JOIN ALGORITHM FOR EFFICIENT QUERY PROCESSING FOR LARGE DATASETS

J. Jayashree*, C.Ranichandra

Department of computer science, VIT University, Vellore, India.

ARTICLE INFO

Corresponding Author:
J. Jayashree,
Department of computer science,
VIT University, Vellore, India.

KeyWords: Dynamic Range Nested-loop join algorithm, Nested-Loop join algorithm.

INTRODUCTION

Join are used for joining records or fields from two or more tables in a database by using a value common to both the tables and the result set can be stored or saved in a table. There are some four types of joins and they are specified by ANSI and they are INNER, OUTER, LEFT, and RIGHT. Inner join are further classified into equi join, natural join and cross join outer join are further classified as left outer join, right outer join and full outer join. Two tables are used as examples and they are Dept ID column of the Dept table

<table>
<thead>
<tr>
<th>Emp Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>LastName</td>
</tr>
<tr>
<td>Aa</td>
</tr>
<tr>
<td>Bb</td>
</tr>
<tr>
<td>Cc</td>
</tr>
<tr>
<td>Dd</td>
</tr>
<tr>
<td>Ee</td>
</tr>
</tbody>
</table>

Dept Table

<table>
<thead>
<tr>
<th>DeptID</th>
<th>DeptName</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Sales</td>
</tr>
<tr>
<td>13</td>
<td>Engineering</td>
</tr>
<tr>
<td>14</td>
<td>Clerical</td>
</tr>
<tr>
<td>15</td>
<td>Marketing</td>
</tr>
</tbody>
</table>

Inner join are considered as a common operation of join and they are also a default type of join. Based on the predicate they combine the values of two table and the results are kept in new table. Inner join has both explicit join notation and implicit join notation.

Outer join does not expect any matching record and they do not require each record in two tables to be joined to have a matching record. Outer join does not have implicit join notation.

Explicit join notation and implicit join notation are the ways of expressing join syntax and they are specified by SQL. Explicit join notation uses the keyword join and on.

```
SELECT * FROM emp INNER JOIN dept
ON emp.DeptID = dept.DeptID;
```

Implicit join notation list the join table and they use select statement

```
SELECT * FROM emp, dept
WHERE emp.DeptID = dept.DeptID;
```

The three categories of join algorithms are
1) Nested-loop join algorithm,
2) Sort-merge join algorithm,
3) Hash-based join algorithms

1) Nested-Loop Join Algorithm

Nested-loop join is considered as one of the simplest algorithm of join where for each record of the first table the entire records of the second table has to be scanned. This process is repeated for each and every record of the first table that is for all the first table records. The execution time is calculated. In this paper Dynamic Range Nested loop Join algorithm (DRNJ) is proposed where the inner relation is going to be blocked using some dynamic range values. Based on the dynamic range values blocks are being formed. Now this block is considered as inner relation and outer table is outer relation and the same nested loop join algorithm is performed and the execution time is calculated. Comparison is made between the two algorithms based on the execution time so that the effectiveness of join algorithms is proposed.

©2011, AJCSIT, All Right Reserved.
outer loop and the inner loop. First table loop is called as outer loop and the second table loop are called as inner loop. As this Nested loop join algorithm has a repeated input/output scans of one of the table they are considered as inefficient.

Let the two tables be A and B, then the algorithm of Nested-loop algorithm are as:

For each record of table A
Read record from table A
For each record of table B
Read record from table B
Compare the join attributes
If matched
Then
Store the records

2) Sort-Merge Join Algorithm

[7] Sort merge algorithm are considered as efficient join algorithm when compared to Nested loop join algorithm. sort merge join algorithm have two operations and they are sorting and merging. In sorting operation the two tables to be joined are sorted in ascending order. In merging operation the two sorted tables are merged.

Sort records of table A based on the join attribute
Sort records of table B based on the join attribute
Let i = 1 and j = 1
Repeat
Read record A(i)
Read record B(j)
If join attribute A(i) < join attribute B(j) Then
i++
Else
If join attribute A(i) > join attribute B(j) Then
j++
Else
Put records A(i) and B(j) into the Qr

3) Hash Based join algorithm

In hash based join algorithm hashing and probing are the two processes. A hash table is created by hashing all records of the first table using a particular hash function. Records from the second table are also hashed with the same hash function and probed. If any match is found, the two records are concatenated and placed in the query result. A decision must be made about which table is to be hashed and which table is to be probed. Since a hash table has to be created, it would be better to choose the smaller table for hashing and the larger table for probing. The hash join algorithm is given as:

Let H be a hash function
For each record in table B
Read a record from table B
Hash the record based on join attribute value using hash function H into hash table
For each record in table A
Read a record from table A
Hash the record based on join attribute value using H
Probe into the hash table
If an index entry is found Then
Compare each record on this index entry with the record of table S
If matched Then
Put the pair into Qr

RELATED WORK

Author [1] proposed a new algorithm namely hash merge join algorithm which is a non block join algorithm. This hash merge join algorithm has been designed to reduce time and to produce join if they are blocked. Two phases of this algorithm are hashing phase and merging phase. In traditional algorithms of join data’s required for join are available in prior but in this paper they have considered about web based application where the data for join are available remotely so that the network traffic is to be considered. In the first phase that is hashing phase, the two tables are first hashed and then joined. In merging phase that is in the second phase the sources which are blocked are joined. The main advantage of this algorithm is that they are applicable in situations where data brought from remote source through network traffic.

Author [2] proposed a new join algorithm that is DINER algorithm which is a adaptive join algorithm and they are used in cases where the data are brought from heterogeneous network intuitive flushing policy are being used where they are used for increasing the productivity. the feature of this DINER algorithm is that they are unblocking and they deal with adaptive. the flushing policy are used in this algorithm which are used for maximizing the tuple productivity. Arriving phase, Reactive phase and Cleanup phase are the three phases of DINER algorithm. Arriving phase is the first phase of this algorithm where the main join operations are being performed. The second phase is reactive phase and they are only triggered where there is a block in both relations. Cleanup phase is the last phase where this phase starts as soon as the two relations are being received.

Author [3] proposes a parallel sort merge join algorithm and to overcome the data skew problem they use divide and conquer approach. To load across multiple processor are being balanced using scheduling phase which are performed after the relation are being sorted. The main thing is to divide the task and the load is to be balanced. The four phases are sort phase, schedule phase, transfer phase and join phase. Sort phase is the first phase where the relation are first sorted. scheduling phase is the second phase where they aim to reduce the time of completion and they are considered as the important phase of this paper where they use the output of the first phase that is the sorting phase and they balance the load. Transfer phase is the third phase where the sorted relation are transferred to the processor. The last phase is join phase where they read the records from disk then join and the joined output are written to the disk.

Author [4] proposed a adaptive join algorithm and the main concept being designed is the agent based adaptive join algorithm and they have designed it for distributed data warehousing. this agent based adaptive join algorithm are known as Ajoin. They are used in the operation of join in online and they are considered to be efficient and they are also used at the run time. they use a intelligent agent for processing the join operations and this agent is also used for the purpose of coordination. processing are taken in four different steps and they are Initialization, Remote adapting, Ripple adaptive join and result output. The main advantage of using this agent is that they filter the unqualified tuples and they reduces the cost.

Author [5] describes about text based information retrieval using hash join algorithm. Different types of hash
join algorithm are namely Xjoin, Hash Merge join and Early Hash join algorithm are described. Xjoin are the extension of symmetric hash join and they are used for less memory and it has principles like to increase the availability and in Xjoin the tuples are partitioned. Cleanup phase is the a part of Xjoin. Early hash join algorithm is also a symmetric hash join. In this algorithm, for each input a hash table is build and they contain a partition number. This partition has a link to a list. Hash merge join algorithm has two phases and they are hashing and merging. In hashing the two relations are brought and then they are stored in memory and they are merged. The results are taken in new table.

Author [6] proposed different join algorithms and they all are based on range join that is non equi join. Four algorithms are proposed in this paper and they are Basic Shifting join algorithm, Buffering Shifting join algorithm, Recursive Hamiltonian cycle join algorithm and Parallel Hamiltonian join algorithm. Two data permutation approaches are proposed in this paper and they are namely shifting approach and Hamiltonian approach. The first two algorithm are based on the first approach and the last two algorithm are based on the Hamiltonian. Basic shifting join algorithm reduces the number of processor subsets yet the data transmissions are more. Buffering shifting join algorithm reduces the number of data transmissions. The main difference in the last two algorithms of this paper which have used Hamiltonian approach are one uses only one processor while the other require all the processor. The last two algorithms require only less storage.

**PROPOSED WORK**

Create student database and connect the database with PHP server side scripting. Create tables under student database and they are mca_academic, mca_placement, mca_academic and mtech_placement. Each table contain twenty five records and the fields of academic table are id,stu_name,gender,sub1 and sub2. the fields of placement are id, stu_name, gender, tenth_per, twelth_per and curr_per.

Authentication is provided for using the database.

Nested-loop join algorithm is considered as one of the general type of join algorithm. Nested-loop join algorithm is implemented for both academic and placement. In this Nested-loop join algorithm implementation for academics mca_academic table are considered as the inner relation and the mtech_academic table are considered as the outer relation. For each of the mca_academics that is the inner relation the entire rows of the mtec_academics that is the outer relation are being scanned and compared using the common attribute and the join result are taken along with their execution time. The same process is repeated for different queries and their execution times is noted. The same Nested Loop join algorithm is implemented for placements also and they are mca_placement table are considered as the inner relation and the mtech_placement table are considered as the outer relation. For each of the mca_placement that is the inner relation the entire rows of the mtec_placement that is the outer relation are being scanned and compared using the common attribute and the join result are taken along with their execution time. The same process is repeated for different queries and their execution times is noted. The proposed Dynamic Range Nested loop join algorithm ( DRNJ) is implemented in two ways. In the first method the records of inner relation are blocked and in the second method blocking is done in both inner relation and outer relation.

**METHODOLOGY**

Nested loop join algorithm and Dynamic Range Nested loop join algorithm are implemented using PHP and mysql. Macromedia DreamWeaver is used for designing. Being a database project authentication are provided. Academic and placement are the two scenarios considered for performing the join operation. Nested loop join algorithm is performed by combining the students who have taken the same subject from both mca_academic table and mtech_academic table. Mca_academic is the inner relation and the mtech_academic is the outer relation. for each record of the inner relation the entire records of the outer relation are scanned and if there is any match then the results are taken in a new table. Execution time is calculated.

```sql
if($pid=="join" && $val=="1")
    $sql = "select * from mca_aca INNER JOIN mtech_aca on mca_aca.sub1=mtech_aca.sub1_1 where mca_aca.id=mtech_aca.id1;"
elseif($pid=="join" && $val=="1")
    $sql = "select * from mca_aca INNER JOIN mtech_aca on mca_aca.sub1_1='$sub.' AND mca_aca.sub2='$sub.' AND mtech_aca.sub2_1='$sub.' WHERE mca_aca.id=mtech_aca.id1;"
```

Dynamic Range Nested loop join algorithm is performed by combining the students who have taken the same subject from both mca_academic table and mtech_academic table. First mca_academic table is broken into blocks using dynamic range values. Now this new block is considered as the inner relation. this blocking can be done either in one table or in both the tables. Nested loop join algorithm are performed that is the new mca_academic is the inner relation and the mtech_academic is the outer relation. for each record of the inner relation the entire records of the outer relation are scanned and if there is any match then the results are taken in a new table. Execution time is calculated. Since the inner relation is being restricted to a fewer number of records the number of scan are reduced.

```sql
$sql = "select * from mca_aca INNER JOIN mtech_aca where mca_aca.id BETWEEN '$m11' and '$m22' AND mtech_aca.id1 BETWEEN '$m11' and '$m22'
AND mtech_aca.sub1_1='$s1' AND mca_aca.sub1='$s1'"
```

```php
if($pid=="join1" && $val=="2")
    $sql = "select * from mca_plc INNER JOIN mtech_plc where mca_plc.id=mtech_plc.id1 and mca_plc.tenth_per>60 and mca_plc.twelth_per>60 and mtech_plc.tenth_per>60 and mtech_plc.twelth_per>60 and mtech_plc.curr_per>60
AND mtech_plc.curr_per>60";
```
**Join Algorithm for efficient query processing for large Datasets**

```
Dynamic Range Nested loop join algorithm code is
$sql = "select * from mca_plc INNER JOIN
mtech_plc where mca_plc.id BETWEEN '$m11' and '$m22'
and mca_plc.tenth_per='".$m33."'and
mca_plc.twlth_per='".$m44."' and
mca_plc.curr_per='".$m55."' and '
'$m22';
echo $sql;
```

**SCREEN SHOTS WITH EXPERIMENTAL QUERIES**

**A. Query 1**

In figure 1, Nested loop join is performed by combining the students who have selected the subject es from both mca_academics and mtech_academic and the execution time is displayed and it is 0.0008 sec.

**B. Query 2**

In figure 2, Dynamic Range Nested loop join is performed by combining the students who have selected the subject es from both mca_academics and mtech_academic and the execution time is displayed and it is 0.0007 sec.

The same procedure is repeated for other subjects and their corresponding execution times are noted. Graph is plotted.

<table>
<thead>
<tr>
<th>Nested loop join algorithm execution time (sec)</th>
<th>Dynamic Range Nested loop join algorithm execution time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>0.0008</td>
</tr>
<tr>
<td>Query 2</td>
<td>0.0003</td>
</tr>
<tr>
<td>Query 3</td>
<td>0.0007</td>
</tr>
<tr>
<td>Query 4</td>
<td>0.0002</td>
</tr>
<tr>
<td>Query 5</td>
<td>0.0003</td>
</tr>
<tr>
<td>Query 6</td>
<td>0.0003</td>
</tr>
</tbody>
</table>

**Fig 3: subject ip in Nested loop join**

In figure 3, Nested loop join is performed by combining the students who have selected the subject ip from both mca_academics and mtech_academic and the execution time is displayed and it is 0.0007 sec.

**Fig 4: subject ip in Dynamic Range Nested loop join**

In figure 4, Dynamic Range Nested loop join is performed by combining the students who have selected the subject ip from both mca_academics and mtech_academic and the execution time is displayed and it is 0.0003 sec.

With the noted execution time of both the algorithms graph is plotted for academic tables. Placement queries are also performed like academics and their execution times are noted and graph is plotted.

<table>
<thead>
<tr>
<th>Nested loop join algorithm execution time (sec)</th>
<th>Dynamic Range Nested loop join algorithm execution time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>0.0009</td>
</tr>
<tr>
<td>Query 2</td>
<td>0.0003</td>
</tr>
<tr>
<td>Query 3</td>
<td>0.0007</td>
</tr>
<tr>
<td>Query 4</td>
<td>0.0007</td>
</tr>
</tbody>
</table>
Join Algorithm for efficient query processing for large Datasets

<table>
<thead>
<tr>
<th>Query</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.0008</td>
</tr>
<tr>
<td>6</td>
<td>0.0007</td>
</tr>
</tbody>
</table>

With the noted execution time of both the algorithms graph is plotted for placement tables.

**CONCLUSION**

From the experimental analysis it's been proved that the new Dynamic Range Nested loop join algorithm is considered efficient. The query response time is also less. In future this Dynamic Range concept can be implemented even in some of the join algorithms.

**REFERENCES**


**AUTHOR PROFILE**

J. JAYASHREE has received B.TECH in INFORMATION TECHNOLOGY from ANNA University, India 2009 and is currently pursuing master's in Networking in VIT University, Vellore. Area of interest is MOBILE COMPUTING. Have published three papers in International Journals.

C. RANICHANDRA has received B.TECH and M.TECH from VIT university, India. The author is currently working as Assistant Professor Senior in VIT and doing research work in Grid Databases.