International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 4, Issue 7(November 2012), PP. 45-52

Vertical Fragmentation, Allocation and Re-Fragmentation in Distributed Object Relational Database Systems-(Update Queries Included)

[1]Surabhi Gupta, [2]Shruti Panda
Computer Science and Engineering, Vellore Institute Of Technology, Vellore, Tamil Nadu, India- 632014.

Abstract- *The efficiency and performance of the distributed systems is mainly determined by the network communications cost between the different sites and execution time of queries. This is achieved by dividing the database into fragments of data and distributing them on various sites. Fragmentation algorithms have been proposed for the relational model, but the object relational data model is not yet implemented. In case of relational database placement of data is comparatively simple. These days Object Oriented Database Management Systems have become very popular, therefore in this paper, an algorithm is implemented for Vertical Fragmentation and Allocation in Distributed Object Database Systems model consisting of simple attributes and simple methods. The key idea of this paper is that it introduces a novel technique that considers re-fragmentation of main database, re-allocation of fragments when it is required and update operation on the server database and corresponding site fragments.*
Keywords- *Distributed Systems, Object Oriented Database, Vertical Fragmentation, Simple Attributes and Simple Methods, Update queries, Re-fragmentation, Re-allocation.*

I. INTRODUCTION

The inception of Distributed Systems took place when storage and management of large amount of data at one site became difficult. Distributed Database Management System governs Storage and processing of logically related data over interconnected computer systems are handled by Distributed Database Management Systems in which data is distributed among several sites. Various advantages of DDBMS are faster data access and processing, reduced operating costs, data is located near "greatest demand" site, reduced danger of single point failure, backup and recovery etc.

Advancement in technology is has made is necessary to handle new types of data such as audio, video, text and graphic data. Examples of such applications are Computer Aided Design and Multimedia Databases. Thus new data models were introduced by database researchers. One of them is object relational model which handles new applications that could not be handled by relational model. Also the object oriented model presents new features such as Inheritance, Encapsulation, Object Identities and Complex Objects and thus requires different techniques for data management.

Fragmentation and distribution of data in distributed database system among different sites has become an important area of research. The fragmentation of data can be either horizontal, vertical or mixed. In Horizontal fragmentation, each row of a relation is assigned to one or more fragments. In Vertical Fragmentation, schema for class is divided into several smaller schemas or subclasses. All schemas should contain a common candidate key (or superkey) to ensure lossless join property. A special attribute row-id attribute should be added to each schema which serves as candidate key. Mixed Fragmentation combines both horizontal and vertical fragmentation.

The problem of data fragmentation has largely been dealt in the relational model, but due to the complex structure of object model the fragmentation of object-relational database is a more difficult and complex process and thus had less attention. In this paper, focus is given on the vertical fragmentation, allocation, re-fragmentation of the object relational database. It has been proved that the performance of a Distributed Object Relational Database System (DORDBS) can be greatly enhanced if the data is stored at local sites in such a manner that many of the user applications running at each site get all needed data at that site without accessing irrelevant data or requiring further communication with other sites. The primary objective of vertical fragmentation in object-oriented databases is to break a class into a set of smaller classes (called fragments). The fragmentation process allows user applications to execute using only one fragment located at local sites which means minimum user application execution time.

List of benefits of fragmentation include: (1) Applications access only portions of classes, thus fragmentation will reduce the amount of irrelevant data accessed by applications. (2)Fragmentation permits parallel execution of a single query by dividing it into a set of sub-queries that operate on fragments of a class. (3) Fragmentation reduces the amount of data transferred when migration is required. (4) Fragmentation replication is efficient than replicating the entire class as it saves storage. (5) Fragmentation is efficient bas it reduces the access time of data. (6) Fragmentation improves usage because the user accesses only uses that data which is useful to him rather than accessing the entire database.

II. BACKGROUND

A. Relational Data Model

Since 1980s, work has been mainly focused on the fragmentation of Relational Database Systems (RDS). Faster query execution and data transfer are the most important factors that affect the performance of distributed database applications. The performance of a DBMS can be improved if adequate proper distribution design including fragmentation, replication and allocation are applied. The relational approach provides two kinds of fragmentation: horizontal and vertical.

Also the hybrid fragmentation is considered another way to partition the data. There are many algorithms developed for horizontal and vertical fragmentation.

B. Object Data Model

Advantages of fragmentation in relational database should also be included in distributed object-relational database (DORDB), too. However, the development of the algorithms are more complicated in object relational database systems. In object-relational databases, different criteria should be considered for fragmentation: hierarchical structure, relationship between classes, usage of attribute of sub class by super class, and so forth.

Vertical fragmentation improves transaction processing cost by decreasing the network communication cost for accessing remote attributes in a distributed environment.

An algorithm is proposed for allocating fragments to the site where fragment had greatest demand. Re-Fragmentation of server database is performed when required. If update queries are entered in any of the sites, it will be redirected to the server. Update operation is performed on the server database and on corresponding site fragments. Execution at sites will be withheld for the duration when update is performed.

C. Vertical Fragmentation – Simple Attributes and Simple Methods

This section presents vertical fragmentation for class model consisting of simple attributes and simple methods.

1. Algorithm for Vertical Fragmentation and Allocation

Input- Text file with a set of queries by users and database.
Output- fragments are found and allocated to suitable sites.
Step 1: User enters a set of queries which consists of both read and update queries.
Step 2: Separation of read and update queries is performed.
Step 3: Method Usage Matrix for each class is formed from the queries.
Step 4: Access frequency matrix of queries for each class is formed for each site.
Step 5: Method affinity matrix is calculated from access frequency and method usage matrix.
Step 6: Clustered method affinity matrix is formed from method affinity matrix by applying bond energy algorithm.
Step 7: Partitions or fragments are obtained by applying partitioning algorithm described below.
Step 8: Partitions are allocated to corresponding sites by allocation algorithm described below.

Method Usage Matrix: (as in [1]) is a matrix which describes the methods(columns) which are used in a query(row). Suppose for query q1 methods m1 and m3 are used, value of m1 and m3 in q1 row will be 1 in method usage matrix and other columns will be zero.

Access Frequency Matrix: (as in [1]) is a matrix which describes the frequency of queries(row) used at sites(columns). For example if q1 is used for 15 times at site 1 the value (q1,s1) will be 15 in access frequency matrix.

Method Affinity Matrix: (as in [1]) is a matrix which describes the affinity between two methods. For example, to calculate affinity between m1 and m2, we should consider all the queries where both m1 and m2 are 1 in method usage matrix and add the access frequency of the corresponding row(of all sites) in access frequency matrix and we should also consider the access frequency of sub-classes where the methods of super-class are used.

2.*Clustered Affinity Matrix (Bond Energy Algorithm)* as in [3]
Input- method affinity matrix
Output - clustered affinity matrix

Initialization – place and fix 1 of the columns of method affinity matrix in clustered affinity matrix
Iteration – place the remaining n-I columns in the remaining i+1 position in the clustered affinity matrix. For each column, choose the placement that makes the most contribution to the global affinity measure
Row order- order the rows according the column order

Contribution of placement
Cont(mi,mk,mj) = 2bond(mi,mk) + 2bond(mk,mj) - bond(mi,mj)

3.Partitioning Algorithm (as in [7],[21])
Input - Order from bond energy algorithm, modified method usage matrix, access frequency matrix
Output – Fragments are obtained

Suppose the order is m4, m1, m3, m2
Algorithm for two fragments:

Consider different cases of two fragments (m4, m1m3m2), (m4m1, m3m2), (m4m1m3, m2).

Calculate split quality for first fragment (VF1), second fragment (VF2), and combined value of VF1 and VF2.

Total split quality for the case is obtained by using the formula VF1*VF2-(combined value*combined value).

The case which has highest total spilt values is considered.
For three fragments: cases (m4, m1, m3m2), (m4, m1m3, m2), (m4m1, m3, m2) . Similarly the split value is calculated and fragments are decided.

Case having the maximum split value gives the desired fragments to be allocated. The fragments are allocated with an extra tuple id to identify each row at the sites. It also helps in re-construction of main database by joining the fragments at sites.

III. IMPLEMENTATION AND ANALYSIS

This section presents an algorithm for allocation of fragments obtained after fragmentation. It also presents technique for re- fragmentation of database if most queries are accessed from server instead of clients and if in case update queries are encountered, they are redirected to server which results in update of both server and corresponding site database. Execution at sites will be withheld for the duration when re-fragmentation as well as update is performed.

A. Allocation Algorithm

Input: Access Frequency Matrix and Method Usage Matrix
Output: Fragments allocated at different sites

Step-1 : Find the maximum value from access frequency matrix and locate the query as well as site for the corresponding maximum value.
Step-2 : Find the methods which have one value in the corresponding row for the query in method usage matrix.
Step-3: Compare the methods collected in step 2 and the methods present in fragments generated by partitioning algorithm.
Step-4 : Allocate the fragment to that particular site if the methods match and initialize the corresponding row of the query and column of the site of access frequency matrix to zero else make that corresponding row of access frequency matrix to zero and find the next maximum value from access frequency matrix
Step-5: Repeat from step 1 to step 5 till all the fragments are allocated.

B. Re-Fragmentation

Fragmentation is based on the access frequency of queries entered by users. The main advantage of performing fragmentation is that, it results in faster execution of queries as output is directly obtained from site than server. Thus total time to get the output reduces as network cost is not included if desired data is present in the site. But fragmentation is performed based on frequency of set of queries entered by the users. Therefore if the user enters different set of queries with different frequency as compared to the previous set of queries, re-fragmentation is required. That is if the users enter a different set of queries with the threshold value more than 40% that is if more than 40% of the queries are executed from server for at least 60% of the sites (Suppose 60 out of 100), a request for re-fragmentation is sent to server by the sites. Re-fragmentation is performed if at-least 60 sites send request for re-fragmentation if at-least 40 queries out of 100 queries are executed from server. Thus re-fragmentation and re-allocation of database is performed which results in better performance. Execution at sites will be withheld for the duration when re-fragmentation is performed which is done by sending a broadcast message to all the sites by server to stop execution and then send a broadcast message to sites to start execution after update operation.

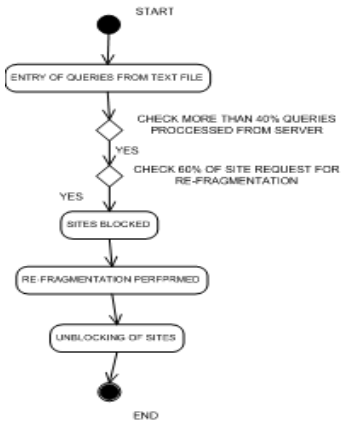


Figure 1 Flow Chart for Re-fragmentation

Threshold value:

One of the major benefits of fragmentation is that major set of queries are accessed from the sites and very few are accessed from the server if not present in the site but if set of queries entered by user changes and most of the queries let us say more than 40% of queries are accessed from server, then there is no use of fragmentation. Thus the limit is termed as threshold value.

C. Update Queries

Fragmentation deals with read queries or data retrieval queries. The user may also want to change the data of server database by executing the write or the update queries. Therefore, change in server database should also be notified to the site databases as when user accesses data from site databases correct data should be retrieved. To ensure the synchronization of server and site databases, when user enters update queries in any of the sites, it will be re-directed to the server.

Update operation is performed on the server database and on corresponding site fragments. Execution at sites will be withheld for the duration when update is performed which is done by sending a broadcast message to all the sites by server to stop execution and then send a broadcast message to sites to start execution after update operation.

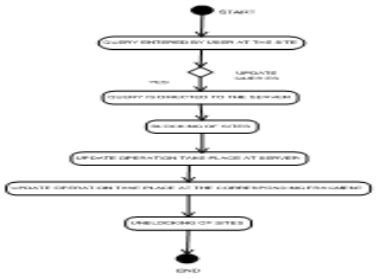
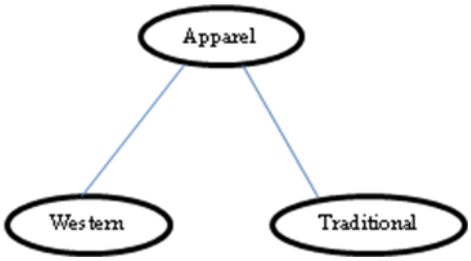


Figure 2 Flow Chart for Update Operation

*D. **Example***

The diagram below shows the parent class "Apparel" which has two child classes "western" dresses and "traditional" dresses.



Apparel class has attributes (dress-id, brand, manufacture-date, cost) and methods (get_dressid(), get_brand(), get_mdate(), get_cost()) designated as m1, m2, m3, m4 respectively.

The set of queries for apparel class are:
Q1: Find manufacture date of apparel where brand="levis";
Q2: List the dress-id, brand of apparel where cost>2000;
Q3: Find the dress-id of apparels where manufacture-date="20-03-2012".

Western class has attributes (western-id, size, style, quantity) and methods (get_westernid(), get_size(), get_style(), get_quantity())

The set of queries for western class are:
Q1: Find the dress-style from western where color="yellow";
Q2: Find the size, color from western where quantity <10 ;
Q3: Find dress-style, quantity from western where western-id="W1";
Q4: Find western-id, manufacture-date from western where size="32";

Traditional class has attributes (traditional-id, culture, instock) and methods (get_traditionalid(), get_culture() , get_instock())

Q1: Find dress-id, manufacture-date from traditional where traditional-id ="T20";
Q2: Find traditional-id, culture, brand of all traditional wear.
Q3: Find traditional-id, manufacture-date if traditional wear is in-stock.

Methods used in Q1 of apparel are m2 and m3. Similarly methods used in Q2 are m1, m2 and m4. Methods used in Q3 are m1 and m3. Thus the resultant method usage matrix for Apparel class is displayed.

Access frequency matrix shows the frequency of the queries entered by user for execution at the sites. Frequency of Q1 at s1 is 10, at s2 is 5 and at s3 is 5.

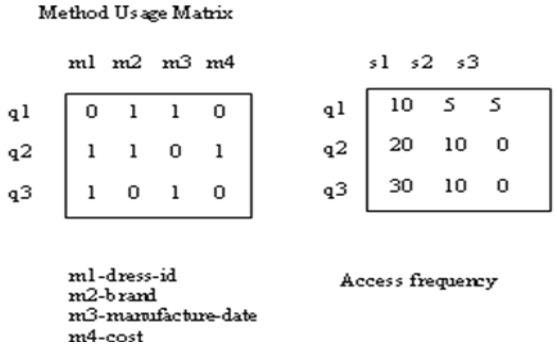


Figure 3 Method Usage Matrix and Access Frequency Matrix of Apparel

Similarly Method Usage Matrix and Access Frequency Matrix for Western and Traditional class is displayed.

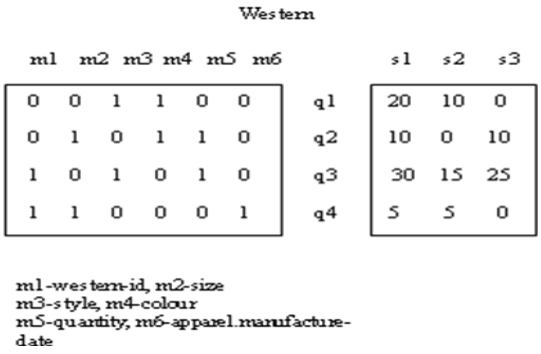


Figure 4 Method Usage Matrix and Access Frequency Matrix of Western

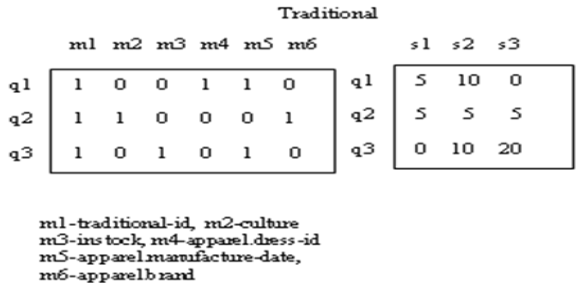


Figure 5 Method Usage Matrix and Access Frequency Matrix of Traditional

Method affinity matrix is displayed and clustered method affinity matrix is displayed by application of Bond-Energy Algorithm.

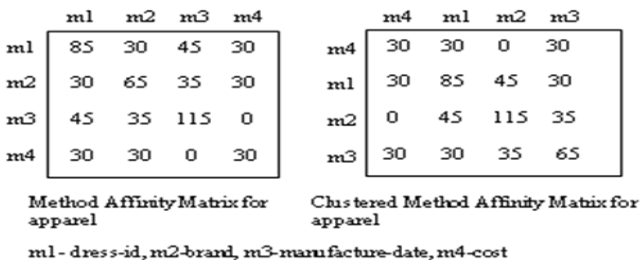


Figure 6 Method Affinity matrix and Clustered Method Affinity Matrix

Fragments are obtained by application of partitioning algorithm. The fragment having the maximum split value is selected. Three cases are formed for two fragments such as case 1: first fragment(m4) second fragment(m1,m3,m2)
case 2: first fragment(m4,m1) second fragment(m3,m2)
case 3: first fragment(m4,m1,m3) second fragment(m2)

Sp value is calculated by application of the formula given below. acc(VF1) is calculated by checking whether only fragment 1 is present in method usage matrix, if present add the access frequency matrix values for the same row. Also check whether that method has been used in child class or not, if used in child class access frequency for that row in child class is also added. Similarly acc(VF2) is calculated. Then combination of VF1 and VF2 is calculated.
Thus sp value= acc(VF1) + acc(VF2) –(acc(VF1,VF2)^2) is calculated.

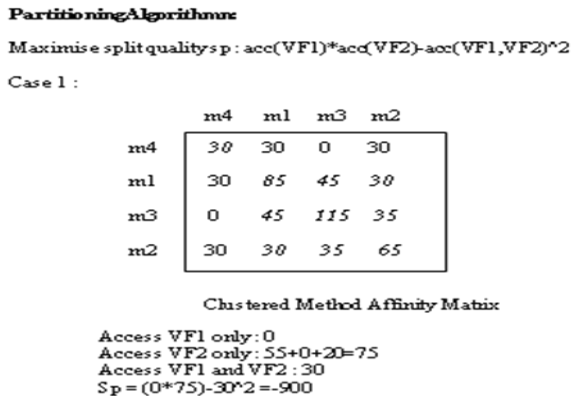


Figure 7 Evaluation of sp value for Case 1 in Partitioning Algorithm

In case of three fragments,
Case 4: VF1(m4,m1) VF2(m3), VF3(m2)
Case 5: VF1(m4) VF2(m1), VF3(m3,m2)
Case 6: VF1(m4) VF2(m1,m3), VF3(m2)
Case 4 has maximum sp value among all the cases. So the fragments are {m4,m1}, {m3}, {m2}

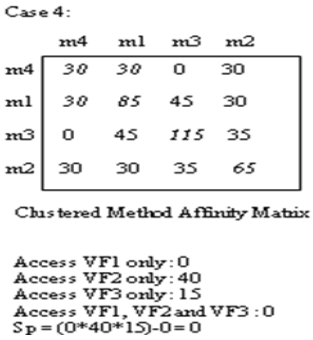


Figure 8 Evaluation of sp value for Case 4 in Partitioning Algorithm

The maximum value from modified access frequency matrix is 30 for row q3 and column s1. The corresponding row in modified method usage matrix has 1 for m1 and m3 and fragment 2 consists of m3 . So fragment 2 along with tuple_id and corresponding attribute is allocated at site1.The row q3 and column s 1 in modified access frequency matrix is made zero and next maximum velue is calculated.

By using allocation algorithm, fragment 1 that is method4 (get_cost()) and method1 (get_dressid()) along with corresponding attributes (dress_id , cost, tuple_id) is allocated at site 2. Similarly method3 (get_mdate()) with attributes (manufacture_date, tuple_id) is allocated at site 1, and method2 (get_brand()) along with attributes (brand, tuple_id) is allocated at site3.

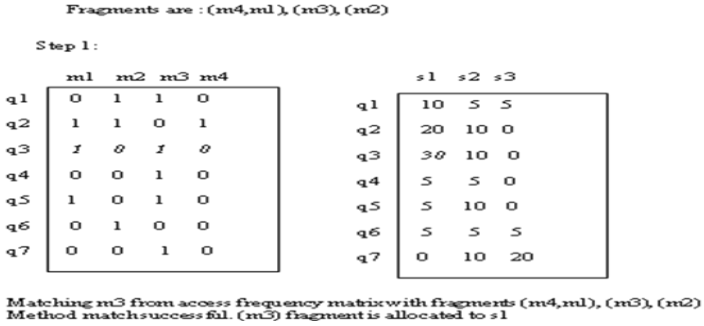


Figure 9 Step 1 of Allocation Algorithm

IV. PERFORMANCE ANALYSIS

With fragmentation in distributed database system, data is distributed on various sites such that data are kept closer where it is needed the most. With this approach, time taken for processing is decreased as the data can be accessed from site databases rather than the server database.

Time taken to obtain the output is evaluated by adding the network communications cost along with query execution cost. Query execution cost is calculated by obtaining the difference between end execution time and start execution time. Network Communications cost is calculated for stand-alone system in this case.

This in turn improves the efficiency and enhances the overall performance of the system. And hence cost of operation is optimized which is determined by processing time.

V. APPENDIX

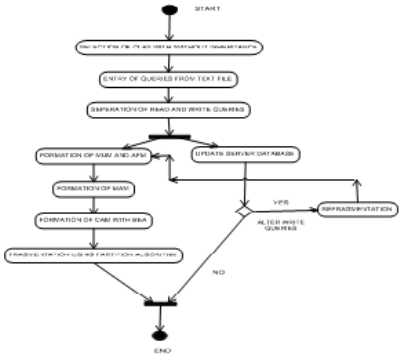


Figure 10 Flow Chart

VI. RESULT AND CONCLUSION

In Distributed Database Systems, Object Relational Model is indeed a generalization of the Relational Model which includes the concepts of Encapsulation and Inheritance. In this project Vertical Fragmentation for Distributed Object Relational Database System with simple attributes and simple methods has been implemented successfully. The top-down approach is followed. The input to the system is the set of queries entered by the users at different sites. Fragmentation is performed based on the frequencies of the queries entered by the users. Fragments are obtained by the application of Bond Energy Algorithm and Partitioning Algorithm. The fragments obtained are allocated to the sites based on the algorithm proposed in the project which allocates the fragments to the sites which has maximum frequency. Processing time for obtaining the output is less for the fragmented data as compared to un-fragmented data. If the users enter a different set of queries and if the threshold value is more than 40% that is if more than 40% of the queries are executed from server for at least 60% of the sites (for example 2 out of 3) , a request for re-fragmentation is sent to server. If update queries are entered in any of the sites, it will be redirected to the server. Update operation is performed on the server database and on corresponding site fragments. Execution at sites will be withheld for the duration when update is performed.

Thus, Fragmentation, Allocation, Re-fragmentation if required and update query has been implemented successfully for a stand-alone system using java as front end and oracle as back-end. Sites are crested using java socket programming. Number of sites is assumed to be three. The sites in which fragments are to be distributed vary from two to three. Distributed Setup is created in the same system using different port numbers.

This project has been implemented for Server-Client Module. It can be also implemented to Peer-to-Peer Module. Only the update queries are dealt with in the project and it can be further extended for the alter and delete queries.

References

[1] Ladjel Bellatreche, Ana Simonet, Michel Simonet,Vertical Fragmentation in Distributed Object Database Systems with Complex Attributes and Methods,IEEE,1996.

[2] Elzbieta Malinowski, Sharma Chakarvarthy, Fragmentation Techniques for Distributing Object Oriented Databases, Chapter in the Book "Conceptual Modeling" in 1997 and Presented in 16th International Conference on Conceptual Modeling Los Angeles , California, USA November 3-5 1997 Proceedings.

[3] M.T. Ozsu, P.Valduriez,Distributed Database System

[4] Mohamed T.Faheen, Amany Sarhan,Fragmentation and Allocation of Object Oriented Database for Simple attributes and complex methods : Cost Based Technique by ,IEEE, December 2005

[5] Ramez Elmarsi, Shamkant B.Navathe, Fundamentals of Database Systems

[6] C.I. Ezeife and Ken Barker ,Distributed Object Based Design : Vertical Fragmentation of classes by , ACM , March 13,1998

[7] Shamkant B.Navathe and Minyoung Ra, Vertical partitioning in database design: graphical algorithm

[8] Kapja Hose and Ralf Schenkel, Distributed database systems : fragmentation and allocation http://www.mpiinf.mpg.de/departments/d5/teaching/ws10_11/dds/slides/DDS-2-print.pdf

[9] David Silberberg , Distributed database system http://hadisusanto.web.ugm.ac.id/semester2/distributed%20dbms/9_Horizontal_Partitioning.pdf

[10] Distributed DBMS concepts and design http://docs.oracle.com/cd/B28359_01/appdev.111/b28371/adobjbas.htm#i456228
[11] Inheritance in SQL Object Types http://docs.oracle.com/cd/B28359_01/appdev.111/b28371/adobjbas.htm#i456228
[12] Oracle 9i Application Developer's Guide-Object Relational Features release 2(2.9) http://docs.oracle.com/cd/B10501_01/appdev.920/a96594/toc.htm
[13] Vertical Fragmentation http://www.scribd.com/doc/55057989/Vertical-Fragmentation-Updated
[14] Embedded PL/SQL http://docs.oracle.com/cd/B14117_01/appdev.101/a97269/title.htm
[15] SQL,PL/SQL and JAVA http://docs.oracle.com/cd/B19306_01/server.102/b14220/sqlplsql.htm
[16] Basic Components of Oracle Object http://docs.oracle.com/cd/B10501_01/appdev.920/a96594/adobjbas.htm
[17] Table Fragmentation http://www.orafaq.com/node/1936
[18] Object type enhancements in Oracle9i Database http://srikanthtechnologies.com/articles/oracle/objecttypes.html
[19] Object Relational Features of Oracle http://infolab.stanford.edu/~ullman/fcdb/oracle/or-objects.html
[20] PL/SQL Object Types http://docs.oracle.com/cd/B10501_01/appdev.920/a96624/10_objs.htm
[21] VerticalFragmentationttp://lsirwww.epfl.ch/courses/dis/2003ws/lecturenotes/week%205%20 Distributed%20Data%20Management%20presentation%20-%20part2%20(1).pdf