International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 1, Issue 2, October 2014

One-to-Many Linkage query using Two Way Adaptive Clustering algorithm

P.Vijayameena[1], R. Chithra Devi[2]

PG Scholar, Department of IT, Dr. Sivanthi Aditanar College of Engineering, Tiruchendur[1]
Assistant Professor, Department of IT, Dr. Sivanthi Aditanar College of Engineering, Tiruchendur[2]

Abstract: De-duplicating one data set or linking several data sets are increasingly important tasks in the data preparation steps of many data mining process. Record linkage is traditionally performed among tables to cluster the data. Old methods are taken long time for one-to-many linkage. Two Way Adaptive clustering algorithm (TWACA) is a new proposed technique, which has a small memory footprint, allowing many such operators to be active in parallel. TWACA is optimized to produce initial results quickly and can hide intermittent delays in data arrival by reactively scheduling background processing. We show that TWACA is an effective solution for providing fast query responses to users even in the presence of slow remote sources. Optimized One-to-Many queryclustering operation and execution is the main aim of this project. The first work of this project is projection. Required column only filtered from various data base. Then, three phases of TWACA named Arriving, Reactive, and Cleanup derived for result rate maximization.

Keywords: Data set, data linkage, clustering, two way adaptive clustering algorithm

I. Introduction

Record linkage is traditionally performed among the entities of same type [2]. It can be done based on entities that may or may not share a common identifier. Optimization is a critical component in the processing of each query submitted to a database management system (DBMS). The difference between a properly optimized query and an improperly optimized one is huge, as a clustering following a suboptimal plan can take exponentially more time than the same query executed following an optimal plan. To determine the most efficient manner in which to execute a given query, the DBMS must evaluate execution strategies and choose among the most favorable calculated.

In recent years, database researchers have begun to rethink the mechanisms for such clustering optimization. In part, this is due to the expansion of the internet and the distributed queries this has made possible, but the techniques uncovered can also be applied to non-distributed environments. The research focuses on the consideration of processing factors at the time that queries are executed, and the adaptation of their execution in response to real-time data sizes and transfer rates. [1] propose will outline the growing research field of adaptive Clustering processing. It will start by first briefly explaining the mechanisms in existing, traditional query processing common today and will then describe the problems with traditional query processing that adaptive query processing seeks to overcome. And, lastly will outline various experimental solutions.

A. Adaptive Query Processing

Adaptive query processing refers to a set of techniques devised to combat the inherent of traditional query processing. It is aimed at producing optimal query plans at times that traditional plans fail. These techniques come in two factors:

i) proactive**operators**, which are coded into an execution plan and autonomously adapt to run-time changes.

ii) reactive**optimizers**, which reform the plan and reorder pre-existing operators due to run-time conditions.

II. Related Work

A. Projection

Given that each input has a handling cost, there are twocases that we have to consider:

1. The cost of performing one projection is less than or equal to the inter-arrival time for input objects (C $1ri$ $\pi \leq$).

2. The cost of performing one projection is greater than the inter-arrival time for input objects ($C\pi>1ri$). In this discussion, we incorporate the transmission cost T into the handling cost. If we want to distinguish between them, we can calculate the cost of handling an individual input as $C\pi+T$. Figure2 shows the consequences of each case. In the

first case, the inter-transmission interval is equal to the interarrival interval, with the only difference being that the first output element appears after *Cπ* time units, so *ro*=*ri*. In the second scenario, the situation is more complicated but we can figure out the inter-transmission interval from Figure 2 by observing that this interval has to be equal to the cost of handling one input. So, the transmission rate is the inverse of that, or *ro*1*C* π =. I most cases we can safely assume, however, that the cost of making a projection is low, so for small values of *C* π.

B. Data Linkage

Data linkage refers to the task of matching entities from two different data sources that do not share a common identifier (i.e., a foreign key). Data linkage is usually performed among entities of the same type. It is common to divide data linkage into two types, namely, one-to-one and one-to many. In one-to-one data linkage, the goal is to associate one record in table TA with a single matching record in table TB. In the case of one-to-many data linkage, the goal is to associate one record in TA with one or more matching records in TB.

One-to-one data linkage was implemented using various algorithms including: an SVM classifier that is trained to distinguish between matching and non matching record pairs ; calculating expectation maximization or maximum-likelihood estimation (MLE) to determine the probability of a record pair being a match; employing hierarchal clustering to link between pairs of entities ; and performing behavior analysis to find matching entities . These methods assume that the same entities appear in the two data sets to be linked, and try to match between records that refer to the same entity. Therefore, these works are less relevant for data linkage between entities of different types. Only a few previous works have addressed one-to-many data linkage. Storkey et al. use the expectation maximization algorithm for two purposes: 1) calculating the probability of a given record pair being a match, and 2) learning the characteristics of the matched records.

A Gaussian mixture model is used to model the conditional magnitude distribution. No evaluation was conducted on this work. Ivie et al. one-to-many data linkage for genealogical research. In their work, they performed data linkage using five attributes: an individual's name, gender, date of birth, location, and the relationships between the individuals. A decision tree was induced using these five attributes. However, the main drawback of this method is that it is tailored to perform matches using specific attributes and, therefore, very hard to generalize.

Christen and Goiser use decision tree to determine which records should be matched to one another. In their work, they compare different decision trees which are built based on different string comparison methods. However, in their method, the attributes according to which the matching is performed are predefined and only one or two attributes are usually used. In this paper, we propose a new data linkage method aimed at performing one-to-many linkage that can match entities of different types. Following . The inner nodes of the tree consist of attributes referring to both of the tables being matched (TA and TB). The leaves of the tree will determine whether a pair of records described by the path in the tree ending with the current leaf is a match or a non match. Data linkage is closely related to the Entity Resolution problem. While in data linkage, the goal is to link between related entries in one or more data sources, the goal of entity resolution is to identify non identical records that represent the same real-world entity, and to merge them into a single representative record. Joint entity resolution attempts to improve entity resolution by using additional information that can be derived by clustering.

Another related research domain is coclustering [13]. Coclustering refers to a two-dimensional clustering process in which the entities (i.e., instances) and the attributes are clustered at the same time. The OCCT model also results in clusters of instances, each may be described with a different set of attributes. The clusters are later modeled in a compact way. In this sense, the proposed OCCT method can also be used for coclustering; however, in this paper, we focus on the linkage task.

Here, Two Way Adaptive Clustering algorithms (TWACA) is a faster, learner and new adaptive clustering algorithm used for result rate optimization. It provides Optimized Query processing. TWACA works in parallel, it done the necessary calculation when the tuple is arriving. It does not affect the received data when any delay occurs while data transmission to destination. Perform Clustering operation for currently received data. TWACA uses less memory space.

III. System Analysis

Data linkage is the task of identifying different entries (i.e., data items) that refer to the same entity across different data sources. Common data linkage scenarios include: linking data when combining two different databases[3]; data deduplication (a data compression technique for eliminating redundant data) which is commonly done as a preprocessing step for data mining tasks[4][5] identifying individuals across different census datasets[6] linking similar DNA sequences[7] and, matching astronomical objects from different catalogues[8].

Existing linkage system [2] uses traditional one-to-many linkage method for combining the result from more than one table. In existing system the clustering query takes too large time to produce the result. It takes more memory space during the execution. The disadvantages are it is not secure, possible for data loss it can able to do one to one data linkage only, it consumes large amount of time, Data Complexity, Change in the database state while clustering the data set and Complex queries used for one-to-one linkage. Data problems while performing Clustering include misspelling, letters or words out of order, fused or split word, missing or extra letters and incomplete words.

In [1] proposed, Two Way Adaptive Clustering algorithms (TWACA) is a faster, learner and new adaptive clustering algorithm used for result rate optimization. It provides Optimized Query processing. TWACA works in parallel, it done the necessary calculation when the tuple is arriving. It does not affect the received data when any delay occurs while data transmission to destination. Perform Clustering operation for currently received data. TWACA uses less memory space. Perform data set linkages operation for more than one data set at a time.

IV. Proposed System

TWAC algorithm for computing the clustering result of two finite relations R_A and R_B, which may be stored at potentially different sites and are streamed to our local system. Given the unpredictable behavior of the network, delays and random temporary suspensions in data transmission may be experienced. The goal of TWACA is twofold. It first seeks to correctly produce the cluster result, by quickly processing arriving tuples, while avoiding operations that may jeopardize the correctness of the output because of memory overflow. Moreover, in the spirit of prior work TWACA seeks to increase the number of cluster tuples (or, equivalently, the rate of produced results) generated during the online phase of the cluster, i.e., during the (potentially long) time it takes for the input relations to be streamed to our system. To achieve these goals, TWACA is highly adaptive to the (often changing) value distributions of the relations, as well as to potential network delays.

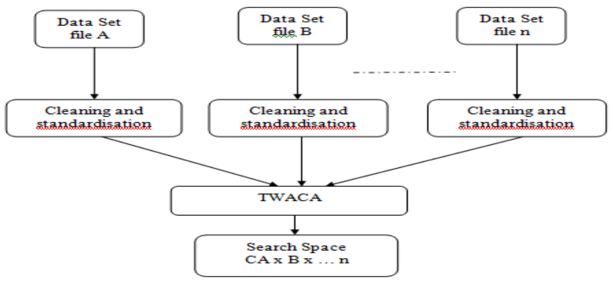
Data Set file A
Data Set file B
Data Set file n
Cleaning and standardisation
Cleaning and standardisation
Cleaning and standardisation
TWACA
Search Space CA x B x … n

Fig.1 System Architecture Design

A. *Algorithm internals and data structures*

Incoming tuples from both relations share the available memory. A separate index, $Index_i$ (i € {A,B}) on the cluster attribute is maintained for the memory resident part of each input relation.

Two index characteristics are important for our framework:

1) Small memory footprint

2) The ability to have sorted access based on the index keys.

Each tuple inserted in memory is augmented with an arriving time stamp (ATS). Each tuple flushed to disk is further augmented with a departure time stamp (DTS). As explained later in this section, these time stamps are used in order to ensure that during the Reactive and Cleanup phases every pair of tuples between the two relations will be clustered exactly once, thus, ensuring the correctness of the produced result.

Each tuple residing in main memory is also augmented with a cluster bit, which is actually part of the index entry for the tuple. This bit is initially set to 0. Whenever an in-memory tuple helps produce a cluster result, its cluster bit is set to 1. As described in Section 3.3, the cluster bits are utilized by a process responsible for evicting tuples that do not produce clusters from some areas of the memory. The latter process seeks to evict those tuples that have their cluster bit set to 0, while clearing the cluster bit of the tuples that it examines. Those tuples that have produced clusters in the past will eventually have their cluster bit cleared and get evicted at a subsequent invocation of the process, if at some point they stop producing cluster results. Thus, the cluster bit serves as a 1-bit approximation to LRU, similarly to the clock algorithm for buffer page replacement. Maintaining a counter (instead of a single cluster bit) for the number of clusters that each tuple has produced would have been more accurate. However, such an approach would also consume more space and an experimental evaluation of both options did not show any significant benefit.

B. *Phases of the algorithm*

The operation of TWACA is divided into three phases, termed in this paper as the Arriving, Reactive, and Cleanup phases. While each phase is discussed in more detail in what follows, we note here that the Arriving phase covers operations of the algorithm while tuples arrive from one or both sources, the Reactive phase is triggered when both relations block and, finally, the Cleanup phase finalizes the cluster operation when all data has arrived to our site.

a. *Arriving Phase*

Tuples arriving from each relation are initially stored in memory and processed as described in Algorithm 1. The Arriving phase of TWACA runs as long as there are incoming tuples from at least one relation. When a new tupleti is available, all matching tuples of the opposite relation that reside in main memory are located and used to generate result tuples as soon as the input data are available. When matching tuples are located, the cluster bits of those tuples are set, along with the cluster bit of the currently processed tuple (Line 9). Then, some statistics need to be updated (Line 11). When the MemThresh is exhausted, the flushing policy picks a victim relation and memory-resident tuples from that relation are moved to disk in order to free memory space (Lines 13-15). The number of flushed tuples is chosen so as to fill a disk block.

The flushing policy may also be invoked when new tuples arrive and need to be stored in the input buffer (Line 2). Since this part of memory is included in the budget (MemThresh) given to the TWACA algorithm, we may have to flush other in-memory tuples to open up some space for the new arrivals. This task is executed asynchronously by a server process that also takes care of the communication with the remote sources. Due to space limitations, it is omitted from presentation. If both relations block for more than Wait Threshmsecs (Lines 18-20) and, thus, no cluster results can be produced , then the algorithm switches over to the Reactive phase. Eventually, when both relations have been received in their entirety (Line 22), the Cleanup phase of the algorithm helps produce the remaining results.

Algorithm 1. Arriving Phase

```
1: while $R_A$ and $R_B$ still have tuples do
2: **if**$t_i$ Є $R_i$ arrived (i Є{A,B}) **then**
3: Move $t_i$ from input buffer to TWACA process space.
4: Augment ti with cluster bit and arrival timestamp ATS
5: j= {A, B} - i {Refers to "opposite" relation}
6: *match*Set = set of matching tuples (found using $Index_j$) from opposite relation $R_j$
7: clusterNum=|jmatchSet| (number of produced clusters)
8: **if**clusterNum> 0 **then**
9: Set the cluster bits of ti and of all tuples in matchSet
10: **end if**
11: UpdateStatistics($t_i$,*numClusters*)
12:*indexOverhead*= Space required for indexing $t_i$ using *Index*$_i$
13:**while**UsedMemory + *indexOverhead*>=MemThresh **do**
14: Apply flushing policy (see Algorithm 2)
15: **end while**
16: Index $t_i$ using *Index*$_i$
17: Update UsedMemory
18: else if transmission of $R_A$ and $R_B$ is blocked more thanWaitThresh**then**
19: Run Reactive Phase
20: **end if**
21: **end while**
22: Run Cleanup Phase
```

b. *Flushing Policy and Statistics Maintenance*

An overview of the algorithm implementing the flushing policy of TWACA is given in Algorithm 2. In what follows, we describe the main points of the flushing process.

TABLE 1. SYMBOLS USED IN TWACA

Symbol	Description (i {A,B})
Ri	Input relation Ri
Ti	Tuple belonging to relation Ri
Indexi	Index for relation Ri
Diski	Disk partition containing flushed tuples of relation Ri
UsedMemory	Current amount of memory occupied by tuples, indexes and statistics
MemThresh	Maximum amount of available memory to the algorithm
WaitThresh	Maximum time to wait for new data before switching to Reactive phase
LastLwVali LastUpVali	Thresholds for values of cluster attribute in lower and upper regions of Ri
LwClustersi MdClustersi UpClustersi	Number of produced clusters by tuples in the lower, middle and upper region, correspondingly, of Ri
LwTupsi MdTupsi UpTupsi	Number of in-memory tuples in the lower, middle and upper region, correspondingly, of Ri
BfLwi BfMdi BfUpi	Benefit of in-memory tuples in the lower, middle and upper region, correspondingly, of Ri

Algorithm 2. Flushing Policy

```
1: Pick as victim the relation $R_i$(i Є{A,B}) with the most in-memory tuples
2: {Compute benefit of each region}
3: $B\ f\ Up_i = UpClusters_i / UpTups_i$
4: $B\ f\ Lw_i = LwClusters_i / LwTups_i$
5: $B\ f\ Md_i = MdClusters_i / MdTups_i$
6: {TupsPer Block denotes the number of tuples required to fill a disk block}
7: {Each flushed tuple is augmented with the departure time stamp DTS }
8: if $B\ f\ Up_i$ is the minimum benefit then
9:     locate Tups_Per_Block tuples with the larger cluster attribute using $Index_i$
10:    flush the block on $Disk_i$
11:    update $LastUpVal_i$ so that the upper region is (about) a disk block
12: else if $B\ f\ Lw_i$ is the minimum benefit then
13:    locate Tups_Per_Block tuples with the smaller cluster attribute using $Index_i$
14:    flush the block on $Disk_i$
15:    update $LastLwVal_i$ so that the lower region is (about) a disk block
16: else
17:    Using the Clock algorithm, visit the tuples from the middle area, using $Index_i$, until Tups_Per_Blocktuples are evicted.
18: end if
19: Update $UpTups_i$, LwTupsi, $MdTups_i$, when necessary
20: $UpClusters_i$, $LwClusters_i$, $MdClusters_i$<- 0
```

c. Reactive Phase

The Reactive phase cluster algorithm [10] termed ReactiveNL, is a nested-loop-based algorithm that runs whenever both relations are blocked. It performs clusters between previously flushed data from both relations that are kept in the disk partitions DiskA and DiskB, respectively. This allowsTWACA to make progress while no input is being delivered. The algorithm switches back to the Arriving phase as soon as enough, but not too many, input tuples have arrived, as is determined by the value of input parameter MaxNewArr. The goal of ReactiveNL is to perform as many clusters between flushed-to-disk blocks of the two relations as possible, while simplifying the bookkeeping that is necessary when exiting and reentering the Reactive phase.

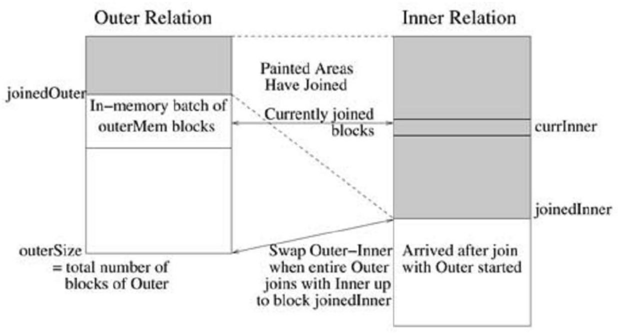
Outer Relation
Inner Relation
joinedOuter
In-memory batch of outerMem blocks
Painted Areas Have Joined
Currently joined blocks
currInner
joinedInner
outerSize = total number of blocks of Outer
Swap Outer-Inner when entire Outer joins with Inner up to block joinedInner
Arrived after join with Outer started

Fig. 3 Status of the algorithm during the reactive phase

V. Conclusion And Future Work

TWACA is a novel adaptive Clustering algorithm that supports both equality and range clusters predicates. TWACA builds on an intuitive flushing policy that aims at maximizing the productivity of tuples that are kept in memory. TWACA is the first algorithm to address the need to quickly respond to bursts of arriving data during the Reactive phase. This paper propose a novel extension to nested loops cluster for processing disk-resident tuples when both sources block, while being able to swiftly respond to new data arrivals.

It provide an extensive experimental study of TWACA including performance comparisons to existing adaptive cluster algorithms and a sensitivity analysis. Our results results demonstrate the superiority of TWACA in a variety of realistic scenarios. During the online phase of the algorithm, TWACA manages to produce up to three times more results compared to previous techniques (Traditional join). The performance gains of TWACA are realized when using both real and synthetic data and are increased when fewer resources (memory) are given to the algorithm. For future work, introduce MWACA (Multi-way Adaptive Clustering Algorithm),a novel adaptive multi-way clustering algorithm that maximizes the output rate, this will dealing the cases where data are held by multiple remote sources.. MWACA surely perform sixty times higher then existing clustering algorithm when a four-way star join is executed in a constrained memory environment

Reference

[1]. S. Babu and P. Bizarro, “Adaptive Query Processing in the Looking Glass,” Proc. Conf. Innovative Data Systems Research (CIDR), 2005.

[2]. P. Fellegi and A.B. Sunter, “A Theory for Record Linkage,”J. Am.StatisticalSoc.,vol. 64, no. 328, pp. 1183-1210, Dec.1969.

[3]. M. Yakout, A.K. Elmagarmid, H. Elmeleegy, M. Quzzani, and A. Qi, "Behavior Based Record Linkage,"Proc. VLDB Endowment, vol. 3, nos. 1/2, pp. 439-448, 2010.

[4]. J. Domingo-Ferrer and V. Torra, "Disclosure Risk Assessment in Statistical Microdata Protection via Advanced Record Linkage," Statistics and Computing,vol. 13, no. 4, pp. 343-354, 2003.

[5]. F. De Comite´, F. Denis, R. Gilleron, and F. Letouzey, "Positive andUnlabeled Examples Help Learning,"Proc. 10th Int'l Conf.Algorithmic Learning Theory,pp. 219-230, 1999.

[6]. M.D. Larsen and D.B. Rubin, "Iterative Automated RecordLinkage Using Mixture Models,"J. Am. Statistical Assoc., vol. 96,no. 453, pp. 32-41, Mar. 2001.

[7]. S. Ivie, G. Henry, H. Gatrell, and C. Giraud-Carrier, "A Metric-Based Machine Learning Approach to Genealogical RecordLinkage," Proc. Seventh Ann. Workshop Technology for FamilyHistory and Genealogical Research,2007.

[8]. A.J. Storkey, C.K.I. Williams, E. Taylor, and R.G. Mann, "AnExpectation Maximisation Algorithm for One-to-Many RecordLinkage," Univ. of Edinburgh Informatics Research Report, 2005.

[9]. P. Christen and K. Goiser, "Quality and Complexity Measures forData Linkage and Deduplication,"Quality Measures in DataMining,vol. 43, pp. 127-151, 2007.

[10]. M.A. Bornea, V. Vassalos, Y. Kotidis, and A. Deligiannakis, "Double Index Nested-Loop Reactive Join for Result Rate Optimization," Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2009.

BIOGRAPHY WITH DATA

P.Vijayameena received the B.Tech degree in Information Technology from Kalasalingam University, Chennai, in 2013. Currently, she is pursuing M.Tech degree in Information Technology from Anna University, Chennai. Her research areas of Interests include Data Mining and Networking.

R.Chithra Devi received the B.Tech degree in Information Technology from Anna University, Chennai, in 2005 and she received M.Tech in IT from Manonmaniam University, Tirunelveli, in 2007.She is an Assisstant Professor in the Department of IT at Dr.Sivanthi Aditanar College of Engineering. Her research interests are Networking and Wireless network.