A Mixed Fragmentation Approach for Solving Big Dimension Problem in Data Warehouse

M. Thenmozhi[1]
Assistant Professor
Department of Computer Science and Engineering
Pondicherry Engineering College
Pondicherry, India

Dr. K. Vivekanandan[2]
Professor
Department of Computer Science and Engineering
Pondicherry Engineering College
Pondicherry, India

***Abstract* - The data warehouse partitioning helps to reduce number of disk I/Os for query execution by minimizing access to irrelevant data. As the queries executed over data warehouse involves fact and dimension tables it is necessary to partition a fact table with respect to a chosen dimension table using referential partitioning technique. Existing works adopt horizontal partitioning approach for referential partitioning as data warehouse contains millions of records. But in real scenarios there may be big dimension which involves larger set of attributes. Hence it is essential to partition the dimension both vertically and horizontally. In this paper we propose an approach which adopts mixed fragmentation depending on the chosen dimension table. When the dimension table is partitioned the number of fragments generated may be large hence the proposed work applies hybrid heuristic algorithm to select optimal set of fragment such that the query cost is minimized.**

Keywords* - *Data Warehouse Partitioning, Horizontal Fragmentation, Vertical Fragmentation, Mixed Fragmentation.

I. INTORODUCTION

Data warehouse assist intelligent decision making that can significantly improve the functioning of an organization [4]. The warehouse data is access by an OLAP (Online Analytical processing) tools in order to present the data in a multidimensional way. The multidimensional model of the data warehouse is typically represented as star schema [10]. Facts, measures, dimensions are the different elements of the star schema. A fact is a focus of interest for the decision-making process, measures are continuously valued (typically numerical) attributes which describe the fact from different points of view and dimensions are discrete attributes which determine the minimum granularity adopted to represent facts. For instance, each fact sale is measured by its revenue and typical dimensions for the sale fact are product, store and date.

Data warehouse integrates massive amounts of data from multiple sources. Hence querying the data warehouse is very complex as it involves lot of joins and aggregate operations. Partitioning is a design technique applied to data warehouse for dividing a single table into two or more partitions such that the combination of the partitions provides the original data without any loss of information. Partitions can dramatically reduce the amount of data retrieved from disk and shortens processing time, thus improving query performance and optimizing resource utilization. *Referential partitioning* [3] is a new partitioning option that allows the partitioning of two related tables to be based on a referential constraint. When there is a parent-child relationship between two tables as in data warehouse having fact-dimension relationship, the parent table can be defined with its reference partitions. Subsequently, the child table can be equi-partitioned by defining the child table to inherit the partitioning key from the parent table, without the need to duplicate the partition key columns. Referential partitioning involves selection of dimension tables based on which a fact table is partitioned. The existing works on referential partitioning uses horizontal partitioning approach. The limitation of these approaches is that in real scenario the dimension table may be wider i.e, it may involve large set of attributes which is called big dimension [5] [13]. Hence applying horizontal partitioning alone might not be effective. Another problem in referential partitioning is when the dimension table is partitioned based on the selected attributes the number of fragments or partitions to be managed in the underlying database may be very complex. Hence the existing work on fragmentation selection adopted evolutionary approach such as genetic algorithm to select optimal set of fragments. As genetic algorithms may need many generations to get a success so running at high performance is important for them. In order to overcome the limitations discussed above the proposed approach makes the following contributions:

- Adopting mixed fragmentation for referential partitioning technique based on chosen dimension table.
- Applying k-means clustering for vertical fragmentation.
- Applying hybrid heuristic such as genetic algorithm with tabu search for horizontal fragmentation selection.

II. RELATED WORK

In [5] they focus on the data partitioning approach to partition the facts tables through all nodes and replicate the dimension tables based on this partitioning. To deal with the limitation to the applicability of the partitioning technique to data warehouses with big dimensions they propose DWS (Data Warehouse Striping) technique. It allows the distribution of large data warehouses through a cluster of computers. With the proposed strategy the performance speed up and scale up obtained in the DWS technique are not

affected by the presence of big dimensions. In [1] they discuss the problem of selecting an optimal fragmentation schema for a data warehouse. They also discuss the benefits of fragmenting fact table based on the partitioning schemas of dimension tables. In this paper, they present a genetic algorithm for schema partitioning selection problem when fact table is partitioned with respect to dimension table. The proposed algorithm gives better solutions since the search space is constrained by the schema partitioning. In [8] the authors propose a new method for horizontal partitioning of relations based on predicate abstraction. The method is formal and compositional: arbitrary fragments of relations can be partitioned with arbitrary number of predicates. They use a genetic algorithm to generate an appropriate solution for this optimization problem. In [7] the authors focus on the horizontal fragmentation of data warehouses. They propose an algorithm, which is aimed at improving the performance of drill-down and roll-up queries by horizontally fragmenting data warehouses organized in different levels of aggregation. The method allows multiple dimensions to be used as a basis for the fragmentation and moreover the algorithm also explores the hierarchical structure of these dimensions. The authors in [13] present the design methods for modelling big dimensions. They use horizontal partitioning, vertical partitioning and hybrid partitioning for partitioning the big dimension. The design methods proposed is formalized by an algorithm that describes the modelling process from OWL ontology to a data warehouse schema. The main difference between [13] and our proposed approach is that for optimizing vertical and horizontal fragmentation we have applied k-means clustering and hybrid genetic with tabu search respectively.

III. PROPOSED APPROACH

In this section we propose an approach to apply referential partitioning over the given data warehouse schema based on the chosen dimension table. The proposed method offers the data warehouse designer to partition a big dimension using mixed fragmentation. Consider a data warehouse modelled as a star schema which consists of set of D dimension tables and a fact table F which need to be referentially partitioned in order to minimize the cost of given set of queries Q. The DWA (Data Warehouse Administrator) can maintain a maximum of N number of fragments in the underlying database. The steps to be followed for a referential partitioning technique are: i) Choosing dimension tables ii) Selecting attributes for fragmentation iii) Applying single table mixed partitioning technique to partition the dimension table iv) Choosing optimal fragments by using hybrid heuristic algorithm v) Apply referential partitioning to fact table based on partitioned dimension tables vi) Compute the overall query cost for the given query set Q. Fig. 1 represents the various steps involved in the proposed approach. Following sections describes the steps in detail.

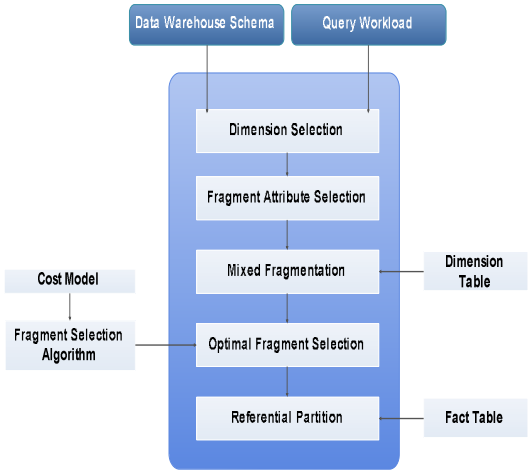

Fig. 1 Proposed Approach

A. *Choosing Dimension Table*

As the data warehouse may involve multiple dimensions it is necessary to select dimension table(s) in order partition the fact table. When the data warehouse schema contains one or more big dimension consisting of numerous attributes then it is consider for partitioning. In the given data warehouse for each dimension we obtain its attributes. When the total number of attributes of a dimension class exceeds a given threshold then we consider the particular dimension as big dimension.

B. *Fragmentation Selection*

The management and maintenance of fragments is very complex if different combinations of attributes and predicates of the dimension(s) are retained. If M_i is the number of fragments of the dimension table D_i, and K is the number of dimension tables fragmented, then the total number of fragments of the fact table is:

$$N = \prod_{i=1}^{K} Mi \ .$$

In order to have limited set of partitions of fact table it is necessary to select optimal set of fragments from the large set of fragments available. This is termed as fragmentation selection problem. In the proposed work we address this issue by applying hybrid heuristic approach for mixed fragmentation techniques.

C. *Mixed Fragmentation Approach*

When the data warehouse involves big dimension containing numerous attributes it is essential to apply mixed fragmentation approach i.e. vertically fragment the table based on the columns (attributes) followed by a horizontal fragmentation on each vertical fragment.

A vertical fragmentation of a relation R produces fragments F1, F2, . . . , Fn each of which contains a subset of R's attributes. Compared to horizontal fragmentation, vertical fragmentation is inherently more complicated. When the horizontal partition consisting of n simple predicates the

possible minterms is 2^n and some of them can be ruled out by existing constraints. Whereas in vertical partitioning for m non-primary key attributes, the number of possible fragments is equal to B(m) (= the mth Bell number), i.e., the number of partitions of a set with m members. For example $B(15) = 10^9$. As the number of fragments is more it is necessary in this case to find optimal solutions (fragments). As in vertical fragmentation attributes usually accessed together are paced in one fragment and hence there is a need for some measure that would define more precisely how closely the attributes are related. The information about the attributes can be obtained from queries and collected in the Query Attribute Matrix (QAM).

For the given query workload Q which is executed over relation R consisting of attributes (A1, . . . , An). The QAM denotes which query uses which attribute. The rows represent workload queries and columns represents attributes from Q. In general term QAM (i, j) is set to one if Qi includes attribute Ai and to zero otherwise as shown in Fig. 2.

	A1	A2	A3	A4
Q1	1	0	0	0
Q2	0	0	1	1
Q3	0	1	1	0

Fig. 2 Query Attribute Matrix (QAM)

Our next step is to derive fragments that optimize data access for a given set of queries. As vertical fragments are built from attributes it is necessary to cluster the attributes with respect to query workload. By using the QAM we can derive the clusters (fragments) by applying k-means algorithm [12] widely adopted for clustering. K–Means clustering is a method of cluster analysis which aims at partitioning of n observations into k clusters. This algorithm inputs vectors of object attributes (columns of QAM) and the k value. It attempts to find the centers of natural clusters in source data by minimizing total intra-cluster variance. Having k as an input parameter we can limit the number of clusters (fragments) of the data warehouse.

Based on the vertical fragments constructed using the previous step, the next step involves partitioning each fragment further applying horizontal fragmentation. For each vertical fragment it is necessary to select the attributes and its corresponding predicates based on which the horizontal partitioning is performed. Given a relation (fragment) R consisting of attributes [A1, A2, …, An], a simple predicate pi is given by:

pi = Ai θ Value. Where, θ is given by {=,<,≤,>,≥,≠}, Value € Di where, Di is the domain of an attribute Ai. For a particular relation R we define Pr = {p1, p2, …,pm}.

By using the query work load we can obtain the fragmentation attributes and it corresponding predicates. The number of fragments generated for a given table (fragment) is exponential to the number of predicates chosen for partitioning. Hence it is necessary to select only minimal set of predicates for partitioning. For this purpose we use com_min algorithm [14] which has been proposed for database fragmentation. The algorithm ensures that only complete and minimal set of predicates are chosen from the given set of predicates. A set of predicates Pr is said to be complete if and only if the accesses to the tuples of the fragments defined on Pr requires that two tuples of the same fragment have the same probability of being accessed by all the queries. The set of predicates Pr is said to be minimal if it is relevant in determining a fragmentation. A predicate pi is relevant to a set of predicates Pr if there are two tuples t and t0 of a fragment F, such that t satisfies pi and t0 does not satisfies pi. Using the resultant set of predicates a fragmentation schema can be constructed. As the possible fragments produced are large in number the fragmentation selection is carried out by using genetic algorithms (GAs) with tabu search (TS).

GAs are a special class of global optimisers, based on the theory of evolution. The genetic algorithm developed by Holland to optimise a function F(x), where x is a vector representing individual solutions [6]. It starts with the initial solution called population and it is filled with chromosome. Each element in chromosome is called gene. Using evolutionary inspired operators such as fitness, crossover and mutation, the best solutions are modified and passed on to the next generation. At any iteration, a fitness value is calculated for each of the current individuals. A GA is terminated after a certain number of iterations or if a certain level of fitness value has been reached.

1) *Representation of Solution:* In order to apply the optimization algorithm it is necessary to represent the solution space. The multidimensional array representation of the initial fragmentation for a sample attributes is given in Fig.3. Each cell in the array represents the subdomains or predicates of the chosen fragmentation attributes.

Gender	M	F	
Age	<30	31-50	>50

Fig. 3 Fragmentation Schema

2) Fitness Function: The quality of a solution is given by the fitness value, which is calculated by the evaluation function. The cost for each fragmentation scheme (estimating the number of inputs outputs required for executing the set of queries) is computed using the cost model. Query Cost = {(Size of each fact fragment)*(Length of each instance of fact)}/Page size of the disk.

3) Genetic Operators: The crossover operator involves the swapping of genetic material (bit-values) between the two parent chromosomes (fragmentation schema). These two parents produce two offspring. The second genetic operator, mutation, can help GA to get a better solution in a faster time. In this model, relocation is used as a key mechanism for mutation. Hence the mutation can introduce diversity without disturbing the sequence.

GAs are able to locate promising regions for global optima in a search space, but sometimes have difficulty finding the exact minimum of these optima. From the literature we find that there

are the solution obtained from GAs can be improved by using a second optimization approach. For the given fragmentation selection problem we use Tabu Search (TS) [9] to refine the solution. We find that GAs are inspired by the process of evolution and work on a group of solutions at a time, whereas TS is based on concepts from artificial intelligence and operates on a single solution at a time. It uses problem-specific operators to explore the given search space and a memory called tabu-list to remember the regions already visited. When the TS explore new areas it can overcome local minima and hopefully reach the global optimum. The steps of hybrid GA with TS algorithm is shown in Fig. 4.

The algorithm begins by selecting initial set of chromosomes (fragmentation schema). Following this, TS is applied to each chromosome in the initial set. As the initial population is improved by TS genetic operators are applied over it. The crossover and mutation operations are performed on the group of chromosomes to generate new chromosomes. The fitness for each chromosome is computed and the fittest chromosomes are retained in the next generation. The proposed algorithm continues by the process of several TS followed by GA steps. The computation of the algorithm stops when maximum number of consecutive GA based steps that do not improve objective criterion value is reached.

1. Generate randomly a set of solutions which forms the initial population
/* Apply Tabu Search */
2. Refine each solution by the neighbourhood routine with respect to fitness value.
   i) A neighbourhood solution is obtained by modifying the value of one element from the solution.
   ii) The neighbourhood solutions are not accepted until they satify given constraint
   iii) A tabu list is updated containing all the fitness values of the solutions that have been explored.
   iv) A new neighbourhood is explored only when its fitness value does not exist to the tabu list.
3. Repeat step 2 until there is no improvement of the best fitness value.
4. Replace the solution by its best neighbourhood.
/* Apply Genetic Algorithm*/
5. Choose two solutions to produce new chromosomes using genetic operators: parent selection and crossover.
   i) These new solution are accepted when they verify the given constraint.
6. Create a new chromosomes using genetic operator: mutation.
7. Update a tabu list of the best chromosomes.
8. Repeat step 1 until there is no improvement of the best chromosome.

Fig. 4 Hybrid Heuristic Algorithm

D. *Fact Fragmentation*

The fact table can be finally be fragmented based on the horizontal fragments obtained in the previous step. As in the proposed work we applied optimization measures for vertical and horizontal fragmentation selection, the number of fact fragments produced is manageable in the underlying database.

IV. EXPERIMENTAL STUDY

We use star schema of the classical Inmon's sales data mart [11], which consists of a fact table and four dimension tables (Date Dimension, Product Dimension, Store Dimension and Customer Dimension as shown if Fig. 5 in order to analyze and test the performance of the proposed approach. Here the Customer dimension consists of 53 attributes, and possibly holds millions of dimension record which is chosen as the big dimension for partitioning. The data warehouse has been populated using synthetic data set. This warehouse has been installed under oracle 11g on a Pentium 1.8 GHz (with a memory of 256 Mo, 60Go) running under windows XP pro. Here we have considered a set of 8 OLAP queries available for the star schema.

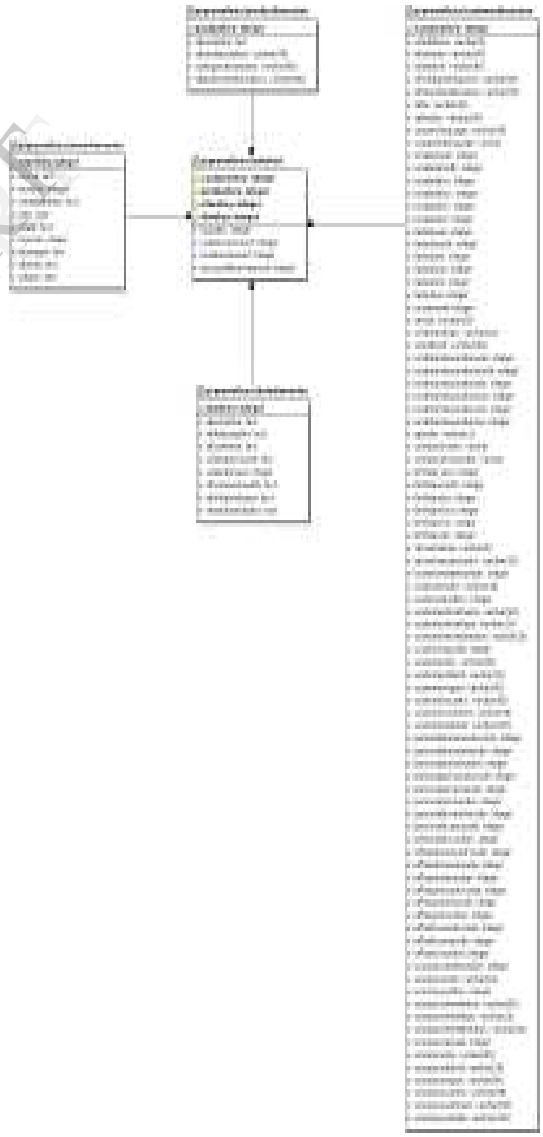

Fig. 5 Star Schema

Vertical partitioning is performed over customer dimension by splitting the big dimension table into multiple tables, each of which contains different number of columns. To apply k-means clustering we construct the QAM i.e query attribute matrix using the customer dimension attributes (A1, A2, A3, A4, A5, A6, A7, A8, A9, A10….A53) =(FirstName, LastName, Salutation, FormalGreetingName, Gender, BirthdayYear, Company, Email, PersonalPhoneNumber, OfficePhoneNumber, StreetName, City, Country, Ethnicity……). Fig. 6 shows the QAM constructed for sample attributes of the customer dimension. After running k-means clustering over the given QAM we obtain two clusters (vertical fragments) with VF1 (A1, A2, A3, A4, A5, A6) and VF2 (A8, A9, A10, A11, A12, A14) for the given matrix given in Fig. 7.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14
Q1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Q2	0	0	0	0	0	0	0	0	0	0	1	1	0	1
Q3	0	0	0	0	0	0	0	1	0	0	1	0	0	0
Q4	0	0	0	0	0	0	0	0	1	1	0	0	0	0
Q5	1	1	0	0	0	1	0	0	0	0	0	0	0	0
Q6	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Q7	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Q8	1	1	0	0	0	1	0	0	0	0	0	0	0	0

Fig.6 A sample QAM for customer dimension

VF1 (Customer)	VF2 (Customer)
FirstName LastName Salutation FormalGreetingName Gender BirthdayYear	Email PersonalPhoneNumber OfficePhoneNumber StreetName City Ethnicity

Fig. 7 Vertical Fragments for Customer Dimension

In order to horizontally partition each vertical fragment we need to select the attributes and the corresponding predicates to be used for fragmentation. From the given query work load we obtain Gender and BirthdayYear as the fragmentation attributes for VF1 (Customer). The predicates of the attributes is obtained (using Com_Min algorithm) is give in Fig. 8. Based on the obtained predicates the fragmentation schema of the VF1 (Customer) is shown in Fig. 9.

Gender	M	F		
BirthdayYear	<1961	1961-1971	1971-1981	>1981

Fig. 8 Predicates for Horizontal Fragmentation

Gender	1	2		
BirthdayYear	1	2	3	4

Fig. 9 Fragmentation Schema for Horizontal Fragmentation

Based on the fragmentation schema given in Fig the total number of fragments is 6. In order to find an optimal fragmentation schema which generates fragments such that the overall query cost is minimized we apply hybrid heuristic algorithm i.e genetic algorithm with tabu search. As the fragmentation schema given in Fig. 10 gives the minimal query cost it is obtained as the optimal fragment.

Gender	M	F	
BirthdayYear	<1961	1961-1971	>1971

Fig. 10 Fragmentation Schema for Horizontal Fragmentation

For partitioning the fact table we first apply horizontal fragmentation given in Fig. 11 to the customer dimension based on the obtained fragmentation schema. Based on this the fact table is referentially partitioned which is given in Fig. 12.

```
CREATE TABLE Customer
(customer_key NUMBER, FirstName Varchar2(20)……., gender CHAR,
birthdayyear Number)
PARTITION BY RANGE (birthdayyear)
SUBPARTITION BY LIST (gender)
SUBPARTITION TEMPLATE (SUBPARTITION Female VALUES ('F'),
SUBPARTITION Male VALUES ('M'))
(PARTITION Cust_1961 VALUES LESS THAN (1961),
PARTITION Cust_1961_71 VALUES BETWEEN (1961-1971)
PARTITION Cust_1971 VALUES GREATER THAN (1971));
```

Fig. 11 Customer Dimension Partition

```
CREATE TABLE Sales (customer_Key NUMBER NOT NULL,
product_key NUMBER NOT NULL,
date_key Number NOT NULL, store_key Number NOT NULL,
constraint Sales_customer_fk foreign key(customer_key)
references CUSTOMER(customer_key))
PARTITION BY REFERENCE (Sales_customer_fk);
```

Fig. 12 Fact Table Partition

Fig. 13 compares the total query execution time (in ms) of different fragmentation technique. From Fig. 13 we infer that the proposed mixed fragmentation approach gives minimal query execution time for the given scenario. The comparison of overall query cost for the given set of queries when applied to different fragmentation techniques is given in Fig. 14. The proposed mixed fragmentation performs better when compared to the horizontal or vertical technique applied for big dimension.

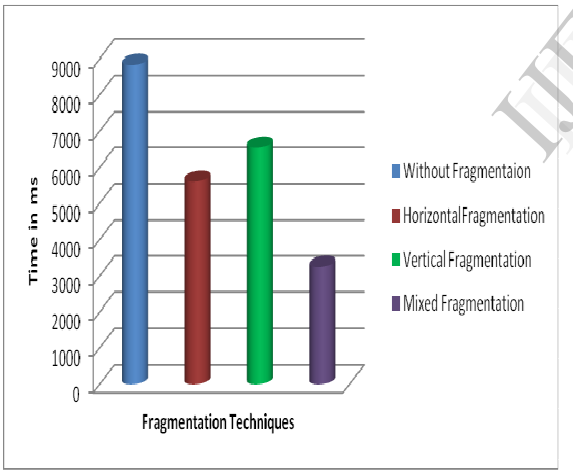

Fig. 13 Comparison of Overall Query Execution Time

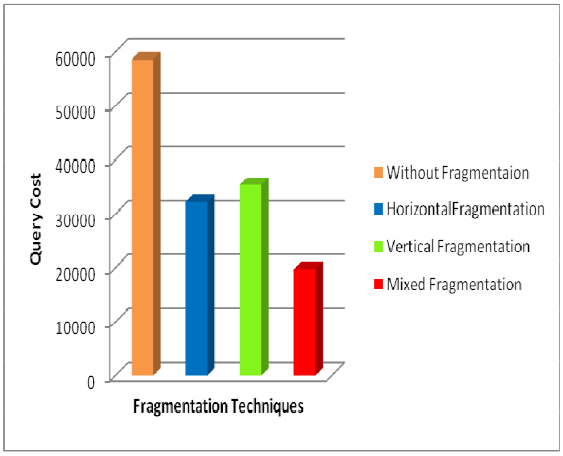

Fig. 14 Comparison of Overall Query Cost

Fig. 15 shows the comparison of individual query execution time for the naïve mixed fragmentation to the proposed optimized mixed fragmentation technique. From the Fig. 16 we observe that the proposed approach helps in reducing the query cost when compared to the existing naïve mixed fragmentation approach. From Fig. 16 we observe that individual query cost is also minimized by the proposed approach.

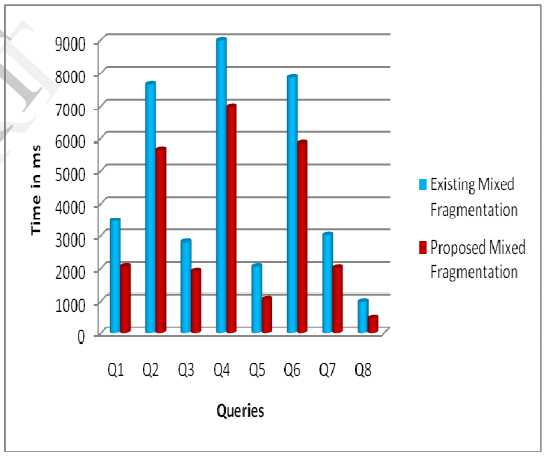

Fig. 15 Comparison of Existing with Proposed Mixed Fragmentation for Query Execution Time

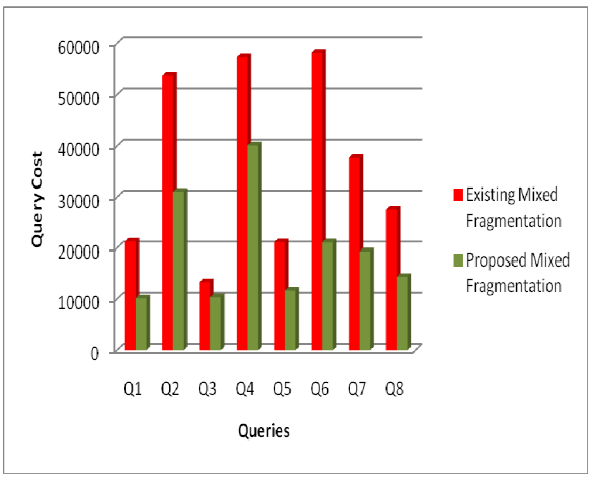

Fig.16 Comparison of Existing with Proposed Mixed Fragmentation for Query Cost

V. CONCLUSIONS

A data warehouse system stores huge volume of data related to a particular business and hence facilitates trend analysis. The query performance is an important requirement of such system. Thus partitioning of fact and dimension table help in minimizing the query execution time and query cost. In data warehouse scenario referential partitioning performs better compared to single table partitioning. When data warehouse involves big dimension consisting of millions of rows and large set of attributes then it is necessary to partition the fact table with respect to the big dimension. In this paper an optimized mixed fragmentation approach is applied which reduces the query execution time and query cost compared to naïve mixed fragmentation approach.

REFERENCES

[1] Bellatreche, Ladjel, and Kamel Boukhalfa. "An evolutionary approach to schema partitioning selection in a data warehouse." In Data Warehousing and Knowledge Discovery, pp. 115-125. Springer Berlin Heidelberg, 2005.

[2] Bellatreche, Ladjel, Kamalakar Karlapalem, Mukesh Mohania, and Michel Schneider. "What can partitioning do for your data warehouses and data marts?." In Database Engineering and Applications Symposium, 2000 International, pp. 437-445. IEEE, 2000.

[3] Bellatreche, Ladjel, Kamel Boukhalfa, Pascal Richard, and Komla Yamavo Woameno. "Referential horizontal partitioning selection problem in data warehouses: Hardness study and selection algorithms." International Journal of Data Warehousing and Mining (IJDWM) 5, no. 4 (2009): 1-23.

[4] Berson, Alex, and Stephen J. Smith. Data warehousing, data mining, and OLAP. McGraw-Hill, Inc., 1997.

[5] Costa, Marco, and Henrique Madeira. "Handling big dimensions in distributed data warehouses using the DWS technique." In Proceedings of the 7th ACM international workshop on Data warehousing and OLAP, pp. 31-37. ACM, 2004.

[6] Davis, Lawrence, ed. Handbook of genetic algorithms. Vol. 115. New York: Van Nostrand Reinhold, 1991.

[7] de Aguiar Ciferri, Cristina Dutra, Ricardo Rodrigues Ciferri, Diogo Tuler Forlani, Agma Juci Machado Traina, and Fernando da Fonseca de Souza. "Horizontal fragmentation as a technique to improve the performance of drill-down and roll-up queries." In Proceedings of the 2007 ACM symposium on Applied computing, pp. 494-499. ACM, 2007.

[8] Dimovski, Aleksandar, Goran Velinov, and Dragan Sahpaski. "Horizontal partitioning by predicate abstraction and its application to data warehouse design." In Advances in Databases and Information Systems, pp. 164-175. Springer Berlin Heidelberg, 2011.

[9] Glover, Fred, and Manuel Laguna. Tabu search. Springer US, 1999.

[10] Golfarelli, Matteo, Dario Maio, and Stefano Rizzi. "The dimensional fact model: a conceptual model for data warehouses." International Journal of Cooperative Information Systems 7, no. 02n03 (1998): 215-247.

[11] Inmon, William H. Building the data warehouse. John wiley & sons, 2005.

[12] Kanungo, Tapas, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. "An efficient k-means clustering algorithm: Analysis and implementation." Pattern Analysis and Machine Intelligence, IEEE Transactions on 24, no. 7 (2002): 881-892.

[13] Liu, Xiufeng, and Nadeem Iftikhar. "Ontology-Based Big Dimension Modeling in Data Warehouse Schema Design." In Business Information Systems, pp. 75-87. Springer Berlin Heidelberg, 2013.

[14] Noaman, Amin Y., and Ken Barker. "A horizontal fragmentation algorithm for the fact relation in a distributed data warehouse." In Proceedings of the eighth international conference on Information and knowledge management, pp. 154-161. ACM, 1999.