International Journal of Modern Computation, Information and Communication Technology, 2018;1(4):83-87.
ISSN: 2581-5954
http://ijmcict.gjpublications.com

Research Article

Query Execution and Optimization in Distributed Database Environment

Muhammad Haroon*

Department of Computing and Information Technology, University of Gujrat Lahore Sub Campus, Lahore, Pakistan.

*Corresponding author's e-mail: haroon.capricorn@gmail.com

Abstract

Query optimization refers to the execution of a query in earliest possible time by consuming a reasonable disk space. A query execution plan is generated before execution and the optimal operator tree is got in the search space. Research with different examples is conducted in the present work by providing up to date information and example work.

Keywords: Query optimization; Query processing; Query execution; Distributed Database.

Introduction

The global success of databases especially at commercial level is very much dependent on the query optimization. SQL queries run on a specified time but as we know, as Joins are very useful in SQL queries on one hand, they are much expensive on the other hand [1]. A join and non-join query imparts a handsome difference in terms of time cost. The query that works according to our requirement is not always useful for us i.e. if data is updating on non-bearable time. We need to optimize query for better performance. This is actually called Query Optimization [2]. It becomes more challenging if this is need to be done in distributed database systems where there are many replications and fragments spread over different servers and platforms.

Query processing

Query processing involves the conversation and translation of high level SQL query into some low level instructions that database engine can read and execute the query [3]. As shown in Fig. 1, the inputted SQL query is first parsed by Query Parser and then translated by Query Translator. It is optimized then and move towards Execution Engine where required result is achieved. In distributed database system environment, the query is broken down in different steps and then forwarded towards the fragments for execution.

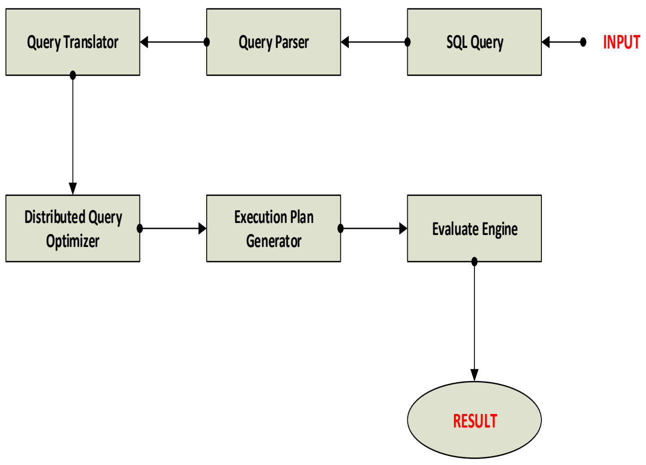

Fig. 1. Query processing

Query optimization

Executing a query by controlling and limiting space and time refers to *Query Optimization*. Every query gives same result but the query with best time and space average is what we need to have and use professionally. Referring to Fig. 1, the step 'Distributed Query Optimizer' is the step where optimization of query is performed. The Query Execution Plan (QEP) is generated which is the plan to optimize query by considering all parameters [3].

Query optimization description

In relational databases, first task is to analyse the SQL query for its possible

Received: 05.09.2018; Received after Revision: 20.09.2018; Accepted: 21.09.2018; Published: 27.09.2018

optimization. There are some cases where query optimization becomes NP-Hard problem i.e. where number of relations in query is not fixed. The second task is to determine the access strategy i.e. sequential scan or index.
Sample join query is:
SELECT admin.*
FROM employee e, admin a, student s, teacher t
WHERE e.id = a.id AND a.id = s.id AND s.id = t.id

The execution plan for this query will be:
To execute join based query, related tables must be located on same server or at least related fragments on same server [4]. With these, one must specify access strategy with redistribution prior to the join.
The module that does query optimization is called Query Optimizer.

Components of query optimizer

Query Optimizer obtains query from query translator after parsing from query parser and works on three components:

i. Search Space
ii. Search Strategy
iii. Cost Model

Search Space

Search Space refers to all feasible sets of operator tree for a join or cartesian product query (Fig. 2). Permutations of the join order in a query imparts a meaningful effect on the execution plan of the query. In comparison of Fig. 3 and Fig. 4, the query gives same result but there is a difference in join permutation that leads to a cost difference between the two. Search space is actually to having all the possible and feasible permutation of operator tree of SQL query [5]. The same case occurs with the cartesian product based queries. Similarly with the joins, the cartesian product is also a costly functionality to use and there need to be very careful to use this. Cartesian product also generates very much permutations and acquire a large search space which requires very carefulness dealing with this.

Limitations of search space

Number of alternative operator trees for a query by applying commutative and associative rules is $O(N!)$ which becomes un-economical in case of a complex SQL queries including many numbers of relations and operators (joins or cartesian products) because factorial elevates exponentially [3].

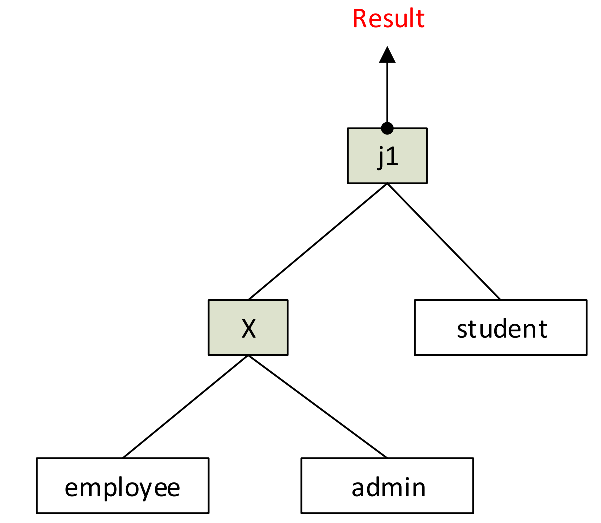

Fig. 2. Operator tree for Cartesian Product Query

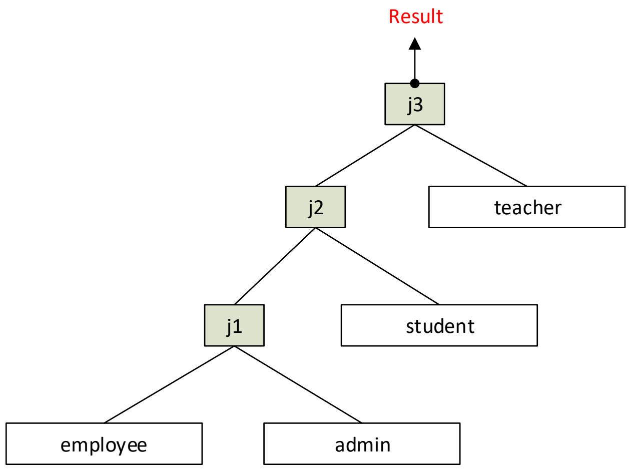

Fig. 3. Operator tree for Join Query (A)

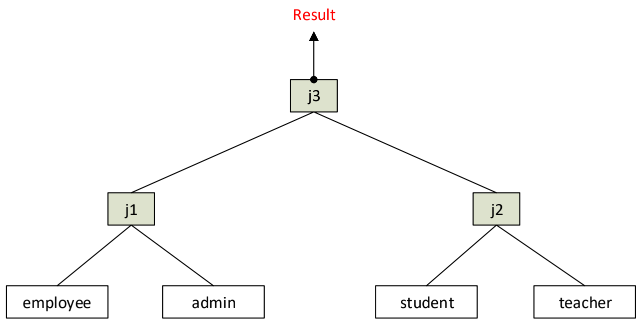

Fig. 4. Operator tree for Join Query (B)

The other limitation or restriction is to apply needless heuristics that generates the useless operator trees in search space which is not recommended at all. For example, the possible operator trees are having joins and cartesian operator trees whereas there may be a case where cartesian product trees are not required.

Orientation of the operator trees is also a limitation. An SQL query produces two operator trees Fig. 3 and Fig. 4. The both trees have different orientations in which one is more economical than other. In this way, there is a restriction over orientation of operator trees. In distributed environment, the tree in Fig. 4 looks more economical due to its parallelism in joins.

Search strategy

Query optimizer uses search strategy to produce the best operator tree by applying the approach of dynamic programming. As the dynamic programming is a deterministic approach, so the technique is to produce operator tree step by step. Dynamic Programming uses breadth first search (BFS) to calculate the best one and discard the rest to avoid the wastage of search space [6]. On contrary, the Greedy Search approach produces only one plan which is the best by not doing exhaustive search. It uses depth first search (DFS).

Fig. 5 shows the query optimizer actions in deterministic strategy done by dynamic programming of the same query we've discussed before. It progresses step by step by executing one join, then second and so on. But this approach becomes expensive in case of too many relations and operators.

Cost model

The model that describes the cost of the query execution in the system. It is the estimated cost of a complete plan for the operator trees in the search space.

Cost model in DBMS

Cost model in Database Management System (DBMS) depend upon various factors that includes: Access Type, Selection of optimal operator tree, CPU and I/O cost etc. The access type refers to the method of accessing data from database i.e. sequential scan or index based [7]. Of course the sequential scan requires more cost than index scan because in sequential scan, there need to be full scan of the relations by looking up tuples one by one that requires more time and space cost. On the other hand, the index based scan does not require that much time and cost space.

The selection of optimal operator tree is also very important in determining the cost. As we discussed, the operator tree can be generated by different means and all the operator trees give the same result but it makes a lot of difference in their cost calculation. This is why, the selection of operator tree means a lot in calculating the cost.

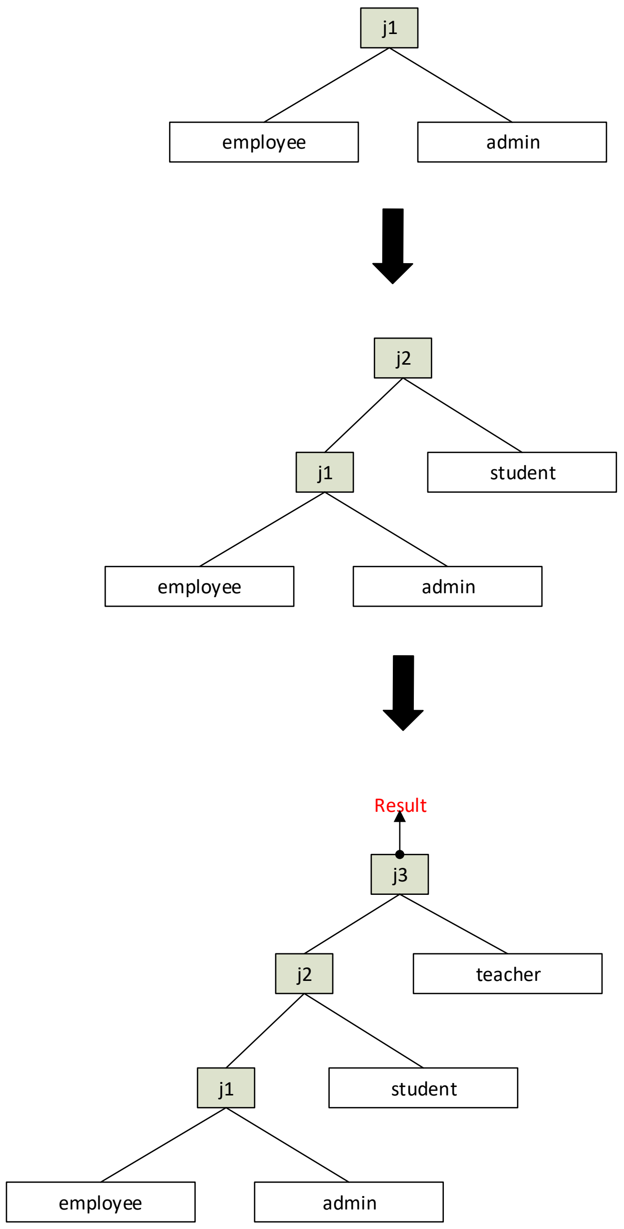

Fig. 5. Query Optimizer Actions in Deterministic Strategy

The peripherals cost like CPU and I/O cost also matters because these parameters are directly proportional to the execution time of the system.

$$Total\ \mathrm{Cos}\,t = CPU\ \mathrm{Cos}\,t + I/O\,\mathrm{Cos}\,t$$

Whereas:

$$CPU\ \mathrm{Cos}\,t = Unit\ Instruction\,\mathrm{Cos}\,t \times Number\ of\ Instructions$$

The total cost is accumulation of CPU Cost and the Input Output Cost and the CPU works on the unit instruction cost by the number of instructions.

Cost Model in Distributed DBMS

Execution of query in distributed database management systems is different from centralized database management system [8]. In distributed environment, an SQL query is sent to different fragments and replicated segments for execution. There are two types of time consumptions in distributed environment i.e. Total Time and Response Time. Total Time is the sum of all time consumptions while executing the query by ignoring the concurrency factor. Whereas the Response Time is the time that a user has to wait for the result of a query by considering concurrency factor [9] Concurrency is one of the most important factors while dealing with database.

A query is executed in various phases. The operator trees in search space is split into various phases. Let there is an operator tree ***T.***
$P(T)$ is the set of phases of operator tree T.
$\delta \in P(T)$ is one individual phase of operator tree T.
$\varphi(\delta)$ is the set of operations i.e. joins or cartesian products of a phase δ.
The total time is the time of all the operations while executing the query. In distributed environment, it is stated as:

$$Total\, \mathrm{Cos}\,t = CPU\, \mathrm{Cos}\,t + I/O\, \mathrm{Cos}\,t$$
$$+ Communication\, \mathrm{Cos}\,t$$
$$CPU\, \mathrm{Cos}\,t = Unit\ Instruction\, \mathrm{Cos}\,t$$
$$\times\ Number\ of\ Instructions$$
$$Communication\, \mathrm{Cos}\,t = Query\ Initialization\, \mathrm{Cos}\,t$$
$$+ Transmission\, \mathrm{Cos}\,t$$

Total cost is very much like the cost of centralized DBMS but there is an extra parameter i.e. communication cost which is the cost of the distributed communication of sending and receiving time from multiple fragments resided on multiple servers or platforms and that communication cost is defines as the sum of query initialization and the transmission cost.
There are some pipelined operations as well in a phase. These are such operations that are in waiting queue and pipelines to be executed in response of some query result. The pipelined operations play vital role while calculating the response time of a query execution.

$$\mathrm{Re}\,sponse Time = \sum_{i=1}^{\varphi(\delta)} Execution Time(i) + Pipeline Wait(i)$$

The response time is the summation of the execution time (time to execute a query) and pipeline wait (time to deliver a phase) of all operations in a query (i=1 to $\varphi(\delta)$).

The Execution Time is the time that is combination of the time to execute an operation i and the transmission time of getting some result and forward to some other process [10]. It also depends upon the selected algorithm. For example, in Fig. 3, the first join operation is performed and rest are pipelined. Then the result of first join operation is executed with the other join and so on. The transmission time is actually the management of this trading. In distributed environment, it is the major challenge to minimize this transmission time. With the reference to Fig. 1, the execution time can be defined as:

$$Execution Time(i) = Transmit Time(i)$$
$$+ \max\left(\cos t_{loop}\left(join(j1, student)\right), Transmit Time(j1)\right)$$

This is an example formula for above mentioned query and operator tree in Fig. 3.

In distributed DBMS, different fragments and the replicated pieces of database are spread over multiple sites and multiple servers so the query execution needs to go to every fragment to check predicates and to get relevant data and of course there are some communication links in between these sites and servers which result in communication and transmission cost [5]. For example, there is a database spread over four sites and there are some communication links in between the sites as shown is Fig. 6.

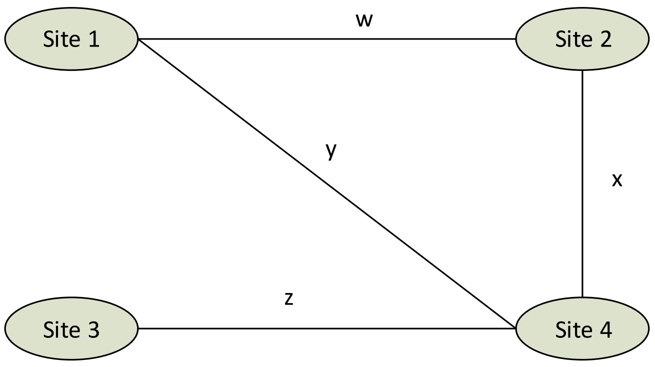

Fig. 6. Distributed Database Fragments Structure over four sites

There are four sites site 1, site 2, site 3 and site 4 with communication links w, x, y and z. The total time of a query execution over this structure will be:

$$Total\,Time = Query\,Initialization\,Time \times 4$$
$$+ Unit\,Transmission\,Time \times (w+x+y+z)$$

And the response time will be:

$$\mathrm{Re}\,sponse\,Time =$$
$$\max\begin{pmatrix} time\ to\ send\ w\ from\ site1\ to\ site2, \\ time\ to\ send\ x\ from\ site2\ to\ site4, \\ time\ to\ send\ y\ from\ site1\ to\ site4, \\ time\ to\ send\ z\ from\ site3\ to\ site4 \end{pmatrix}$$

$$time\ to\ send\ w\ from\ site1\ to\ site2 =$$
$$Query\,Initialization\,Time + unit\,transmission \times w$$
$$time\ to\ send\ x\ from\ site2\ to\ site4 =$$
$$Query\,Initialization\,Time + unit\,transmission \times x$$
$$time\ to\ send\ y\ from\ site1\ to\ site4 =$$
$$Query\,Initialization\,Time + unit\,transmission \times y$$
$$time\ to\ send\ z\ from\ site3\ to\ site4 =$$
$$Query\,Initialization\,Time + unit\,transmission \times z$$

This is how one can calculate the costs in distributed environment.

Conclusions

Query optimization is always been under discussion and research for years. This is one of the important and challenging tasks in database systems area. In addition to distributed environment, it becomes more interesting and difficult too. Query processing is also somehow complex with distributed DBMS and hence query optimization too. The understanding of distributed system is required to tackle this issue and write about it. Minimize the cost factor is actual challenge to deal with.

Conflicts of interest

Authors declare no conflict of interest.

References

[1] Ozsu MT. Valduriez, P. Principles of Distributed Database Systems. Prentice-Hall, Inc., NJ, USA: 1999.

[2] Hasan W. Optimization of SQL Queries for Parallel Machines.LNCS 1182, Springer-Verlag, 1996.

[3] Zloof MM. Query-by-Example: A Data Base Language. IBM Systems Journal 1977;14:324-43.

[4] Bhuyar PR. Horizontal Fragmentation technique in Distributed database. International Journal of Scientific and Research Publications 2012;2(5):1-7.

[5] Abadi D, Carney D, Cetintemel U, Cherniack M, Convey C, Lee S, Stonebraker M, Tatbul N, Zdonik S. Aurora: A new model and architecture for data stream management. The International Journal on Very Large Data Bases 2003;12:120-39.

[6] Chaudhuri S, Shim K. Optimization of Queries with Userdefined Predicates. In Proc. of VLDB. Mumbai. 1996.

[7] Cheng CH, Lee WK, Wong KF. A genetic algorithm-based clustering approach for database partitioning. IEEE Transactions on Systems, Man, and Cybernetics 2002;32:215-30.

[8] Abuelyaman ES. An optimized scheme for vertical partitioning of a distributed database. International Journal of Computer Science and Network Security 2008;8:310-16.

[9] Ţâmbulea L, Horvat-Petrescu M. Redistributing Fragments into a Distributed Database. International Journal of Computers Communications and Control 2008;3(4):384-94.

[10] Ganski RA, Long HKT. Optimization of Nested SQL Queries Revisited. In Proc. of ACM SIGMOD. San Francisco, USA: 1987.

****