The Comprehensive analysis and Evaluation of Physical join operators in Relational Database

Ashfaq Ahmed
Department of Computer Science
University of Management and Technology
Lahore, Pakistan
ashfaq.ahmed024@gmail.com

Dr.Muhammad Shoaib Farooq
Department of Computer Science
University of Management and Technology
Lahore, Pakistan
Shoaib.farooq@umt.edu.pk

Muhammad Ishaq Raza
Department of Computer Science
University of Management and Technology
Lahore, Pakistan
ishaq.raza@nu.edu.pk

Adnan Abid
Department of Computer Science
University of Management and Technology
Lahore, Pakistan
Adnan.abid@umt.edu.pk

Ali Raza
Department of Computer Science
University of Management and Technology
Lahore, Pakistan
aliirazii@gmail.com

Abstract— Join is a most significant operation in the relational database that provides the combination of two or more relation based on a common key. Join is most expensive operation and an efficient development will increase the performance of many database queries. There are three common join algorithms, nested loop, hash and sort-merge join. SQL server supports three type of join algorithms .In this research, we propose the taxonomy of joins with our possible sub-topics which cover physical join operators. The aim of this research is the evaluation of physical join operators for query optimization. Three major types of join methods were investigated for query optimizer. The optimizer determines the best joining method and builds an optimized plan for query execution. The optimizer evaluates and analyzes the joins operator type, a number of rows in table and indices on the table column when it picked the best plan. Every join method has its own advantages and disadvantages, and it's difficult to say that which one is the best, based on different circumstances. The optimizer will decide to choose best join algorithm depends on the data statistic, indexes, and demographics if any of them are available. Our research work can help find out what join methods are to be adopted for best performance with the lowest cost.

Keywords— Joins, logical operators, physical join, operation, SQL

I. Introduction

Join is a most significant operation in a relational database that provides the combination of two or more relation based on a common key. Join is most expensive operation and an efficient development will increase the performance of numerous database queries. There are three common join algorithms, nested loop, hash and sort-merge join. SQL server supports three type of join algorithms [30] .

A join can be multi-way or two-way [1].A two-way join is a combination of two relation and multi-way when joined more than two relations in a relational database [1]. A join among n relations normally executed as a sequence of (n-1) two- way relation [3]. Join is very important as it is used in most of the queries in the database [11]. In SQL, join keyword is used to joining two or more table [13].

Joining more than two tables**:** When the number of table increase by two or more then in this condition we have to use the minimum number of joins and this use of joins is largely dependent on the number of tables. When we have n number of tables then we need (n-1) number of joins [10].

Joining Conditions:Join query operation have relation operator including these (=, =!>, <), conditions and these are used to compare the table attributes also known as columns. These conditions are called join conditions [11]. For the execution of the join, database application combined pair of a dataset and each dataset belongs to each one table which has one data. This is how joining conditions are satisfied.These joining conditions are optional but only in certain conditions [11].

In more than one table when we process the join query, then SQL machine merge the two table which is solely based on the joining conditions. Then it matches the columns from both tables and then a new resultant table is created of match results. The SQL machine carries on the procedure with every table until the required join output is not reached. The SQL optimizer explains the order which is based on the joining conditions and every available statistics for the table and indexes on the table [2].

$$R1 \bowtie_c R2 = \sigma\,(R1 \times R2)$$

Lastly with reference to join condition, specifically talking about WHERE part of joins can have a new condition which is going to address column of the single table in that address. The conditions specified can return most records when the query is done using join operation [2].

Join Operation: The join operation play a significance role for any relational database with two-way or multi-way relation, as it enables us to solve relationships between different relations [11].The general syntax of join is $R \bowtie_c S$ [37].

The rest of the paper is organized as follows: Section II consist of literature review, Section III consist of taxonomy of physical join operators, Section IV

II. LITERATURE REVIEW

In the recent years, people have tried to develop join which are efficient. The following are the join method: nest loop join, sort-merge and hash join as the efficient and these strategies also compute the equi join and non equi join of two relations. The sort-merge join method was dominantly used in early relational database systems[32].There are two joins which are specifically designed to overcome the disk I/O overhead related to general has-base join, the two joins are Grace and Hybrid hash joins [32] [20]. Shatdal et al.[22] Explained methods for an increase in the performance of hash join relevant to cache [23].

MISHRA et al.[31] In this paper different type of implementation techniques and join operators are surveyed. These techniques are classified based on how they divide rows from different tables. Some want that one rows compared to be all the rows from other relation.in further ,some join methods require implicit partitioning while other are explicit.

Barber et.al [38] analysis the random access hash join characteristics, and renew the non-partitioned hash join, present a variants of partitioned joins in which make only the partitioned, for large outer table this is more efficient than partitioned joins. Blanas et.al [21] this paper focused on the analysis of hash join algorithms regarding recent multi-core processor in the environment of main memory. This paper presents execution of hash joins in the main memory of DBMS and the operations among these hash joins are also discussed. More complex queries related to processing is considered for future work and the combined impact of load balancing, synchronization cost, computation, and cache behavior.

Kitsuregawa et al. [33] introduce GRACE hash join algorithm and more refinements of this algorithm have been proposed for the sake of avoiding I/O through keeping as numerous intermediary partitions in main memory as possible[18][19][20]. Syrdal et al. [7] calculate the joins cost that intended to guess the extent of discontinuity of audible explain by the combination of two particular units.

Swami [9] Produce result of optimization of large join queries based on combinatorial and heuristic technique. Yang et al. [12], compare the performance of all types of join methods and provide opportunity to choosing the best one based on performance and cost.

Blanas et al. [4] were the start to evaluate that partitioned-join working slow just as non-partitioned join specifically on hyper threaded machine.CAT and CHT can be apply both non-partitioned and partitioned join. Proposed a portioned SQL join that reduced inter-stock reads [6]. Recently a latch-free hash table implant and design for scalable NUMA-aware at develop stage. It is very easy hash join to a series of DIRA lookups will develop the hardware acceleration simpler [5] [3].Chen et al.[24] apply hash join algorithm to improve cache performance through perfecting method in CPU.We implements a general model for overcome the complexities successfully include with hash join algorithm. Balkesen et al.[25] Explained and compared the job processing of main memory, multicore and parallel join algorithms, which focus on radix-hash and sort-merge join. As the experimental study show that approaches relevant to sort-merge in comparison with radix-hash join only when huge information included, radix-hash join showed superiority.

Schneider et al. [26] this paper is based on comparison of results between hash join and sort merge joins and finally summarized that hash join is better if memory was limited. In parallel database systems hash join was also the main choice [27].Hemalatha et al [28] this paper performed the analysis of hah join. Nested loop join, sort-merge methods using random record generation techniques and display the result considered that nested loop was most expensive join techniques due to its number of iterations.

Yang et al.[8] survey of join methods and shows the believe that no intensely performance improvements in three main methods hash ,nested ,sort-merge in relational database.in relational database the performance improvements of join in future lie in more radical approach, join index, parallel join and layer database. Yuan et al.[31] this paper explore the map reduce framework considered in terms of optimization for multicore CPU relevant to hash , hash join relevant to partition is involved and hash join without any partition is also involved. Firstly develop algorithm for hash join with map reduce environment on multicore CPU in partition phase, and then build and probe phases. Chen et al.[32] In this article, implement two types of joins which are sort merge and block nested joins. Experimented result shows that sort merge algorithm outperform than block nested loop join on execution time in term of dissimilar amount of buffer with similar result after join.

III. THE TAXONOMY OF PHYSICAL JOIN OPERATOR

There are three main type of join methods, nested loop, hash and sort-merge join and variations of these join methods are evaluated, in addition Product join and Exclusive join were investigates in this taxonomy. SQL server supports three type of join methods [30]. The Figure 1 below explains joins methods with its variations.

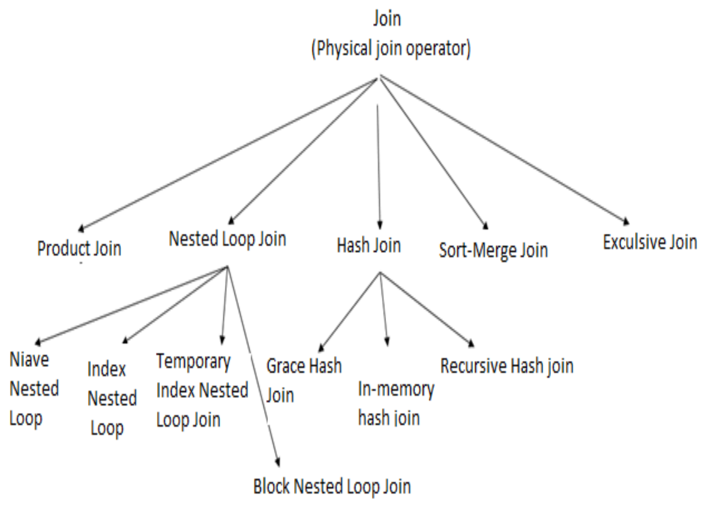

Figure 1.Taxonomy of physical join operators

Join methods: Join strategies is the mechanism for joining the two or more tuple sources. Based on statistical analysis the optimizer chooses the best join algorithm with the minimum estimated cost. We are going to discuss join methods in detail in the sub sections below.

1. Nested loop

Nested loop join is the easiest way of implementing a join [15].As the name recommends, nested loop join contains two or more loops that are nested into each other. In specific, assuming two relations the outer relation read first tuples and compare with the each tuple of inner relation. Later that, the next outer relation tuple is read and again matched to every tuple of inner relation [16].The computational complexity of nested loop join is O(n*m).Hence, nested loop join is a good starting point for comparison, thus nested loop join is brute-force strategy for a join [16].The nested loop join is hardly applicable for huge volume of data, because of its high complexity. However, it contains great potential regarding parallel execution especially for new hardware which makes it still considerable. The optimizer choose nested loop joins to execute another join in the following conditions.

- It is possible to drive inner table from the outer table.
- The amount of data is low enough to make nested loop join technique efficient.

Nevertheless, Teradata takes advantage of its index structure, "nested loops join "is enhance version of nested join in order to make nested loop optimize for selection, the following conditions must be fulfilling.

- The joining condition is based on equivalence.
- For single table the join column is unique index.
- For another table join column is any index.

1.1 Block-Nested-Loops Join

Block nested loop is improving version of nested loop join that reduce I/O cost. Specifically when considering the access gaps.it is a good way to design algorithms for a well I/O behavior.it is more efficient when considering nested loop join to take advantages of the latent hard disk through combining rows that are operate into chunks of rows ,known as blocks[16].The block size depends on the number of available main memory. For comparison the outer relation a take more space as possible, whereas the inner relation B can read page by page. Already fetched page can be used more efficiently with this algorithm. For further enhancement considering the number of rows in both relations. The outer loop relation should be smaller one, because of maximum page missing occur in outer loop relation and rows of the inner relation are read sequentially.
The block nested loop computational complexity is still O (n*m), because still every row of one relation is compared to every row of other relation [16]. To avoid some unnecessary comparison one way is to represent hash join [16].

1.2 Naive nested loop join

In this situation, the inner relation has no index that do not fulfill the join column required criteria in the nested loop join operator [34]. In this case ,SQL server will hardly choose nested loop join and instead of this tends to resort hash joins, but this is might be display on smaller join column that have a small data type or smaller data-set [34].

1.3 Index nested loop join

In this situation the inner relation tuples are compared using SEEK operator done by an index that is pre-built on latent (underlying) data-set [34]. It provide you the best nested loop performance with the required data-set [34]. This is clearly what you require your nested loop join operator.

1.4 Temporary index nested loop join

In this situation ,SQL Server to create the temporary index on the latent(underlying) tables in order to fulfill the nested loop join criteria [34].SQL will necessarily decide the cost of creating temporary indexing to perform nested loop join operation more significance than running cost of naive nested loop join or other kind of join operation[34]. The missing index that is being temporarily creating by SQL to fulfill the join kind can be identify in the misplace index:

DMVs (such assys.dm_db_missing_index_details and sys.dm_db_missing_ index_groups) [34].

2. Hash Join

Today's Hash join is a most frequently used in commercial database system to efficiently implement equijoins, hash join extensively studies over the past few years[17][18][19][20].

Hash join is simplest algorithm, the algorithm divides in two phases, first phase builds a hash table on the smallest relation and then using tuples of larger relation probe this hash table to find matches.

Conversely, the random access pattern inherits in the hashing process have slight temporal locality or spatial locality. When the available main memory for hash join is too small to hold the hash table and build relation, the hash join algorithm suffers too much random disk accesses. Grace join algorithm used to avoid this problem [17].

This algorithm known as "hash join "gets its name from "hash table "the fact that one smallest table build as hash table and possibly matched tuples from second relation(table) are searched through hashing using the smaller table[14].

Commonly the optimizer will initially identify a smaller table, then sort it through join attribute row hash order. The performance of hash join algorithm will be best if the smaller table is surely smallest table and can fit in the memory. Otherwise the database partition the row sources and join proceeds partition by partition. Then by doing the binary search of smaller table for a match the larger table is proceeded one row at a time [14].hash join is also based on equi-join. Hash join uses a dynamic hash match function and hash table to match tuples. The complexity of hash join algorithm is O ($N*h_c+M*h_m+J$).

How the optimizer considers a hash join? Optimizer consider the hash join when the following conditions are true:

- A large fraction of a small table can be joined or a relatively larger quantity of data must be join.
- The join is based on equi-join.

A hash join is most cost effective if the smaller table is surely smallest table and can fit in the memory. In this case the performance of hash join algorithm will be best [14]. In general hash join performance is batter then sort-merge joins due to sorting is most expensive.

Hash join is feasible for the relations (table) having no index or huge table has indexed [36].it is best for the case in which huge table having no indexing and execute the parallel query and return the best performance [36]. Most of the experts says that its extensive lifter join [36]

2.1 In-Memory Hash Join

This type of hash join initially compute or scans the whole small input (build input) and then build a "hash table "in memory [29]. Every tuple put into a hash bucket based on a hash value calculated for a hash key. If the whole build input is lesser then the current available memory, then all tuples can be inserted into the hash table. This phase (build) is tracked by the probe phase. The whole probe input is calculated or scanned one tuple at a time and for every probe tuple computed hash key value, the scanned related hash bucket and produced all the matches [29].

2.2 Grace Hash Join

In this type if the smaller data set does not fit into main memory, a hash join proceeds in few steps known as grace hash join. Every step has a probe phase and build phase. Firstly the whole probe and build inputs are used and then database partition row sources using hash function based on hash key into numerous records[29].if we use the hash function on hash keys assuredly that joining data set must be in same pair of record or files. So the task of joining two big inputs reduced to many but lesser cases of same tasks, then hash join applied to each partitioned pair [29].

2.3 Recursive hash join

In this kind, if the build input is very huge that inputs for an external merge would require multiple merge partitioning steps and partitioning levels. Additional partitioning steps are used for specific partitions if only some of the partitions are large [29]. To make partitioning steps as fast as possible asynchronous I/O process are used in which single thread can keep so much disk drive busy [29].

2.4 Hybrid hash-join

The hybrid hash join is a clarification of grace join algorithm in which take more benefit of extra available memory. Through the phase of partitioning, the hybrid hash join used the extra memory for two goals [35].

- To contain the present resulted buffer page for every partitions.
- To contain a whole partition in-memory ,is called "partition 0"

Because of partition 0 is never read and written from disk, hash join normally execute some I/O operations than the grace join[35].One thing is notable that this algorithm is memory-delicate, since there are two calculating demands for memory first the hash table for potation 0 and resulted buffer for other partitioning. Picking excessively big data could be the cause the algorithm to recourse, because of non-zero partition is excessively big to fit into memory [35].

3.0 Product Join

This is the most simple and basic join technique. To find a match among two relation based on join condition which is not based on equality (>, <, <>) [14].the reason why this is known as product join due to it number of comparisons needed is the "product" of the number of tuples of both relations. For example table R has 20 tuples and table S has 25 tuples, then it would needed 20 x 25=500 comparisons to identify the matching tuples. If the WHERE clause is not used then it will cause a cross join or Cartesian join which will return all the combination of tuples from both relation like above example 500 tuples are returned as a result[14].The vendor like IBM and oracle referred to as nested loop join which also make sense when it mapping to algorithms[14].this is known as a product join in Oracle, IBM and Microsoft but in Teradata it is known as the counter part of nested loop join in the further RDBMS[14].

4.0 Sort Merge Joins

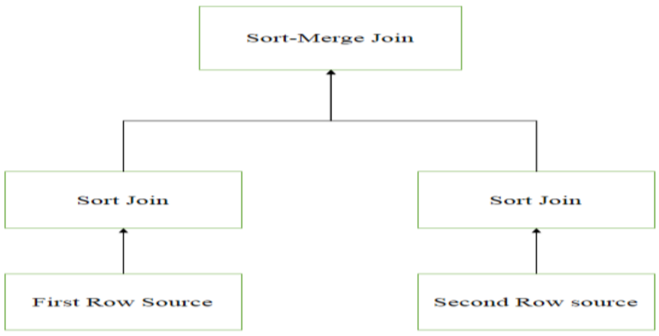

Figure 2.show the sort merge operation.

The sort merge join integrate two sorted list like a zipper.it require both row sources of the join must be sorted through join predicate. A sort merge may be performed well when the selectivity of join column is low or clustering factor is very high, and there is no index on join attributes (columns). If the outer join cannot drive from preserve outer relation to inner relation, it cannot be used a nested loop or hash joins .In this scenario sort merge outer join are used [1].The optimizer uses the sort merge from the following scenario. If the nested loop join is inefficient due to its large data volume, a hash join when the hash table not fit into memory, sort merge can be more cost-effective than hash join [1].

5.0 Merge Join

The merge join is more efficient join method. The optimizer choose merge join when the join conditions are based on equality (=). There is a precondition although the two relation must be sorted based on join attributes in earlier.
In an interleaved way, both relations only required to be scanned once [14]. The merge join is not essentially always superior to product join, because the fact that merge join is required sorting. If both relations are very large, very huge effort can be required for sorting [14].

- The time complexity of merge join is O(N+M)
- On the join key both inputs are sorted
- Must be based on equality operator
- Outstanding for very huge relation

The merge join feasible for the relation having join columns based on an index [36]. The index also non-clustered or clusterd.it is the best join in this case because it required an index for both tables [36].so it's presorted and easily compared and give the resultant data.

6.0 Exclusion Join

This join technique is used to identify the non-matching rows. The optimizer will choose the exclusion join when the query contains "EXCEPT" or "NOT IN". Exclusion join is behave like same as anti-join. In fact this type of join can be completed as either product join or Merge join[14].In general, exclusion join is based on set subtraction operation, and (TRUE,UNKNOWN,FASE) three value logic will be used if the compression is based on temporary result set or null able columns[14].

2 EVALUATION OF PHYSICAL JOIN OPERATOR

Case Study: Consider the following tables and statistics which are part of a student system.

Student (RollNo, Name, DegreeID, BatchID, …..);
Attendance (RollNo, CourseCode, Semester, AttFlag, …..);
Block Size (B) = 32 KB; Available Memory (K) = 100 Blocks;
Assume that there are 1200 matching rows in *Attendance* table per *Student* table row.

Table Name	**Row Count (*r*)**	**Row Width (*R*) (in bytes)**	**Table Size (*b*) (in Blocks)**
Student →	128,000	256	1,000
Attendance →	1,280,000	256	10,000

Example of high selectivity query#1 that returns small number of rows:
SELECT * FROM student INNER JOIN attendance ON student.rollno=attendance.rollno
WHERE student.rollno IN (1, 2, 3, 4, 5)

Example of low selectivity query#2 that returns large number of rows:
SELECT * FROM student INNER JOIN attendance ON student.rollno=attendance.rollno
The best/worst case scenarios of the following:
Nested loop join: We are explaining two cases of nested loop join, best and worst in the coming sub section.

Best case scenario: It is efficient for high selectivity query (i.e. a query that returns small number of rows).When outer table has small number of qualifying rows and inner table has large number of qualifying rows.
I/O Cost = O (outer table qualifying rows * inner table blocks)
= O (Qualifying $r_{Student}$ * $b_{Attendance}$)
= O (5 * 10,000)
= 50,000
Worst case scenario: When outer table has large number of qualifying rows.

I/O Cost = O ($b_{Student}$ * $b_{Attendance}$)
= O (1000 * 10,000)
= 10,000,000

Sort merge join: We are explaining two cases of sort merge join Best and Worst in the coming sub section.

Best case scenario: It is efficient for low selectivity query (i.e. a query that returns large number of rows).
When both operand tables are pre-sorted.
I/O Cost = O($b_{Student}$ + $b_{Attendance}$)
= O(1000 + 10,000)
= 11,000

Worst case scenarios: When both operand tables are not pre-sorted.
I/O Cost = O ($b_{Student}$ * log($b_{Student}$ /k)) + O ($b_{Attendance}$ * log($b_{Attendance}$ /k)) + O($b_{Student}$ + $b_{Attendance}$)
= O (1000 * $\log_2$(1000/100)) + O (10,000 * $\log_2$(10,000/100)) + O (1000 + 10,000)
= 80,722

Hash join: We are explaining two cases of Hash join Best and Worst in the coming sub section.

Best case scenarios: It is efficient for low selectivity query (i.e. a query that returns large number of rows).When available memory is sufficient to store at least one of the operand table.

I/O Cost = O($b_{Student}$ + $b_{Attendance}$)
= O(1000 + 10,000)
= 11,000

Worst case scenarios: When available memory is not sufficient to store at least one of the operand table (smaller).
I/O Cost = O ($b_{Student}$ * $\log_2$($b_{Student}$ /k)) + O ($b_{Attendance}$ * $\log_2$($b_{Student}$ /k)) + O($b_{Student}$ + $b_{Attendance}$)
= O (1000 * $\log_2$(1000/100)) + O (10,000 * $\log_2$(1000/100)) + O (1000 + 10,000)
= 47,542

Table 47 shows the performance of physical join operators in term of I/O.

Table 1: Physical joins methods input output cost

Join methods	Best case	Worst case
Nested loop join	I/O cost=50,000	I/O cost =10,000,000
Hash join	I/O cost=11,000	I/O cost=47,542
Sort-merge join	I/O cost=11,000	I/O cost=80,722

Table 1 shows the performance of physical join operators in term of I/O.

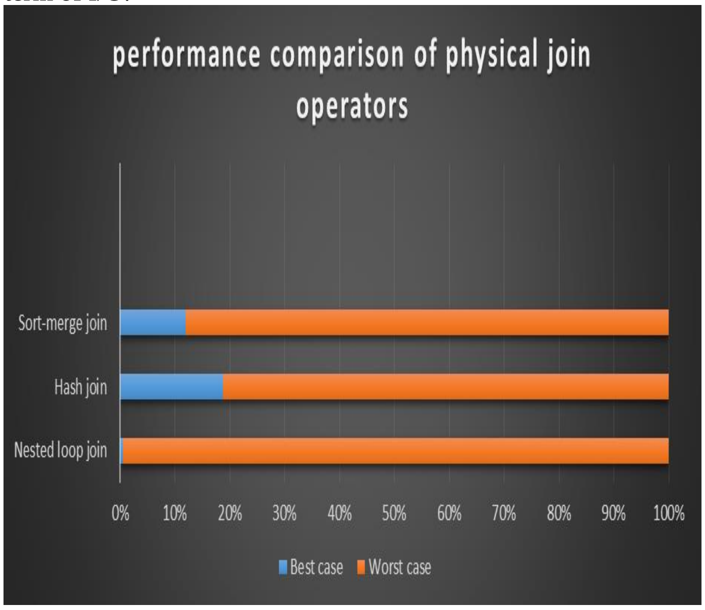

Figure 3: Performance comparison of Physical join operators

In above figure 3 shoes the performance of physical join operators in term of I/O cost.
Note: When there is a significant difference between table sizes then hash join performs well as compared to sort merge join. Otherwise when there is no significant difference between table sizes then the performance of both sort merge and hash join will be same.

3 Conclusion

In this research, we examined one of the core topics in the Database system which includes logical and physical join operators. The physical join operators were investigated its performance using a case study. Physical join operator's performance was explained and we found that each physical join operator is better than the other in some scenario. So this will help the optimizer for choosing the best approach while performing the execution. The optimizer determines the best, build an optimized plane for running the query. The optimizer evaluates and analyzes the joins operator type, a number of rows in table and indices on the table column when it picked the best plan.

• Nested loop join is considering to chosen for the small amount of data or smaller tables and if it is feasible to do seek index in the inner relation to confirming better performance.

• Merge join is considering to be excellent performance when the larger table is pre-sorted data.it only require only one comparisons and does not require a lot of comparison.

• Hash join is considering to be suitable for larger table having no indexing. It require a lot of memory, lesser I/O but require more CPU.

• The Exclusion join method is used to identify the non-matching rows. The optimizer will choose the exclusion join when the query contains "EXCEPT" or "NOT IN".

• A Sort Merge may be performed well when the selectivity of join column is low or clustering factor is very high, and there is no index on join attributes (columns).

For future direction we recommend that Optimized algorithm should be designed to handle physical join operators.

ACKNOWLEDGMENT

I am very thankful to department of computer science at University of Management and Technology for providing me peaceful and ambient environment. I am also very thankful to all the reviewers who took time out of their busy schedule for reviewing this article.

REFERENCES

[1] Mishra, P., & Eich, M. H. (1992). Join processing in relational databases. ACM Computing Surveys (CSUR), 24(1), 63-113.

[2] Oracle® Database SQL Reference 10g Release 1 (10.1), Documentation. https://docs.oracle.com/cd/B12037_01/server.101/b10759/toc.htm .Accessed-05 November 2015.

[3] MACKERT, L F, AND LOHMAN, G. M 1986 R* Optimizer: Validation and performance evaluation for distributed queries. In Proceedings of Conference on Very Large Data Bases, pp. 149-159,

[4] Blanas, S., Li, Y., & Patel, J. M. (2011, June). Design and evaluation of main memory hash join algorithms for multi-core CPUs. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (pp. 37-48). ACM.

[5] Lang, H., Leis, V., Albutiu, M. C., Neumann, T., & Kemper, A. (2015). Massively parallel NUMA-aware hash joins. In In Memory Data Management and Analysis (pp. 3-14). Springer International Publishing.

[6] Mane gold, S., Boncz, P. A., & Kersten, M. L. (2000, September). What happens during a join? Dissecting CPU and memory optimization effects. In Proceedings of the 26th international conference on very large data bases (pp. 339-350). Morgan Kaufmann Publishers Inc.

[7] Syrdal, A. K., & Conkie, A. (2005, September). Perceptually-based data-driven join costs: comparing join types. In INTERSPEECH (Vol. 5, pp. 2813-2816).

[8] Yang, Y., & Singhal, M. (1997). A comprehensive survey of join techniques in relational databases. Computer and Information Science TR, 48.

[9] Swami, A. (1989, June). Optimization of large join queries: combining heuristics and combinatorial techniques. In ACM SIGMOD Record (Vol. 18, No. 2, pp. 367-376). ACM.

[10] Pratt, Phillip J (2005), A Guide to SQL, Seventh Edition, Thomson Course Technology, ISBN 978-0-619-21674-0

[11] Ramez Elmasri, Shamkant B. Navathe, Fundamentals of Database Systems, Edition 5, Addison Wesley Pub Co Inc, 2010, ISBN 0136086209, 9780136086208, Page 183 – 184

[12] M. Tamer Özsu; Patrick Valduriez (2011). Principles of Distributed Database Systems (3rd ed.). Springer. p. 46.ISBN 978-1-4419-8833-1.

[13] http://www.studytonight.com/. Accessed-04 march, 2016.

[14] terdata online documentation join stratgies http://teradata.weizheng.net/2012/02/join-strategies-in-teradata.html. Accessed 06 December 2015

[15] Priti Mishra and Margaret H. Eich. Join Processing in Relational Databases. ACM Computing Surveys , 24(1):63{113, March 1992. (cited on Page 13, 14, 15, 16, 17, 18, 38, and 68).

[16] David broneske,"one the impact of hardware on relational join processing" ,Mastrer thesis University of Magdeburg School of Computer Science ,agust 19,2013.

[17] Kitsuregawa, M., Tanaka, H., & Moto-Oka, T. (1983). Application of hash to data base machine and its architecture. *New Generation Computing*, *1*(1), 63-74.

[18] SHAPIRO, L. D. 1986. Join processing in database systems with large main memories. ACMTrans.Datab. Syst. 11, 3, 239–264.

[19] NAKAYAMA, M., KITSUREGAWA, M., AND TAKAGI, M. 1988. Hash-partitioned join method using dynamic destaging strategy. In *Proceedings of the 14th International Conference on Very Large Data Bases* (Los Angeles, CA). 468–478.

[20] ZELLER, H. AND GRAY, J. 1990. An adaptive hash join algorithm for multiuser environments. In *Proceedings of the 16th International Conference on Very Large Data Bases*. (Brisbane, Queensland,Australia). 186–197.

[21] Blanas, S., Li, Y., & Patel, J. M. (2011, June). Design and evaluation of main memory hash join algorithms for multi-core CPUs. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data* (pp. 37-48). ACM.

[22] SHATDAL, A., KANT, C., AND NAUGHTON, J. F. 1994. Cache conscious algorithms for relational query processing. In *Proceedings of the 20th International Conference on Very Large Data Bases* (Santiagode Chile). 510–521.

[23] MANEGOLD, S., BONCZ, P. A., AND KERSTEN, M. L. 2000. What happens during a join? Dissecting CPU and memory optimization effects. In *Proceedings of the 26th International Conference on Very Large Data Bases* (Cairo, Egypt). 339–350.

[24] Chen, S., Ailamaki, A., Gibbons, P. B., & Mowry, T. C. (2007). Improving hash join performance through prefetching. *ACM Transactions on Database Systems (TODS)*, *32*(3), 17.

[25] Balkesen, C., Alonso, G., Teubner, J., & Özsu, M. T. (2013). Multi-core, main-memory joins: Sort vs. hash revisited. *Proceedings of the VLDB Endowment*, *7*(1), 85-96.

[26] D. A. Schneider and D. J. DeWitt. A performance evaluation of four parallel join algorithms in a shared-nothing multiprocessor environment. SIGMOD '89, pages 110{121, 1989.

[27] S. Fushimi et al. An overview of the system software of a parallel relational database machine grace. In VLDB, 1986.

[28] Hemalatha, G., & Thanuskodi, K. (2010, September). Optimization of joins using random record generation method. In *Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India* (p. 28). ACM.

[29] [Understanding physical joins SQL online documentation. http://www.sqlrelease.com/nested-loop-merge-and-hash-joins-in-sql-server , Accessed May 4, 2016

[30] nested loop joins https://blogs.msdn.microsoft.com/craigfr/2006/07/26/nested-loops-join/ accessed 27 april,2016.

[31] Tong, Y. U. A. N., Zhijing, L. I. U., & Hui, L. I. U. (2016). Optimizing Hash Join with MapReduce on Multi-Core CPUs. *IEICE TRANSACTIONS on Information and Systems*, *99*(5), 1316-1325.

[32] Chen, M., & Zhong, Z. (2014). Block Nested Join and Sort Merge Join Algorithms: An Empirical Evaluation. In *Advanced Data Mining and Applications* (pp. 705-715). Springer International Publishing.

[33] understanding physical join operator , http://use-the-index-luke.com/sql/join ,accessd accessed 30 May 2016.

[34] understanding the types of nested loop join, http://thinknook.com/nested-loop-join-sql-server-graphical-execution-plan-2012-03-25/ accessed 22 June 2016.

[35] DeWitt, D. J., Katz, R. H., Olken, F., Shapiro, L. D., Stonebraker, M. R., & Wood, D. A. (1984). *Implementation techniques for main memory database systems* (Vol. 14, No. 2, pp. 1-8). ACM.

[36] understanding physical join opertaors , http://www.sqlserverblogforum.com/2011/10/merge-join-vs-hash-join-vs-nested-loop-join/ ,accessed 22 June 2016.

[37] C. J. Date (2011). SQL and Relational Theory: How to Write Accurate SQL Code. O'Reilly Media, Inc. pp. 133–135. ISBN 978-1-4493-1974-8.

[38] Barber, R., Lohman, G., Pandis, I., Raman, V., Sidle, R., Attaluri, G., ... & Sharpe, D. (2014). Memory-efficient hash joins. *Proceedings of the VLDB Endowment*, *8*(4), 353-364