SIADBDD: An integrated tool to design distributed databases

SIADBDD: Una herramienta integrada de ayuda al diseño de bases de datos distribuidas

Abel Rodríguez Morffi[1], Carlos Ernesto García González[1], Wilfried Lemahieu[2], Luisa Manuela González González[1]

[1] Universidad Central "Marta Abreu" de Las Villas Carretera a Camajuaní km. 5.5. C.P. 54830. Santa Clara, Cuba

[2] Katholieke Universiteit Leuven, Research Center for Management Informatics (LIRIS) Faculteit ETEW, Room: 03.105, Naamsestraat 69 - bus 3500, 3000 Leuven, Belgium

Abstract

This paper addresses an architectural and functional overview of an implemented tool that aids designers to design DDBs in a relational context. Conceptual design and fragmentation issues are considered as well as the allocation problem. The tool applies metaheuristics for solving many design problems to obtain outputs in reasonable time. They use cost models and are targeted at globally minimizing these costs.

----- *Keywords:* Distributed Database Design, CASE tools, global conceptual schema, fragmentation, allocation.

Resumen

Este trabajo presenta un resumen sobre la arquitectura y las funciones de SIADBDD, una herramienta integrada de ayudas al diseño de BDD en un contexto de bases de datos relacionales. Estas ayudas consideran desde la modelación conceptual de esquemas globales hasta la localización de los fragmentos de datos a los sitios de procesamiento donde residirá la BDD objeto de diseño.

-----*Palabras clave:* Diseño de bases de datos distribuidas, herramientas CASE, modelación conceptual, fragmentación, ubicación.

* Autor de correspondencia: teléfono: + 53 + 42 + 281 5 15, fax: + 53 + 42 + 202 1 13, correo electrónico: luisagon@uclv.edu.cu (L. González).

Introduction

With the increasing demand of database applications that are accessed by users from different geographical locations, database distribution design becomes an essential part of the database design, which targets at increasing the overall system performance. From the early 1980s, the problem of database distribution design has attracted interest from many researchers. It has first been discussed in the context of the relational data model, then in the object oriented data model. With the current popularity of web information systems, there is an increasing need for Distributed Database Systems (DDBS) to provide back-end support for Web-based database applications. The aim of database distribution design is to make applications that access the database more efficient and effective. Therefore, the global queries have to be analyzed in order to design an adequate distribution of the data. The design of DDBs enhances application performance by reducing the amount of irrelevant data accessed by the applications [1], and the amount of data transferred unnecessarily between distributed sites during application processing [2]. There are two ways by which the performance of applications can be enhanced: grouping sites and fragment allocation. Grouping sites of distributed databases holds relevant data accessed by an application into a group of sites. It determines whether or not a set of sites is assigned to a certain cluster, and it considered as a fast way to determine the data allocation to a set of sites rather than site by site. Grouping sites into clusters minimizes the communication costs between the sites and improves the system performance. On the other hand, fragment allocation is the process of allocating the fragments to the sites of distributed databases to minimize the data transfer cost and the number of messages during application processing. This work aims at dividing entities into fragments, which are later distributed to the machines in a computer network in such a way that the total cost is minimized as much as possible [3]. This approach emphasizes methods that minimize the transactions' communication cost, increase data availability and integrity by allocating database fragments replicated over the sites where possible or necessary, and minimize the transactions' total response time. In the study of DDBs, several key disciplines are converging: databases, algorithms, operating systems, networks, software engineering, etc. Furthermore, the efficient implementation of a design is an optimization problem that requires solutions to several interrelated problems such as data fragmentation and allocation. Each problem phase can be solved with several different approaches thereby making the DDB design a very difficult task. Traditionally, database design has been heuristic in nature.

In this article we concentrate on designing DDBs in the context of the relational data model. Having in mind the characteristics and complexity of DDB design, we have been motivated to develop a tool that addresses the problem of designing DDBs. The paper is organized as follows. In section 2 we complete the introduction by briefly reviewing previous work on DDB design. The paper's main contribution is in sections 3 and 4. Section 3 provides the CASE tool's architectural overview and section 4 contains a functional overview. Section 5 draws conclusions about the important features of this paper. Finally, section 6 suggests future research directions.

Related work

Several approaches have been proposed for database partitioning and fragment allocation in DDBs. Navathe et al [4] have proposed a mixed fragmentation methodology, as well as the necessary components of a prototype of the mixed fragmentation Distributed Database Design Tool (D^{3}T), which has been under development. It allows the optimal partitioning of global relations in a distributed database by using a grid approach, i.e. partitioning a single relation by simultaneously applying both horizontal and vertical partitioning in the same algorithm and supports the investigation of the effects of the different sequences of partitioning. This report has motivated our work, wherein the tool allows designers to make distribution design decisions using

horizontal, vertical and/or mixed fragmentation. They do not address allocation with or without replication. Tamhankar *et al.* [5] have developed a comprehensive methodology for fragmentation and distribution of data across multiple sites such that design objectives in terms of response time and availability for transactions, and constraints on storage space are adequately addressed. Daudpota *et al.* [6] have constructed a formal model of data allocation and have derived an algorithm to fragment and allocate the relations. This model is not applied to distributed applications in networks with different connectivity (LAN/WAN). Peddemors *et al.* [7] have described the first phase realization of a DDBS in which an iterative process is used to build the DDBS. Each phase has a set of objectives, spans a limited amount of time, ads functionality, and the output of every phase serves as input for the next phase. This paper has motivated our work in the way that each phase has a set of objectives and its outputs serve as inputs for the next phase of the design process. Bellatreche L. *et al.* [8] formulated the combined methods and class allocation problem and developed a model to calculate the total data transfer cost incurred. Their allocation algorithm generates near optimal solutions to the problem. Lee *et al.* [9] have proposed a heuristic methodology for determining file and workload allocation simultaneously on a LAN. This method minimizes the response time for processing transactions. Only transactions with the same properties are routed to the same server, which does not guarantee the minimization of the communication cost. Their assumption of non-redundant allocation decreases the reliability of the system, and the impact of storing fragment copies on the sites of the LAN is not very well clarified. Huang *et al.* [10] have proposed a heuristic algorithm that reflects transaction behavior in distributed databases. Their model determines the number of replicates for each fragment and finds a near optimal allocation of all fragments in a WAN such that the total communication cost is minimized. The fragments accessed by a transaction are all assumed independent, which is not the case in the real world. This method neglects site information like storage

and processing capacity and it is applied only on a WAN network. They consider the CPU processing time and I/O access time as minor factors in minimizing the total cost in a WAN environment. Son *et al.* [11] have introduced an adaptable vertical partitioning method in distributed systems. Our previous work in this field dealt with components and tools concerning DDB design [12-18]. In the latter work [18], we have analyzed and implemented diverse methods to tackle combinatorial optimization problems in distribution design, which are very complex problems. These methods include exact and heuristics approaches which have been very useful in solving real life problems. The main issue with exact methods is their applicability to large problems, specifically for the type of NP-complete problems for which there is no guarantee to find an optimal solution in a polynomial time [19]. A good alternative for NP-complete combinatorial optimization problems of large size is to find a reasonable solution in a reasonable time [20]. This is the idea of the heuristic methods which are in general quite simple and based on intuitive and common sense ideas [21]. The general problem with many heuristics is that they may get stuck in local optimal solutions. More recently a number of metaheuristics have evolved that define ways to escape local optima. Metaheuristics are higher level heuristics designed to guide other processes towards achieving reasonable solutions, and do not guarantee in general that one will finish with an optimal solution, though some of them present convergence theories. However they have been successfully applied to many problems. Here we explore Genetic Algorithms and a much more recent approach, Reinforcement Learning (RL) for solving the harder problem, namely allocation. RL may be interpreted as a conjunction between machine learning and decision making problems.

Tool Architectural Overview

The problem of DDB design comprises first, the fragmentation of database entities and second, the allocation of these fragments to distributed sites. Two approaches are possible in a DDB de-

sign: top-down and bottom-up. This paper uses the top-down design approach where the input to the design process is the global conceptual schema (GCS). Statistical information collected from the design activities includes access patterns of user applications, and information about sites and the network. The output from the design process is a set of local conceptual schemas (LCS) over distributed sites [3]. The input to the design process is obtained from system requirements analysis which defines the system environment and collects an approximation of both the data and processing needs of all potential database users. Providing an easy user interface for entering the distribution requirements as well as facilitating user control in driving the distribution process are topics that we addressed when implementing an integrated tool to support the entire DDB design cycle. Ceri *et al.* [22] give an outline of the overall DDB design methodology that deviates from conventional centralized database design only in the distribution aspect (see figure 1).

Distribution Design
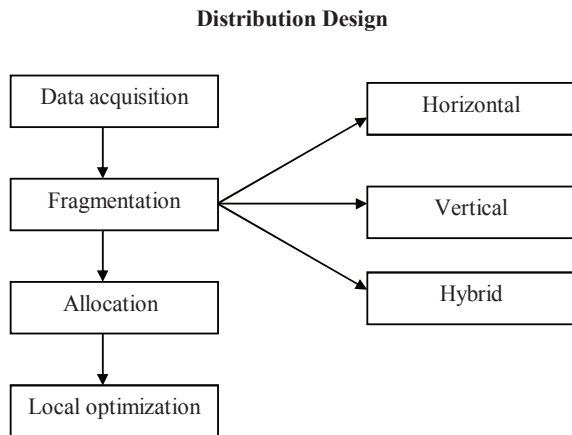
Figure 1 Distribution design activities. (Ceri *et al.*, 1983) [22]

The distribution design involves data acquisition (ERECASE, APPWIZARD, NETWIZARD), database partitioning (FRAGMENTER), allocation and replication of fragments (ALLOCATOR, DISTRIBUTOR), and local optimization (not considered herein). As a result, we have created a DDB design tool that integrates various methods for each component of distribution design.
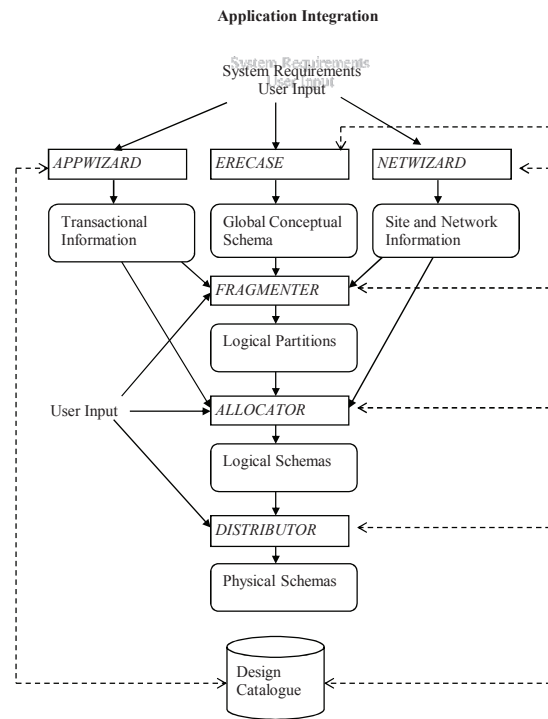
Figure 2 Application integration through the design process

Figure 2 shows an abstract representation of the integrated design process for the tool where information is vehicled from one tool to another by feeding the output of one application as input to the next, but not exactly in a linear fashion. Rather, the common information is stored into a shared database, namely the design catalogue, which can be accessed by each tool through one common interface and is embedded within the integrated tool. From the end-user's perspective, application integration has been successful, if the user is not able to differentiate the sources of data and functionality he accesses from the user interface. Figure 3 depicts an architectural overview of the proposed tool, namely SIADBDD.

Unfortunately, collecting the large amount of required information is a hard task and requires time and effort. Some of the drawbacks of today's integration technology at the user interface can be reduced through process-level integration. The idea is to provide suitable data through a catalogue as part of a workflow (see figure 2). Preceding steps

in the workflow fetch data from the catalogue and use them as inputs to the algorithms involved in the design process. Outputs are placed back to the catalogue, so that they can be used for further steps. Unfortunately, the risks of application integration are often discarded. Because integration means to create dependencies between applications this may reduce the ability to adapt to changes. On the other hand, dependencies are good, because they save time and effort to a great extent.
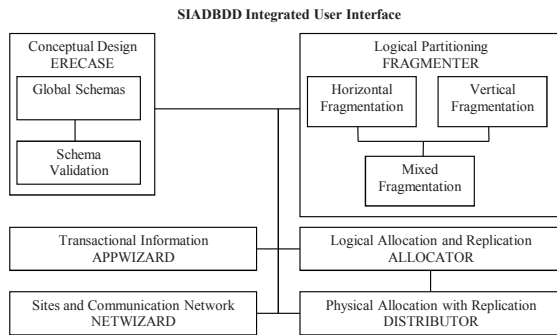

Figure 3 Architectural overview of the integrated tool

Tool Functional Overview

SIADBDD architecture is built up of the following seven main modules: Integrated user interface, Conceptual design, Transactional information, Sites and communication network information, Partitioning, Logical Allocation and Physical Allocation.

Integrated user interface

This is the central component of the tool. The integrated user interface is responsible for activating any needed tool through the design process workflow. It provides appropriate modules for configuring any aspect of the integrated tool (see figure 4).
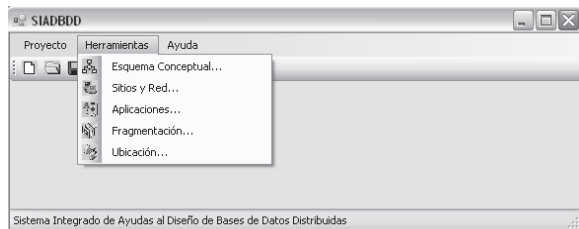

Figure 4 Integrated user interface of SIADBDD

Conceptual design

The design process is logically iterative and exploratory. ERECASE is the implemented tool that helps us characterizing global conceptual schemas [12, 16]. This component provides appropriate features for the definition and redefinition of global conceptual schemas with a variety of constructs from the Extended Entity-Relationship Model [23] It uses the notation from [24]. Additionally, this component can check schemas for correctness by means of structural validations, uniqueness of names, use of identifiers, etc. When the conceptual schema is correct, a visualization of the logical schema by means of relations [25] and a script for relations creation is generated. ERECASE was implemented with an easy user interface for collecting schema information as well as user control in driving the design process (see figure 5).
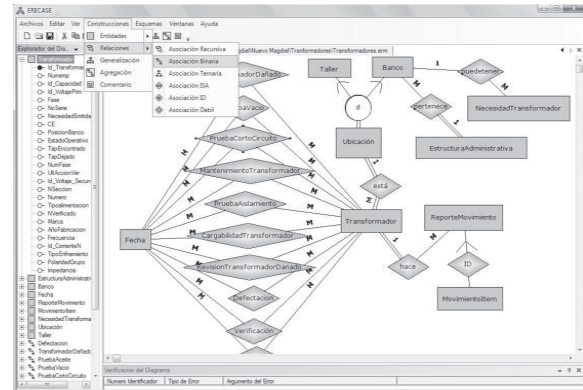

Figure 5 Sample view of ERECASE tool

Transaction information

APPWIZARD is a tool for collecting applications (transactions) access patterns. The tool was implemented with an easy user interface for collecting the distribution requirements as well as user control in driving the design process (see figure 6a).

This tool provides the user with advanced features for getting supplementary information of transactions relevant to the distribution. It is not necessary

to collect information on 100% of the expected transactions (that would of course be impossible). Since the 80-20 rule [22] applies to most practical situations, it is adequate to supply the 20% of the heavily used transactions which account for about 80% of the activity against the database.
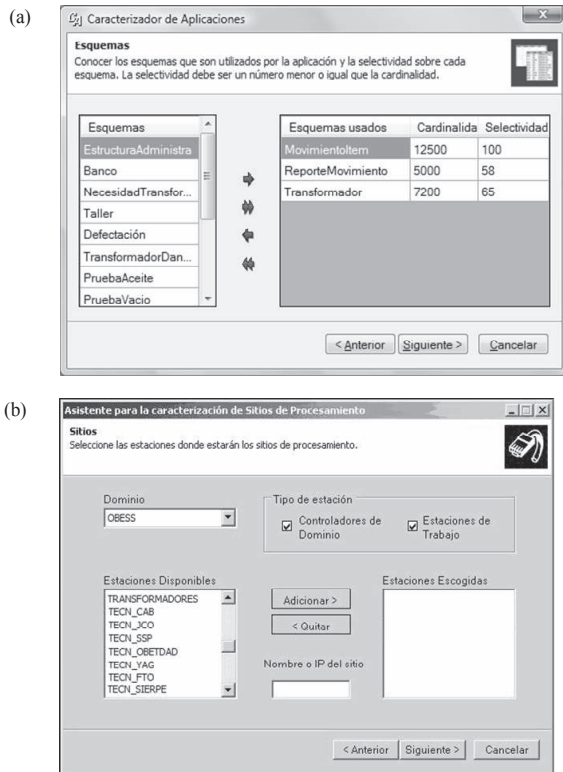
**Figure 6** Sample views of (a) APPWIZARD and (b) NETWIZARD

Sites and communication network information

NETWIZARD is the tool that collects information about sites and communication network required for the DDB design process. This tool provides a number of reports presenting required parameters values of the node in any given simulation. The tool was implemented with an easy user interface for collecting the distribution requirements as well as user control in driving the design process. Figure 6b displays an example view of NETWIZARD. Because of space restrictions, more detailed views are not given.

Partitioning

FRAGMENTER is the tool that allows designers to make distribution design decisions using horizontal, vertical and hybrid fragmentation. Relation instances are essentially tables, so the issue is one of finding alternative ways of dividing a table into smaller ones named fragments.

Three fragment types are defined on a database entity. Horizontal fragmentation is the breaking up of a table into a set of horizontal fragments with only subsets of its tuples [3, 22, 26]. Vertical fragmentation is the breaking up of a table into a set of vertical fragments with only subsets of its attributes [5, 27, 28, 29, 30]. Hybrid (also called mixed) fragmentation is the breaking up of a table into a set of hybrid fragments with both subsets of its tuples as well as subsets of their attributes [4]. A lot of research work has been published on fragmentation and allocation in the relational data model [3, 4, 27, 31] (see figure 7).
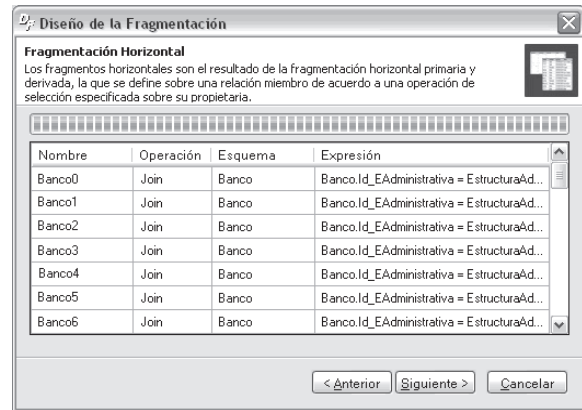
Figure 7 Sample view of FRAGMENTER

Allocation

While fragmentation is an important issue, our main concern in this section is with how the data should be allocated around the network once it has been partitioned by whatever criteria. Data allocation is a critical aspect of DDBSs: a poorly designed data allocation can lead to inefficient computation, high access costs and high network loads [3, 32] whereas a well designed data alloca-

tion can enhance data availability, diminish access time, and minimize overall usage of resources [3, 33]. It is thus very important to provide DDBSs that find a good solution in a reasonable amount of time, achieving data allocations that minimize the cost of answering the given queries. This section addresses the problem of determining where to place a given set of fragments on a network in order to minimize the cost of answering a given set of queries Q. We assume that fragmentation of the original relations has been carried out before the data allocation phase. ALLOCATOR is a tool that supports allocation with replication (see figure 8). Since the allocation problem is pretty complex and involves combinatorial optimization problems, the tool implements Genetic Algorithms and a Q-Learning method for mapping fragments to sites. Outputs of these methods can be compared and then selected for materialization.
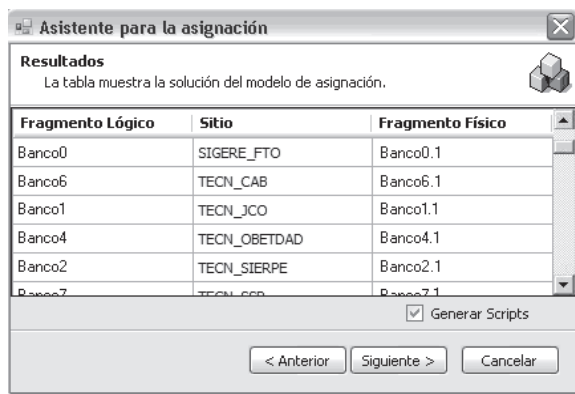

Figure 8 Logical fragments allocation by ALLOCATOR

Physical allocation

DISTRIBUTOR materializes physical designs by allocating distribution partitions obtained by ALLOCATOR to sites on the network by means of replication under the publish-distribute-subscribe model. Data replication is a key technology in distributed systems that enables higher availability and performance. Physical designs are completed over the generation of script using calls to Transact-SQL store procedures.
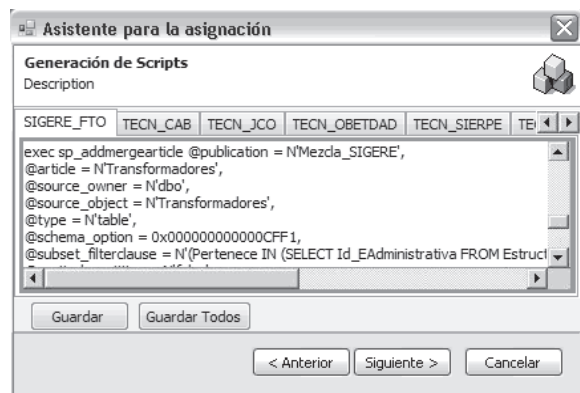

Figura 9 Scripts that materialize physical designs

Conclusions

This paper outlines issues involved in the conceptual design, fragmentation and allocation in a DDBS. The paper proposes a novel integrated tool for aiding designers in initial distributed database designs. The contribution of this work is the implementation of the integrated tool, built up of a variety of applications and methods for performing distributed database designs. This enables designers to easily design and validate designs with minor time and effort consumption. The algorithms necessary to support the design process are implemented and their complexities are polynomial. A description of the architecture and functions is also provided. The utility of this tool is clear cut. Unfortunately, many design parameters need to be entered by designers, and their estimation is sometimes difficult.

Future work

Further research could study the tuning of the parameters involved in algorithms for allocating fragments within ALLOCATOR. At this moment we work on the integration of several algorithms for this allocation problem, specifically Q-Learning, Genetic Algorithms, Bird Flocks and some other tools developed by our research group. The main goal is to help in the design of Distributed Databases in a more efficient way by using less effort and time. Furthermore, we are working on semantic schema validation, and improving algorithms for distributing data over sites within DISTRIBUTOR.

Acknowledgements

References

1. C. I. Ezeife, K. Barker. "Distributed Object Based Design: Vertical Fragmentation of Classes". *Distributed and Parallel Databases*. Vol. 6. 1998. pp. 317-350

2. K. Karlapalem, S. B. Navathe, M.M.A. Morsi. *Issues in distribution design of object-oriented databases.* M. T. Özsu, U. Dayal, P. Valduriez (editors.) Distributed Object Management. Ed. Morgan Kaufmann. San Mateo, California, USA.1994. pp. 148-164

3. M. T. Őzsu, P. Valduriez. *Principles of Distributed Database Systems,* 2nd ed. Ed. Prentice-Hall. Upper Saddle River, New Jersey. 1999. pp. 80-166.

4. S. B. Navathe, K. Karlapalem, M. Ra. *A mixed fragmentation methodology for initial distributed database design.* College of Computing. Georgia Institute of Technology. Atlanta, Georgia. USA. 1995. pp. 1-34.

5. A. M. Tamhankar, S. Ram. "Database fragmentation and allocation: an integrated methodology and case study". *IEEE Transactions on Systems, Man, and Cybernetic Part A*. Vol. 28. 1998. pp. 288-305.

6. N. H. Daudpota. "Five Steps to Construct a Model of Data Allocation for Distributed Database Systems". *J. Intell. Inf. Syst.* Vol. 11. 1998. 153-168.

7. A. J. H. Peddemors, L. O. Hertzberger. "A High Performance Distributed Database System for Enhanced Internet Services". P. M. A. Sloot, M. Bubak, L.O. Hertzberger (editors.) *Proceedings of the International Conference and Exhibition on High-Performance Computing and Networking, Europe* 1998. Ed.Springer. Amsterdam. 1998. pp.469-478.

8. L. Bellatreche, K. Karlapalem, Q. Li. "Complex Methods and Class Allocation in Distributed Object-Oriented Database Systems". *International Conference on Object Oriented Information Systems.* 1998. pp. 239-256.

9. H. Lee, Y. K. Park, G. Jang, S. Y. Huh. "Designing a distributed database on a local area network: a methodology and decision support system". *Information & Software Technology*. Vol. 42. 2000. pp. 171-184.

10. Y. F. Huang, J. H. Chen. "Fragment Allocation in Distributed Database Design". *Journal of Information Science and Engineering.* Vol. 17. 2001. pp. 491-506.

11. J. H. Son, M. H. Kim. "An adaptable vertical partitioning method in distributed systems". *Journal of Systems and Software*. Vol. 73. 2004. pp. 551-561.

12. C. E. García, A. Rodríguez, L. M. González, W. A. Álvarez. *ERECASE, una herramienta con validación de diagramas entidad relación. 6to. Simposium Iberoamericano de Computación e Informática SISI 2005.* Instituto Tecnológico de Nuevo León, Monterrey. NL. México. 2005. pp. 1-10.

13. A. Morell, L. M. González, A. Rodríguez. *Un enfoque a la fragmentación vertical en bases de datos distribuidas.: Congreso Internacional Informática 2000.* Nuevas Tecnologías Informáticas. La Habana. Cuba. 2000. pp. 1-10.

14. A. Morell, L.M. González, A. Rodríguez. "Algoritmos para la fragmentación vertical en bases de datos distribuidas". *COMPUMAT 2000. 7mo Congreso de la Sociedad Cubana de Matemática y Computación.* Manzanillo. Cuba. 2000.

15. A. Rodríguez, L. M. González, L. S. Águila. "Asignación de fragmentos en Bases de Datos Distribuidas mediante la aplicación de Algoritmos Genéticos". *Boletín de la Sociedad Cubana de Matemática y Computación*. Vol. 3. 2005. pp. 1-6.

16. A. Rodríguez, L. M. González, L. Cabrera, A. Morell. "ERECASE:Una herramienta de ayuda a la modelación de esquemas conceptuales globales". *I Workshop de Bases de Datos, Jornadas Chilenas de Computación JCC 2002*. Cámara Chilena del Libro A.G. Universidad de Atacama. Copiapó. Chile. 2002. pp. 49-58.

17. A. Rodríguez, L. M. González, A. Morell, L. Cabrera, M. Artiles, L. S. Águila, Á. Valdés. *Integración de herramientas de ayuda al diseño de bases de datos distribuidas. I Workshop de Bases de Datos, Jornadas Chilenas de Computación JCC 2002*. Cámara Chilena del Libro A.G., Universidad de Atacama. Copiapó. Chile. 2002. pp. 111-120.

18. A. Rodríguez, D. Rosa, M. Mainegra, L. M. González. *An Intelligent Agent using Reinforcement Learning to Solve the Allocation Problem in a Distributed Database with Replication.* Technical Report. Universidad Central de Las Villas. Santa Clara. 2007. pp. 1-10

19. X. Lin, M. E. Orlowska, Y. Zhang. "On Data Allocation with the Minimum Overall Communication Costs in Distributed Database Design". O. Abou-Rabia, C.K. Chang, W.W. Koczkodaj (eds.): *Proceedings of the Fifth International Conference on Computing and Information - ICCI'93*. IEEE Computer Society, Sudbury. Ontario. Canada. 1993. pp. 539-544.

20. C. H. Papadimitriou, N. P. Completeness, A. Retrospective. P. Degano, R. Gorrieri, A. Marchetti (eds.). *Proceedings of the 24th International Colloquium on Automata, Languages and Programming, ICALP'97*. Springer. Bologna. Italy. 1997. pp. 2-6.

21. X. Lin, M. E. Orlowska. "An Integer Linear Programming Approach to Data Allocation with the Minimum Total Communication Cost in Distributed Database Systems". *Inf. Sci.* Vol. 85. 1995. pp. 1-10.

22. S. Ceri, S. B. Navathe, G. Wiederhold. "Distribution Design of Logical Database Schemas". *IEEE Trans. Software Eng.* Vol. 9. 1983. pp. 487-504.

23. P. P. Chen. "The Entity-Relationship Model - Toward a Unified View of Data". *ACM Trans. Database Syst.* Vol. 1. 1976. pp. 9-36.

24. R. Elmasri, S. B. Navathe. *Fundamentals of Database Systems*. 2nd ed. Ed. Benjamin-Cummings. Menlo Park. CA. 1994. pp. 28-107

25. E. F. Codd. *The Relational Model for Database Management*. Version 2. Ed. Addison-Wesley 1990. pp. 1-89.

26. S. Ceri, G. Pelagatti. *Distributed Databases: Principles and Systems*. Ed. McGraw-Hill. New York. 1984. pp. 128-136.

27. F. A. Baião, M. Mattoso, J. W. Shavlik, G. Zaverucha. "Applying Theory Revision to the Design of Distributed Databases". T. Horváth (ed.): *Inductive Logic Programming: 13th International Conference*. ILP 2003, LNAI 2835. Ed. Springer. Szeged. Hungary. 2003. pp. 57-74.

28. J. Pérez, R. A. Pazos, J. F. Solís, D. Romero, L. Cruz. "Vertical Fragmentation and Allocation in Distributed Databases with Site Capacity Restrictions Using the Threshold Accepting Algorithm". O. Cairó Battistutti, L. E. Sucar, F. J. Cantu (eds.): MICAI 2000: *Advances in Artificial Intelligence, Mexican International Conference on Artificial Intelligence*. Ed. Springer. Acapulco. 2000. pp. 75-81.

29. L. Bellatreche, A. Simonet, M. Simonet. *Vertical Fragmentation in Distributed Object Database Systems with Complex Attributes and Methods*. 8. th. Inter. Conf. on Database and Expert Systems. 1996. pp. 15-21

30. S. B. Navathe, M. Ra. "Vertical Partitioning for Database Design: A Graphical Algorithm". *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon*. 1989. ACM Press. 1989. pp. 440-450

31. S. Ceri, B. Pernici, G. Wiederhold. "Distributed Database Design Methodologies". *IEEE Database Eng. Bull.* Vol. 75. 1987. pp. 533-546.

32. D. Saccà, G. Wiederhold. "Database Partitioning in a Cluster of Processors". *ACM Trans. Database Syst.* Vol. 10. 1985. pp. 29-56.

33. P. M. G. Apers. "Data Allocation in Distributed Database Systems". *ACM Trans. Database Syst.* Vol. 13. 1988. pp. 263-304.