DYNAMICAL PROCESSING TECHNIQUE FOR DATA FRAGMENTATION IN DISTRIBUTED DATABASE SYSTEMS: DESIGN AND TOOL SUPPORT

ISMAIL OMAR HABABEH
SCHOOL OF COMPUTING-LEEDS METROPOLITAN UNIVERSITY, LEEDS, U.K
E-MAIL: ISMAIL.HABABEH@UAEU.AC.AE
MUTHU RAMACHANDRAN
SCHOOL OF COMPUTING-LEEDS METROPOLITAN UNIVERSITY, LEEDS, U.K.
NICHOLAS BOWRING
DEPARTMENT OF ENGINEERING & TECHNOLOGY-MANCHESTER METROPOLITAN UNIVERSITY, MANCHESTER, U.K.

(Received 23 May 2005)

ABSTRACT. Data fragmentation is one of the primary techniques that has been used in partitioning and developing distributed database systems (DDBs). Database partitioning technique improves the performance of the system throughput and reduces the amount of irrelevant data to be accessed and transferred among different sites in DDBs. We propose a technique that splits the database relations into pair-wise disjoint fragments, which would be allocated to the DDBs sites during the process of allocation. This technique reduces the data transferred between the sites during the applications execution, minimizes the communication cost needed for processing applications, handles the database queries and meets their future needs, guarantees the ability of processing a given portion of a given transaction in all sites, and improves the applications speed and efficiency by minimizing the number of fragments to be accessed. We have also developed a fragmentation tool that is interactive, full functional and easy to use, based on our fragmentation life cycle algorithms to create the database segments and generate the minimum number of disjoint fragments. A case study and some results in tabular and graphical means will be presented to demonstrate the efficiency of the method.

AMS Classification: 68P15
Keywords: Distributed database, Data segmentation, Data fragmentation, Application definition, Disjoint fragments, algorithms, Fragment life cycle.

1. INTRODUCTION

The design of database partitioning improves the performance of applications by reducing the amount of irrelevant data accessed by the applications and the amount of data transferred during applications processing [11]. Fragmentation is the process

JOURNAL OF DYNAMICAL SYSTEMS & GEOMETRIC THEORIES
VOL. 4, NUMBER 2 (2006) 103-116.
©TARU PUBLICATIONS

of partitioning the database relations into disjoint fragments and describes how each fragment in the DDBs is derived from the global relations.

Various methods already exist describe data fragmentation in distributed DDBs. Naturally, there are benefits and drawbacks to all schemes. Some methods need to incorporate performance evaluation ways for merging grid cells [9, 8] or may not minimize the transactions response time [6]. Some have an exponential time complexity for their allocating methods [3] and [1]. Other doesn't guarantee the ability of all sites to process a given portion of a given query [10]. Some methods applied on specific type of network connectivity or small number of sites [2, 7, 12].

This paper presents a method for partitioning data in a relational database into pair-wise disjoints fragments. We propose a method to define a horizontal partitioning technique that split the database relations into pair-wise disjoint fragments, which would be allocated to the sites of DDBs during the process of allocation [6]. The records of a relation are assigned to different disjoint fragments such that union of the disjoint fragments can obtain the relation. In this type of fragmentation, all of the information in the record is used, or else none is.

The remainder of this paper is organized as follows. In section 2, we will define the fragment life cycle processes. Description of our partitioning technique will be done in section 3. A case study for our partitioning algorithm will be illustrated in section 4. In section 5, a description of our tool support will be presented. Performance evaluation will be done in section 6. Finally and in section 7, we will have the conclusion.

2. THE FRAGMENT LIFE CYCLE

The fragment life cycle in the dynamical distributed database systems is identified by four main processes; defining the database application, segmentation, fragmentation, and fragment allocation. We define the fragment life cycle processes in the following sequence:

- The sites requesting data in forms of database applications, which determine the specific information that should be extracted from the database.
- The database segments are created to extract the records that match the requirements of each application, the segments could have the same records or it could be overlapped.
- The minimum number of disjoint fragments are generated from the segments, the fragments should not have intersection records.
- The disjoint fragments would be allocated to the DDBs sites requesting that data.

Figure1 describes the design of the fragment life cycle in our method.

3. DATABASE PARTITIONING

Our partitioning technique is designed to avoid allocating unnecessary records to the fragments for a given application and reduces the amount of irrelevant data to be accessed and transferred among different sites in the DDBs.

3.1. **Defining Applications.** The data that are requested by DDBs sites should be identified by the database administrators who categorize the data into applications , set their requirements, and determine their input parameters. Table 1 shows

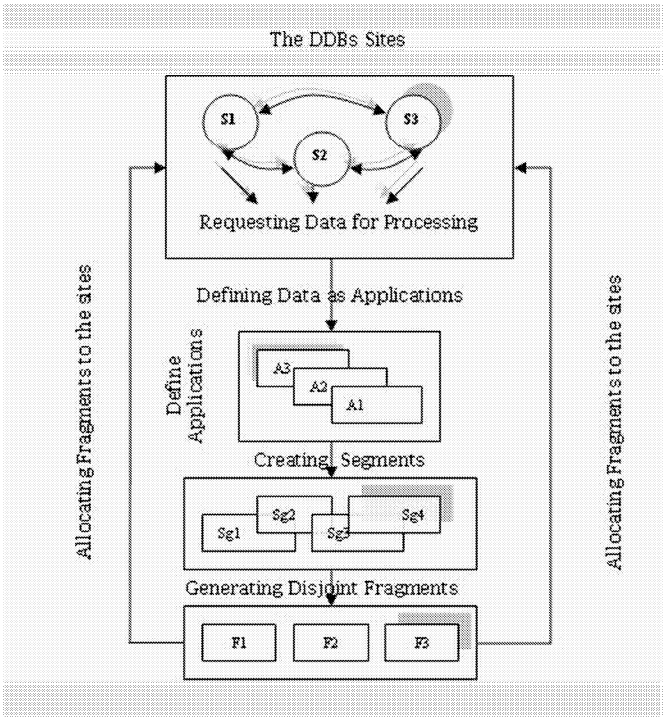


FIGURE 1

an example for some applications requested by DDBs sites.

Table1. Database Applications

Site #	Application #	Application definition
S1	A1	List Of All Employees
S2	A7	Employees Who Have Major M3 At Site 2
S3	A13	Classes Of Employees At Site 3
S7	A26	Employees Who Have Classes $\leq$ R4
S7	A27	Sites At Locations Between L-11110 And L-11115
S8	A30	Employees Who Have Basic Salary $>$ \$ 3500
S9	A33	Employees Who Have Allowances $\geq$ \$ 500
S10	A38	Classes Who Have Allowances $\leq$ \$ 450
S11	A40	Employees Who Have Major M1 At Site 11
S12	A44	List Of Sites Where Site# $\leq$ 7

3.2. **Creating Segments.** Segmentation is the process of extracting the database records that satisfy certain condition(s) according to the definition of each database

application; it is necessary for generating the database fragments. SELECT, JOIN, and SEMIJOIN relational operators [11] are used to generate the database segments. The following algorithm defines the process of creating database segments.

Segmentation Algorithm

Input: Nmax: Number of applications in the DDBs
Database applications definitions
Output: The set of database segments
Step 1: For N = 1, Nmax, do step (2)
Step 2: Segment (N) = Records used by application(N)
Step 3: End For
Step 4: Stop.

Since we use a horizontal partitioning technique, then the number of created segments equals to the number of database applications definitions.

3.3. **Generating Fragments.** We define the following horizontal partitioning technique, by which the global database is segmented into pair-wise disjoint fragments through the DDBs sites. A pair of joint segments S_i, S_j are replaced by three fragments:

- The first fragment consists of the records resulted from the intersection between S_i and S_j. $F_k = S_i \bigcap S_j$.
- The second fragment consists of the records in segment S_i but not in the intersection fragment F_k. $F_{k+1} = S_i - F_k$.
- The third fragment consists of the records in segment S_j but not in the intersection fragment F_k. $F_{k+2} = S_j - F_k$.

The union of all fragments that form the relation could reconstruct the split relation. The segments of each relation are determined by the queries performed on the database system, and the segment itself consists of complete records. The empty segments are removed and the non-empty segments are referred as fragments, which would be allocated to the DDBs clusters and sites [6]. The following algorithm defines the process of generating database fragments.

fragmentation Algorithm

Input: K: Number of the last fragment in the DDBs
Rmax: Number of database relations in the DDBs
Nmax: Number of records in each relation
Output: The set of disjoint fragments in DDBs
Step 1: Set 0 to K
Step 2: For R = 1, Rmax, do steps (3) - (16)
Step 3: For I = 1, Nmax, do steps (4) - (15)
Step 4: For J = 1, Nmax, do steps (5) - (14)
Step 5: If I $\neq$ J go to step (6)
Else, go to step (13)
Step 6: If $S_i \bigcap S_j \neq \phi$ do steps (7) - (13)
Else, go to step (13)

Step 7: Add 1 to K
Step 8: Create new fragment $F_k = S_i \bigcap S_j$ in relation R
Step 9: Create new fragment $F_{k+1} = S_i - F_k$ in relation R
Step 10: Create new fragment $F_{k+2} = S_j - F_k$ in relation R
Step 11: Set $S_i = \phi$
Step 12: Set $S_j = \phi$
Step 13: End IF
Step 14: End For
Step 15: End For
Step 16: End For
Step 17: Stop.

3.4. **Free Records Fragments.** In case of remaining segments ($S_i \bigcap S_j = \phi$) that are not belonging to any fragment, a new fragment should be created and added to the collection of fragments in that relation. The following algorithm describes the process of creating new fragments for free records.

Free records fragmentation algorithm

Input: K: Number of the last fragment in the DDBs
Rmax: Number of database relations in the DDBs
Nmax: Number of records in each relation
F: The generated fragments for all relations in DDBs
Output: The new fragments for free records in all relations
Step 1: For R = 1, Rmax, do steps (2) - (13)
Step 2: For I = 1, Nmax, do steps (3) - (12)
Step 3: For J = 1, Nmax, do steps (4) - (11)
Step 4: If $I \neq J$ go to step (5)
Else, go to step (10)
Step 5: If $S_i \bigcap S_j = \phi$ do steps (6) - (10)
Else, go to step (10)
Step 6: Add 1 to K
Step 7: Create new fragment $F_k = R_j - \bigcup F$ in relation R
Step 8: End IF
Step 9: If $F_k \neq \phi$ Add F_k to the collection fragments of the relation
Step 10: End IF
Step 11: End For
Step 12: End For
Step 13: End For
Step 14: Stop.

3.5. **Allocating Fragments.** The generated disjoint fragments could be allocated to the site requesting the data composing the fragments, to the group of sites sharing that data, or to all DDBs sites if there is a clear positive benefit of the computation of allocating each fragment to every site [5]. Allocating fragments to clusters and their sites [6] enhances the DDBs performance.

In our partitioning method, all transactions are processed, the redundant transactions are eliminated, and the applications speed and efficiency are improved by getting the minimum number of fragments to accessed.

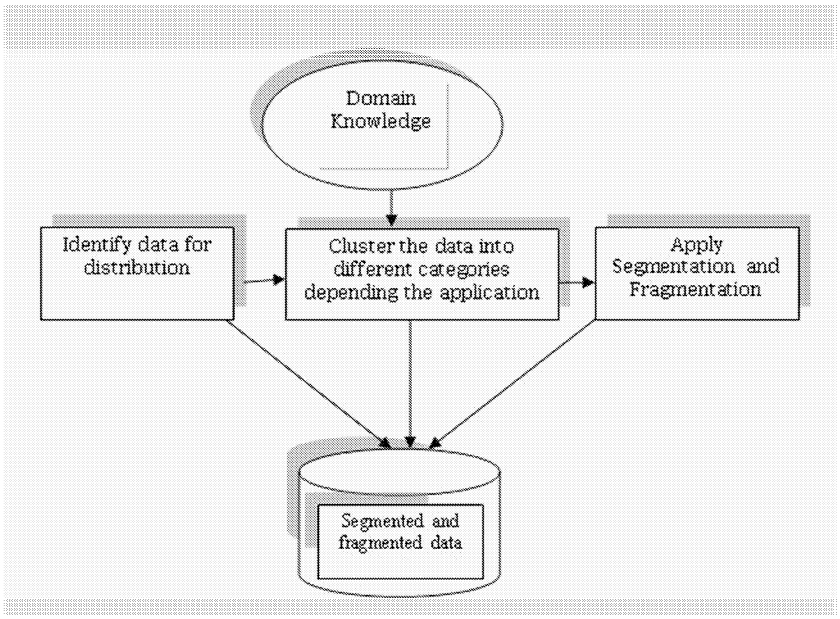


FIGURE 2

Figure 2 shows our data fragmentation methodology.

4. A CASE STUDY

We will apply our fragmentation technique on a full-distributed relational database; with a complete connected network topology that is assumed for the information needs of a public institution. For simplicity, we propose twelve sample sites distributed graphically; site (S1) is used for executing and developing applications, each one of the sites (S1, S2, S3, S4) serves as a major node for communication with other sites. In addition, the sites (S5, S6, S7, ..., S12) serve as local terminals for facilities which produce administration and financial information for the institution's employees. Figure 3 shows the distributed database sites over the network.

4.1. **Database Relations.** Five global relations are assumed in this case study. Table 2 describes the name, number of records, data fields, and the record size of each relation.

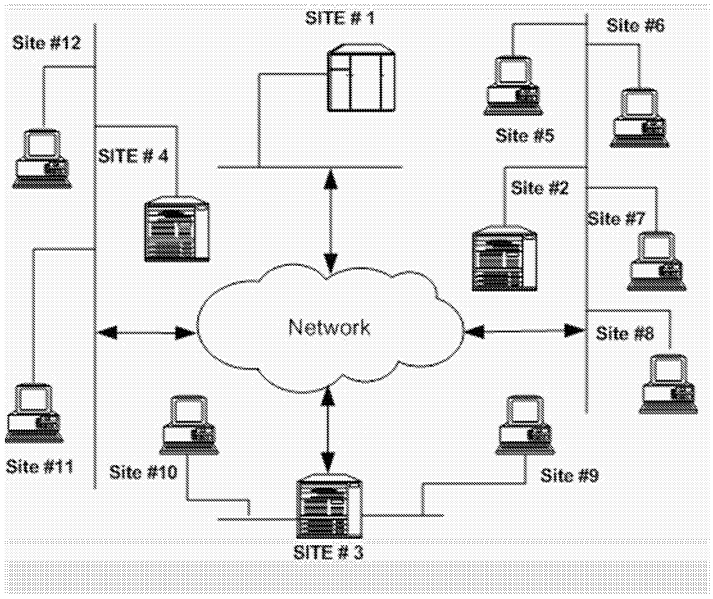


FIGURE 3

Table2. Database relations.

Relation #	Relation Name	Record Size (Bytes)	# of Records	Data Fields
R1	EMPLOYEE	40	20	Emp #, Emp Name, Rank #, Site #, Major #
R2	SALARY	32	7	Rank #, Basic sal., Allowances, Deductions
R3	SITES	32	12	Site #, Site Name, Location
R4	MAJOR	24	4	Major #, Major Name
R5	CLASSES	24	7	Rank #, Rank Name

4.2. **Database Applications Definition.** We propose 45 different applications, table 3 shows the description of each application definition and its requirements in SQL format.

Table3. The Database applications requirements

Site #	Application #	Application Requirements
S1	A1	Select * From Employee
S1	A2	Select * From Classes

S1	A3	Select * From Salary
S1	A4	Select * From Sites
S1	A5	Select * From Majors
S2	A6	Select * From Employee
S2	A7	Select * From Employee Where Major # = M3 And Site # = 2
S2	A8	Select * From Classes, Employee Where Classes.Rank# = Employee.Rank# And Employee.Site# = 2
S2	A9	Select * From Salary, Employee Where Salary.Rank# = Employee.Rank# And Employee.Site# = 2
S2	A10	Select * From Sites
S3	A11	Select * From Salary
S3	A12	Select * From Employee Where Major # = M2 And Site # = 3
S3	A13	Select * From Classes, Employee Where Classes.Rank# = Employee.Rank# And Employee.Site# = 3
S3	A14	Select * From Sites
S4	A15	Select * From Majors
S4	A16	Select * From Employee Where Major # = M4 And Site # = 4
S4	A17	Select * From Classes, Employee Where Classes.Rank# = Employee.Rank# And Employee.Site# = 4
S4	A18	Select * From Sites
S5	A19	Select * From Employee
S5	A20	Select * From Classes, Employee Where Classes.Rank# = Employee.Rank# And Employee.Site# = 5
S5	A21	Select * From Employee Where Major # = M4 And Site # = 5
S6	A22	Select * From Majors
S6	A23	Select * From Sites
S6	A24	Select * From Classes, Employee Where Classes.Rank# = Employee.Rank# And Employee.Site# = 6
S7	A25	Select * From Classes
S7	A26	Select * From Employee Where Rank# In(R1, R2, R3, R4)
S7	A27	Select * From Sites Where Location In (L-11110 ,L-11111,L-11112,L-11113,L-11114,L11115)
S8	A28	Select * From Sites
S8	A29	Select * From Salary
S8	A30	Select * From Employee, Salary Where Employee.Rank# = Salary.Rank# And Salary.Basic > $ 3500
S9	A31	Select * From Salary
S9	A32	Select * From Sites

S9	A33	Select * From Employee, Salary Where Employee.Rank# = Salary.Rank# And Salary.Allowances $\geq$ \$ 500
S9	A34	Select * From Employee Where Rank# In (R5,R6,R7)
S10	A35	Select * From Majors
S10	A36	Select * From Employee Where Site# = 10
S10	A37	Select * From Salary
S10	A38	Select * From Classes, Salary Where Classes.Rank# = Salary.Rank# And Salary.Allowances $\leq$ \$ 450
S10	A39	Select * From Employee, Salary Where Employee.Rank# = Salary.Rank# And Salary.Deductions $\geq$ \$ 300
S11	A40	Select * From Employee Where Major # = M1 And Site # = 11
S11	A41	Select * From Sites
S11	A42	Select * From Classes
S12	A43	Select * From Employee Where Rank# = R3 And Site # = 12
S12	A44	Select * From Sites Where Site# $\leq$ 7
S12	A45	Select * From Employee Where Site# = 4

5. A TOOL SUPPORT

We developed an implementation tool that is interactive, full functional and easy to use, based on our partitioning technique to create the database segments and produce the minimum number of fragments.

The architecture of our tool known as DiSF (Distributed Segmentation and Fragmentation) is novel in the sense that it supports the use of knowledge extraction technique to help achieve the effective use of our algorithm. Figure 4 illustrates the architecture of DiSF.

The database segments are created according to their applications definition. Figure 5 shows the process of creating database segments.

Figure 6 shows the result of the segment execution.

Each database segment executed is used as a source for the next processing step; the data fragmentation process.

The following procedures illustrate the steps of generating database fragments in our tool. Table 4a-n shows the detailed procedures to generate some fragments for the salary relation.

- The first segment in relation number 2, Seg.3 intersects with the next segment in the same relation Seg.9.

Table4-a

Segment #	Site #	Application #	Relation #	Record(s)#
Seg. 3	S1	A3	RL2	1,2,3,4,5,6,7
Seg. 9	S2	A9	RL2	2,3,4

- The result is new disjoint fragments F1 and F2.

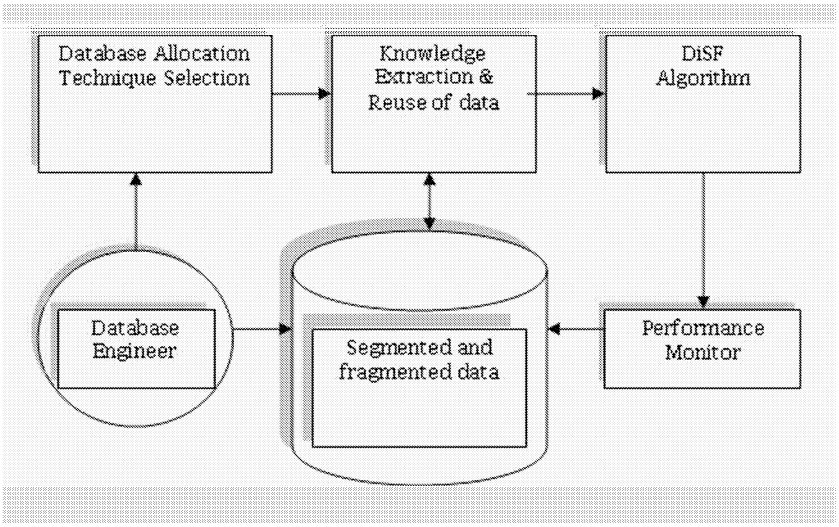


FIGURE 4

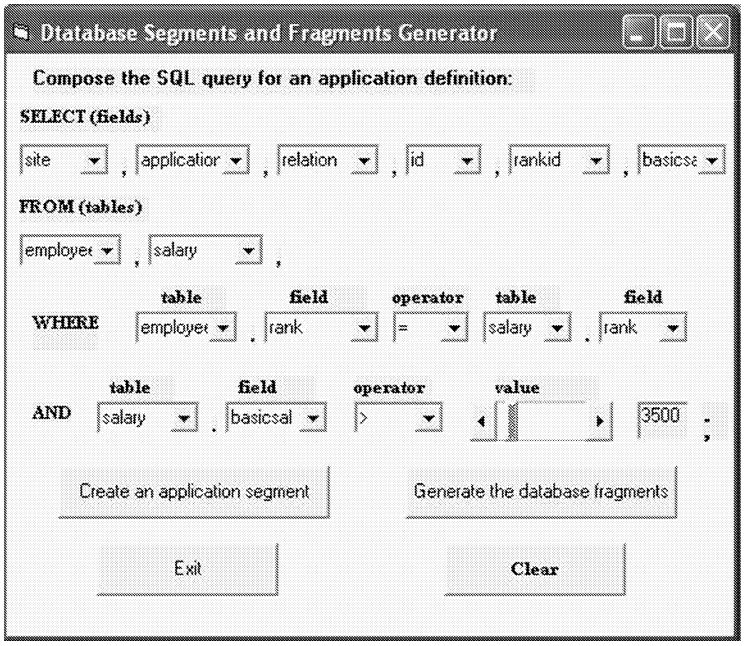


FIGURE 5

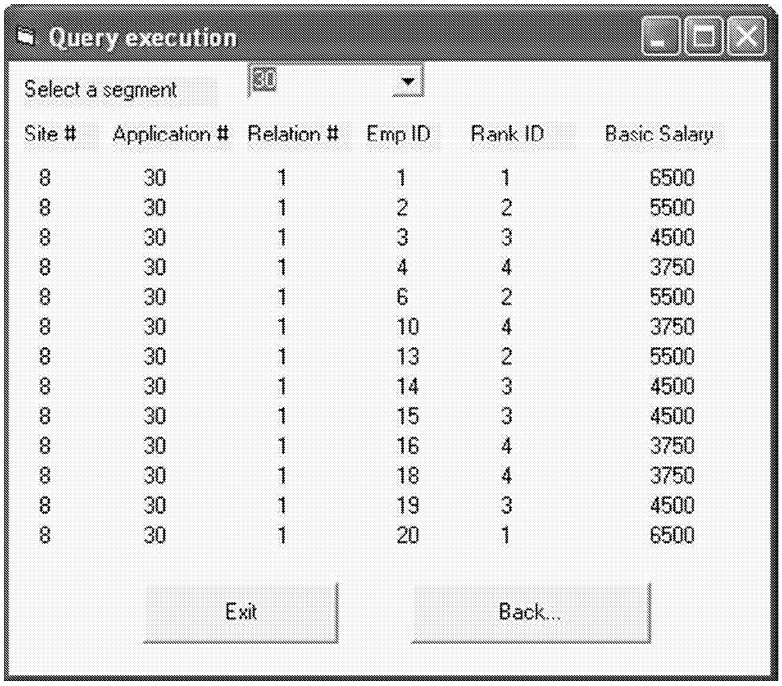


FIGURE 6

Table4-b

Fragment #	Site #	Application #	Relation #	Record(s)#
F1	S1,S2	A3,A9	RL2	2,3,4
F2	S1	A3	RL2	1,5,6,7

- Seg.3 and Seg.9 are omitted, and the next two segments in the same relation; Seg.11 and Seg. 29 are intersected.

Table4-c

Segment #	Site #	Application #	Relation #	Record(s)#
Seg. 11	S3	A11	RL2	1,2,3,4,5,6,7
Seg. 29	S8	A29	RL2	1,2,3,4,5,6,7

- The result is new disjoint fragment F3.

Table4-d

Fragment #	Site #	Application #	Relation #	Record(s) #
F3	S3,S8	A11,A29	RL2	1,2,3,4,5,6,7

- Seg.11 and Seg.29 are omitted, and tool continue the fragmentation process for the remaining segments in this relation.
- When all segments in this relation are fragmented, the tool start fragmentation process for any intersection between the generated fragments in this relation. In this case F1 intersects with the next fragment F3). (Notice that the intersection between F1 and F2 is empty).

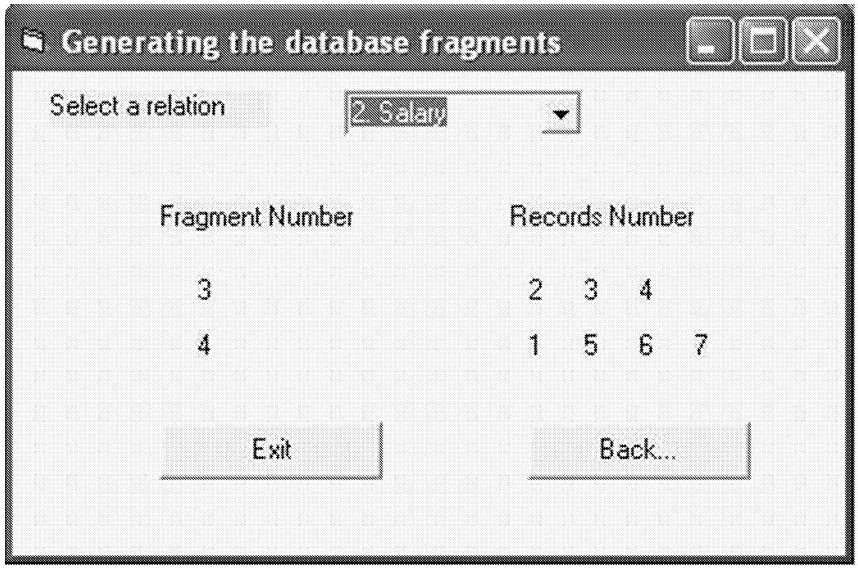


FIGURE 7

Table4-e

Fragment #	Site #	Application #	Relation #	Record(s) #
F1	S1,S2	A3,A9	RL2	2,3,4
F3	S3,S8	A11,A29	RL2	1,2,3,4,5,6,7

- The result is new disjoint fragments F5 and F6.

Table4-f

Fragment #	Site #	Application #	Relation #	Record(s) #
F5	S1,S2,S3,S8	A3,A9,A11,A29	RL2	2,3,4
F6	S3,S8	A11,A29	RL2	1,5,6,7

- F1 and F3 are omitted, and the fragmentation process continue for the remaining fragments in this relation.
- The final fragments in this relation are renamed sequentially and the free disjoint segments are renamed as fragments and added to the list of the final fragments.

Figure 7 shows the generated fragments for salary relation and their corresponding records in the final fragmentation process.

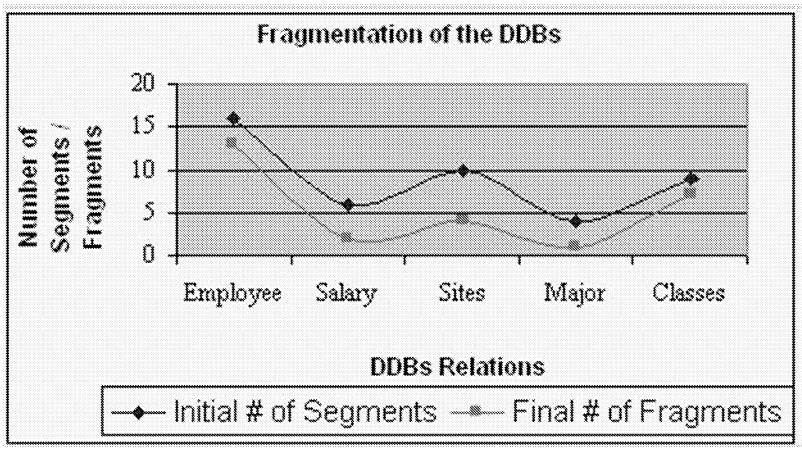


FIGURE 8

6. PERFORMANCE EVALUATION

Data fragmentation improves the performance of DDBs by reducing the amount of irrelevant data accessed by the applications. We applied our partitioning methods on 45 different applications. All relations are partitioned according to the applications requesting them and 45 segments are generated. After applying our partitioning method we generate 27 fragments, so 18 segments are omitted from the DDBs and the system performance increased by 40%. Table 5 shows the performance improvement in the DDBs after applying our partitioning algorithm.

Table5. The performance improvement of the fragmentation algorithm

Relation #	Relation Name	Initial # of Segments	Final # of Fragments	Improvement%
1	Employee	16	13	+ 18.75%
2	Salary	6	2	+ 66.67%
3	Sites	10	4	+ 60%
4	Major	4	1	+ 75%
5	Classes	9	7	+ 22.22%
TOTAL		45	27	+ 40%

Figure 8 shows the improvement of the performance achieved by our fragmentation algorithms.

7. CONCLUSION

In this approach, we met the requirements of determining data fragmentation in DDBs, removing the redundant records from the DDBs, and enhancing the system performance. This application method can be implemented even if the input parameters are very large. In the future we will focus on finding new computational

methods for allocating fragments to the sites of the DDBs, and adding an adaptive algorithm to incorporate fault tolerance constraints during the determination of fragment allocation.

References

[1] Cheng Chun-Hung, Wing-Kin Lee, Kam-Fai Wong (2002). A Genetic Algorithm-Based Clustering Approach for Database Partitioning. IEEE Transactions On Systems, Man, And Cybernetics-Part C: Applications and Reviews, Vol. 32 No. 3, August 2002.

[2] Daudpota N.H. (1998). Five steps to construct a model of data allocation for distributed database systems. Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies. vol.11, no.2; Sept.-Oct. 1998. p.153-68.

[3] Gen Yee Wai and Michael J. Donahoo and Shamkant B. Navathe, "A Framework for Server Data Fragment Grouping to Improve Server Scalability in Intermittently Synchronized Databases", CIKM 2000, November, 2000.

[4] Hababeh Ismail, Bowring Nicholas, and Ramachandran Muthu. "A Strategy for Clustering and Fragment Allocation Design in DOODBs". The 2nd International Conference on Computer Science and its Applications ICCSA 2004". June 28th - 30th, 2004. National University & US Education, San Diego - California, USA. PP 345 - 352.

[5] Hababeh Ismail, Bowring Nicholas, and Ramachandran Muthu. 2005 "An Integrated Mathematical Model for the Computation of Data Fragmentation and Allocation in the Distributed Database systems". International Journal of Pure And Applied Mathematical Sciences, GBS Publishers & Distributors. Vol. 2, No 1.

[6] Huang Yin-Fu, Chen Jyh-Her (2001). Fragment Allocation in Distributed Database Design. Journal of Information Science and Engineering. 17 (2001). P. 491-506.

[7] Lee H.,Y.-K.Park, G.Jang, S.-Y.Huh, Designing a distributed database on a local area network: A methodology and decision support system. Information and Software Technology. 2000, 42 P. 171-184.

[8] Minyoung, Ra & Park, Yang-sun (1992) Data fragmentation and allocation for PC-based distributed database design, Korea: Science & Engineering Foundation.

[9] Navathe, Karlapalem, and Minyoung (1995). A mixed fragmentation methodology for initial distributed database design. Journal of Computer and Software Engineering. v.3, no 4; 1995. P395-425.

[10] Stonebraker, Michael, Aoki, Paul, Devine, Robert, Litwin, Withold and Olson, Michael (1994) Mariposa: A new architecture for distributed data, IEEE Database Engineering. PP. 54-65.

[11] Tamer M. Ozsu & Patrick Valduriez (1999). Principles of Distributed Database Systems. Prentice Hall, 1999.

[12] Tamhankar, AM & Ram,S (1998). Database Fragmentation and Allocation: An Integrated Methodology and Case Study. IEEE Transactions on Systems, Man. and Cybernetics-Part A. Systems and Humans. Vol. 28. No 3. May 1998. PP. 288 - 305.