A Method for Join Ordering Problem using Harmony Search Algorithm

Chintal Upendra Raval[1], Prof. Kaushal M Madhu[2]

[1] Student, Information Technology, LJ Institute of Technology, Gujarat, India
[2] Assistant Professor, Computer Engineering, LJ Institute of Technology, Gujarat, India

ABSTRACT

Finding the optimal join ordering for a database query is a complex combinatorial optimization problem which has been approached by a wide variety of strategies and algorithms, ranging from simple deterministic search to complex hybrid algorithms based on genetic search and incorporating domain-specific heuristics. In this report we review a set of join ordering algorithms and classify them according to the nature of the search strategy they implement. We also briefly discuss the relative advantages and applicability of different algorithms. In this report, a meta-heuristic method based on the Harmony Search Algorithm will be adapted to resolve the join-ordering problem.

Keywords:- *Query optimization; join ordering, relational databases: query execution plan; deterministic algorithms; randomized algorithms; genetic algorithms; hybrid algorithms, Multi Join Query Ordering, Harmony Search Algorithm.*

1. INTRODUCTION

Query Optimization is one of the most important and expensive stages in executing database queries. If these queries involve join operator between several tables, join ordering process will have a considerable effect on lowering costs of execution. This operator relates two tables through their common attributes. It is the responsibility of optimizer to find an execution plan with minimum costs or almost close on that. In queries with maximum 5 or 6 relations, the best order is easily reachable using evaluation and search in the whole possible space, and this can be accomplished in shorter than a second. But, in the case of more than 8 relations, it is not possible to find the best plan easily [4].

In traditional relational database, the number of relations in join queries is usually less than 10. In these cases, such methods as dynamic programming have been used in order to cope with this difficulty. However, in such systems as decision backup systems, data mining and OLAP, sometimes, there are more than 100 tables in join operator [4].

A central issue in relational query optimization is the selection of an effective join ordering, i.e., an order for evaluating efficiently the join predicates of a given query. For example, when joining 3 tables A, B, C of size 10 rows, 10,000 rows, and 1,000,000 rows, respectively, a query plan that joins B and C first can take several orders-of-magnitude more time to execute than one that joins A and C first [5].

The number of execution plans, among which only the most appropriate one would be adopted, increases as The size of data gets bigger and the number of relations participating in join operator arises, but traditional methods do not give proper answer for this problem [4].This problem is generally considered as NP-Hard problem [4].

The Join Ordering Problem (JOP) has been approached by several classes of algorithms. It is a generalization of the classical combinatorial Traveling Salesman Problem (TSP). The problem of finding the shortest Hamiltonian cycle in a complete graph. The TSP is among the best-studied comma trial optimization problems and dozens of algorithms have been proposed for it. Most of these algorithms are directly applicable to the JOP (which is

considerable newer). In this review however, we consider only algorithms already applied to the JOP. An extensive survey of general global optimization algorithms is not in the scope of this work.

2. JOIN ORDERING PROBLEM

2.1 Description of Join Ordering Problem

When query command entered by user, at first, Parser analyses the commands syntactically, and transforms it into a standard form for next time if no error has been recognized. Next, optimizer receives the standard form and finds an execution plan, and transmits it to query execution engine. Finally, when execution of codes completed, the query results will be returned. **"Fig-1"** demonstrates these stages [4].

In optimization problem, input is a query graph (join graph) including all participant relations (tables) in join. These relations are considered as graph nodes. Search space or solution space is a set of plans providing same results for the problem. A solution is described by processing tree showing examination of join statements. Processing tree is a binary tree which its leaves and internal nodes are basic tables and join operator, respectively. Edges determine current of data movement from vertices to root [4].

Although finding an optimum execution plan for query is theoretically possible, in most cases, optimizers provide one effective and acceptable execution plan [4].

In addition to the join ordering, type of join operation is another parameter having considerable effect on costs of final execution. Join can be of Nested Loop Join, Merge Join, or Hash Join type. In the present study, Nested Loop Join is the only type to be assessed.

The goal of optimization is to find a point with the minimum cost in search space. Cost can be measured as tuples which should be read from disc and/or written on disc. In this study, it is assumed that execution environment is not distributed and that the content of database is much bigger than the capacity of main memory [4].

Five relations, known as R1, R2, R3, R4, and R5 have been taken into account, which participate in a query command Q in join operator. Processing tree can be left-deep, right-deep or bushy type. In left-deep trees, right part of each node is consistently a basic relation. Generally, with n relations, it may create $\binom{2(n-1)}{n-1}(n-1)!$ processing tree and n! Left-deep tree. **"Fig-2"** shows three kinds of join tree [4].

2.2 Cost Function

To estimate the cost of join process between several relations, a simple model is used: the sum of the tuples number about intermediate results decides the cost of QEP. The required parameters are:

N(R): Number of Tuples in Table R.
D (A, R): Numbers of Distinct Values of Attribute A in Table R.

It is assumed that the values of fields in the tables are distributed evenly. Considering two relations R1 and R2 and their common attributes X, the number of tuples resulting from their join process is

$$n(T) = \frac{n(R1) \times n(R2)}{\max(D(X,R1), D(X,R2))} \qquad (2.1)$$

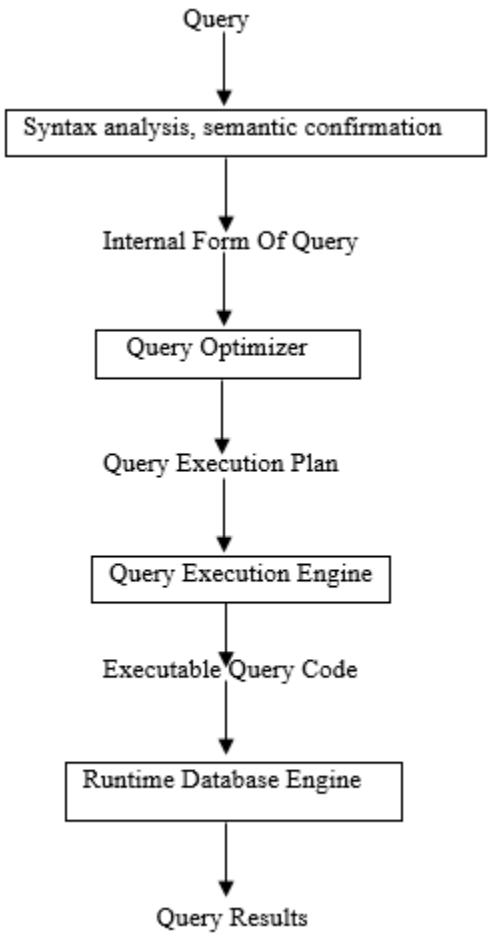

Fig-1: Process of query execution [4].

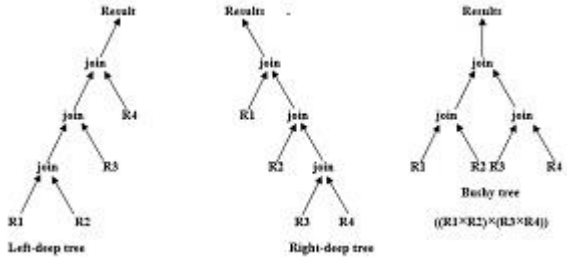

Fig-2: Three Kinds of Join Tree [4]

3. HARMONY SEARCH ALGORITHM

The harmony search algorithm (HS) is a meta-heuristic approach inspired by natural processes of musical performances. It was developed by Geem and al. [8] in 2001, and has been studied by many researchers as Lee and al. [9].

The algorithm consists of finding a perfect state of harmony in a musical orchestra in which each musician plays a note, to find a better harmony. In a similar manner, each musician plays a note in the broadest possible to form a band with other musicians. If all the notes played by all the musicians are seen as harmonious, then it is stored in the memory of each of the musicians in order to get the same optimal result for the next time.

The HS algorithm consists of five main steps. The following algorithm (Algorithm.1) represents the optimization of the search algorithm band procedure.

Step 1. Initialize the algorithm parameters.
Step 2. Initialize the harmony memory.
Step 3. Improvise a new harmony.
Step 4. Update the Harmony Memory.
Step 5. Check the stopping criterion.

These steps are described in the next five subsections.

3.1 Initialization of the Parameters

In this step, we initialize the algorithm parameters: the number of solutions generated (HMS), the rate of memory considered (HMCR), the adjustment rate (PAR) and the other stopping criteria like the maximal number of iterations.

3.2 Initialization of the Harmony Memory

The initiation of the band memory HM is to generate the HMS solutions in a random way; each x solution is consisting of N elements. For each solution the objective function f is calculated, the equation (3.1) presents the general structure of the HM.

$$HM=\begin{bmatrix} x_1^1 & \cdots & x_n^1 | f(x^1) \\ \vdots & \ddots & \vdots \mid \vdots \\ x_1^{hms} & \cdots & x_n^{hms} | f(x^{hms}) \end{bmatrix} \quad (3.1)$$

3.3 Improvising a New Harmony

Generate a new vector x'. For each component x_i' with probability hmcr (harmony memory considering rate; 0 ≤ hmcr ≤ 1), pick the stored value from HM: $x_i' \leftarrow x_i^{int(u(0,1)*hms)+1}$ with probability 1-hmcr, pick a random value within the allowed range.

Perform additional work if the value in Step 2 came from HM. With probability par (pitch adjusting rate; 0 ≤ par ≤ 1), change x_i' by a small amount: $x_i' \leftarrow x_i' + \delta$ or $x_i' \leftarrow x_i' - \delta$ for discrete variable; or $x_i' \leftarrow x_i' + fw.u(-1,1)$ for continuous variable with probability 1-par, do nothing. If x' is better than the worst vector x^{worst} in HM, replace x^{worst} with x'.

3.4 Update the Harmony Memory

The new generated harmony replaces the worst one stored in the memory band (HM), only if its physical condition (measured in terms of the objective function) is better than the worst harmony.

3.5 Checking the Stopping Criterion:

The execution of the algorithm terminates when the maximum number of repetitions is reached or when the algorithm finds the right harmony.

4. ADAPTATION OF THE HARMONY SEARCH ALGORITHM TO THE JOIN ORDERING PROBLEM

The search algorithm is based on the following parameters:

HM: The harmony memory that contains the entire solution.
HMCR: Probability to choose a node in the memory.
PAR: Probability of adjustment to choose a neighboring node.

The first phase of adaptation shown in Algorithm.2 is to initialize the HMCR, PAR, and HMS settings.

The second phase consists of initializing the HM harmony memory of the HM solutions in a random way so that each solution represents a Hamiltonian cycle of relations.

The third phase consists of starting to look for a solution that depends on values of parameters until the stop condition is satisfied.

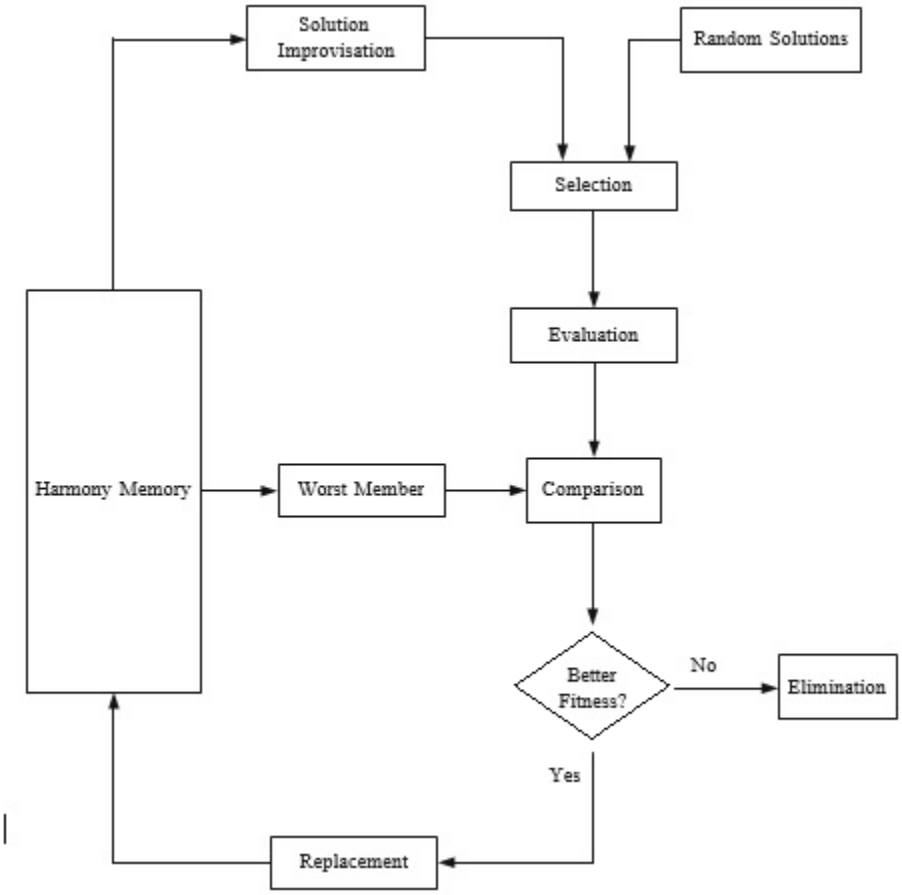

Fig-3: Working of Harmony Search Algorithm.

In this adaptation, the algorithm stops looking for new solutions when the number of iterations of searches exceeds the maximal number, or when the algorithm manages to find the optimum of the problem's instance.

The next step is to generate a new cycle. For each index from 1 to n (number of relations) of the new solution randomly selects a number between 0 and 1 (PHMCR). If the number is strictly less than HMCR, then memory is considered, otherwise the algorithm choses a relation in a random way. If the memory is considered then the algorithm selects a relation from the memory. Once the relation is obtained, we generate another random number between 0 and 1 (PPAR), if the latter is less than the value of the adjustment's parameter PAR, then the algorithm places the relation obtained by one of his neighbors. Once the relation is recovered, the algorithm inserts it into the current index of the new solution. Before each insertion into the new solution, the algorithm checks for the selected relation, to avoid closing the cycle.

After having generated the new solution, the algorithm applies the descent local search method on this solution to obtain a solution for the problem S, if the S solution is better than the wrong solution of the HM memory, then it releases the wrong solution of the memory and integrates the new one. Once the memory updates itself, the algorithm determines the position of the poor solution of the memory, increases the number of iterations, then restarts the process.

The approach of this article is applied to solve the JOP problem using the local search method as the descent method.

Algorithm: Adaptation of Harmony Search Algorithm for Join Ordering Problem

```
Initialization of the parameters: HMS, HMCR and PAR.
Initialization of the HM memory by HMS solutions.
While ((iteration <maximum number of iterations) and (the algorithm has not reached the best problem)) do
    For each i relations do
        If (PHMCR <HMCR) then
            Choose a relation from the HM column i.
            If (PPAR<PAR) then
                Replace the selected relation with one of its neighbors.
            end If
            else
            Select a relation randomly.
        end If
        Place the chosen relation in the current position of the new graph.
    end for
    Improve the new solution by the local search method.
    Update HM memory.
end while
Return the best solution of the HM.
End
```

5. EXPERIMENTAL RESULTS AND DEBATE

The adaptation of the proposed algorithm is coded into a program language C+ + on visual studio 2013, the results are executed on a computer Intel (R) Core (TM) i3 CPU T6570@2.10GHZ 2.10GHz and 6.00 GB of RAM. Instances used belong to JOPLib library. The cost of join between the relations are registered in a matrix, the initial solution are randomly generated for each cycle, and the time of creation of the cost matrix are not included in the execution time of the algorithm. In the algorithm, there are three key parameters that influence the performance results: the size of the memory band (HMS), the rate of the memory consideration (HMCR) and the rate of the pitch adjustment (PAR). The values of the HMS and HMCR parameters used are 40 and 0.95 [9]. For the value of the rate adjustment we applied a combination of values in the interval [1, 40] with steps of 1.

Table-1: Values of Adaptation Parameters

Parameter	Value
HMS	40
HMCR	0.95
PAR	0.55

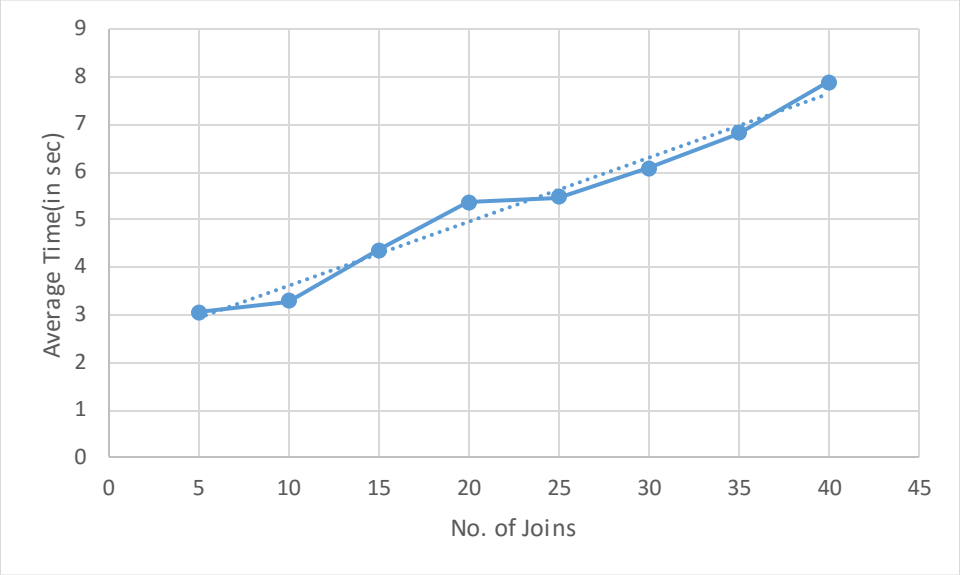

Chart-1: Average execution time for Harmony Search Algorithm for JOP.

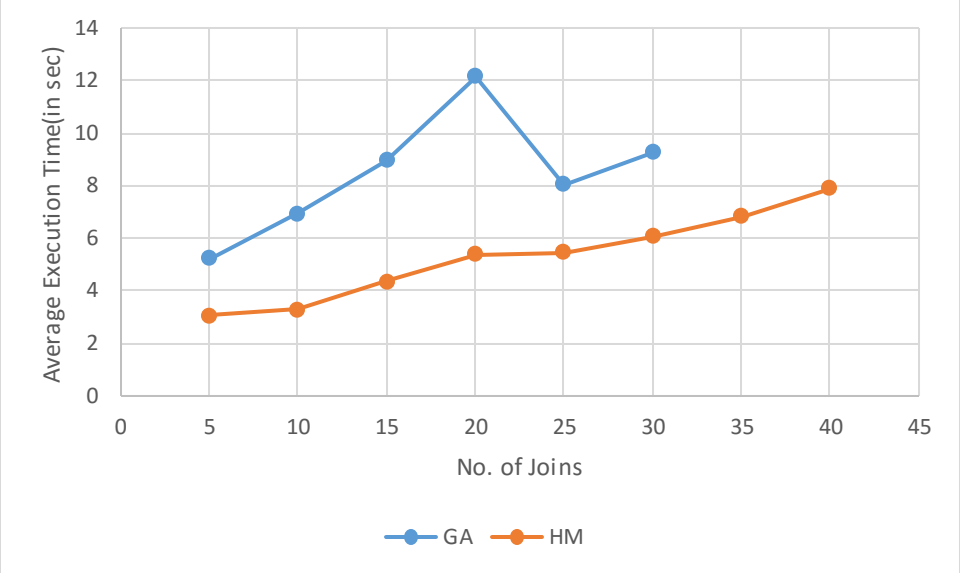

Chart-2: Comparative Analysis of Harmony Search and Genetic Algorithm for JOP.

4. CONCLUSION

Multi join query optimization useful and motivating research problem in the field of database .The propose method can be used to find Reasonable solution more efficiency than other algorithm, which fastest convergence rate among all known solution for JOP. The success of any database management system (DBMS) depends on how the query model is exploited. MJQO is very important in database research field. A good optimization algorithm not only improves the efficiency of queries but also reduces query execution time.

It reduces the response time of query processing .Harmony Search Algorithm towards the optimization of DBMS queries is still a novice field. There are still many opportunities to generate optimized solutions and to refine search strategies using of Harmony Search Algorithm for the Queries in RDBMS especially when the size and complexity of the relations increase with a number of parameters influencing the query.

The success of any database management system (DBMS) depends on how the query model is exploited. MJQO is very important in database research field. A good optimization algorithm not only improves the efficiency of queries but also reduces query execution time.

5. ACKNOWLEDGEMENTS

I am very grateful to **Dr. A. C. Suthar,** Director of L. J. Institute of Engineering and Technology, for providing facilities to achieve the desire milestone.

I also extend my thanks to Head of Department **Prof. Gayatri S. Pandi** for her inspiration and continuous support.

I wish to warmly thank my guide**, Prof. Kaushal M Madhu** for all his diligence, guidance, encouragement, inspiration and motivation throughout. Without his valuable advice and assistance it would not have been possible for me to attain this landmark**.** He has always been willingly present whenever I needed the slightest support from him. I would not like to miss a chance to say thank for the time that he spared for me, from his extremely busy schedule. I am very obliged to all my dear friends for their continuous livelihood and comfort in each and every phase of my life.

I would like to thank all of them whose name are not mentioned here but have played a significant role in any way to accomplish the work. Grace of the almighty God and blessings of my parents have formed the path to reach my desire goal.

6. REFERENCES

PAPERS

[1] Ahmed Khalaf Zager Al saedi, Prov Madya Dr.Rozaida bt.Ghazali and Prof. Dr. Mustafa Bin Mat Deris, "An Efficient Multi Join Query Optimization for DBMS Using Swarm Intelligent approach," Information and Communication Technologies (WICT),DOI: 10.1109/WICT.2014.7077312, pp.113-117,IEEE, Dec 2014.

[2] Mukul Joshi and Praveen Ranjan Srivastava, "Query Optimization: An Intelligent Hybrid Approach using Cuckoo and Tabu Search," International Journal of Intelligent Information Technologies, DOI: 10.1155/2014/727658,IJIIT,2013.

[3] Miguel Rodríguez, Daladier Jabba, Elias Niño, Carlos Ardila and Yi-Cheng Tu, "Automata Theory Based Approach to the Join Ordering Problem in Relational Database Systems," 2nd International Conference on Data Management Technologies and Applications, DOI: 10.5220/0004433802570265, Research Gate, Sep. 2013.

[4] HamidReza Kadkhodaei and Fariborz Mahmoudi, "A combination method for Join Ordering Problem in relational databases using Genetic Algorithm and Ant Colony," Granular Computing (GrC), 2011 IEEE International Conference, DOI: 10.1109/GRC.2011.6122614, pp. 312-317, IEEE, Nov 2011.

[5] Swati V. Chande and Madhavi Sinha, "Genetic Optimization for the Join Ordering Problem of Database Queries," India Conference (INDICON), 2011 Annual IEEE, DOI: 10.1109/INDCON.2011.6139336, pp. 1-5, IEEE, Dec. 2011.

[6] S. Vellev, "Review of Algorithms for Join Ordering Problem in Databse Query Optimization," Information Technologies and Control,Research Gate,2011.

[7] Morad bouzidi and Mohammed essaid riffi, "Adaptation of the Harmony Search Algorithm to Solve the Travelling Salesman Problem", Journal of Theoritical and Applied Information Technology, Vol. 62 No. 1, ISSN: 1992-8645, Nov 2014.

[8] X.-S. Yang, "Harmony Search as a Metaheuristic Algorithm", in: Music-Inspired Harmony Search Algorithm: Theory and Applications (Editor Z. W. Geem), Studies in Computational Intelligence, Springer Berlin, vol. 191, pp. 1-14 (2009)

[9] Xiaolei Wang, Xiao-Zhi Gao and Kai Zenger, "An Introduction to Harmony Search Optimization Method," Springer, DOI 10.1007/978-3-319-08356-8,pp.13-17,Jan 2015.

[10] Florian Waas, Arjan Pellenkoft, "Join Order Selection (Good Enough Is Easy)", Springer, DOI 10.1007/3-540-45033-5_5,pp 55-57,Dec 2000.

[11] Michael Steinbrunn, Guido Moerkotte, Alfons Kemper, "Heuristic and Randomized Optimization for the Join Ordering Problem", Springer, DOI 10.1.1.53.6198,pp 11-27,,Dec 2010.

WEBSITES

[12]http://www.intechopen.com/books/advanced-technologies/toward-optimal-query-execution-in-data-grids

[13] http://www.benjaminnevarez.com/2010/06/optimizing-join-orders/

[14] http://www.benjaminnevarez.com/2010/06/optimizing-join-orders/

[15] https://docs.oracle.com/cd/F49540_01/DOC/server.815/a67781/c20c_joi.htm

[16] http://blog.tanelpoder.com/2010/10/06/a-the-most-fundamental-difference-between-hash-and-nested-loop-joins/

BOOKS

[17] Database System Concepts Sixth Edition by Avi Silberschatz,Henry F. Korth,S. Sudarshan,McGraw-Hill ISBN 0-07-352332-1