Data Search Using Hash Join Query and Nested Join Query

Junus Sinuraya¹, Syarifah Fadillah Rezky², Makmur Tarigan³
¹Department of Computer Engineering, Medan State Polytechnic, Medan, Indonesia
²Department of Informatics Management, Ganesha Polytechnic Medan, Medan, Indonesia
*junussiburaya12@gmail.com

Abstract. In accessing or searching data by using Query or Join on applications that are connected to a database, the usefulness of the implementation of the data and the process time is important to note. There are many ways that a system management database can do in processing and generating answers to a query. All methods will eventually produce the same answer (output) but they would still have different values, such as the time to respond to data. Some queries that are often used for processing data are Hash Join and Nested Join, both queries have different algorithms but produce the same output. A client-based or network-based application was employed to test both algorithms or queries with running time parameters or the time to respond to data. The test was conducted by connecting the number of tables and the number of rows/records. The result of the study is that the query hash join's speed in response to small amounts of data is better than the large data in the nested join.

1. Introduction

Database Management System (DBMS) is an intermediary for users with a database. A database language that has been determined by the DBMS company is used to interact with a DBMS (database). User commands interacting with the System Management Database include entering data, changing data, deleting and displaying data, or also called as Data Manipulation Language (DML). Structure Query Language (SQL) is a standard language for interacting with databases using common operations in the database. A query is a kind of ability to display data from a database where it retrieves from tables in the database, but not all tables are displayed as desired.

Join table is a combination of tables using queries that are done through certain columns/keys that have related values to get a data set with complete information. Join is needed because designing tables in transactional systems is mostly normalized, it is done to reduce redundancy.

In accessing or searching data by using Query or Join on applications that are connected to a database, the usefulness of the implementation of the data and the process time is important to note. There are many ways that a system management database can do in processing and generating answers to a query. Data search or data processing can be done by accessing data contained in the database. Data access is carried out by performing queries on databases with database management systems. The speed of data access can be increased in many ways, apart from the hardware, it can also be done from the software, more specifically in the application program.
There are several algorithms that are often used in the join, namely the Hash Join Algorithm and Nested Join Algorithm. The Hash Join works with Optimizer that makes a Hash Table based on the predicate Join. Each table in the Inner and Outer is each made a code with a Hash Function then each Hash code from the Inner will be compared to the Hash Code from the Outer. If the Hash code from the Inner and Outer is the same then a check value of the column will be carried out which will eventually be included in the result if the column value is the same. The Nested Join algorithm is an effective Join if the merged subset is small and if the conditions in the join command are efficient to combine the 2 (two) tables.

2. Research Methods

The data source for this study is a database that has 8 interconnected tables. All tables that do not have an index or cluster were used in data search.

2.1. Query Testing Scenario

Data search test used two query algorithms, Hash Join Query and Nested Join Query. Query testing in terms of data search based on a large amount of data to search data was divided into several stages of relations and data groups. The following data groups were as a trial and based on the number of different data for the research following the relation stage (table 1) and table data group (table 2). The number of records/rows of data for each table used dummy data and the addition of data using the query command to do query algorithm testing.

| No | Number of Relations |
|----|---------------------|
| 1  | 1 Relations         |
| 2  | 2 Relations         |
| 4  | 3 Relations         |
| 5  | 4 Relations         |
| 6  | 5 Relations         |

| NO | The amount of Data (Record) | Data group |
|----|-----------------------------|------------|
| 1  | 10                          |            |
| 2  | 20                          |            |
| 3  | 40                          |            |
| 4  | 80                          | Small      |
| 5  | 160                         |            |
| 6  | 320                         |            |
| 7  | 640                         |            |
| 8  | 1280                        |            |
| 9  | 2560                        | Medium     |
| 10 | 5120                        |            |
| 11 | 10240                       |            |
| 12 | 20480                       |            |
| 13 | 40960                       |            |
| 14 | 81920                       |            |
| 15 | 163380                      | Large      |
| 16 | 655360                      |            |
| 17 | 1.310.720                   |            |

Table 1. Stages of Relations

Table 2. Research Test Data Group Table
2.2. *Hash Join and Nested Join Query*

Queries of the two algorithms to be tested in this study were carried out in several stages of relations and produced the same information. To calculate the time needed to display the data sought, a client-server-based application is created.

2.3. *Query Testing Parameters*

The parameter used to test the above queries in data search is the time executing and displaying information in data search. To calculate the time needed to access data, an application is created.

![Flowchart of the Query Testing Scenario](image)

2.4. *Research Tools*

This study used research tools in the form of hardware and software as follows:

a. Hardware
   1) Intel Pentium Core Duo Processor.
   2) 2GB RAM.
   3) 500 GB hard drive.
   4) Monitor with a resolution of 1024 x 768 pixels 32 bit color.
   5) Mouse and keyboard.

b. Software
   1) Windows 7 Operating System
   2) Visual Basic.Net 2010.
   3) Microsoft SQL Server 2008.
3. Results And Discussion

Experiments on peer to peer networks with the following hardware specifications:

**Server**
- Intel Core I3 2.20 GHz processor
- 360 GB hard drive
- 2 GB of memory
- Windows 7 Operating System

**Client**
- Intel Core Duo 2.10 GHz processor
- 360 GB hard drive
- 2 GB of memory
- Windows 7 Operating System
- Microsoft Visual Basic.Net 2010

In order to make it easier to see the comparison of hash join and nested join then the data were grouped:
- Small data group ranging from 10-320 lines (Record)
- Medium data group ranging from 640-20480 lines (Record)
- Large data group ranging from 40960-1310720 lines (Record)

### 3.1. Query Test Results 1 Relations

The results of query testing by connecting 2 (two) tables in the form of queries and the different amount of data based on the group of data are the speed of the query time in conducting data searches. The results of query testing are as follows:

![Figure 2. Comparison of Query Running Time 1 Relations](image)

From the data above, it can be seen that small data group or records that have little speed (seconds) queries in performing data searches for the three queries above have almost the same speed even though the hash query join is sometimes rather slow but not too much while the medium data group has the same time with large data group. Accessing data time is different where the Nested Join Scalar query shows better than the other two queries but the time difference is not that big.

From Figure 2 above, it can be concluded as follows:

1. Hash Join query access 1 table data in full from the application with a small amount of data the time needed to access is unstable while the larger the data amount, the longer time needed to access the data.
2. Nested Join Scalar queries to access data 1 table in full from the application of a small amount of data the time needed is almost the same as the three queries and a large amount of data required the longer time than other queries.

3. Correlated Nested Join queries access 1 table data in full from a small number of data applications the time needed to access data is the same as Scalar Nested Join Queries but for the large amounts of data, it is faster than Hash Join Queries and longer than Nested Join Scalar Queries.

3.2. Query Test Results 2 Relations

Query testing by connecting 3 (three) tables displays in full 1 table to search data or access data from applications by using 3 queries namely Hash Join Query, Scalar Nested Join Query, and Correlated Nested Join Query. The following are the results of the three queries to access data from the application:

![Comparison of Query Running Time 2 Relations](image)

Figure 3. Comparison of Query Running Time 2 Relations

From the results of testing the query based on the above data, it can be explained as follows:

1. Small data group Query, the Hash Join query takes longer time than Nested Join query, and the Nested Join scalar query is better than Correlated Nested Join.
2. Medium data group Query, the Hash Join query takes longer time than nested join query, nested join scalars query is more stable than others.
3. large data Query group, the nested join are better than hash join query while nested join scalars query is more stable than correlated nested join query.

The figure 3 above can be concluded as follows:

1. Hash Join query accessed 1 table data in full from the application. The small data has an unstable time in accessing while the large data takes a long time. The larger the amount of data tested, the longer the time needed to access the data.
2. Nested Join Scalar query accessed 1 table data in full from the application. The time needed for small data is almost the same as the three queries and the large data time is better than other queries.
3. Correlated Nested Join query accessed 1 table data in full from the application. The time needed to access small data is the same as Scalar Nested Join Query but in large data, the time is faster than Hash Join Query and longer than Nested Join Scalar Query.

3.3. Query Test Results 3 Relations

The query testing connects 4 tables at once in accessing one table data in full. The following is a comparison of the time needed for the query to access data.
Figure 4. Comparison of Query Running Time 3 Relations

From the results of the query testing based on the above data, it can be explained as follows:

1. For the small data group, hash join query and nested scalar query join are more stable in the speed of accessing data while correlated nested join query is not much different from the other 2 queries but the correlated nested join is not stable.
2. For the medium data group, the hash join query has better speed than both scalar and correlated nested join.
3. The large data group time is different between hash join query and nested join query where nested join query is much faster than join hash query.

From Figure 4 above, it can be concluded as follows:

1. Hash Join query accessed 1 table data in full from the application. The small data has an unstable time in accessing while the large data takes a long time. The larger the amount of data tested, the longer the time needed to access the data.
2. Nested Join Scalar query accessed 1 table data in full from the application. The time needed for small data is almost the same as the three queries and the large data time is better than other queries.
3. Correlated Nested Join query accessed 1 table data in full from the application. The time needed to access small data is the same as Scalar Nested Join Query but in large data, the time is faster than Hash Join Query and longer than Nested Join Scalar Query.

3.4. Query Test Results 4 Relations

The query testing connects 5 tables at once in accessing one data table in full. The following is a comparison of the time needed for the query to access data.
The result of the query testing above is that there is no difference in the time needed to access in small data. The large data changes in a significant way where the time needed to access data in hash join query is greater than the Nested Join.

From Figure 5 above it can be seen that in the small data group, hash join query, scalar and correlated nested join has the same speed. For the medium data group, hash join's speed is better than other queries and in the large data group, the nested join correlated's speed is better than other queries.

### 3.5. Query Testing Results 5 Relations

The results of the query testing of the Hash Join and Nested Join query by connecting 6 (six) tables at the same time in accessing one table's data in full. The following is a comparison of query running time needed to access data.

![Figure 5. Chart of Data Search Query Comparison 3 Table Relations](image)

In the figure above, the results of testing the Hash Join query, the small data need a shorter time to access data and the large data need longer time in accessing. Nested Join Scalar Query Test Results for small data, it needs shorter time but when the amount of data is medium the results of the query tend to rise, while for the large data, it gets longer. Testing Results of a Correlated Nested Join Query are,

![Figure 6. Comparison Results Testing for Query Running Time 5 Relations](image)
for small data, the time needed is almost the same as the Hash Join Query but is slightly better than Scalar Nested Join Query for medium data and a large amount of data the time needed is also longer. From Figure 6 above, it can be explained that in the small data group hash join, correlated Nested join is better than Nested Scalar Query but in the medium data group, hash join query is better than the other two queries but in the large data group, Nested Join Scalar is far superior to than other queries.

4. Discussion of Research

Query speed to access data, especially data search, has many factors that affect the query speed, but in this study, the researchers examined queries from the speed of time to access data from client-server-based applications or network-based applications.

4.1. Analysis of Query Test Results 1 Relation

Testing data search by connecting with both tables using query hash join and nested join. The nested join is divided into two, namely scalar and correlated

The above query is executed with inner join returning all rows from both tables where there is a match or is the second slice of the table. If there is something that is not matched in the tables, then the row is not displayed and a match is found by table scan, which is to find a line match between one table and the table that is connected, all matching data will be stored in the memory so that for fewer data, it will be fast, if the data is large then the memory will be full so the process will take longer

Based on the Estimated Execution Plan, Nested Join correlated is not different with non-correlated ones and will only be longer in the data search process because before data search is processed, the subquery is processed first and it makes a long process. So large data takes a long time.

For Hash Join query, the small data group needs a shorter time to access data even though sometimes the time needed is not stable. The medium data group is better in speed than small data group where there is no time needed to access data. In the large data group, the time needed to access data is greater than other groups because the data accessed is also getting bigger.

For Nested Join Scalar query, the small data group and the medium data group have almost no value in accessing data while large data group needs more time to access larger data.

For Correlated Nested Join Query, the small data group and the medium data group have almost no value in accessing data while large data group needs more time to access larger data.

Based on the experimental data above, it can be concluded as follows:
1. Hash Join query on large data 1 relation, the time needed to access data is faster than both scalar and correlated query nested join.
2. Nested Join queries both Scalar and Correlated on small and medium data is faster than Hash join query.

4.2. Analysis of Query Test Results 2 Relations

Hash Join query that connects 3 (three) tables in a small data group, there is a change in time when accessing data compared to the relations of 2 (two) tables to be longer. Medium data group 1 relation is better than 2 relations in accessing data. Large data group 1 relation are faster than 2 relations.

Nested Join Scalar query that connects 3 (three) tables in a small data group 1 relation and 2 relations no change in time to access data and the medium data group is also almost the same there is no change in 1 relation and 2 relations remain stable but the large data group is better in relation 2 than relation 1.
Correlated Nested Join Query that connect 3 (three) tables in a small data group change occurs where 1 relation is more stable than 2 relations, and the medium group is also 1 relation is more stable than 2 relations but large data group is better in 2 relations compared to 1 relation.

Based on the experimental data above, it can be concluded as follows:

1. Small data group nested join scalars query is faster than Correlated Hash Join and Nested Join.
2. Medium data group nested scalar join query is faster than hash join and nested correlated.
3. Large data group nested scalar join query is also faster than correlated hash and nested join.

4.3. Analysis of Query Test Results 3 Relations

Hash Join Query small data group in this relation is faster than the previous relations, the medium data group is also almost the same as small data group and still better than previous relations and in the large data group, there is a greater change in the time needed to access data.

Scalar Nested Join Query for the small data group have no difference with previous relations which have changed in the medium group which is longer than previous relations and large data group is faster than previous relations.

Correlated Nested Join Query in small data group is better than the prior relations and so are medium and large data groups.

Based on the experimental data above, it can be concluded as follows:

1. In the small data group, The Hash Join query and the Nested Join Scalar query are better in terms of speed when accessing data.
2. In the medium data group, Hash Join query is faster than both nested join scalars and nested correlated queries.
3. In the large data group, Nested Join Scalar is better than the Hash Join Query and there is no big difference with the Correlated Nested Join Query.

4.4. Analysis of Query Test Results 4 Relations

Hash Join Query in the small data group, there is no difference in the previous relations in the time of accessing data and the medium data group is also the same, there is no change in time to access data but large data group requires longer time compared to previous relations.

Scalar Nested Join Query in the small data group, the time needed is no change from the previous relation for small amounts of data remains consistent in the speed of data access but there is a change in the medium data group where the time is longer than before and large data group takes longer than previous relations.

Correlated Nested Join Query in the small group is faster than previous relations and in the medium data group, there is no difference in time but there are major differences in large data group where the time is longer than the previous ones but better than other queries.

Based on the experimental data above, it can be concluded as follows:

1. Small data group, the three queries are almost the same in the speed of time to access data.
2. Medium data group, query hash join is better compared to scalar nested join and correlated nested join.
3. Large data group, nested join correlated query is better in time than the nested join scalar and hash join.
4.5. Analysis of Query Test Results 4 Relations

Hash Join query in small data group is not different from the previous relations, it remains consistent in the speed of accessing data, the medium data group is also almost the same, it remains stable in time but large data group is faster than previous relations.

Scalar Nested Join Query in small data group is different from previous relations but not too large in the difference, the medium data group takes longer time than previous relations but the large data group is faster than previous relations.

Correlated Nested Join query in small and medium data groups remain consistent as before, there is no change in time, in large data group, the time is faster than the previous relations.

Based on the experimental data above, it can be concluded as follows:

1) Small data group hash join and nested join are almost the same in speed time to access data but better than nested join scalar query.
2) Medium data group hash join query is better in the speed compared to scalar nested join and correlated nested join.
3) Large data group nested scalar join is better than correlated nested join and hash join.

5. Conclusion

After the researchers conducted the study by testing the query Hash join and Nested Query with application designed using Microsoft Visual Study 2010 and Microsoft SQL Server 2008 as a database. Running time testing of responding to data carried out for data processing, especially data search based on the number of tables realized, the number of rows/records, then the researchers make the following conclusions:

Small data group between 10-320 lines with Query Hash join is better than Nested Join. Large data group between 40960-1310720 lines with Nested scalar join is better than Hash Join query.

References

[1]. Aris A.b, (2008), Optimisasi Query Pada Sistem Database Paralel, Tersedia: http://xuyas.wordpress.com
[2]. Bratbergsengen K., (1984), Hashing Methods and Relational Algebra Operations, Proc. 10th VLDB.
[3]. DeWitt D.J., Katz R.H., Olken F., Shapiro L.D., Stonebraker M., Wood D., (1984), Implementation Techniques for Main Memory Database Systems, Proc. ACM SIGMOD Conference.
[4]. DeWitt D.J., Gerber R.H., (1985), Multiprocessor Hash-Based Join Algorithms, Proc. VLDB.
[5]. A. Gerber R. H., (1986), Dataflow Query Processing Using Multiprocessor Hash-Partitioned Algorithms, Dissertation, University of Wisconsin-Madison, Computer Sciences Technical Report.
[6]. Immanuel Chan, (2008), Oracle Database Performance Tuning Guide, 10g Release 2 (10.2), Redwood City: CA.
[7]. Kitsuregawa M., Tanaka H., Moto-oka T., (1983), Application of Hash to Data BaseMachine and Its Architecture, New Generation Computing 1, pp.
[8]. Korth, H.F dan Silberschatz, A, (1991), Database System Concepts, McGraw Hill, Singapura.
[9]. Kusrini, (2006), Optimasi Query Untuk Pencarian Data dengan Subset Query, Bandung.
[10]. Metta Santiputri, Mira Chandra Kirani, Anni, (2010), Perbandingan Cross-Product Dan Subset Query Pada Multiple Relasi Dengan Metode Cost-Based, Seminar Nasional Informatika UPN Veteran, Yogyakarta.
[11]. Nielsen, Paul., Mike White, and Uttam Parui, (2009), *Microsoft SQL Server 2008 Bible*, Wiley Publishing, Inc.,

[12]. Shapiro L.D., (1986), *Join Processing in Large Database Systems with Large Main Memory*, ACM TODS 11,3.

[13]. Nakayama M., Kitsuregawa M., Takagi M., (1988), *Hash-Partitioned Join Method Using Dynamic Destaging Strategy*, Proc. 14th VLDB, pp.

[14]. Patrick Valduriez, Georges Gardarin, (1984), *Join and Semijoin Algorithms for a Multiprocessor Database Machine*, ACM TODS 9,1.

[15]. Pong M., (1988), *NonStop SQL Optimizer: Query Optimization and User Influence*, Tandem Systems Review 4,2 pp: Tandem Computers, Part. No. 13693

[16]. Sagi Arsyad, (2008), *Pengenalan.NET dan Microsoft Innovation Center*, Universitas Indonesia, Jakarta.

[17]. Sandra Cheevers, (2006), *Oracle Database Product Familiy, An Oracle White Paper*, Redwood Shores, CA USA.

[18]. Schneider D.A., DeWitt D.J., (1989), *A Performance Evaluation of Four Parallel Join Algorithms in a Shared-Nothing Multiprocessor Environment*, Proc. ACM SIGMOD Conference: Portland, Oregon.

[19]. Setiawan, M.A., (2004), *Optimasi SQL Query untuk Informasi Retrieval pada Aplikasi Berbasis Web*, Proceedings Seminar Nasional Aplikasi Teknologi Informasi UII, Yogyakarta.

[20]. Stonebraker M. et.al., (1989), *Parallelism in XPRS*, UC Berkeley, Electronics Research Laboratory, Report M89/16.

[21]. Tom Best dan M.J. Billings, (2005), *Oracle Database 10g: Administration Workshop I, Electronic Presentation*, Redwood Shores, California USA.