OPTIMIZATION TECHNIQUE FOR MULTI-JOIN OPERATION IN ACTIVE DATA WAREHOUSE

1. Paramasivam . K  
Research scholar, 
Dept. of Computer Science, 
Periyar University, Salem, 
TamilNadu -India.

2. C. Chandrasekar  
Associate Professor, 
Dept. of Computer Science, 
Periyar University, Salem, 
TamilNadu - India.

ABSTRACT

In current scenario of fast changing data content in most dynamic heterogeneous application, data format transformation and data updates needs to be handled in traditional data ware housing. Many researches have been made under active data warehousing which keep the latest updates of more dynamic large multi-format data warehouses. Previous work presented a mesh join operation algorithm a frequently encountered operation which join a fast stream of source updates with a disk-based relation under the constraint of limited memory. The access cost of two join inputs is done by relying on fast sequential scans of disk-based relation and shared I/O cost of accessing disk-based relation across multiple tuples of source stream updates. Since the source streams receives. It develop a systematic cost enabled tuning of MESHJOIN for maximizing throughput under a specific memory budget, or minimizing memory consumption for a specific throughput. Since source streams have more number of relations in different directions, the previous work does not support multi-way joins.

To overcome the issues obtained from the previous work, here we are going to present a new technique optimization scheme for multiple relations of joins being generated by streams from different direction in active data warehousing retrieval of data based on multiple queries. Here, multiple on-line aggregations play a major role in the efficient and more precise user query data extraction. It evolves an optimization technique to have a more cohesive multi-relation joins of the stream tuples. Experimental simulations are carried out to evaluate the performance of proposed optimization technique with both synthetic and real data sets in terms of Execution time, and operational overhead being used to get optimal threshold for multi-relational joins generation.

Keywords: Active Data Warehousing, Optimization scheme, Multi-join operation
1. INTRODUCTION

Data warehouses are normally revived in a batch (or, offline) fashion: If data sources are met with updates, then they are defended and loaded through the Extraction-Transformation-Loading (ETL) process. This partition between querying and updating is a basic theory of traditional data warehousing applications, and obviously shortens numerous aspects of the implementation. But the main concern over traditional application is that the warehouse is not always up-to-date with respect to the recent updates, which in turn entails that the outcomes of the queries are basically an old.

To deal with this issue, current works have initiated the idea of active data warehouses. In these circumstances, all updates to the production systems are propagated immediately to the warehouse and incorporated in an on-line fashion. This model has numerous confronts, while it entails that transformations need to be achieved constantly as update tuples are streamed in the warehouse.

An active warehouse wants the estimation of the tuples of source which are streamed from the operational sources, in order to guarantee that the updates are broadcasted in a timely fashion. The main concern over active data warehouse is that it is necessary to update the tuples based on incoming tuples in the source streams. Besides, the join algorithm has to activate under partial memory because the enclosing transformation is chained to other transformations.

This paper presents an optimization technique for a multi-join operations taking part from source streams in different directions. The optimization technique for join operation improves the reliability, scalability of active data warehouses. The data retrieval might be an optimal one. The proposed optimization technique uses limited memory to allow multiple operations to operate simultaneously; the join condition is arbitrary (equality, similarity, range, etc.); the join relationship is general (i.e., many-to-many, one-to-many, or many-to-one); and the result is exact.

2. LITERATURE REVIEW

Join algorithms have been considered broadly because of the database development, and previous works have initiated a host of proficient techniques for the case of finite disk-based relations. Active or real-time data warehousing has recently
emerged [D. Burleson 2004], which provides the up-to-date information about the specific activities taken by the user. This also briefly described by [Oracle Corp 2003].

Research in Active data warehouse Neoklis Polyzotis, Spiros Skiadopoulos et. al., 2008 has presented some algorithms for definite tasks with the recognition of duplicates, the resumption from failure and the incremental loading of the warehouse. These algorithms are intended to activate in an off-line fashion. In current years, the case of uninterrupted data streams has increased in popularity in active data warehouse and researchers have observed techniques and issues for join processing over streaming infinite relations.

Earlier studies S. Chandrasekaran et.al., 2003 have presented generalizations of Symmetric Hash-Join to a multi-way join operator for an efficient handling of join queries over multiple unbounded streams. These works, also proposed by L. Golab and M. T. O’zsu 2003 with different techniques, imagine the purpose of window operators (time- or tuple-based) over the source streams, limits each stream to a restricted tuple-set in main-memory. M. Hammad, M. Franklin 2003 presented a scheduling for shared window over source streams, but this conjecture does not relate to the problem, where the working memory is smaller than the large disk-based relation.

B. Seeger, D. Taylor et.al., 2002 presented a streamed bounded relations, like the Progressive Merge Join Y. Tao, M. Yiu, 2005, and the more recent Rate-based Progressive Join presented join algorithms that uses the streaming inputs constantly and sustain the expected tuples to produce results as early as possible; but the algorithm flushes a separation of the data to disk and access it later. Obviously, this model does not compete well for on-line refreshing.

3. OPTIMIZATION TECHNIQUE FOR MULTI-JOIN OPERATIONS IN ACTIVE DATA WAREHOUSES

In active data warehouse, the information should be up-to-date with the recent values. Normally, the source streams can get the values from different directions. So, to make the recent values to be added into the warehouse, many techniques have been proposed earlier, but those techniques met with the negative effects. To make the multi-join operation more efficient in an active data warehouse, the proposed optimization
technique is used. The architecture diagram of optimization technique for multi-join operation is shown in fig 3.1.

Fig 3.1 Architecture diagram of optimization technique for multi-join operation

Active Data warehouse consists of most updated information and the multi-join operation over the tables maintained under the source streams from different directions. The source streams maintain different tables. If in case, we want to join one or more table to form a one unique table, we are going to implement a multi-join operation to perform the join operation into the tables in the source streams. After performing the multi-join operation, for a particular table, there might be the several paths to perform the join operation. To choose the best path for multi-join operation on the tables in the source stream, optimization technique is used.
An Optimization technique refers to the selection of a best element from some set of available alternatives based on multi-join operations. For a given function, \( f: T_1 \rightarrow R \), where \( T_1 \) is the table which is to be joined with the other table in the relation. If the table \( T_1 \) wants to join with \( T_3 \), then \( T_1 \) will identify the generic attributes which are present in both the tables \( T_1 \) and \( T_3 \).

### 3.1 Optimized Multi-Join algorithm

#### 3.1.1 Algorithm for building index values for a given relation

Let \( R_1, R_2, R_3, \ldots, R_n \) denote the relations and let \( A_0, A_1, \ldots, A_m \) denote the categories for \( R_1 \) and \( B_0, B_1, \ldots, B_m \) denote the categories for \( R_2 \) and \( C_0, C_1, \ldots, C_m \) and so on.

Let key denotes the key value to perform join operation.

**Step 1.** Process \( R_1, R_2, \ldots, R_n \) to determine the number of categories.

**Step 2.** The categories determined are \( A_0, A_1, \ldots, A_m \), \( B_0, B_1, \ldots, B_m \) and \( C_0, C_1, \ldots, C_m \).

**Step 3.** Build a triple index with \( 3^i \) entries \( i \) entries for \( R_1 \), \( i_1 \) entries for \( R_2 \) and \( i_2 \) entries for \( R_3 \).

**Step 4.** Obtain a hash function \( H(key) \) to evaluate the tuple \( T \) based on the key value (key) which acts as the join attribute.

Identify the index, Index= \( f (key, TupleLength) \) ……………..(eqn 3.1a)

\[
HF = \sum_{i=0}^{3^n} f_i(k, TL) \quad \text{..............................................}(eqn 3.1b)
\]

Where \( k \) is the data which contains the key value

\( TL \) is specifies the tuple length

\( f_i \) is the function to be computed

**Step 5.** For each tuple \( T \) in \( R_1 \\n
   **Step 5.1.** Find category \( A_j \) of \( T \)

   **Step 5.2.** Apply hash function \( H \).

   (Use eqn 3.1a & 3.1b)

   **Step 5.3.** Add an entry of the three fields which are the key and tuple number in the triple index in an empty place designated to category \( A_j \).

**Step 5.4.** End for

**Step 6.** For each tuple \( T_1 \) in \( R_2 \)
Step 6.1. Find category Bj of T1
Step 6.2. Apply the hash function H.
    (Use eqn 3.1a & 3.1b)
Step 6.3. Add an entry of the three fields which are the key and tuple number in
the triple index in an empty place designated to category Bj.
Step 6.4. End for

Step 7. For each tuple T2 in R3
    Step 7.1. Find category Cj of T2
    Step 7.2. Apply the hash function H.
        (Use eqn 3.1a & 3.1b)
    Step 7.3. Add an entry of the three fields which are the key and tuple number in
the triple index in an empty place designated to category Cj.
Step 7.4. End for

Once the relations are obtained from an active data warehouse, the hash function
is computed for each tuple in the relations. Then the triple index values are computed for
a given relational table.

3.1.2 Algorithm to form join index
Let R1,R2,R3…Rn denote the relations and let A0,A1,…Am denote the categories for R1
and B0,B1,…Bm denote the categories for R2 and C0,C1,…Cm and so on.
Step1. Process the first category (A0, A1,…Am) and form join operation using nested
join.
Step2. Perform
    Join operation
Step3. Until (all similar categories)
    After getting the triple index, perform join operation, until all the similar
categories are identified.

3.1.3 Algorithm to perform multi-join operation
Let the first join be Join1 with sources as A0, A1,…Am represented by A, Join2 with
sources as B0, B1,…Bm represented by B and Join3 with sources as C0, C1,…Cm
represented by C
Step1. Perform
    Step2. Join1 with A and B
    Step3. Join2 with inputs AB and new source C
    Step4. Join3 with inputs ABC and new source D
Step5. Until (operation for n relations to obtain a resultant join)

Perform multi-Join operation for all the sources obtained from relational table. The optimized multi-join operation is done efficiently for the given relational tables retrieved from an active data warehouse.

4. PERFORMANCE EVALUATION

The proposed optimization technique for multi-join operation in active data warehouse is implemented in oracle. The experiments were run on an Intel P-IV machine with 2 GB memory and 3 GHz dual processor CPU. We present an experimental study to estimate the effectiveness and performance of our proposed optimization technique for multi-join operation in active data warehouse. The disk-based relations are accumulated on a 7200RPM disk and the machine is unloaded for each experiment.

We estimate the effectiveness of the proposed optimization technique for multi-join operation on real-life insurance data with varying characteristics. The insurance data set consists of 9000 instances and 86 attributes. But here, we have taken into the consideration of 10 attributes as sample to show the performance of the proposed optimization technique.

The optimization technique which has been applied to the multi-join operation in active data warehouse, improves the efficiency of the active data warehouse to maintain as new warehouse which has updated information about the new activities taking part in the warehouse. The time taken by optimization technique, to choose the best multi-join operation for the tables to form a unique relation, is low and the it reduces the negative effects affected the tables present in the active data warehouse. The proposed affords updated tuples as soon as they are requested. This provides exact updated values with low execution time. The performance of the proposed optimization technique for multi-join operation in active data warehouse is measured in terms of

i) Execution time
ii) Operational overhead

iii) Efficiency

5. RESULTS AND DISCUSSION

When compared to an existing Meshing Streaming Updates with Persistent Data in an Active Data Warehouse, the proposed optimization technique for multi-join operation is effective in terms of execution time, operational overhead and reliability of the active data warehouse by keeping the up-to-date information in the source streams in a successful manner. Since we are using an optimization technique for multi-join operation, the active data warehouse has updated information and it also consumes less time to perform the multi-join operation.

<table>
<thead>
<tr>
<th>Relational Tables</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proposed OT for multi-join</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
</tr>
<tr>
<td>20</td>
<td>0.4</td>
</tr>
<tr>
<td>30</td>
<td>0.8</td>
</tr>
<tr>
<td>40</td>
<td>0.5</td>
</tr>
<tr>
<td>50</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 5.1 Comparison table describes the execution time

Table 5.1 described the execution time taken to perform multi-join operation for the given relational tables. The outcome of the proposed OT is compared with an existing mesh join streams for multi-join operation.

Fig 5.1 Relational tables vs. execution time
Fig 5.1 describes the execution time taken to perform the multi-join operation over the relational tables in active data warehouse. The optimization technique which has been used here which chosen the best path to perform the multi-join operation without any loss of data. Even the relational tables in the active data warehouse increases; the time taken to execute the multi-join operation on those tables is less in the proposed optimization technique for multi-join operation contrast to an existing Meshing Streaming Updates with Persistent Data in an Active Data Warehouse.

<table>
<thead>
<tr>
<th>Multi-join operation for relational tables</th>
<th>Operational overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proposed OT for multi-join</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>40</td>
<td>8</td>
</tr>
<tr>
<td>50</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 5.2 Comparison table describes the operational overhead

Table 5.2 described the operational overhead occurred while performing multi-join operation for the given relational tables. The outcome of the proposed OT is compared with an existing mesh join streams for multi-join operation.

Fig 5.2 Multi-join operation vs. Operational overhead

Fig 5.2 describes the operational overhead arises while multi-join operation takes place on relational tables in the active data warehouse. The existing Meshing Streaming
Updates with Persistent Data in an Active Data Warehouse has high operational overhead since it used mesh join operation; it might not an efficient one to perform the same operation over the active data warehouse. That is, when compared to an existing Meshing Streaming Updates with Persistent Data in an Active Data Warehouse, the proposed optimization technique for multi-join operation has less operational overhead.

<table>
<thead>
<tr>
<th>Information available in active data warehouse</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proposed OT for multi-join</td>
</tr>
<tr>
<td>10</td>
<td>22</td>
</tr>
<tr>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
</tr>
<tr>
<td>50</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 5.3 Comparison table describes the efficiency rate

Table 5.3 described the efficiency of multi-join operation for the given relational tables in an active data warehouse. The efficiency rate of the proposed OT is compared with an existing mesh join streams for multi-join operation.

Fig 5.3 Updated information vs. efficiency

Fig 5.3 describes the efficiency of active data warehouse in both the proposed optimization technique for multi-join operation and an existing Meshing Streaming Updates with Persistent Data in an Active Data Warehouse. When we implement an optimization technique on active data warehouse, the efficiency of the system increases,
contrast to an existing Meshing Streaming Updates with Persistent Data in an Active Data Warehouse.

Finally, we observed that the proposed optimization technique for multi-join operation in active data warehouse provides effectiveness in terms of reliability, efficiency, execution time and operational overhead. The proposed technique is the best one to perform the multi-join operation in active data warehouse without any negative effects.

6. CONCLUSION
In this work, we efficiently handled the issues raised over active data warehouse for multi-join operation by professionally introducing the proposed optimization technique. The proposed optimization technique performed the join operation between the source streams and the relation under a limited memory and it consumed less time to achieve the multi-join operation. We have successfully developed a systematic approach for multi-join operation without any loss of data. The experimental results showed that the proposed optimization technique for multi-join operation in active data warehouse worked efficiently by improving the efficiency and reliability of the join operation with less overhead and less execution time.

7. AUTHORS PROFILE
Mr. K. Paramasivam obtained M.Sc., Computer Science from P.S.G College, Bharathiyar University, Coimbatore, Tamil Nadu, India, in 1999, and M.Phil., Computer Science from Manonmaniam Sundaranar University, Thirunelveli, Tamil Nadu, India in 2003. He was working as IT analyst, in Department of Software development, TATA Consultancy Services, Bangalore, Karnadaka, India. Currently he is working as Lead Engineer in Department of Software development, Razorsight (US Based Company), Bangalore, Karnadaka, India. He is pursing Ph.D in Data warehouse.

Dr. C. Chandrasekar completed his Ph.D in Periyar University, Salem at 2006. He worked as Head, Department of Computer Applications at K.S.R. College of Engineering, Tiruchengode, Tamil Nadu, India. Currently he is working as Associate Professor in the Department of Computer Science at Periyar University, Salem, Tamil Nadu. His research interest includes Mobile computing, Networks, Image processing and Data mining. He is a senior member of ISTE and CSI.
REFERENCES

1) Neoklis Polyzotis, Spiros Skiadopoulos et. Al., ‘Meshing Streaming Updates with Persistent Data in an Active Data Warehouse’, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 20, NO. 7, JULY 2008


