E-LEARNING AND DISTRIBUTED DATABASES

Adrian Runceanu, *University Constantin Brancusi of Targu-Jiu, ROMANIA*
Ilie Borcoși, *University Constantin Brancusi of Targu-Jiu, ROMANIA*
Marian Popescu, *University Constantin Brancusi of Targu-Jiu, ROMANIA*

ABSTRACT. In this paper we present a parallel between the educational content information located in a site of e-learning and information fragmentation in a distributed database.

Keywords: distributed databases, e-learning, fragmentation, educational technology, algorithm

INTRODUCTION

In this paper I try to show a parallel between the concepts of distributed databases and e-learning. In this regard, I will present in the first part some aspects of distributed databases: definition, the architecture of distributed databases management system, distributed database fragmentation and, in the second part, I will present what is the acceptance general concept of e-learning. As a personal opinion, I think it can make a connection between distributed databases and an educational system for e-learning. Finally, I try to present the implementation of an educational website based on a CMS (Content Management System) open-source that contains the necessary modules for didactic activities where I used some editing features, implementation and execution of programming applications.

1. ABOUT DISTRIBUTED DATABASES

"A distributed database is a database in which portions of the database are stored on multiple computers within a network. Users have access to the portion of the database at their location so that they can access the data relevant to their tasks without interfering with the work of others. A centralized distributed database management system (DDBMS) manages the database as if it were all stored on the same computer. The DDBMS synchronizes all the data periodically and, in cases where multiple users must access the same data, ensures that updates and deletes performed on the data at one location will be automatically reflected in the data stored elsewhere." [10]

The distributed databases design is an optimization process that requires obtaining solutions to several interpenetrating problems, namely data fragmentation, data allocation and local optimization. Each problem can be solved through different approaches, and therefore the design of distributed databases becomes a very difficult task. Usually the design process is heuristically defined.

The distributed databases design derived from the non-distributed databases design only in terms of distribution. The design involves data acquisition, database partitioning, the allocation and replication of partitions and local optimization. The database partitioning can be performed in

several different ways: vertical, horizontal and hybrid (also called mixed) partitioning.
This aspect of a database design will be highlighted in this thesis and namely the developing of an algorithm for checking some various algorithms proposed in literature for distributed databases partitioning (fragmentation).
Basically, it will be considered the problem of vertical data partitioning (fragmentation), also known as the attributes partitioning. This technique is used for the databases design to improve the performance of transactions. In vertical partitioning, the attributes of a relation R are together in groups that do not overlap and the relationship R is designed on relations fragments in accordance with these attributes groups.
In the distributed databases, these fragments are allocated on different sites. Here comes the aim of vertical partitioning to create vertical fragments of a relationship to minimize the cost of the data accessing during the transaction process. If the fragments are closer as possible to the needs of the transactions set, then the transactions processing cost can be reduced. For the distributed databases design, the transaction cost is reduced by increasing the local transactions (on a site) and at the same time by reducing the amount of data views which are not local. The aim of vertical partitioning technique (and in general, partitioning techniques) is to find a partitioning scheme to meet the objective outlined above. Note that the partitioning problem can be tackled on different levels of detail considering some additional information.

1.1 FRAGMENTATION IN DISTRIBUTED DATABASES

The relational database management systems do not support complex, strongly structured and multimedia data very well. Object oriented database management systems (OODBMS) meet the requirements of these new applications [7]. Distribution of data across a distributed system is a major design problem that impacts on system performance directly. A good distribution design enhances application performance by reducing communication cost overhead due to references to nonlocal data [2].
The distributed database design problem consists of fragmentation of database entities followed by allocation of these fragments to distributed sites. There are two ways to design a distributed database: top-down and bottom-up approaches [5]. With the top-down approach, input to the design process is the global conceptual schema (GCS), which is obtained from the conceptual design step. The GCS describes the global database entities and their relationships, e.g., the database classes, the inheritance, class composition and method nesting hierarchies between them. The conceptual design is the process by which the enterprise is studied to determine entity types like class types and relationships among these entities. The second input consists of the access pattern information, which shows the queries accessing database classes and how frequently they access them. The output from the distributed design process is a set of local conceptual schemas (LCSs) [6]. LCSs describe database entities at each local site. With the bottom-up approach, since a number of databases already exist, the design task

involves integrating them into one global database.

There are different ways to perform fragmentation and allocation. In [1] consider data fragmentation and allocation as a single problem. Another approach is fragmentation followed by an allocation phase. The allocation scheme distributes fragments in such a way that overall system throughput is maximized while minimizing the overall cost of maintaining the system. The work in this paper treats fragmentation and al location of database objects as separate design steps.

An object base is a collection of database objects classified into a number of classes.

Objects with common attributes and methods belong to the same class. An object based system supports an object oriented data structure including features of encapsulation and inheritance [6]. Encapsulation bundles together the methods and the attributes of an object. Inheritance allows reuse and incremental redefinition of new classes in terms of existing ones. A distributed object based system is a collection of local object bases located at different local sites, interconnected by a communication network [3].

With a relational database system, the locality of accesses of database applications is usually defined not on the entire relations but on their subsets because the application views are subsets of relations. Relation instances are essentially tables, and there are three methods for dividing (fragmenting) a table into smaller ones, namely, horizontal, vertical and hybrid fragmentation [6]. Horizontal fragmentation partitions a relation along its tuples such that each horizontal fragment has a subset of the tuples in the relation. Vertical fragmentation partitions a relation along its attributes such that each vertical fragment has a subset of attributes in the relation. Hybrid Fragmentation partitions a relation along its tuples and then along its attributes or partitions a relation along its attributes and then along its tuples such that each hybrid fragment be-longs to only one horizontal and one vertical fragment. In the case of object databases, horizontal fragmentation keeps sub-sets of instance objects of classes in horizontal fragments, while vertical fragments contain subsets of class attributes and methods, hybrid fragments contain grid of only one horizontal fragment and only one vertical fragment.

Fragmentation of data improves overall performance by reducing the amount of data that needs to be moved between sites or replicated in order to answer user queries. Thus, distributed databases benefit greatly from fragmentation because of the following reasons:

(1) Fragmentation reduces the amount of irrelevant data ac-cessed by applications.

(2) It allows greater concurrency and parallel execution of a single query.

(3) It reduces the amount of data transferred when migration is required.

(4) It allows replicating fragments rather than replicating the entire class.

Existing algorithms dealing with fragmentation obtain their inputs only from static requirements analysis. Major changes in a domain would entai1 a full re-analysis of the system and re-running of the distributed design algorithms. For reduced cost of system input, re-analysis and increased user confidence in distributed design techniques, a dynamic fragmentation system is required. This paper work proposes an approach for measuring the system performance of a distributed object based system that is horizontally fragmented. Changes in the global conceptual schema and access pattern information are periodically monitored and used to assess the performance of the system based on the current horizontal

fragments. The system performance is measured in terms of:
(1) The amount of irrelevant local access, which yields a measure of the local processing cost of transactions due to irrelevant instance objects of fragments, and
(2) The amount of relevant remote access due to relevant instance objects of fragments that are accessed remotely by transactions.

2. COST MODEL FOR ALLOCATION

In [7] is presented an allocation model that minimizes the total cost of storage and processing, trying to comply with certain restrictions imposed on response time.
It consider a decision x_{kj} variable, defined as:

$$x_{kj} = \begin{cases} 1, & \text{if fragment } F_k \text{ is placed on site } S_j \\ 0, & \text{in the other cases} \end{cases} \quad (1)$$

Total cost function has two components: processing applications and storage. This can be expressed as:

$$TOC = \sum_{\forall q_i \in Q} QPC_i + \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} STC_{jk}, \quad (2)$$

where QPC_i is the cost of processing the application q_i and STC_{jk} fragment F_j is the cost of storage per station S_k.
We can define the cost of storage with the formula:

$$STC_{jk} = USC_k * size(F_j) * x_{jk} \quad (3)$$

The cost of processing applications (QPC) can be separated into two components: the actual processing cost (PC) and the cost of transmission (TQ).

$$QPC_i = PC_i + TC_i \quad (4)$$

Processing component consists of three cost factors: access cost (CA), the integrity constraint (IC) and control competitor (CC):

$$PC_i = AC_i + IC_i + CC_i. \quad (5)$$

Detailed specification of these three cost factors depend on the algorithms used for these tasks. For example, AC has the form:

$$\sum_{\forall F_j \in F} STC_{jk} \le \text{storage capacity at site } S_k, \quad \forall S \in S. \quad (6)$$

The first two terms from the previous formula calculates the number of accesses to the query qi the fragment F_j and UR + RR value provides the total number of accesses that executes work | retrieve and update them. Local cost of processing them is assumed to be identical, given LPC_k multiplying the total cost of access to station S_k and multiplying x_{jk} show that will select only those values of the cost for stations where parts are stored. Access cost function requires that each application can be divided into subqueries that run on each fragment stored on a station, the results are returned to the station from which the request was sent. This is a simplistic view that ignores the complexity of database processing. For example, it was revealed the cost of execution join or cost competitiveness and integrity. The cost of transmission can be made analogous to the cost of access. The queries that are updating is required to inform all stations where the replicas, while the queries that are restored, it is sufficient to access only one of the descendants, in addition, after the application update, no data transmission back to state of origin (except a confirmation message), while the queries that are restored may result in transmission of data.

Component update transmission function has the form:

$$TCU_i = \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} u_{ij} * x_{jk} * g_{o(i),k} + \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} u_{ij} * x_{jk} * g_{k,o(i)}, \quad (7)$$

Where the first term is to send a message to update the initial station 0 (0 demand all copies of fragments that must be updated and the second term is to confirm transmission). Component retrieve can be specified as follows:

$$TCR_i = \sum_{\forall F_j \in F} \min_{S_k \in S} (r_{ij} * x_{jk} * g_{o(i),k} + r_{ij} * x_{jk} * \frac{sel_i(F_j)}{fsize} * g_{k,o(i)}). \quad (8)$$

The first term in TCR is the cost of forwarding the application to recover all stations contains copies of the fragments to be accessed, and the second term is the cost of transmission results from these stations to the original station.

The cost of transmission for q_t query can be described as:

$$TC_i = TCU_i + TCR_i \quad (9)$$

Constraint functions can be analysed similarly. For example, response time constraint can be described thus: execution time of the call q_i maximum response time of applications qi, $\forall q_i \in Q$, and storage constraints may be described by the relation:

$$AC_i = \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} (u_{ij} * UR_{ij} + r_{ij} * RR_{ij}) * x_{jk} * LPC_k, \quad (10)$$

The heuristic methods must be sought that produce sub-optimal solutions and must test the closeness of the results of heuristic algorithm for optimal allocation. Unfortunately there is insufficient information to determine the "near" optimal solution.

3. E-LEARNING AND A CASE-STUDY IMPLEMENTATION

3.1 ABOUT E-LEARNING

E-learning refers to the use of electronic media and information and communication technologies (ICT) in education. E-learning is broadly inclusive of all forms of educational technology in learning and teaching. E-learning is inclusive of, and is broadly synonymous with multimedia learning, technology-enhanced learning (TEL), computer-based instruction (CBI), computer-based training (CBT), computer-assisted instruction or computer-aided instruction (CAI), internet-based training (IBT), web-based training (WBT), online education, virtual education, virtual learning environments (VLE) (which are also called learning platforms), m-learning, and digital educational collaboration. These alternative names emphasize a particular aspect, component or delivery method. [11]

The implementation of the e-learning system can be achieved using one of three approaches:

1. Using the technologies to support or supplement the traditional face-to-face course
2. Integrating online activities into a traditional course to enhance the learning experience
3. Delivering a course that is entirely online

The link between an e-learning website and a distributed database consists of using the

specified fragmentation in distributed databases that have educational content and teaching management in the context of e-learning website. Thus, allocating dynamic information only where "there is the greatest demand" ensure the increase of the speed access to information and a better management of the local users.
From here, the information is accessed by each user faster because they are closer.
There is a big disadvantage that refers to updating data in a distributed system such as any change in the database involved in all copies or modifications of existing lines and hence the discount rate is low. In addition, when is necessary information about all users of such an educational website, gather data from multiple places and so it increases the time to obtain summary information.

3.2 CASE-STUDY IMPLEMENTATION

I could give an example that will help to understand: consider that you are designing a distributed database for your university. The university has one central site and five peripheral sites managing all the didactics. Each peripheral site is independent from the others: just use the information in the database about its students, its exams, its teachers, and so one, it is easy that a few of the relationships of the database will be fragmented in one fragment for each peripheral site (e.g. the student relationship) and, because of the "independence" of the sites, the only solutions for allocation with sense will be to allocate the fragment on its site or on the central one: none of the other sites will use it; is easy that a lot of relationship will be fragmented in this way, think just to the students, the exams and the teachers relationships, each one fragmented in 5 fragments: not considering others fragmentation 15 fragments are created. Using the "one for each site" (a more correct name would be "one fragment for each peripheral site") is possible to reduce the number of the fragments that the user has to manage: the input for the allocation won't be 3 text files with 5 fragments each one, but 3 text files with 1 fragment each one. But there's also the possibility that a relationship is not fragmented in one fragment for each peripheral site (because is not possible to relate a tuple to a single peripheral site or simply because there's no need). In this case there aren't no-sense solutions for the allocation of the relationship (or the fragments obtained from other fragmentations), it can be allocating in all the sites central or peripherals; in this case must be used the "one for all sites" policy (one fragment for all sites).

CONCLUSION

A first conclusion is that using the principles of distributed databases in implementing websites with educational content improves access to information provided to users.
A second conclusion relates to reducing the amount of data that should be accessed remotely through local access specific data fragmentation and memorizing them where it is used more. The proposed implementation is a particular case of our university's website, but can be used as a case study for other implementations.

REFERENCES

[1] Apers, P.M.G. Data Allocation in Distributed Database Systems. ACM Transaction on Database Systems 1988.
[2] Bhar, S., Barker, K, Static allocation in distributed object-base systems: a graphical approach. Information Systems and Data Management. 6th Conference, CISMOD '95, Proceedings.
[3] Ezeife, C.I. and Barker, K. Vertical Class Fragmentation in a Distributed Object Based System. Technical Report -

Department of CS, University of Manitoba, TR 9402.
[4] Ezeife, C.I. and Barker, K. A Comprehensive Approach to Horizontal Class Fragmentation in a Distributed Object Based System. International Journal of Distributed and Parallel Database, Kluwer Academic Publishers, V1, 1995.
[5] Ezeife, C.I. and Barker, K. Class Fragmentation in a Distributed Object Based System. A Thesis Presented to the University of Manitoba in partial fulfilment of requirements for the degree of PhD in CS
[6] Ozsu, M.T., Valduriez, P. Principle of Distributed Database Systems. Prentice Hall, Second Edition, 1999.
[7] Ravat, F. and Zurfluh, G. Issues in the Fragmentation of object oriented database. Proceedings, Basque International Workshop on information Technology, 1995.
[8] Runceanu A., Fragmentation in distributed databases, (2008), International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE 2007) Conference, Conference Proceedings book, December 3-12, 2007, University of Brigeport, USA, publish in Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering, Springer Science, ISBN 978-1-4020-8734-9 (Print) 978-1-4020-8735-6 (Online), DOI 10.1007/978-1-4020-8735-6_12.
[9] Runceanu A., Popescu M., An algorithm for replication in distributed databases, International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (C I S S E 2 0 1 1) Conference, CONFERENCE PROCEEDINGS, December 3-12, 2011, University of Brigeport, USA.
[10] http://searchoracle.techtarget.com/definition/distributed-database Quoted April 14 2013
[11] http://en.wikipedia.org/wiki/E-learning Quoted April 14 2013