Computers in Human Behavior 30 (2014) 427–435

Contents lists available at SciVerse ScienceDirect

Computers in Human Behavior

journal homepage: www.elsevier.com/locate/comphumbeh

A synchronized design technique for efficient data distribution

Hassan I. Abdalla *

Computer Science Department, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia

ARTICLE INFO

Article history:
Available online 26 May 2013

Keywords:
Synchronization
Distribution
Fragmentation
Replication
Allocation

ABSTRACT

One of the important features of database fragmentation and allocation techniques is the fact that they depend not only on the entries of a database relation, but also on their empirical frequencies of use. Distributed processing is an effective way to improve performance of database systems. However, for a Distributed Database System (DDBS) to function efficiently, fragments of the database need to be allocated carefully at various sites across the relevant communications network. Therefore, fragmentation and proper allocation of fragments across network sites is considered as a key research area in distributed database environment. However, fragments allocation to the most appropriate sites is not an easy task to perform. This paper proposes a synchronized horizontal fragmentation, replication and allocation model that adopts a new approach to horizontally fragment a database relation based on attribute retrieval and update frequency to find an optimal solution for the allocation problem. A heuristic technique to satisfy horizontal fragmentation and allocation using a cost model to minimize the total cost of distribution is developed. Experimental results are consistent with the hypothesis and confirm that the proposed model can efficiently solve dynamic fragmentation and allocation problem in a distributed relational database environment.

1. Introduction

Usually fragmentation is performed without having allocation in mind. While allocation, on the other hand, is always dependant on fragmentation and is performed on the assumption that fragmentation is obtained a priori.

Vertical fragmentation (*VF*) allows relation *R* to be partitioned into disjoint sets of columns or attributes except the primary key. Horizontal fragmentation (*HF*), on the other hand, allows relation *R* to be partitioned into disjoint tuples or instances. Previous works in the area *VF* and *HF* have one problem in common: that most of them concentrated only on fragmentation problem and overlooked allocation problem to reduce complexity.

The complexity of horizontal fragmentation is $O((K^2)^m)$ where m is the number of simple predicates and K is the network sites whereas the complexity of vertical fragmentation is $O((K^m)^m)$ where k is the network sites and m is the number of non-primary key attributes. In the context of relational data model, the existing techniques for *HF* fall into following two classes: Min-term-predicates techniques (Özsu & Valduriez, 1999) and affinity based techniques (Bellatreche, Karlapalem, & Simonet, 1996).

When fragmentation process is complete, then fragment allocation follows. There are four data allocation strategies applicable in a distributed relational database: these are centralized, fragmentation (partition), full replication, and partial replication (selective) Iacob (2010). Our technique adopts the fragmentation strategy and sometimes partial replication. However, allocation is the process of assigning the database fragments to DDBSs network sites. When data is allocated, it might either be replicated or maintained as a single copy. So, fragment allocation can be either non-redundant or redundant. Under a non-redundant allocation scheme, exactly one copy of each fragment will exist across all the sites, while under redundant allocation schema, more than one copy of each fragment will exist across all the sites (Hauglid & Ryeng, 2010).

In Ceri, Negri, and Pelagatti (1984), a method was proposed to use min-term predicate to fragment relations. It did that to perform primary horizontal fragmentation, a set of disjoint and completed predicates $P = \{P_1, \ldots, P_n\}$ should be determined. In Özsu and Valduriez (1999), the iterative algorithm called Com-Min was presented to generate a complete and minimal set of predicate *Pr* from a given set of simple predicates. In Zhang (1993), affinity-based vertical fragmentation method was adapted to horizontal fragmentation. Ra (1993) proposed a graph-based algorithm for horizontal fragmentation such that the predicates clustering based on the predicates affinities. However, Khalil, Eid, and Khair (xxxx) presented a horizontal transaction-based partition algorithm which used predicates usage matrix as input, and Mahboubi and Darmont (2009) used predicate affinity for HF in data warehouse.

Several researches, have already been submitted the data allocation technique, are restricted in their theoretical and

* Tel.: +966 506103423.
E-mail address: habdalla@ksu.edu.sa

0747-5632/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.chb.2013.04.025

implementation parts and other techniques are ignoring the optimization of the transaction response time. DDBSs design aims to enhance the applications performance by minimizing the amount of irrelevant data accessed by the applications (Ezeife & Barker, 1998), and by minimizing the amount of data transferred in processing the applications (Karlapalem, Navathe, & Morsi, 1994). In Mahmood, Khan, and Fatmi (1994), many management methodologies have been presented to manage database and either of these methods could be merged with a mathematical modeling for data allocation in DDBSs that was proposed using greedy approach in (Abdalla, 2008) to effectively treat allocation problem. It considered a network communication, a local processing and data storage cost to allocate fragment site by site. This method supports the best location of data fragmentation in DDBSs based on fragment access pattern and cost of moving fragments from one site to other with site constraints maintaining as site capacity (C) and the fragment limit (FL). In Daudpota (1998), formal model of data allocation had been constructed and had derived an algorithm to fragment and allocate the relations.

Some authors have considered several aspects of the allocation problem into account in different contexts. Mei, Mancini, and Jajodia (2003) Combined security considerations into the fragment allocation process. Lin and Veeravalli (2003) proposed an algorithm for allocation and replication that adapts to the changing patterns of online requests. Grouping partitioning into an automatic design framework is considered in Agrawal, Narasayya, and Yang (2004), and Chin (2002), considered incremental allocation and reallocation based on changes in workload. In Lin, Orlowska, and Zhang (1993), data allocation algorithms presented to achieve the minimum overall communication cost. SAGA approach is proposed in Abdalla (2011) where the optimal places are determined for data fragments individually. Data access pattern and transmission cost were used in this approach to help for specifying that places.

In Surmsuk (2007), *CRUD* matrix is proposed. Its rows indicate relations attributes and columns indicate several applications locations. DDBSs analysts and designers used *CRUD* matrix in the requirement analysis phase of system development life cycle for making decision of data mapping to different locations (Surmsuk, 2007; Whitten, Bentley, & Dittman, 2004). In this paper, *CRUD* has been modified according to our requirement of HF and named it Attribute Retrieval and Update Matrix (*ARUM*). We address horizontal fragmentation and allocation simultaneously in the context of the relational model. So, a heuristic technique to satisfy a synchronized horizontal fragmentation and allocation using cost model is a key contribution of this paper. However, based on the rule of thumb, if fragment is queried more frequently than it is modified, then replication is advisable as it is already done in this work. But, this replication benefit comes with the additional cost of keeping all copies identical. This cost, which could potentially be high, consists of total storage cost, cost of local processing, and communication cost (Rahimi & Haug, 2010).

In this paper, a novel approach is proposed to perform fragmentation horizontally based on sites update and retrieval query frequencies. A properly gathered information about queries will enable the database designer to establish fragmentations that allows an efficient and synchronized (redundant and non-redundant) allocation.

1.1. The paper organization

The rest of this paper is organized as follows: In Section 2, we will introduce our proposed model that includes the fragmentation model, the allocation model and cost functions. In Section 3 the fragmentation/allocation algorithm is presented. The implementation results that justify the developed technique are given in Section 4. Section 5 discusses the model performance evaluation that supports the assumptions. Section 6 briefly compares our model with similar existing ones. Finally, the conclusions are presented in Section 7.

2. The proposed model

2.1. Problem description and notations

In this work a new optimal fragmentation and allocation method have been proposed to perform horizontal fragmentation (*HF*) and allocation in a synchronized fashion using a cost model. The *HF* divides the intended relation according to the Attribute Retrieval and Update Frequency (*ARUF*) matrix (Table 7).

ARUF is a table constructed by calculating a given retrieval and update frequencies for attribute predicates individually. The *ARUF* calculation is performed using *ARUM* (Table 4), distance cost matrix (Table 5), and a cost function. An *ARUM* matrix is a table constructed by placing predicates of a relation attributes as columns and site queries as rows. Our system development phases are shown in Fig. 1.

The proposed model will be performed in two phases: in the first phase, site constraints will be relaxed, while they will be enforced in the second phase.

2.2. The heuristic approach

This work introduces a simple cost model that collects retrieval and update information at the sites where queries execute. Then it evaluates the cost of fragmenting the intended relation/fragment and the cost of allocating the resulted fragments across different network sites. Finally, the heuristic model chooses the site to store the candidate fragment by selecting the site that sustains the highest query cost for the fragment. The fragmentation and allocation process is performed in a synchronized fashion. In summary, the model adopts the maximum update and retrieval cost heuristic to find the optimal fragmentation and allocation scenarios.

2.3. The proposed fragmentation model

The proposed fragmentation model is based on a set of simple predicates Pr $[P_1,\ldots,P_n]$ where n is the number of relation attributes. These predicates have been assigned to the relation attributes $A[A_1,\ldots,A_n]$. Attributes with a numeric value will comprise a predicate Pr that have one of three states: ($Pr_i > V$), ($Pr_i < V$) or ($Pr_i = V$). However, attributes with alphabetical value will comprise a predicate Pr that have only one state: ($Pr_i = V$). For each predicate Pr_i there are $Pr_i{\cdot}RF_i$ value and $Pr_i{\cdot}UF_i$ value representing the predicate retrieval and update frequency values respectively for the corresponding attribute issued by the site queries $Q[Q_1,\ldots,Q_k]$ executed under a site query strategy where k is the number of site queries that are assumed to be the most frequently used queries in DDBS accounting for say more than 75% of the processing in DDBS. Each query Q_k can be executed from any site with a certain frequency (*FREQ*) to access the intended fragment. The execution frequencies of k queries at m sites can be represented by $m \times k$ matrix, QF_{km}. The attribute predicate "*Pr*" is represented as follows: $A_i{\cdot}value\ \theta\ V$, $RF_i{\cdot}value1$, $UF_i{\cdot}value1$.

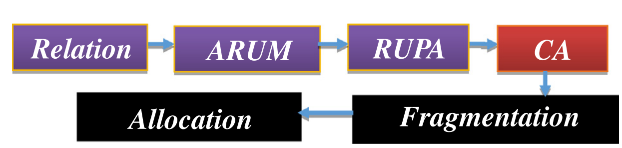

Fig. 1. The proposed system phases.