Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.199

IJCSMC, Vol. 8, Issue. 7, July 2019, pg.52 – 64

Selection of Optimal Materialized Views in Data Warehouse Using Hybrid Technique

Wesal Adeeb Abdullah[1]; Naji Mutar Sahib[2]; Jamal Mustafa Al-Tuwaijari[3]
[1]Department of Computer Sciences, College of Science, Diyala University, Iraq
[2]Department of Computer Sciences, College of Science, Diyala University, Iraq
[3]Department of Computer Sciences, College of Science, Diyala University, Iraq
[1] wesaladeeb2@gmail.com, [2] al.sahib@sciences.uodiyala.edu.iq, [3] altuwaijari@yahoo.co.uk

Abstract— Decision making is the main purpose of DW. Typically, decision making queries are analytical, complex, recurring and include aggregation functions or many join operations posed over DW. A critical issue in designing DW is answering these queries efficiently. Many ways have been proposed to address this problem, one of them is materialize views while due to space constraints all views cannot be materialized in DW. In this paper we have design an efficient methodology for selecting an optimal MVs based on three factors (MV response time, MV storage area and MV frequency) using bitmap index to minimize the total time of creating MVs, then using hybrid technique (Firefly algorithm and Quantum Particle Swarm Optimization algorithm) to select optimal MV which has low response time, low storage area and high frequency. The results proved that bitmap index achieved good results because it take less time and storage area in recoupoing results since bitmap indices have the ability of accomplishing processes on index level before recouping the base relations where, the total time of applying these functions over bitmap index was found 168 milleseconds, while directly over base tables was found 216 milleseconds. Also, the hybrid technique was more efficient in term of optimal MVs selection time since, FA presents optimal initial point to QPSO algorithm.

Keywords— data warehouse, materialized views, processing time, firefly algorithm, quantum particle swarm optimization algorithm.

I. INTRODUCTION

Now a day's, big enterprises must process and analyse huge amounts of data. The traditional on-line transaction processing (OLTP) system become inadequate to meet the deep analysis requirements of multidimensional data due to the complexity of data queries, and the quick increase in data volume [1]. Data warehouse is a unified repository of data collected from an enterprise's different business systems for analysis purpose. It is a copy of transaction data that have been structured for analysis and query [2]. A basic requirement of data warehouse's success is the ability for providing the decision makers with timely and accurate information besides quick query response times [3]. In order for accelerating access to data in DWs and answering the analytical queries efficiently efficient query processing techniques and access methods are required [2]. For the purpose of providing best response time and higher information the common method which is used in practice is the concept of materialized views, in which query is more speedily answered [3]. Before seen the concept of MV, we define a view, view is a derived relation and its definition is store in the database system only therefore its value at a given time is the result of evaluating specific relational expression at that time. A view thus is a function typically recomputed every time it is mentioned. Materialized views are defined over the base relations and involve the DW contents after treatment. It can be designed according to the user's requirements. It is significant because the data access through MVs is just a cache that is a copy of data which can be accessed

rapidly [4]. Selecting views to materialize is one of the most significant decisions in DW design. The problem of view selection can be defined as select a collection of derived views which decrease the total query processing time to materialize them in DW. Therefore the goal is select an appropriate collection of views to materialize to decrease the total query processing time [3]. In this paper MVs are selected under three constraints (MV response time, MV storage area and MV frequency). Another alternative method to access data which required much less time in DW is bitmap index technique, since it is efficient especially for read-mostly data. Bitmap indices in most DW applications are shown to perform good results than other indexing schemas like: B-tree or R-tree [5].

II. RELATED WORKS

Sapna and Roshna in 2014 [4] proposed framework for achieving high query performance by detecting which query is more beneficial for creating the MVs. They determine the performance of any query depending on the processing time required for this query submitted directly to DW with respect to this query through selected MVs. The implementation results illustrate that the processing time of queries directly selected to DW was found 32 milliseconds whereas these queries through MVs was found 15 milliseconds. They also using mathematical model for preserving the selected MVs depending on the storage required by these MVs to be stored. Dengfeng Yao et. al (2015) [1] present a methodology for improving the cost model and the polynomial greedy algorithm to address the storage space constraints of the databases for selecting the views that have minimum cost from the candidate views for materializing them in DW. This algorithm firstly, calculates the available storage space to decide adding or deleting the candidate MVs by selecting a lower cost. The implementation results show that the algorithm was effective. Amit Kumar and T.V. Vijay Kumar (2017) [2] presented a technique for solving the view selection problem. This technique dependent on the hybrid properties of the Discrete Genetic operator based Particle Swarm Optimization algorithm which is a combination of a PSO algorithm and genetic operators like: mutation and crossover for selecting optimal collection of views from a multidimensional lattice. The implementation results illustrate that the proposed technique was effective as comparative with other most existing technique used to select views to materialize also the proposed technique decreased the analytical queries processing time.

III.MATERIALIZED VIEWS

Materialized views include pre-computed and summarized information used for answering queries posed on DW for saving query response time and storage. In DW, typically MVs are designed according to the user's requirements (e.g., frequently mentioned queries) [6]. Materialized views are schema objects which can be used to re-compute, summarize distributed data and replicate. In DW, MVs can be stored in another database or in the same database as its base relations for using it to improve the query performance by rewriting these queries through MVs rather than base tables. In data warehouse MVs store the query results physically in a table therefore when this query is posed to DW there is no need to re-calculate it again because such a view is already materialized in DW transparently to the DB application or the end users and can be used at any time this query is posed on the system [7]. Therefore, MVs optimize the DW performance by reducing the query processing time which is significant in big OLAP systems since queries are directed to the MVs rather than the base relations [8]. One of the disadvantages of MVs is that they need extra space because they stored physically in DW, also they must be synchronous with the base relations, so when the base tables are changed at any time after the MVs were defined these MVs must be maintained or re-calculated [4]. Due to these downsides all views cannot be materialize and most important issue in designing DW is to select an appropriate set of views to materialize in DW. There are three types of MVs in DW, one contains aggregation functions like (SUM, AVG, MIN, MAX, COUNT, STDEV, VAR, etc…), one contains joins only and the third type is hybrid MVs which contains joins and aggregation functions [9], in this paper MVs with aggregation functions have been created.

IV.BITMAP INDEX TECHNIQUE

Bitmap indexes are efficient indexing technique used for read only or read mostly data for speeding up multidimensional range queries [10]. In DW there are huge amounts of data and ad-hoc queries posed on these data, so bitmap index is widely used for these applications, because it reduce the processing time for large categories of ad-hoc queries, also it requires less storage area as compared with other existing indexing techniques and its simple to represent. Bitmap indexes improve the complex queries performance by implementing low-cost operations like: NOT, AND and OR in the selection of predicate on several indexes simultaneously for reducing the search space before going to the primary source tables [11],[12]. Bitmap indexes are used with low cardinality columns. Cardinality is the number of distinct groups stored in a column, if the number of distinct groups reaches 1% or more of the total number of rows in the table then the column is

said to has low cardinality. The gender attribute is an example of low cardinality attribute because it has only two values 'male' and 'female' [13].

V. FIREFLY ALGORITHM

Firefly algorithm is a nature inspired, metaheuristic algorithm that is based on the behaviour of the social flashing of fireflies in summer sky in the regions that have tropical temperature [14]. Most fireflies produce rhythmic and short flashes and the patterns of these flashes are unique for most times for a particular region. Females of a specific region respond to the male individual pattern of the same region [15]. The characteristics of flashes can be idealized based on the following three rules [15]: any firefly can be attracted to another firefly regardless of their sex because all fireflies are unisex, the attractiveness of each firefly is proportional to its brightness which reduces when the distance among fireflies increases, the firefly will moves randomly when there is no firefly brighter than a particular firefly, the firefly brightness is circumscribed by the objective function value [14][15]. Firefly algorithm seems more effective in optimization because it utilizes real random numbers and also, the communication among particles is global [14]. The attractiveness between any two fireflies which is proportional with the light intensity seen by neighboured fireflies can be defined by eq. (1) [16], [17].

$$\beta = \beta_0 e^{-\gamma d2} \quad (1)$$

Where: d is the distance between any two fireflies.

β_0 is the attractiveness at d=0 and γ is the light absorption coefficient.

The distance (d) between each two fireflies k and m at x_k and x_m can be calculated using the Cartesian distance as in eq. (2) [16], [17].

$$d_{km} = / X_k - X_m / = \sqrt{\sum^{s}_{i=1} (X_{k,i} - X_{m,i})^2} \quad (2)$$

The movement of firefly k which is attracted by brighter firefly can be computed using eq. (3) [16], [17].

$$X_k = X_k + \beta_0 e^{-\gamma d2} (X_k - X_m) + \alpha (rnd - 0.5) \quad (3)$$

Where: X_k is the previous position, the second term is used to calculate the attractiveness and the third term is used in case of movement in random direction when there is no brighter one, and rnd is random number uniformly distributed between [0, 1]. For most cases in practice $\beta_{0=1}$ and $\alpha \in [0, 1]$ [16], [17].

VI. QUANTUM PARTICLE SWARM OPTIMIZATION ALGORITHM

Quantum particle swarm optimization algorithm has been proposed by Sun and others in 2004, this algorithm is probability searching algorithm and it is based on the quantum mechanics and relies on the delta potential well [18]. The basic idea in QPSO algorithm is the combination of standard PSO algorithm in which the positions of particles are updated based on the particles current location information of local and global optimum position information with quantum physics [19]. Because in quantum space, the particle's velocity and position cannot be circumscribed simultaneously, the function known wave function $\Psi(X,t)$has been used to describe the particle's movement state [20]. In QPSO model the particle can be emerges in random position of the all feasible search space based on certain probability and better value of fitness function, thus the QPSO model is superior in term of the global search performance as the comparison with standard PSO model [21]. The QPSO algorithm consists random location between global best and Pbest position known mean best position in the algorithm's iterative equation for optimizing the ability of the particles global search [22]. The QPSO algorithm can be converges when the particle converges to its local attraction point ($L_{i,t}$) which can be calculated using eq. (4) [].

$$L^j_{i,t} = W^j_{i,t} L^j_{i,t} + (1- W^j_{i,t}) G^j_t \quad (4)$$

Where: $W^j_{i,t}$ is random number normally distributed in [0, 1].

when the particles move in one dimensional delta potential well them new position can be calculated using eq. (5) [23].

$$Y^d_{i,t} = L^j_{i,t} \pm M^j_{i,t} /2 \ln 1/s \quad (5)$$

Where: s is random number uniformly distributed on interval [0, 1], and the value of $M^{j}_{i,t}$ can be calculated using the following two equations eq. (6) and eq. (7) [24].

$$M^{d}_{i,t} = 2\alpha \,/\, Y^{d}_{i,t} - L^{d}_{i,t} \,/ \qquad (6)$$

$$M^{d}_{i,t} = 2\alpha \,/\, Y^{d}_{i,t} - G^{d}_{i,t} \,/ \qquad (7)$$

By using eq. (6) and eq. (7) with eq. (5) the particle's updated position can be calculated using eq. (8) and eq. (9) [24].

$$Y^{d}_{i,t} = L^{j}_{i,t} \pm 2\alpha \,/\, Y^{d}_{i,t} - L^{d}_{i,t} \,/\, \ln (1/ s^{d}_{i,t+1}) \qquad (8)$$

$$Y^{d}_{i,t} = L^{j}_{i,t} \pm 2\alpha \,/\, Y^{d}_{i,t} - L^{d}_{i,t} \,/\, \ln (1/ s^{d}_{i,t+1}) \qquad (9)$$

In eq. (8) and eq. (9) the probability of using ± based on the random number value use – when the random number value is greater than 0.5 and use + otherwise [18], α is known contraction expansion factor and its very significant factor in this algorithm because it used to balance the local and global search of the algorithm [24].

VII. THE PROPOSED SYSTEM

This section introduce the main stages of the proposed work (selection optimal MVs using hybrid technique) as shown in fig. 1.

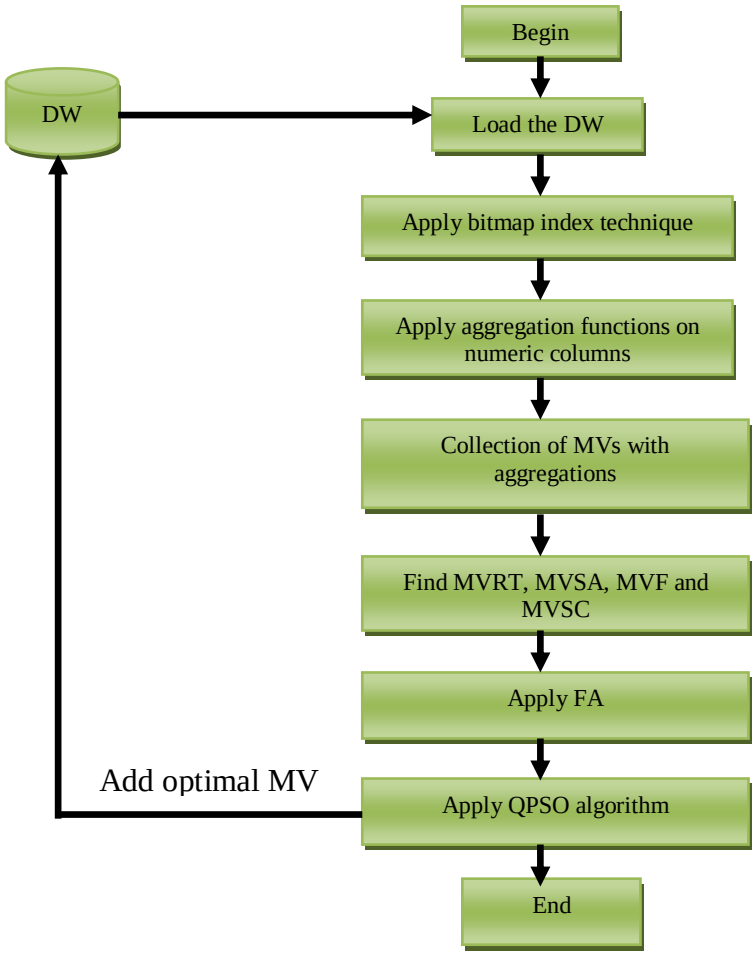


Fig. 1 The proposed system for selecting optimal MVs using hybrid technique.

A. Apply Bitmap Index Technique

The first step in the proposed work is apply the bitmap index on the determined column after computing the column cardinality to decide whether bitmap index is appropriate for this column or not using algorithm in Table 1.

TABLE I
ALGOROTHM TO CALCULATE THE COLUMNS CARDINALITY

Input: selected table from DW. **Output:** the cardinality of each column in determined table.
Begin **Step (1):** select required table from DW. **Step (2):** select the determined columns and wait for the answer from the server. **Step (3):** the server will calculate the value of cardinality which is the number of distinct groups in the determined column and return it to the end user. **End**

B. Apply Aggregation Functions

The second step in the proposed work is applying the aggregation functions (AVG, MAX, SUM, MIN, STDEV, VAR and COUNT) on the selected column with two ways (over bitmap index and directly on base tables) using algorithm in Table 2.

TABLE II
ALGORITHM FOR APPLYING AGGREGATION FUNCTIONS

Input: Selected table. **Output:** Execution time of aggregation functions on selected column with and without bitmap index.

Begin
Step (1): input table.
Step (2): Do {
Calculate (SUM);
Calculate (AVG);
Calculate (MIN);
Calculate (MAX);
Calculate (COUNT);
Calculate (STDEV);
Calculate (VAR);
}
While (all columns in table)
Step (3): show the execution time of implementing aggregation functions with and without bitmap index.
End

C. Find Materialized Views Parameters and Selection Costs

After applying aggregation functions in the last step, we have a set of MVs, in this step we will calculate the response time which is the time required for processing this MV, the storage area which is the storage required by current MV and computed by multiplying the MV columns and rows, and the frequency which is how many the current MV requested as MVs information parameters and stored them in MVs information table for calculating the selection cost to each MV in the MVSET using eq. (10)

$$MVSC= W1* FC+ W2* RTC+ W3* (1- SAC) \quad (10)$$

Where: W1, W2 and W3 are weights given to FC, RTC and SAC consecutively and their summation is equal to 1. The cost of each parameter has been calculated by dividing the current value of each parameter on the maximum value of this parameter.

TABLE III
ALGORITHM TO FIND MVS SELECTION COSTS

Input: MVs set. **Output:** The MVs Selection Costs.

Begin
Repeat
For k ← 1 to MVS
MVIP ← find MVF;
MVIP ← find MVSA;
MVIP ← find MVRT;


```
    MVIT  ← MVIP
end repeat
Repeat
    For k ← 0 to NRMVIT (number of rows in MV information table) -1
    MVIP ← MVIP[k]
    RT  ← MVIP
    F  ← MVIP
    SA  ← MVIP
    RTC ← RT/MMVRT (maximum MV response time)
    FC  ← F/MMVF (maximum MV frequency)
    SAC  ← SA/MMVSA (maximum MV storage area)
    MVCT (MV cost table)  ←RTC
    MVCT ← FC
    MVCT  ← SAC
    end repeat
    calculate MVs selection costs
    Repeat
    for k ← 1 to MVCT
    MVSC= WT1*AFC+WT*PTC+WT3*(1-SPC);
    MVCT ← MVSC
End repeat
End
```

D. Apply Firefly Algorithm

In this step FA that shown in Table VI has been used to produce optimal initial point to the QPSO algorithm. Firefly algorithm find the brightest one (in this optimization problem the brightest one is the maximum selection cost value), then calculating the distance and attractiveness among the brightest one and other fireflies, finally it reorder the fireflies according the amount of movement of each firefly towards the brightest one until the termination condition is met.

TABLE IV
FIREFLY ALGORITHM

Input: two-dimensional matrix of MVs selection costs, maximum number of iterations (M), light absorption coefficient (γ), α.
Output: selection costs reordered based on their movements towards the brightest one.

Begin
Step (1): set A=1.
Step (2): begin from the value in the middle, then find the brightest one.
Step (3): compute the distance among the brightest one and other selection costs using eq. (2).
Step (4): compute the attractiveness between the brightest one and each selection costs using eq. (1).
Step (5): if the brightness of (xk) < brightness of (xm) then firefly (xk) will move towards firefly (xm) using eq. (3).
Step (6): Ranks MVs selection costs based on their movement.
A=A+1
Step (7): if stopping condition (A>M) not satisfied then
Repeat from step (2)
Else
Take the sorted selection costs.
End

E. Apply QPSO Algorithm

The QPSO algorithm which illustrated in Table VI has been used to find optimal MVs which have low response time, low storage area and high frequency for using them to reduce the complex queries response time. At the first step the QPSO algorithm calculates the fitness value of each particle using eq. (11) and eq. (12), and determines the Pbest position. At each iteration, the QPSO algorithm calculates the mean best position of the swarm's particle (mbest), as well as evaluates the fitness value of each particle and updates its Pbest and current gbest position before updating its current position until the termination condition is met.

$$\text{Mean} = \sum Ni=1\ (Xi)/N \qquad (11)$$

$$\text{Variance} = (Xi\text{-mean})2\ /N \qquad (12)$$

Where:
Xi : is the value in the current position, and N is the number of MVs selection costs in a given search space.

TABLE V
QUANTUM PARTICLE SWARM OPTIMIZATION ALGORITHM

Input: Search space result from FA (MVs selection costs reordered by FA), r1, r2, r3, parameter α, number of iterations w.
Output: Optimal MV.

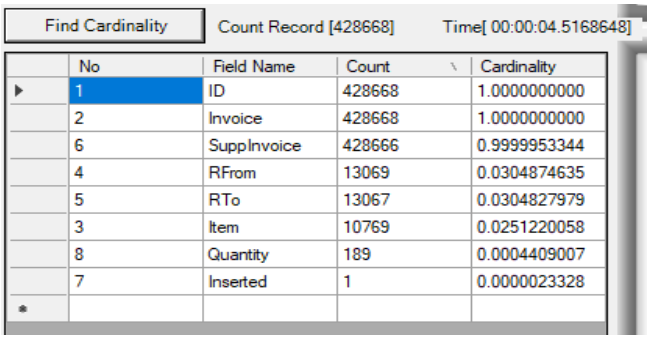
Begin
Step(1): set d=1
Step(2): calculate the fitness function for each MV selection cost using eq. (11) and eq. (12).
Step(3): determine Pbest position and gbest position.
Step(4): if the fitness of the particle f(x)> fitness of the best particle f(pbest) then go to step(5) else go to step(6).
Step(5): f(pbest) = f(x) and pbest = x
F(global) = fitness of the best neighbours of f(x).
Step(6): if f(x)> f(global) then
F(global) = f(x) and global = x.
Step(7): update the position of the Pbest particle according eq.(2), calculate the mean best position for new Pbest position
Step(8): if (r3<0.5) then
Update the particle position according eq.(8) with (+) sign.
Else
Update the particle position according eq.(8) with (-) sign.
d=d+1
Step (9): if stopping condition (d>w) not satisfied then
Repeat from **step(2)**
Else
Take optimal MV.
End

VIII. THE IMPLEMENTATION OF THE PROPOSED SYSTEM

The sales system data warehouse which built by one of the master's project [25] has been taken as a case study. It contains seven dimension tables and one fact table (Suppliers table, Items table, Clients table, Invoices table, Invoices-details table, SuppInvoices table, SuppInvoices-details table and Fact-Sales table).

A. Implementing Bitmap Index

The button "Find Cardinality" in Fig. 2 has been used to calculate the selected column cardinality to decide whether bitmap index is appropriate for this column or not.



No	Field Name	Count	Cardinality
1	ID	428668	1.0000000000
2	Invoice	428668	1.0000000000
6	SuppInvoice	428666	0.9999953344
4	RFrom	13069	0.0304874635
5	RTo	13067	0.0304827979
3	Item	10769	0.0251220058
8	Quantity	189	0.0004409007
7	Inserted	1	0.0000023328

Fig. 2 Interface to calculate the columns cardinality.

B. Implementing Aggregation Functions

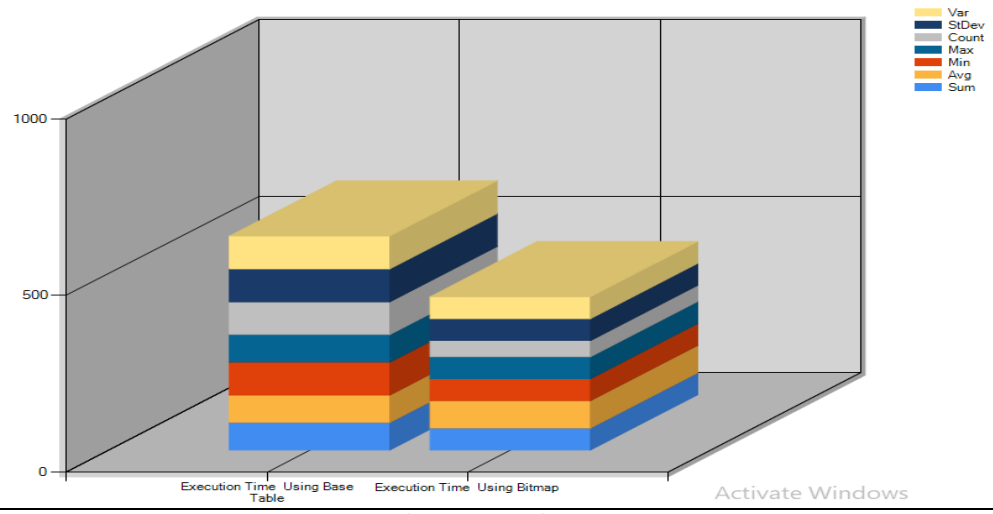
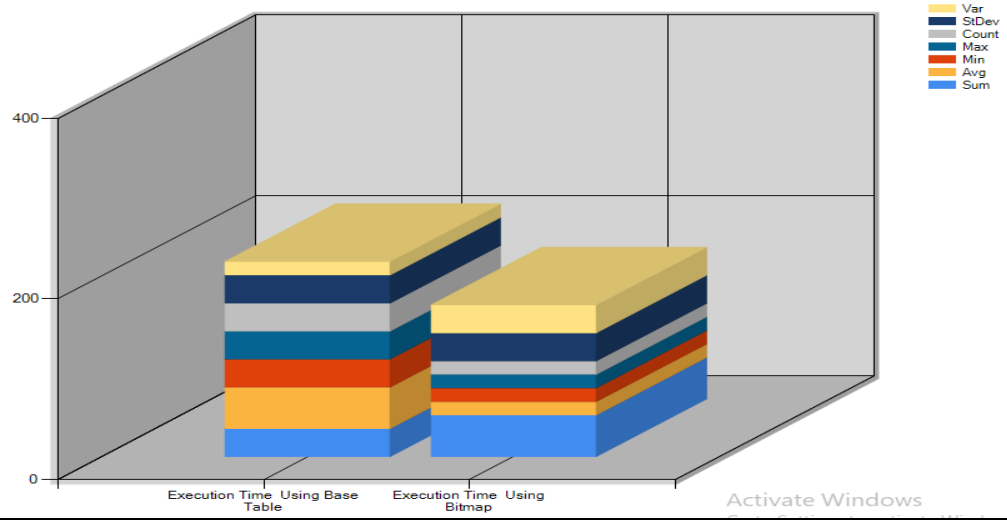
After implement bitmap index in the last step, in this step aggregation functions (AVG, MAX, SUM, MIN, STDEV, VAR and COUNT) have been implemented on the selected column with two ways (directly over base tables and over bitmap index). The following Table VI shows the aggregation functions in relational algebra expressions and the execution time of implementing these functions on many columns in various tables before and after implementing bitmap index technique.

TABLE VI
ILLUSTRATES THE IMPLEMENTATION OF AGGREGATION FUNCTIONS ON MANY COLUMNS IN VARIOUS TABLES AND THE RESPONSE TIME OF THESE AGGREGATIONS WITH AND WITHOUT BITMAP INDEX TECHNIQUE.

Case NO	Filed name	Aggregation		Time	
		Aggregation in relational algebra	Function	Time using base tables	Time using bitmap index
Case 1	Quatity	Quatity □sum_Quatity (TBL_INVOICE_DETAILS)	Sum	00:00:00:078	00:00:00:062
		Quatity □Avg_Quatity (TBL_INVOICE_DETAILS)	Avg	00:00:00:078	00:00:00:078
		Quatity □Min_Quatity (TBL_INVOICE_DETAILS)	Min	00:00:00:093	00:00:00:062
		Quatity □Max_Quatity (TBL_INVOICE_DETAILS)	Max	00:00:00:078	00:00:00:062
		Quatity □Count_Quatity (TBL_INVOICE_DETAILS)	Count	00:00:00:093	00:00:00:046
		Quatity □StDev_Quatity (TBL_INVOICE_DETAILS)	StDev	00:00:00:093	00:00:00:062
		Quatity □Var_Quatity (TBL_INVOICE_DETAILS)	Var	00:00:00:093	00:00:00:062
Case 2	Price	price□sum_price (TBL_ITEMS)	Sum	00:00:00:031	00:00:00:046
		price □Avg_price (TBL_ ITEMS)	Avg	00:00:00:046	00:00:00:015
		price □Min_price (TBL_ ITEMS)	Min	00:00:00:031	00:00:00:015
		price □Max_price (TBL_ ITEMS)	Max	00:00:00:031	00:00:00:015
		price □Count_price (TBL_ ITEMS)	Count	00:00:00:031	00:00:00:015
		price □StDev_price (TBL_ ITEMS)	StDev	00:00:00:031	00:00:00:031
		price □Var_price (TBL_ ITEMS)	Var	00:00:00:015	00:00:00:031

The following Table VII shows the charts of implementing aggregations in Table VI with and without bitmap index (each colour represents a single function).

TABLE VII
SHOWS THE CHARTS OF IMPLEMENTING AGGREGATIONS IN TABLE VI CASE'S.

Case No	Chart
Case 1	The implementation of aggregation functions on quantity column. 
Case 2	The implementation of aggregation functions on Price column. 

C. Calculating the MVs Selection costs

In this step the response time, storage area, frequency and them costs have been calculated of each MV as shown in Fig. 3 for using them in calculating the selection cost for each MV in MVSET using eq. (10) as illustrated in Fig. 4.

	No	MV	MV storage area cost KB (SAC)	MV response time cost(RTC)	MV frequency cost (FC)
▶	1	Cost_View1	0.0008176614881...	0.53901035942743	1
	2	Invoice_View1	0.2730989370400...	0.4919927696779...	0.625
	3	Item_View1	0.0147179067865...	0.3214262708208...	0.8125
	4	ItemID_View1	0.0147179067865...	0.3988672365295...	0.75
	5	Price_View1	0.0008176614881...	0.3650668472945...	0.9375
	6	Quantity_View1	0.0008176614881...	0.3439741785366...	0.375
	7	RFrom_View1	0.0228945216680...	0.4258355179811...	0.8125
	8	RTo_View1	0.0228945216680...	0.3669232285808...	0.375

Fig. 3 The costs of MVs parameters.

W1	W2	W3	MVs selection costs
0.836702582815989	0.8148456056671...	0.79362501660065	2.06888890559305
0.109478580816406	0.7692058942137...	0.189537885221437	0.5846431415957...
0.631188419010112	0.8872162359241...	0.952249895293382	1.73624996666744
0.44073956759681	0.6086965001228...	0.944404527984748	1.50384863675984
0.791907959986435	0.0987645779264...	0.956750837134547	1.73443792439656
0.356963755263464	0.8950202655489...	0.0995455529073	0.5411894271820...
0.670507110036214	0.4771992645585...	0.376542143699034	1.11591681433914
0.522246747055206	0.7407114150657...	0.979533217372155	1.42473402691081

Fig. 4 The selection costs of each mv in MV set.

D. Implementing Firefly Algorithm

After calculating the MVs selection costs which is two-dimensional matrix contains 625 MVs selection costs as shown in Fig. 5. Firefly algorithm has been applied on this two-dimensional matrix to produce optimal initial point to the QPSO algorithm as shown in Fig. 6.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0.4281...	1.0684...	1.7886...	0.8455...	0.6938...	1.9825...	1.0465...	1.7499...	1.2695...	1.3312...	1.7291...	0.9057...	0.4101...	0.2582...	0.5927...	0.1209...	0.5519...	0.6086...	0.6257...	0.3395...	0.3944...	1.4349...	0.6434...	1.6160...	0.6575...
0.2681...	0.2022...	0.4564...	0.8872...	0.7734...	0.7196...	0.6824...	1.1714...	0.6246...	0.3278...	0.6475...	0.4024...	1.5845...	1.2284...	0.2352...	0.7441...	0.8657...	0.2703...	1.1170...	0.8518...	0.5063...	0.3708...	0.9887...	0.7129...	0.3577...
0.6360...	2.0148...	1.2476...	1.2462...	0.8014...	0.9884...	0.5252...	0.8572...	0.2955...	0.5077...	0.1995...	0.8377...	0.5889...	0.2596...	0.8245...	0.8418...	0.3343...	0.5400...	0.6893...	0.2511...	0.3691...	0.7764...	0.1812...	0.4681...	0.7884...
0.5080...	0.7404...	1.0389...	0.5271...	1.0690...	1.7411...	0.9591...	0.2672...	0.4196...	0.6355...	1.0703...	2.2237...	0.3354...	0.1864...	0.3531...	1.1562...	0.9657...	1.3268...	0.6443...	0.9680...	0.1620...	0.3149...	0.9172...	1.0463...	0.4245...
0.7542...	0.6595...	0.7300...	0.1853...	0.7751...	0.4932...	0.2822...	0.7472...	0.3682...	0.6905...	0.7700...	0.4935...	0.5740...	0.3920...	0.2968...	0.9055...	0.6553...	0.1446...	0.3267...	0.4775...	0.3446...	0.4455...	0.6830...	0.5956...	0.7510...
0.9239...	0.9374...	0.7301...	0.8357...	1.0449...	0.7837...	1.0543...	1.1254...	0.0877...	0.9759...	0.3595...	1.3119...	0.7002...	0.6797...	0.3704...	0.3275...	1.1899...	1.3923...	0.7888...	0.5437...	0.9669...	0.7778...	0.9459...	0.3905...	0.2410...
1.0764...	0.1977...	0.6881...	1.2865...	1.0599...	0.8757...	1.9447...	0.5847...	0.0794...	0.9845...	0.2540...	1.1807...	1.0301...	0.6390...	0.4828...	0.9965...	0.6057...	1.0004...	0.2598...	0.5775...	0.9996...	0.1482...	1.7462...	0.9954...	0.8653...
0.4886...	1.0106...	1.1037...	1.1755...	0.7761...	1.0473...	0.8482...	0.2585...	0.9909...	0.4491...	0.2193...	0.2826...	0.8833...	0.5248...	0.7202...	0.6675...	0.2335...	1.0226...	0.5591...	0.6695...	0.3558...	0.3971...	0.5650...	0.2607...	0.2735...
0.8672...	0.9709...	1.0219...	0.3786...	1.1185...	0.5675...	1.0417...	0.5214...	0.6574...	0.3335...	1.9909...	0.7522...	0.2461...	0.7522...	0.7898...	0.9523...	0.2710...	0.2982...	0.9935...	0.3217...	1.0428...	1.1630...	1.0481...	0.2326...	1.4919...
0.7541...	0.3323...	0.5162...	0.3724...	1.2212...	0.6241...	0.7136...	0.0812...	0.3485...	1.2811...	1.1291...	1.2311...	1.4231...	0.3422...	0.5913...	0.4643...	0.5439...	0.4421...	0.4973...	1.0272...	0.7402...	1.0197...	0.5791...	1.1388...	1.3697...
0.3549...	0.8014...	0.1751...	0.2733...	0.1362...	0.5072...	0.7272...	0.4429...	0.6331...	1.1066...	0.7504...	0.6320...	0.4231...	0.3425...	1.1165...	1.3333...	0.4023...	0.6730...	1.3950...	0.5863...	0.8777...	1.1087...	0.2545...	1.1936...	1.0547...
0.6733...	0.2677...	0.6733...	0.6261...	1.2114...	0.6802...	0.3200...	0.3096...	0.8900...	0.7825...	0.7265...	0.6570...	0.4581...	0.7931...	1.4377...	0.4788...	0.2536...	0.9955...	1.2968...	0.9105...	0.8998...	0.3722...	1.2411...	0.5722...	1.1704...
1.4362...	0.5057...	0.6480...	0.9902...	0.7422...	0.8084...	0.1567...	0.2186...	0.8324...	1.2842...	0.4419...	0.9022...	0.6188...	0.7994...	0.7049...	0.6936...	1.2995...	1.4367...	0.2921...	0.8096...	1.3727...	0.3480...	1.0527...	1.0504...	1.3366...
0.9432...	0.2884...	0.7743...	0.5611...	1.1775...	0.1578...	0.8576...	0.3642...	0.3616...	0.8596...	0.5021...	0.4224...	0.7381...	0.8028...	0.0848...	0.5090...	0.5554...	0.6391...	1.5099...	1.2976...	1.2606...	1.1846...	1.5378...	0.9280...	0.9035...
1.3358...	0.8466...	0.6117...	0.8268...	0.6980...	0.7859...	0.5517...	1.1414...	0.7825...	0.6511...	0.9114...	0.4121...	0.5214...	0.6009...	1.0502...	0.8350...	1.3129...	0.2703...	0.9818...	0.1903...	0.4069...	0.4265...	0.4330...	1.2965...	0.7399...
0.7661...	0.3746...	1.1435...	0.6770...	0.9196...	1.0327...	0.1433...	0.5578...	0.9896...	1.1060...	0.5441...	1.3808...	0.6469...	0.5547...	0.5902...	0.3376...	0.5943...	1.2123...	0.9677...	0.7805...	0.3507...	1.2738...	0.3780...	0.5666...	0.9454...
0.4191...	0.4302...	0.5487...	0.4674...	0.8466...	0.6301...	0.9280...	0.8648...	1.0779...	0.8854...	0.4469...	0.4008...	0.9998...	0.9915...	1.2550...	0.5531...	1.2026...	0.9496...	1.2251...	1.3186...	0.8259...	0.3541...	0.9872...	0.3132...	1.3358...
0.7132...	1.1022...	1.3835...	0.4368...	0.4913...	0.9042...	0.7647...	0.9125...	0.6983...	0.4389...	0.2159...	0.6353...	0.9769...	0.4380...	1.2622...	0.7389...	1.0375...	0.4779...	1.2531...	0.7899...	0.6436...	1.2663...	0.4099...	0.1846...	1.2853...
0.1972...	0.4758...	0.3453...	0.6492...	0.8633...	0.3471...	1.0449...	1.5232...	0.6696...	0.5653...	0.3009...	0.8167...	1.5250...	0.5588...	0.5591...	0.6111...	0.7466...	0.7363...	0.5607...	0.2627...	0.2158...	0.8903...	0.7992...	0.1624...	0.9438...
0.7319...	0.4577...	0.3468...	0.6725...	1.0289...	1.1710...	0.2929...	0.7539...	0.5597...	0.2542...	0.8743...	0.9309...	1.1954...	1.0997...	0.9769...	0.9498...	0.6070...	0.9977...	0.3030...	1.0186...	1.1359...	0.2961...	0.6666...	0.7040...	0.1563...
0.6404...	0.9069...	1.0777...	0.6024...	0.1159...	0.2791...	0.7430...	0.5730...	0.7702...	0.2931...	0.1192...	1.0119...	1.3645...	0.3695...	0.4400...	0.6116...	1.2271...	1.1738...	0.7524...	1.0413...	0.3944...	1.1061...	0.3787...	0.4476...	1.7740...
0.8052...	0.9619...	0.2613...	0.8195...	0.1741...	1.9830...	0.5036...	1.2166...	0.7352...	0.5258...	1.2966...	0.3078...	0.6550...	1.0513...	0.2680...	0.1796...	1.1502...	0.9649...	0.6742...	1.1896...	0.0239...	2.0278...	0.7685...	1.2352...	0.6170...
1.2988...	0.4959...	0.7854...	0.7820...	1.1373...	0.5512...	0.7082...	0.8724...	0.5882...	0.1036...	1.5465...	0.3175...	0.2940...	0.4017...	1.4742...	0.6214...	0.7783...	0.5090...	0.5392...	0.4420...	0.5331...	1.0426...	0.3831...	0.4491...	1.4112...
0.5076...	0.9710...	0.2521...	1.7989...	1.2312...	0.6034...	0.8032...	0.2026...	0.5892...	0.8119...	1.1703...	0.0395...	0.7937...	0.9237...	0.9052...	0.5303...	0.7456...	2.2338...	0.5326...	1.8617...	0.5299...	1.2080...	1.2244...	0.7333...	0.8887...
1.4937...	0.7398...	0.7918...	1.6299...	0.4662...	0.5511...	1.3032...	0.9148...	0.9181...	0.4455...	1.4171...	1.6946...	0.9326...	0.8592...	1.1021...	1.4216...	0.4828...	0.3123...	0.4479...	0.4587...	1.5220...	0.8434...	0.8220...	0.8135...	1.2491...

Fig. 5 Two-dimensional matrix of mvs selection costs.

	No	Xi	Location	Brighness
▶	1	0	567	2.11143589626...
	2	943.758982277...	38	1.72228556143...
	3	926.176173305...	82	1.75873401958...
	4	907.411081566...	56	1.68365132656...
	5	884.899141017...	1	1.56211294485...
	6	871.138042670...	249	1.96665292854...
	7	845.181616044...	73	1.59448639657...
	8	816.022539846...	58	1.52030584930...
	9	815.290203431...	81	1.55059886210...
	10	809.950252535...	271	1.85413856459...
	11	776.385634219...	108	1.51439059098...
	12	774.040618161...	236	1.69465585126...
	13	773.09421323485	293	1.80252680254...
	14	760.426771014...	34	1.38785597592...

Fig. 6 The implementaion of FA

E. Implementing QPSO Algorithm

The last step in the proposed work is using QPSO algorithm to find optimal MVs which have low response time, low storage space and high frequency as shown in Fig. 7.

	No	i	j	ValueB	Location
▶	2	3	2	0.55320241...	55
	3	4	2	1.68365132...	56
	4	5	2	0.25847996...	57
	5	6	2	1.52030584...	58
	6	7	2	0.45660596...	59
	6	6	2	1.52030584...	58
	6	7	2	0.45660596...	59
	7	7	3	0.20871029...	85
	8	8	3	0.84849458...	86
	9	9	3	0.77243825...	87
	10	10	3	0.57240399...	88
	11	11	3	0.65581114...	89
	12	12	3	0.80946172...	90
	13	13	3	0.79477296...	91

Fig. 7 The implementation of QPSO algorithm.

Quantum Particle Swarm Optimization algorithm used three random numbers one of them is normally distributed on [0,1] and the other are uniformly distributed on [0,1]. The implementation of Gaussian distribution and uniform distribution of random numbers in quantum PSO algorithm have been illustrated in Fig. 8.

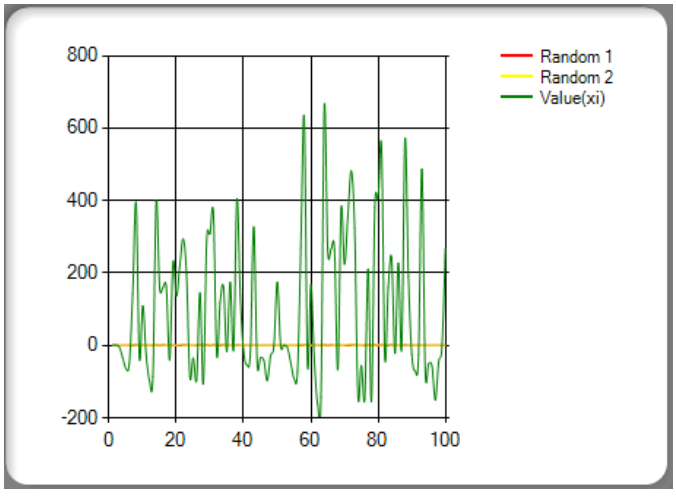


Fig. 8 The distributions of random numbers.

The location of each particle in the search space resulted by QPSO algorithm is illustrated in fig. 9.

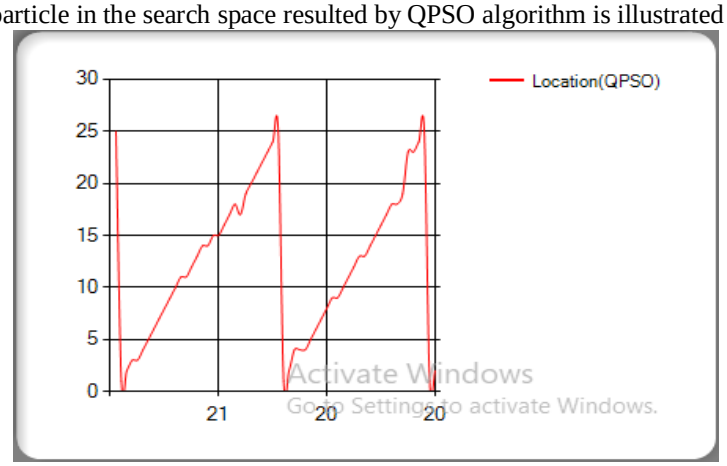


Fig. 9 The position of each particle in QPSO algorithm.

IX. RESULTS DESCUSION

The implementation results proved that bitmap index reduce the MV creation total time. The time difference of applying aggregation functions on one column is sales system DW over base table and over bitmap index is illustrated in Fig. 10, where the total time of applying these functions over bitmap index was found 168 milleseconds, while directly over base tables was found 216 milleseconds.

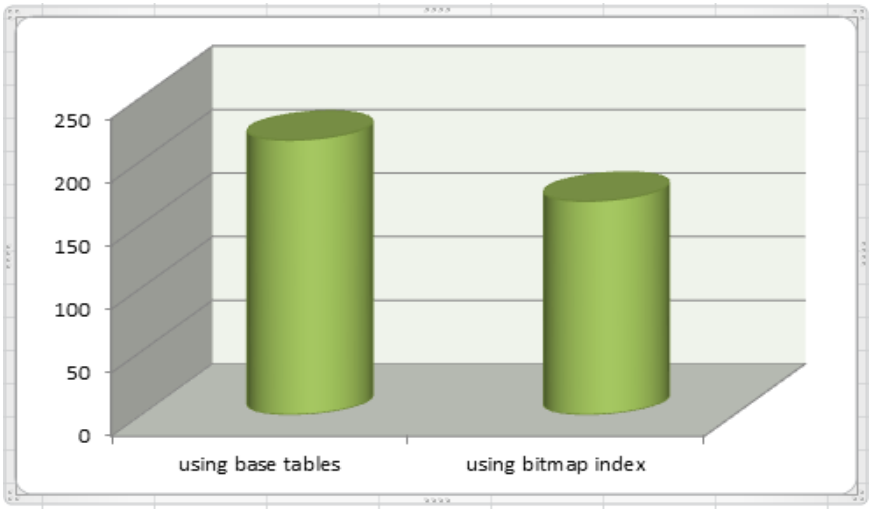


Fig. 10 The time difference of aggregation functions over bitmap index and over direct access

The optimal MVs resulted by hybrid technique which evaluated based on the three factors (MVRT, MVSA and MVF) to find optimal one which has low response time, low storage area and high frequency have been illustrated in Table VIII.

TABLE VIII
OPTIMAL MVS RESULTED BY HYBRID TECHNIQUE

Iteration NO.	Location	MVSC	SA	RT	F
60, 197	441	1.2717	×	✓	×
74, 166, 168	395	1.3202	×	✓	✓
76, 78	397	0.2530	×	✓	×
79	398	0.6829	×	×	✓
80	399	0.9762	✓	✓	×
33, 101, 183	458	0.7567	×	✓	×
160	414	1.1459	×	×	×
29, 95, 177	454	0.7611	×	✓	×
61, 67,200	442	0.4616	✓	✓	✓

The optimal MV which has low response time, storage area as well as high frequency has been found in iteration 61 and the system came back to it again in iterations 67 and 200.

X. CONCLUSIONS

The proposed study introduce an efficient technique to select most beneficial MVs that have low response time and storage area besides high frequency. The implementation results proved that bitmap index achieves good results because it takes less time and storage area in recoupoing results since bitmap indices have the ability of accomplishing processes on index level before recouping the base relations. Also, the hybrid technique (FA and QPSOA) was more efficient in term of optimal MVs selection time since, FA presents optimal initial point to QPSO algorithm.

REFERENCES

[1] D. yao, A. Abulizi, and R. Hou, *"An improved algorithm of materialized view selection within the confinement of space,"* IEEE Fifth International Conferenc on Big Data and Cloud Computing , 2015.

[2] I. Kumar and T. V. V. Kumar, *"Materialized View Selection using Discrete Genetic Operators based Particle Swarm Optimization,"* International Conference on Inventive Systems and Control (ICISC), IEEE , 2017.

[3] Mr. P. P. Karde and Dr. V. M. Thakare, *"SELECTION OF MATERIALIZED VIEW USING QUERY OPTIMIZATION IN DATABASE MANAGEMENT : AN EFFICIENT METHODOLOGY,"* International Journal of Database Management Systems (IJDMS), Vol.2 (4), November 2010.

[4] S. Choudhary, and R. Mahajan, *"A Design and Implementation of Efficient Algorithm for Materialized View Selection with its Preservation in a Data Warehouse Environment,"* International Journal of Science and Research (IJSR), Vol. 4, Issue 11, November 2015.

[5] J. Rajurkar, and T. K. Khan, *"Efficient Query Processing and Optimization in SQL using Compressed Bitmap Indexing for Set Predicates,"* IEEE Sponsored 9th International Conference on Intelligent Systems and Control (ISCO), 2015.

[6] Ashadevi. B, *"Analysis of View Selection Problem in Data Warehousing Environment ,"* International Journal of Engineering and Technology ,Vol.3 (6), pp.447-457, Dec 2011- Jan 2012.

[7] N. Bagheri and M. Sadeghzade, *"Using Bat Algorithm for Materialized View Selection in Data Warehouse,"* International Journal of Advanced Research in Computer Science and Software Engineering , Vol. 4, Issue 10, October 2014.

[8] M. A. Saleh*," Optimal Selection of Materialized Views and Indexes for Quality Data Warehouse,"* A thesis submitted to the Department of Computer Science- College of Computers Science and information technology- University of Anbar In a partial fulfillment of the requirements for the degree of Master in computer science, 2016.

[9] S. U. Khan, W. M. Shah, M. Haris and R. Naseem , *"Data Warehouse Enhancement Manipulating Materialized View Hierarchy,"* IEEE , 2013.

[10] K. Stockinger and K. Wu, *"Bitmap Indices for Data Warehouses,"* Computational Research Division Lawrence Berkeley National Laboratory , University of California, 2007.

[11] B. P. S. Chattopadhyay, S. Mitra, R. Chowdhury and Dr. M. De, *"Study & Comparison of Indexing Models in Data Warehouses,"* International Journal of Software Engineering & Practices, Vol.2, Issue 3, July 2012.

[12] N. Sharma and A. Panwar, *"A Comparitive Study of Indexing Techniques in Data Warehouse,"* International Journal of Multidisciplinary Sciences and Engineering, Vol. 3(7), July 2012.

[13] M. M. Hamad and M. Abdul-Raheem, *"EVALUATION OF BITMAP INDEX USING PROTOTYPE DATA WAREHOUSE,"* International Journal of Computers & Technology, Volume 2 (2), April 2012 .

[14] Asokan K and A. Kumar R, *"Application of Firefly algorithm for solving Strategic bidding to maximize the Profit of IPPs in Electricity Market with Risk constraints,"* International Journal of Current Engineering and Technology, Vol.4(1), February 2014.

[15] S. K. Pal , C.S Rai and A. P. Singh *"Comparative Study of Firefly Algorithm and Particle Swarm Optimization for Noisy Non-Linear Optimization Problems ,"* I.J. Intelligent Systems and Applications, 2012.

[16] J. KWIECIEŃ and B. FILIPOWICZ, *"Firefly algorithm in optimization of queueing systems",* BULLETIN OF THE POLISH ACADEMY OF SCIENCES TECHNICAL SCIENCES, Vol. 60, No. 2, 2012.

[17] Z. T. M. Al-Ta'i and O. Y. Abd Al-Hameed, *"Comparison between PSO and Firefly Algorithms in Fingerprint Authentication"*, International Journal of Engineering and Innovative Technology (IJEIT), Volume 3, Issue 1, July 2013.

[18] L. Jing-hui and X. Wen-bo, *"The Optimization about PID controller's parameter based on QPSO Algorithm,"* IEEE, 10th International Symposium on Distributed Computing and Applications to Business, Engineering and Science, 2011.

[19] R. Li and D. wang, *"Clustering Routing Protocol for Wireless Sensor Networks Based on Improved QPSO Algorithm,"* IEEE, international conference on advanced mechatronic systems, Xiamen, China, December 6-9, 2017.

[20] M. Chen, J. Ruan, D. Xi, *"Micro Grid Scheduling Optimization Based on Quantum Particle Swarm Optimization (QPSO) Algorithm,"* IEEE, 2018.

[21] Z. Dongming, X. Kewen, W. Baozhu and G. Jinyong, *"An Approach to Mobile IP Routing Based on QPSO Algorithm,"* IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, 2008.

[22] I. R. Mohammed, *"Image Steganography based on the behavior of Particle Swarm Optimization,"* A Thesis Submitted to the Council of the College of Computer - University of Diyala as a Partial Fulfillment of the Requirements for Degree of Master, 2018.

[23] M. Clerc and J. Kennedy, *"The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space,"* IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 6(1), FEBRUARY 2002.

[24] J. Sun, C. Lai and Xiao- Jun, *"Particle Swarm Optimization Classical and Quantum Perspective,"* A Chapman & Hall/CRC Press, 2011.

[25] Y. A. Turky," *A Dynamic Warehouse Design Using Simulation Modeling Approach,"* A Thesis Submitted to the Council of the College of Computer - University of Anbar as a Partial Fulfillment of the Requirements for Degree of Master,2016.

