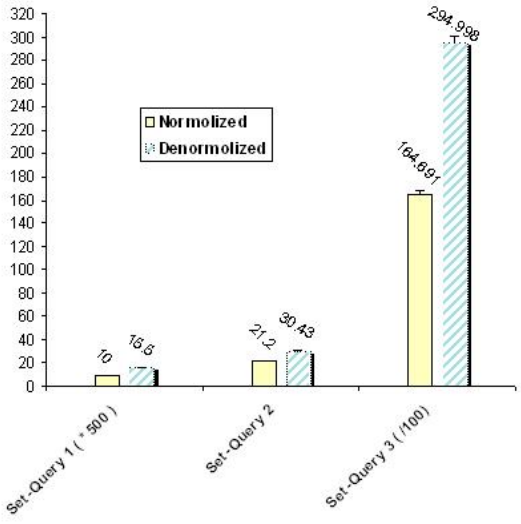

Fig. 4. Query elapse times for one dimension queries

C. 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 results in slower system response time and performance will degrade as the table size increases. This query shows that if the First Normal Form tables is transformed by denormalization method, the query response time can be extraordinary decreased as the number of join operations is reduced.

Data aggregation is a frequent operation in data warehouse and OLAP environments. Data are either aggregated on the fly or pre-computed and stored as materialized views [21] . Since, data aggregation and data computation can really grow to be a complex process through time, having a good and well-defined architecture support dynamic business environments has 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 that of Q5A. Thus, we claim that two of the components 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 decreased compared to 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.

Fig 5 shows the query elapse time for multi-dimensional hierarchical queries which have been applied on first and second schemas. This Fig shows that using hierarchical de-normalization method can improve system response time when the queries are unanticipated ad hoc queries.

VI. Conclusion

The present experimental study was an evaluation of denormalization following necessary guidelines in a data warehouse design. The researchers demonstrated how set queries for the measurement of denormalization can affect using hierarchical implementation. With the objective of elucidation of the effects of hierarchical denormalization in further detail, we attempted to prepare a real test environment to measure query retrieval times, system performance, ease of utilization and bitmap index effects. The tests made a comparison between normalized and hierarchical denormalized data structures in terms of aggregation and computation costs. The findings confirm that most probably hierarchical denormalization have the capability of improving query performance since they can

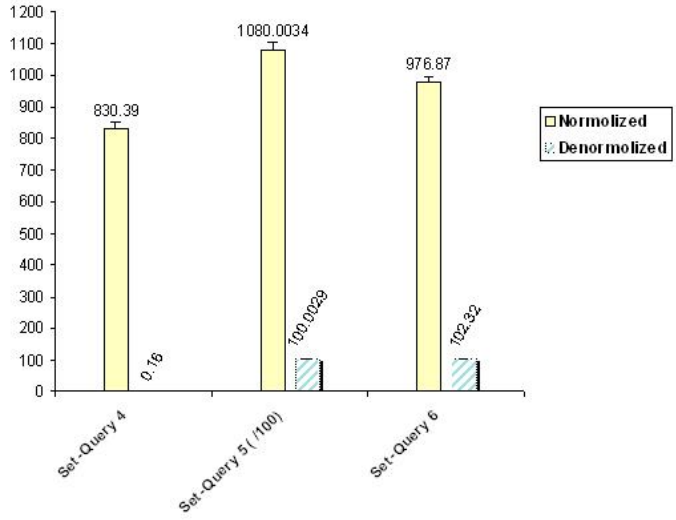

Fig. 5. Query elapse times for multi dimension queries which are involved by join operations

reduce the query response times when the data structure in Data warehouse is engaged in several joins operations. The results can help researcher in the future to develop general guidelines which can be applicable to a majority of database designs. Finally, hierarchical denormalization can be regarded as a fundamental phase in a data warehouse data modeling which is rarely dependant on applications requirements in which data warehouse does not frequently have to be updated.

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] J. Mamcenko and I. Sileikiene, *I*ntelligent Data Analysis of E-Learning System Based on Data Warehouse, OLAP and Data Mining Technologies. Proceedings of the 5th WSEAS International Conference on Education and Educational Technology, Tenerife, Canary Islands, Spain, December 16-18, 2006 pp. 171

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

[5] C. DELLAQUILA and E. LEFONS and F. TANGORRA, *D*esign and Implementation of a National Data Warehouse. Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, February 15-17, 2006 pp. 342-347

[6] 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.

[7] 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

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

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

[10] 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

[11] 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.

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

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

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

[15] M. Klimavicius, *D*ata warehouse development with EPC. Proceedings of the 5th WSEAS International Conference on Data netwrks, Communications and Computers, Romaina 2006

[16] 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

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

[18] 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.

[19] 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

[20] 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.

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

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

[23] 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

[24] M. Zaker and S. Phon-Amnuaisuk and S. Haw, *I*nvestigating Design Choices between Bitmap index and B-tree index for a Large Data Warehouse System. Proceedings of the 8th WSEAS International Conference on APPLIED COMPUTER SCIENCE (ACS'08) Venice, Italy, November 21-23, 2008, pp.123

[25] 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.

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

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

Morteza Zaker is a research student in Advanced Databases and Data Warehouse area. He is a system analyst and skilled in database with more than one decade.

Somnuk Phon-Amnuaisuk received his B.Eng. from King Mongkut Institute of Technology (Thailand) and Ph.D. in Artificial Intelligence from the University of Edinburgh (Scotland). He is currently an associate Dean for the faculty of Information Technology, Multimedia University, Malaysia where he also leads the Music Informatics Research group. His current research works span over multimedia information retrieval, polyphonic music transcription, algorithmic composition, Bayesian networks, data mining and machine learning.

Dr.Su-Cheng Haw's research interests are in XML Databases and instance storage, Query processing and optimization, Data Modeling and Design, Data Management, Data Semantic, Constraints and Dependencies, Data Warehouse, E-Commerce and Web services