Journal of Population Therapeutics & Clinical Pharmacology

RESEARCH ARTICLE
DOI: 10.47750/jptcp.2023.30.16.029

Optimal Data Cube Computation In Serial Processing

Ajay Kumar Phogat[1]*, Suman Mann[2]
[1]Maharaja Surajmal Institute, GGSIPU, Delhi, 110075, India
[2]Maharaja Surajmal Institute of Technology, Delhi, 110058, India
***Corresponding author:** Ajay Kumar Phogat, Maharaja Surajmal Institute, GGSIPU, Delhi, 110075, India, Email: ajaykumarphogat@gmail.com

Submitted: 22 April 2023; Accepted: 15 May 2023; Published: 12 June 2023

ABSTRACT

In the business intelligence area there is a need for multidimensional data analysis which makes it fast and interactive. Data warehousing and online analytical processing policies have come in existence for this purpose, in which the data may see as a Multi-dimensional data cube, which allows interactive and different ways of analysis of data at different levels and displayed in image form so that calculating the data cube efficiently is great and significance greatly reduces the response time of the entire system. There are various basis to do this but widely used map-reduction algorithms, are effective data cube calculation procedures that utilize parallel systems for faster computation but their basic drawback is they are tied to the system's hardware environment and fail to be efficient on a single thread system, so we introduce the lowest first method, which is a sequential algorithm, which works efficiently in a single thread environment and performs data cube calculations in linear time this is possible for each query, without the need for complex systems such as distributed clusters by focusing on intermediate cuboids and better calculating their values as opposed to

Result: MR-cube policies based on dividing search space into batch areas may fail to lead to the proper division of work and sub-optimal results when executed sequentially. It took only 168.23 milliseconds to generate the total cuboid presented in this research work enabled by the non-heuristic nature of the lowest-first approach making it outperform other optimizers.

Conclusion: The preprocessing time complexity of the algorithm is O(nlogn) and query time complexity is O(n) which gives the best path of computing a cuboid starting from base cuboid representing data cube and for the preprocessing it may extended its use cases to bulk path finding queries over the prescribes data warehouse where q >> log(n).

Keywords: *Cuboid computation, data cube, multidimensional model, OLAP*

INTRODUCTION

Data warehouse (DW) is used for the repository that assembles data from various heterogeneous origins, handles it for efficient and effective retrieval, storage, and provides it to multiple consumers for business intelligence requirements [37]. Data Warehousing emerges as an alternative to address different data quality concerns for supporting decision-main processes in a given subject (being time-variant but not volatile), and the information that is stored is a multidimensional model [12]. They are of two types DW and data mart. While DW is the place where information from various destination materializes.

These are online analytical processing (OLAP) tools for analyzing that is highly interactive for multidimensional data[16]. OLAP is one of those technologies that analyze and evaluate data from the data warehouse. In OLAP, data is characterized by measurements. One major consistent structured data is the data cube, used to refer to multi-dimensional aggregation in data warehouse environments. Data cube represents to the three dimensional values used to visualize data from multidimensional models from which data is computed according to requirement and saved. The cuboid i.e. data cube used is identified by majorly two components, facts (quantitative) and dimensions (qualitative) [17]. Organization stores its records by the measure attribute which is mostly represented by numerical value. In a data warehouse system, response time of given query mainly based on the systematic and effective calculation of the given data cube [18]. Although, generating a data cube takes time and memory[38][39].

Section 2 discusses related work. Section 3 presents the state of the art approach to cube computation and its limitations. Section 4, presents the proposed approach Lowest First and its advantages over state of the art along with a mathematical proof to its validity, algorithm itself and implementation. Section 5 then gives experimental results and a comparison between both the approaches, last section is for conclusion.

BACKGROUND

Most prevalent study on the problem of computing and selection of data cube is Map Reduce based MR Cube approach that reduces the Data Cube computation process as a variant of map reduce and defines map steps and reduce steps [22]. [Mark] first proposed the concept of Mr. Cube and since then many variants of it have been designed and developed namely [33]. Other works include Bottom-up Cubing Algorithm (BUC) method. BUC method compute cube according to bottom-up from the most aggregated cuboids to least aggregated. In the paper [32] author theorized an algorithm known as greedy which finds the right ideas to implement, subject to various barriers to selection. There is an efficiency problem, the solution for this is given by the author in [33]. PBS (select by size), its function is proposed to select the cube in respect of the cube size. H. Gupta et al. [36] proposed a multidimensional greedy heuristic structure and view (each scene has a unique evaluation) Graf. Shukla et al. [34] considering the view selection problem for the multi-cube data model, Simple Local, Simple Global and Complex Global, three different algorithms have been proposed for selecting aggregates from the Multi-Cube schema. The heuristic algorithm was proposed by researchers [35], to identify a group of materialized views which is depend on the concept of reusing temporary results by implementing major questions with the support of the Multiple View Processing Plan [24]. For the particular case of OR View Graph, researcher [34] proposed an approximate algorithm which was developed. To prove that the materialist approach is systematic and effective, in comparison with the researcher [35] a genetic algorithm was used. W. Yang [7] greedily proposed a genetic algorithm to repair a set of physical cubes and identified that this solution could greatly reduce the amount of Curie maintenance as well as the cost of cube maintenance. M.P. Deshpande [6] proposed a sorting-based algorithm for cube computing that overlaps various group-activity calculations, using minimal memory for each calculation of the cube in the cube's network. Lavanova and R. Boris [9] created a data cube based on an object-oriented conceptual model. S. Sen [3] offered a way to find the optimal routes in a cuboid lattice based on two major transactions roll-up and drill-down to find convenient routes to travel between the data cubes. Number of authors is now focusing on developing effective algorithms for the exact cube [33].

Lowest First Approach

Map-Reduce paradigm is generally meant to implemented over parallel and distributed systems [32] while majority of the algorithms ever implemented are single threaded even though they might be managed by operating systems that are multi-processing, yet instructions are executed in a linear order hence its most likely that with existing data warehousing systems a sequential algorithm is much easier to integrate and upgrade to compared to its MR Cube [32][27] counterparts without loss of generality and increased efficiency[32].

For Cube Computation we propose an algorithm named Lowest First which is a sequential algorithm, previously the algorithms that have been developed for materialization were not targeted towards sequential execution but lowest first assumes sequential execution and is validated by a proof using mathematical induction that proves the algorithm is most efficient at computing materialization, that computes cube by exploiting and finding those intermediary cuboids that contain minimum rows with comparatively lower space and time complexities. As an example assume a cuboid have 4 dimensions named as (X, Y, Z, W) having values X=12, Y=22, C=7, D=17. To materialize data cube X have (X,Y,Z,W) dimensions then for pre materialization there are 2 options to do this:

One way is by starting from (X, Y, Z, W) to (X, Y, Z) and further to (X.Y) and ultimately we reach (X).

Second way may be to start from (X, Y, Z, W) to (X,Z,W) to (X,W) and finally to (X).

In totality there will be 3C2*2C1 ways, All possible ways are tried to make a difference and find the better path but the proposed algorithm here will give the effective option by ordering and hashing the multiple of number of dimensions in each data cube which may not be needed to materialize in-between cuboids and give the effective path. First, here represented the path cost as equation by which as a criteria it may compare the efficiency of different options and further optimization can take place [23]. The equation contains the number of dimensions each in-between cuboid has and cardinality is the number of elements in the set containing all distinguishable values a dimension of data cube may has. For elaboration, above explained example of a cuboid with 4 dimensions (X, Y, Z, W) is used and try to find an better path starting from (X, Y, Z, W) to (X) by noticing this may say that the track will involve two intermediate cuboids having 3 and 2 dimensions(D) materializing both will result in our goal cuboid, and both in-between cuboids must have all dimensions present in the goal cuboid which is (X) in this case.

$$c1c2c3+c4c5 \qquad \text{(i)}$$

here is the expression for the cost where c1, c2, c3 are the number of rows of the D available in in-between cuboid having 3 D and c4, c5 are the number of rows of D exist in intermediate cuboid having 2 D. It is verified that c4, c5 must be present in c1, c2, c3 so for easiness here assume c1=c4 and c2=c5 so that the equation becomes

$$c1c2c3+c1c2 \qquad \text{(ii)}$$

The goal dimension (X) must be available in both in-between cuboids hence it suppose c1 to be X and take it common, so equation then converted to

$$X(c2c3 + c2) \qquad \text{(iii)}$$

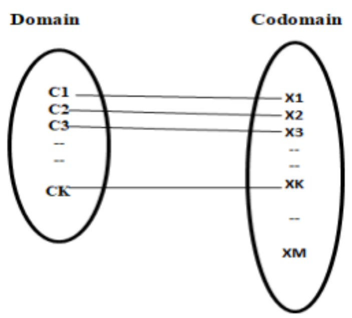

FIGURE 1: Example of Optimal Mapping

If we abstract away the materialization part from the approach discussed in this paper there has been substantial research work that has been done around mathematical optimizers [33] that can be used via interface to solve the problem of optimizing paths [34]. But a straight comparison can't be drawn between lowest approach and other optimizers [39] simply due to the fact that the development of lowest first was done by trying to optimize materialization and then backtracking to roots by formalizing the methodology [40]. There are limitations to the State of the Art methodology, as MR Cube is based upon the MapReduce framework which is implemented on parallel, distributed clusters. These clusters are mostly complicated to setup and defining the division of data and work among these clusters and managing the combining phase of the results of each cluster onto a final cluster where overall result will be generated is a complex and cumbersome intense process[14]. There may arise situations where we don't have enough resources to invest into the cube computation process, example use cases might include prototyping or low tier to mid- tier data warehouses at such situations Lowest First approach can be an efficient and easy to implement system [32]. Also a large part of MR Cube approach is the allocation of mappers to batch areas which need to be divided optimally and in case of lack of distributed systems the runtime boils down to suboptimal values [16].

where X=c1 Now it is required to exploit an optimal injection between the domain set c2, c3

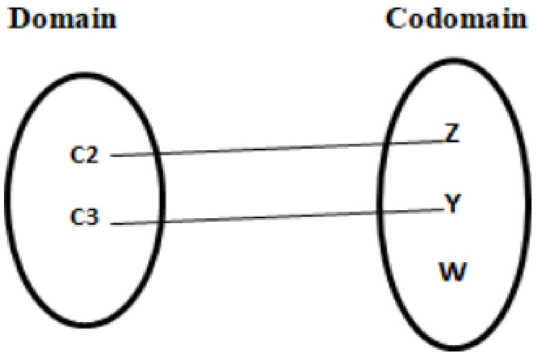

FIGURE 2: Set Mapping for Inductive assumption

The ways the given algorithm generate optimal injection by exploiting the following operations. So the resultant mapping for the example of data cube and particular path finding will look like is given figure 2, where the ordered difference set in increasing order of Z,W,Y based on their values 7,17,22 accordingly and the domain set is c2, c3 so the equation finally converts to

Z *W + Z (iv)

It shows the optimized option from <X,Y,Z,W> to <X> is

< X, Y, Z, W >=>< X, Z, W >=>< X, Z >=>< X > (v)

and the in between data cube are <X, Z, W> and <X, Z> which require least space and time to materialize with all other combinations. Now here is the validation by mathematical induction to show that algorithm finds an optimal path.

Lemma 1 *the optimal materialization of assumed generic data cube would be the minimum possible value of the following equation*

Proof

c1c2c3:::cn + c1c2c3:::cn = 1 + ::: + c1c2 + c1 (vi)

Co-domain mapping is

X1;X2;X3…………;Xm (vii)

Suppose it is available in ascending order so

X1X2X3X4………..Xm (viii)

Base step exist if n=1 when n=1 the domain becomes c1 and the equation is converted to a single term having only c1 now as per given lemma it reflect the variable c1 to the start element of the sorted co-domain X1 which is the minimum possible value of the equation c1 as X1 is the minimum value with all elements of set (vii), hence initial step is true.

Inductive step. Foe n=k assume the lemma is not false for the condition n=k which then gives the equation

c1c2c3:::ck + c1c2c3:::ck = 1 + ::: +c1c2 + c1 (ix)

will output in minimum possible value is as in figure 2 and the best value of equation for n=k becomes

X1X2X3:::Xk + A1A2A3:::Xk = 1 + ::: + X1 (x)

For n=k+1 Here it is required to present using Figure 3 that the numerical value of equation is

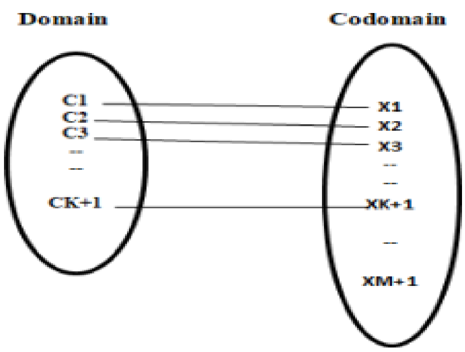

FIGURE 3: Optimal Mapping of n =k+1

minimum when the tracking/mapping is as shown in figure 3 and to prove this it divide the equation (10) into 2 divisions, second part of the equation is as it is as (x) which by inductive step for n=k is lowest. Now it is required to present that part 1 is minimum possible which is true as to take k+1 positive integers from a given set such that their multiple is minimum is only feasible when minimum k+1 terms are taken and this is exactly doing by selecting X1, X2, X3,. ,Xk.

c1c2c3:::ck + 1 + c1c2c3:::ck + ::: + c1c2 +c1 (xi)

Therefore for the condition n=k+1 the lemma proved to be right and (Figure. 4) is the effective reflection of items which proves lemma to be correct. Now algorithm is described in a two part with series of steps which may be implemented in any king of programming language.

Proposed Algorithm

Let us dry-run the above algorithm on a sample case, take a lattice of cuboids of a data cube having 3 dimensions named as Time, Market and Location here cardinality of Time is 4 (T1, T2, T3, T4), cardinality of Market is 2 (M1, M2) and cardinality of Location is 6 (L1, L2, L3, L4, L5, L6) and Sales is the measure which is quantitative.

Algorithm: Lowest First Function

```
Create an array A having dimensions.
Sort array A in non-decreasing order.
Materialize(S, T)
S is the source cuboid and T is the target cuboid.
Let H be a hash set for the dimensions of S and D for elements exclusive to T.
for all Ti in T do
if ti does not exist in H then
insert into D.
end if
end for
Let Sd be sorted difference set, which will contain dimensions in sorted order which
could be present in intermediate cuboids.
for all Ai into A do
if Ai exists in D then
insert into Sd
end if
end for
while x is not S do
print x
end while
```

Here focus is to get the best available option of materializing cuboid <Q> begin from <T, M, L>. Now above defines procedure is used to generate the best option of path for the specified data cube, source and objective cube. It generate an array named 'sorted dimensions' by ordering the dimensions T, M, L based on the given values of all the dimensions in increasing order and the output is according to ordered dimensions = [T, M, L] as values of T=3, M=5 and L=7.

For Example we are Querying < T, M, L > => <T>

1: Given Cuboid = < T, M, L >

2: Goal Cuboid = > <T>

3: Verify that dimensions in target cuboid must be at lower level from the specifies cuboid.

4: For generating difference Set it will firstly make a hash mapping of specified e dimensions and repeat upon goal data cube verify that this dimension available in the hash map or not and if not exist then insert it into differenceSet. So after making it differenceSet = [M, L].

5: Generate a hash map of differenceSet.

6: loop over sortedattr generated while preprocessing and for every dimension in it verify that it is available in the hash map of differenceset if exist then insert it to a different array diffsetsorted. After this process, a named array diffsetsorted which contains the elements of differenceset is sorted in increasing order.

diffsetsorted = [M,L].

The track now is begin from goal cuboid <Q> and to the in-between data cube having dimensions of goal cube and the starting dimension in diffsetsorted ie, M and then the next dimension and this is continue. So option obtained by algorithm <T> => <T,M> => < T, M, L > is the best available path to materialize data cube <T> starting from < T, M, L > Now let's validate the result optimal path shows with darkline from < T, M, L > to <T> by comparing it will all other combinations. Other possible path for materializing cuboid

<T> are:

< T, M, L > => <T, L> => <T>, here cost is the total sum of materialization of in between cuboids ie, cost(<T, L>) it is the product of given values of dimensions available in the cuboid = 3*7=21.

< T, M, L > => <T,M> => <T>, cost = 3*5 =15 calculated in the same way as for previous path

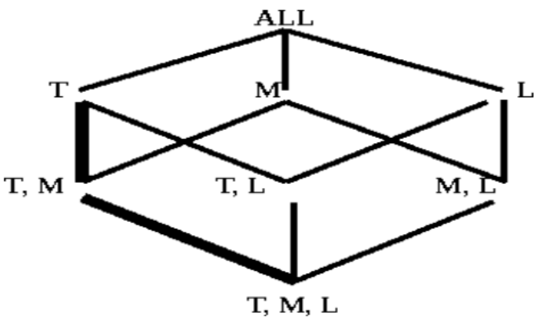

FIGURE 4: Three Dimensions T, M, L Lattice of cuboids having optimal highlighted path

IMPLEMENTATION AND RESULTS

There are considerable cons using an empirical approach to estimate the performance of a given set of methods, methods are platform independent in the sense, method or a function can be executed in any programming language on an any computer running on an arbitrary operating system. The same is notified in the above theoretical and empirical evaluation. As par the practical evaluation is considered we implemented the experimental analysis of both algorithms on the above discussed Data Cube with 3 dimensions namely Q, R, B on a system with 8GB memory and 1.3GHz processing speed and Quad Cores locked to a single thread to mimic sequential environment. Data generated are average of three successful runs. There has been a 17% decrease in running time for the sample case taken when switching from MR-Cube to Lowest First, this decrease is because we have limited the environment to single thread while if it were multi-threaded or distributed system MR-Cube would have outperformed the lowest first approaches but due to the simplistic and single thread while if it were multi-threaded or distributed system MR-Cube would have outperformed the lowest first approaches but due to the simplistic and non-heuristic nature of the lowest first approach, it theoretically beats the other optimizers when raw time complexity comparisons are made.

TABLE 1: Performance Statistics

Source	Target	Time(ms) MR-Cube	Time(ms)using lowest first
T	TM	7.62	6.35
T	TM	6.41	7.89
T	TML	28.30	23.45
M	TM	6.98	5.78
M	ML	5.92	6.40
M	TML	24.23	20.29
L	TL	18.42	8.98
L	ML	15.23	7.67
L	TML	29.62	30.45
TM	TML	19.38	15.43
TM	TML	22.19	18.56
ML	TML	19.80	16.98
Total		**204.10**	**168.23**

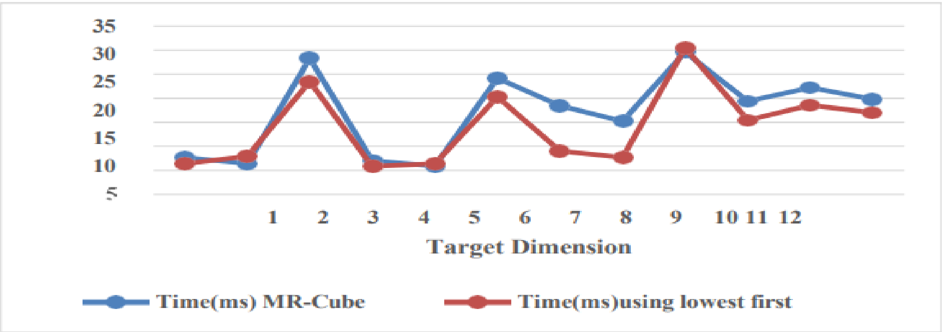

FIGURE 5: Comparison of time(ms) taken by MR-Cube & lowest first approach

CONCLUSION

State of the art algorithms like MR-Cube in order to perform effectively require optimal allocation of mappers to each batch area which is dependent on proper division and creation of batch areas and later data aggregation becomes a cumbersome process if not done properly and more importantly these perform well only in distributed parallel environments but such distributed and parallel systems and fail to be effective in simple single threaded systems, hence the need for simpler and easy to implement algorithms for non- parallel single processing systems is fulfilled by the proposed algorithm. The preprocessing time complexity of the algorithm is O(nlogn) and query time complexity is O(n) which gives the best path of computing a cuboid starting from base cuboid representing data cube and for the preprocessing it may extended its use cases to bulk path finding queries over the prescribes data warehouse where q >> log(n):

The limitation of the above study is that the proposed algorithm fails to exploit multi-threaded environments, our next study would be to parallelize the effort in an efficient manner such that the time taken for finding optimal materialization path is close to T/n here n is the number of physical threads available in the system and T is the total time taken in a single threaded environment.

Funding

This research received no specific grant from any funding agency.

Author information Affiliations

Ajay Kumar Phogat

Research Scholar, GGSIPU, Maharaja Surajmal Institute, Delhi, India

Suman Mann

Department of Information Technology, Maharaja Surajmal Institute of Technology, Delhi, India

Ethics declarations Conflict of interest

The authors declare no competitive interest regarding the publication of this paper.

REFERENCES

1. X. Li, J. Han, and H. Gonzalez, "High dimensional OLAP: A Minimal cubing approach", 30th International Conference on Very Large Databases (VLDB'04), Toronto Canada, Aug. 2004, pp.528-539.
2. V. Harinarayan, Rajaraman, A, Ullman, "Implementing Data Cubes Efficiently", In ACM SIGMOD International Conference on Management of Data, ACM Press, New York, 1996 pp.205-216.
3. S. Sen and N. Chaki, "Efficient Traversal in Data Warehouse Based on Concept Hierarchy Using Galois Connections," Second International Conference on Emerging Applications of Information Technology, 2011, pp. 335- 339, doi: 10.1109 EAIT.2011.69.
4. S. Sen, N. Chaki and A. Cortesi, "Optimal Space and Time Complexity Analysis on the Lattice of Cuboids Using Galois Connections for Data Warehousing," Fourth International Conference on Computer Sciences and Convergence Information Technology, 2009, pp. 1271-1275, doi: 10.1109/ICCIT.2009.185.
5. Stefanovic, N., Han, J., Koperski, K. "Object-Based Selective Materialization for Efficient Implementation of Spatial data cubes", IEEE Transaction on Knowledge and Data Engineering, 2000, pp 938-958
6. M.P. Deshpande, S. Agarwal, J.F. Naughton, R. Ramakrishnan, "Computation of Multi-dimensional Aggregates" University of Wisconsin Madison, Technical Report, 1997
7. L.Y. Wen, K.I. Chung, "A genetic algorithm for OLAP data cubes" Knowledge and information systems, January 2004, volume 6, Issue1, pp 83-102.
8. Aouiche K., Jouve PE., Darmont J., "Clustering-Based Materialized View Selection in Data Warehouses", https://doi.org/10.1007/11827252_9
9. I. Antoaneta, R Boris, "Multidimensional models constructing data cube" Int. conference on computer systems and technologies-CompSysTech2004, V-5 pp1-7
10. Mami and Z. Bellahsene," A survey of view selection method" SIGMOD Record, March 2012 Vol. 41, No. 1, pp20-30.
11. I. Mami, R. Coletta, and Z. Bellahsene, "Modeling view selection as a constraint satisfaction problem", In DEXA, pp396-410, 2011
12. S. Mann, A. Gosain, and S. Sabaharwal. "OO Approach for Developing Conceptual Model for a Data Warehouse." Journal of Technology and Engineering Science", 2009, volume1 pp. 79-82.
13. H. Gupta, I.S. Mumick, "Selection of views to materialize under maintenance cost constraint" In 7th International Conference on Database Theory (ICDT'99) Jerusalem, Israel, pp. 453-470, 1999

14. Anjana Gosain, Suman Mann, “Object Oriented Multidimensional Model for a Data Warehouse with Operators", International Journal of Database Theory and Application, 2010, Vol. 3, No. 4.
15. Gosain, A., Mann, S. “An object-oriented multidimensional model for data warehouse", In Fourth International Conference on Machine Vision (ICMV 2011): Computer Vision and Image Analysis; Pattern Recognition and Basic Technologies, Vol. 8350, p. 83500I.
16. Gosain, Anjana, and Suman Mann., “Space and Time Analysis on the Lattice of Cuboid for Data Warehouse." International Journal of Computer Applications, 2013, volume 77 No.3.
17. C. Zhang and J. Yang, “Genetic algorithm for materialized view selection in data warehouse environments," Proceedings of the International Conference on Data Warehousing and Knowledge Discovery, LNCS, vol.1676,
18. pp. 116-125, 1999
19. Gosain, A., Mann, S. Empirical validation of metrics for object oriented multidimensional model for data warehouse. International Journal of System Assurance Engineering Management 5, 262–275 (2014). https://doi.org/10.1007/s13198-013-0155-8
20. Antoaneta Ivanova, Boris Rachev, “Multidimensional models constructing data cube", International Conference on Computer Systems and Technologies CompSys-Tech '2004.
21. S. Mann, Bharti and P. Singh, "Empirical validation of multidimensional model for data warehouse," Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, 2014, pp. 1-6
22. A. Shukla, PM Deshpande, JF Naughton, “Materialized view selection for multidimensional datasets", Proceeding of the 24th international conference on very large databases, New York, August 1998, pp 488-499.
23. A. Gosain, S. K. Khatri and S. Mann, "Multidimensional modeling for data warehouse using object oriented approach," Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, 2014, pp. 1-6.
24. A. Shukla, Deshpande PM, Naughton JF, “Materialized view selection for multi-cube data models" 7th International Conference on extended database technology, Germany, March 2000, Springer, pp 269-284.
25. Suman Mann Ajay Kumar Phogat, “Dynamic construction of lattice of cuboids in data warehouse", Journal of Statistics and Management Systems, 971-982.
26. Dinesh Mankad, Preyash Dholakiya, "The Study on DataWarehouse Design and Usage" International Journal of Scientific and Research Publications, March 2013.
27. Prashant R., Suman M., Eashwaran R., “Efficient Data Cube Materialization" Advances in Communication and Computational Technology, Lecture Notes in Electrical Engineering, vol 668. Springer, Singapore ,2020.
28. Mukund Sundarajan, Quqi Yan, “Simple and Efficient Map Reduce Algorithm for Data Cube Materialization", Google Research 1600 Amphitheatre Pkway, Mountain View, CA, 94043, USA, 2017.
29. Phogat, A. K., & Mann, S. (2023). Hyper Lattice Structure for Data Cube Computation. In Proceedings of Data Analytics and Management: ICDAM 2022 (pp. 697-705). Singapore: Springer Nature Singapore
30. Chen, W., Wang, H., Zhang, X., “ An optimized distributed OLAP system for big data” In 2nd IEEE International Conference on Computational Intelligence and Applications 2017.
31. Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The Arithmetic Optimization Algorithm. Computer Methods in Applied Mechanics and Engineering, 376, [113609]. https://doi.org/10.1016/j.cma.2020.113609
32. Laith Abualigah and Dalia Yousri and Mohamed {Abd Elaziz} and Ahmed A. Ewees and Mohammed A.A. Al- qaness and Amir H. Gandomi, “Aquila Optimizer: A novel meta-heuristic optimization Algorithm” computer and industrial engineering” vol 157, 2021
33. Abualigah, L., Diabat, A. Advances in Sine Cosine Algorithm: A comprehensive survey. Artif Intell Rev 54, 2567–2608 (2021). https://doi.org/10.1007/s10462-020-09909-3
34. Abualigah, L., Elaziz, M. A., Hussien, A. G., Alsalibi, B., Jalali, S. M. J., & Gandomi, A. H. (2021). Lightning search algorithm: a comprehensive survey. Applied Intelligence, 51(4), 2353-2376.
35. Phogat, A. K., & Mann, S. (2022). Optimal Data Cube Materialization in Hyper Lattice Structure in Data Warehouse Environment. Journal Of Algebraic Statistics, 13(1), 149-158.
36. Phogat, A. K., & Mann, S. (2022, April). Analysis of Materialization View Selection Approaches in Data Warehouse. In 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE) (pp. 1418-1422). IEEE.