American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-5, Issue-12, pp-200-206
www.ajer.org
Research Paper Open Access

# An Object-Oriented Approach for Optimizing Query Processing In Distributed Database System

E. E. Ogheneovo[1], E. Oviebor[2]
[1,2]*Department of Computer Science, University of Port Harcourt, Port Harcourt, Nigeria.*

***Abstract:*** *Query processing in distributed databases involves the transfer of query from one site to another. As a result of this complexity, additional storage space and time may be needed, which could result in cost overhead since cost is often associated with query execution especially in distributed database. These costs could arise from input/output (I/O), CPU used at each site, cost of transferring data between sites, etc. With increasing volume and complexities of data and information been processed in large databases these days, it becomes very paramount to have a query execution strategy and optimization technique for efficient retrieval of information. In order to minimize the response time and resource utilization of systems, it is necessary to optimize query processing in distributed databases. Various techniques have been used in the past but these techniques all require that the user have a good knowledge of the whole data thus making them unsuitable for autonomous distributed databases where nodes are unaware of each other. This paper proposed an object-oriented technique that can be used for processing information with minimum response time and resource usage in distributed databases. The result of our technique actually shows that data can be retrieved with minimal delay.*
***Keywords:*** *Database, distributed database, object-oriented approach, optimization, query processing*

## I. INTRODUCTION

As databases become larger and complex, it becomes increasingly difficult to keep the entire database in a single physical location [1] [2]. Thus there is need to keep them in more than one location or site [3]. This is necessary as a result of a number of issues such as storage capacity, security, performance, technical problems (e.g., disk crash), natural catastrophes, and many other considerations [4]. Consider an organization with several branches scattered around the globe, if the database is centralized (i.e., a single large database), then users will have to connect to the centralized database in order to access data or information irrespective of their distances or geographical locations. Two problems may arise: (i) moving large amount of data may not be feasible or efficient; (ii) secondly, if the centralized database develops problem(s) as a result of technical hitches, users may not be able to access data from the centralized database. Hence, there is need for a distributed database for the organization.

The fundamental part of any DBMS is query processing and optimization [5] [6]. The results of any query have to be made accessible at a time required by the user who submitted the query. Object-Oriented database system [7] [8] [9] which gives more strong data modeling capabilities are usually found in Data shipping technique, whereby, the needed data or information is sent into a client machine which will be worked on, and perhaps stored, within. That is why relational vendors are moving towards integrating object-oriented features into their systems, for example, the emerging SQL3 standard [10]. That is, if a database system which has relational functionalities is later integrated with object-oriented functionalities, the database system will be more efficient and more beneficial.

Query optimization [11] [12] is the most important stage in query processing where the database optimizer has to choose a query-evaluation plan with minimized cost and maximized performance [13] [14] [15] [16]. Query optimization is a complex job in distributed client and server machine while data position becomes a main issue. For us to optimize queries correctly, enough information has to be obtained to ascertain what data access methods are most efficient (for instance, relation and field cardinality, group information, and catalog availability). Optimization algorithms have a significant effect on the operations of distributed query processing. The execution of query in distributed system is seriously subjected to the competence of the optimizer to get effective query evaluation plan. It shows that query optimization is one of the most critical phases in the execution of queries in distributed environment. Query optimization is a difficult task in distributed environment because of numerous factors like data allocation, speed of communication channel, indexing, availability of

memory, size of the database, storage of intermediate result, pipelining, and size of data transmission [17].. The execution plan which is produced by the query optimizer denotes an execution strategy as a minimum cost for any query. The query optimizer is an application unit which accomplishes optimization of query on three essential parts of the plan which are:

Relational models [18] and its offspring are usually built upon the *query shipment or shipping technique*, whereby, most of the queries being processed are carried out in the server's machine, while, vendors that have object-oriented systems are adding more powerful query functionalities. If this is achieved the database system that has an object-oriented functionalities, will be more effective as well, when query facilities is added. Because of these differences, and complementary strengths, it has become apparent that database systems combining the best aspects of the relational and object-oriented approaches are likely to gain acceptance across a large range of applications [19] [20]. This paper proposes an object-oriented technique that can be used for processing information with minimum response time and resource usage in distributed databases.

## II. DISTRIBUTED DATABASE SYSTEM

A distributed is a single logically database that is spread across computers in multiple sites that are connected by a data communications network [21]. That is, a distributed database consists of multiple, logically interrelated databases stored at different computers network sites [22]. Usually, a distributed database appears to a user as if it is centralized but in the real sense, parts of the database are located at different sites [23]. A distributed database is a collection of autonomous computers, connected through a network and distribution middleware, which enables computers to coordinate their activities by sharing the recourses in the distributed database such that users sees the it as a single, integrated computing system [24] [25]. It allows applications or users to access data from local and remote databases. Therefore, a distributed database is truly a database; it is not a loose collection of files as may be perceived.

Figure 1 shows a distributed database in which the network users to share the data available in the databases. For example, a user at one node must be able to access (and perhaps update) data at node. As seen from figure 1, every node usually have its own local DBMS that manages data and information at that node and the DBMS helps to co-ordinate the activities between various nodes and between the server and the client machines. Thus the distributed database even though can be accessed (and perhaps updated) at different locations, it is still centrally administered as a corporate resource while providing local flexibility and customization to users in organizations and enterprises [26].

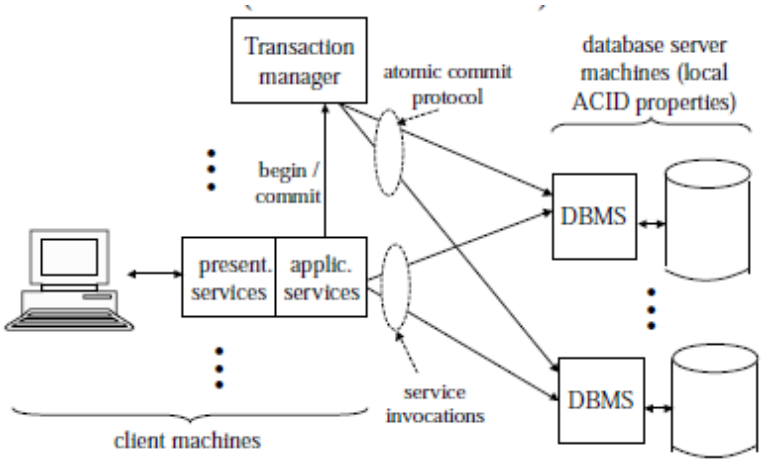


**Fig. 1:** Transaction processing in distributed database architecture

However, it is important to differentiate between a distributed database and decentralized database. Although, a decentralized database is a database that is also stored on computers at different locations; these computers are not interconnected by network and database software that make the data appear in one logical database. Thus in a decentralized database, users at various sites or locations cannot share data. A decentralized database is just a collection of independent databases, rather than having a single database distributed in different geographical locations.

### 2.1 Cost Model

The objective of query optimization in distributed database environment is to minimize the total cost of computer resources. In a logical cost model with cost coefficients is developed for relational systems. The cost function may be stated in regard to the response or total time. Total time involves the local processing overhead

(central processing unit time plus input / output overhead), communication overhead (fixed time to initiate a message plus time to transmit a data). If we reduce the total time, it means there will be an increase in the utilization of the system resources, hence, the throughput of the system increases [27]. The response time is evaluated as the time elapsed between initiation and completion of a query including parallelism [28]. It is the time spent, when a query is submitted, and after execution, in order to get the required result.

Query processing is more complex and difficult in distributed environment in comparison to centralized environment [29] [30]. As large number of parameters affect the performance of distributed queries, relations may be fragmented and/or replicated, and considering many sites to access, query response time may become very high [31] The distributed query optimization has many difficulties concerning the cost model, bigger set of queries, optimization interval, and optimization cost. Primarily, the cost of queries depend on the volume of transitional relations which can be generated in the course of implementation and that can be sent across the net for query processing in several nodes. The importance is the approximation of the volume of intermediate tables centered on join method, and join order, in order to minimize the number of data sent, therefore reduce the total overhead, and total period of distributed query evaluation [32] [33].

However, to ensure that the query processing cost is reduced to minimal level, there is need to minimize the amount of data transmission as shown in figure 2. In this case, the cost of selection operation is defined as:

$$\text{Cost}(S_2) = \log_2(B_R) + \frac{SC(A,R)}{F_R} - 1$$

where $\log_2(B_R)$ = cost of locating the first tuple using binary search,
second term = blocks that contain records satisfying the selection
Thus A is primary key, then SC(A, R) = 1. Hence, $\text{cost}(S_2) = \log_2(B_R)$

## III. METHODOLOGY

The methodology we used in this application is Object-Oriented-Analysis-Design (OOAD). We adopted this methodology because it improves the quality of the system due to program reuse and maintenance. This succeeding application development methodology, includes three features namely: Object Oriented Analysis, that works on the design needs as well as the total architecture of the client machine, which also focuses on explaining what the client machine can do as regard to the key objects within the problem domain; Object Oriented Design, which transform a client machine architecture to programming constructs (that is; classes, method descriptions, and interfaces); and Object Oriented Programming that executes these programming constructs. Figure 2 shows the model for the proposed system. The administrative department is directly linked to the server machine while the employee department is directly linked to the client machine. However, the client and server machines can communicate directly with each other such that the server can provide information needed by the client machine. Both the server and client machines have user interfaces through which user can login to the database and retrieve the needed data or information by querying the databases in each site and the database in the server machine and then display the information in the user interface of client machines.

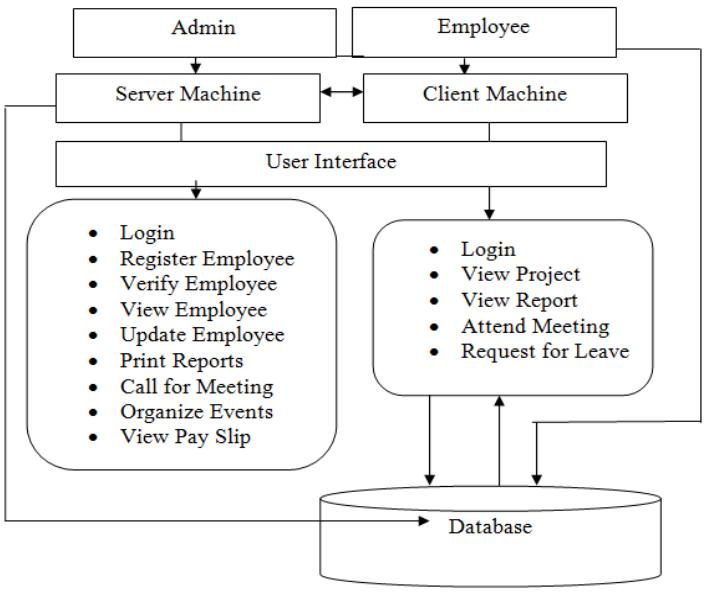


**Fig. 2:** Model of the proposed system

Suppose there is a relation storing information about Employee in a Server located in Delta branch. The Employee relation has the following schema: Db_emp(EmpSno, EmpName, IndustryName, EmpAge). Fields EmpSno and EmpAge are long integer and integer given to the employee serial number and employee age, respectively. Field EmpName and IndustryName are strings given to employee name and industry name in the database server.

Figure 3 shows the class diagram of the proposed system. The class diagram shows the relationship between the employee, the utility, login page, payroll, and the leave. It shows the interconnection between the various classes.

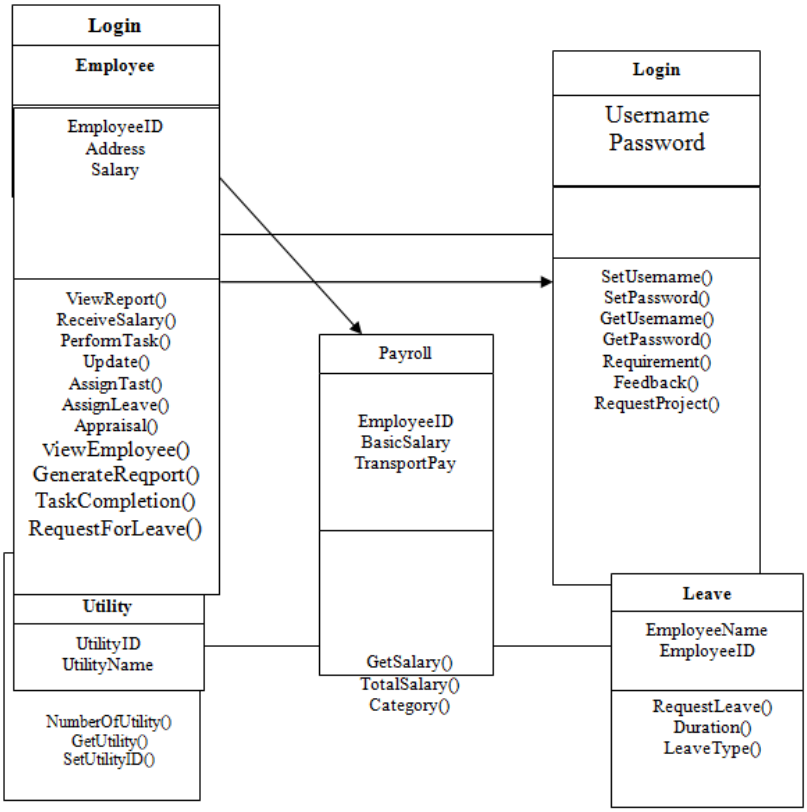


**Fig. 3:** Class diagram of the proposed system

Figure 4 shows the use case diagram of the proposed model. It shows the relationship between the server and client machines, that is, between the administrator and the employee. The client and server machines communicate with each other such that the server provides information needed by the client machine.

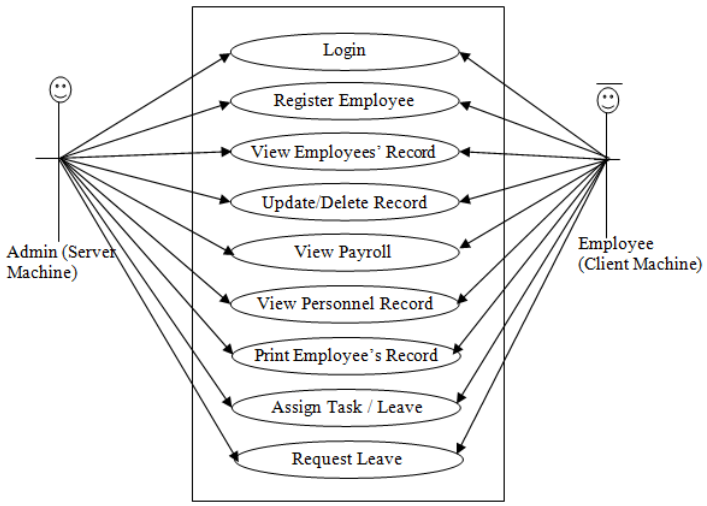


**Fig. 4:** Use case diagram of the proposed model

Both the server and client machines have user interfaces through which user can login and submit their queries to the database and retrieve the needed data or information by querying the databases in each site and the database in the server machine and then display the information in the user interface of client machines. It contain such interfaces such as login, register employee, view employee's record, update/delete record, view payroll, view personnel record, print employee's record, assign task/leave, and request leave. These menus or interfaces help users to interact very well with the database in the server machine by clients or employee.

## IV. RESULTS AND DISCUSSION

This paper proposed an object-oriented technique that can be used for processing information with minimum response time and resource usage in distributed databases. The result of our technique actually shows that data can be retrieved with minimal delay. In this technique, we consider a relation that stores information about Employee in a server located in Delta branch. The Employee relation has the following schema: Db_emp(EmpSno, EmpName, IndustryName, EmpAge). Fields EmpSno and EmpAge are long integer and integer given to the employee serial number and employee age, respectively. Field EmpName and IndustryName are strings given to employee name and industry name in the database server.

Figure 5 shows the result of the proposed technique when query is submitted in the client machine functionalities providing two sets of data $X_1$ and $X_2$ for corresponding values of Y. From the figure, it can be seen that that using the same set of data on the y-axis, different data can be generated for different times. In this case, the different times are represented as $X_1$ and $X_2$.

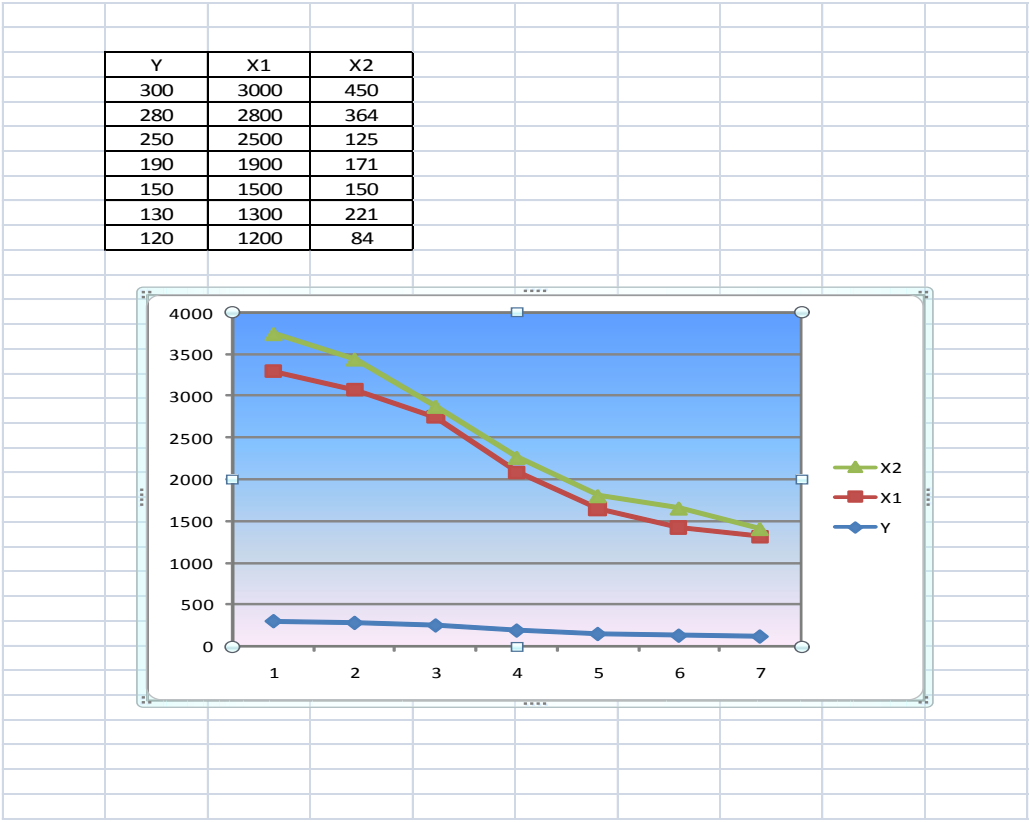


**Fig. 5:** Outputs generated using varying times for optimized queries

Thus with varying times, we can generate different outputs signifying that the query processing can be optimized such that with better optimization, it takes less time to retrieve data from a distributed database. The implication is that we can conclude that retrieval time is inversely proportional to quality of the optimized query. That is, the better optimized a query engine, the lower (i.e., faster) the retrieval time

Figure 6 depicts the volume of disk space spent when the proposed technique is used to optimized data entries in the client system. From the figure, it can be seen that when a query is submitted in the client machine with two sets of data $X_1$ and $X_2$ for corresponding values of Y, it can be seen that using different data sets, different memory sizes are used to store the data as represented in $X_1$ and $X_2$.

| Y | X1 | X2 |
|---|---|---|
| 300 | 16 | 2.4 |
| 280 | 14.9 | 1.9 |
| 250 | 13.1 | 0.7 |
| 190 | 10.1 | 0.9 |
| 150 | 8 | 0.8 |
| 130 | 6.9 | 1.2 |
| 120 | 6.4 | 0.4 |

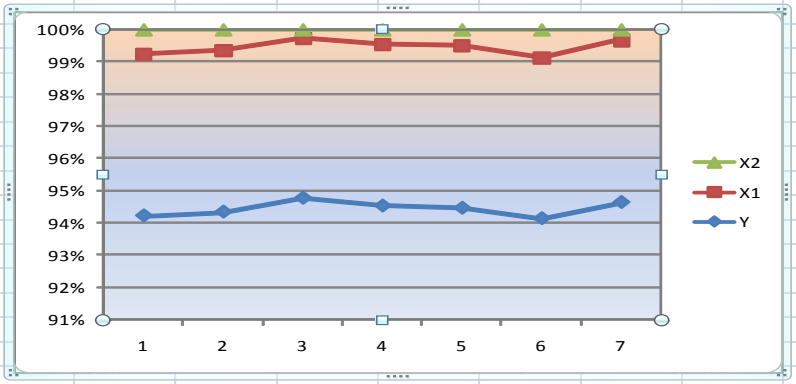


**Fig. 6:** Volume of disk space used optimized queries

Thus the smaller the space occupied by the query, the better the performance of the query processing and the faster the result of a query can be retrieved. The implication is that we can conclude that the smaller the space occupied by a query, the faster the result can be generated.

## V. CONCLUSION

This paper proposed an object-oriented technique that can be used for processing information with minimum response time and resource usage in distributed databases. Query processing in distributed databases involves the transfer of query from one site to another. The situation can even be more complicated especially when a particular site might require certain data from another site before in can execute its query. With increasing volume and complexities of data and information been processed in large databases these days, it becomes very paramount to have a query execution strategy and optimization technique for efficient retrieval of information. In order to minimize the response time and resource utilization of systems, it is necessary to optimize query processing in distributed databases. It has been suggested that selection of the most cost effective solution is needed for query execution. Thus with varying times, we can generate different outputs signifying that the query processing can be optimized such that with better optimization, it takes less time to retrieve data from a distributed database. The implication is that we can conclude that retrieval time is inversely proportional to quality of the optimized query. That is, the better optimized a query engine, the lower (i.e., faster) the retrieval time. Therefore, the smaller the space occupied by the query, the better the performance of the query processing and the faster the result of a query can be retrieved. The implication is that we can conclude that the smaller the space occupied by a query, the faster the result can be generated.

## REFERENCES

[1]. Aljanaby, A. Abuelrub, E. and Odeh, M. (2005). A Survey of Distributed Query Optimization, The International Arab Journal of Information Technology, Vol. 2, No.1, pp. 48-54.

[2]. Amiri, K., Petrou, D., Ganger, G. and Gibson G. (2000). Dynamic Function Placement for Data-Intensive Cluster Computing. In Proceedings of the USENIX Annual Technical Conference, June, 2000.

[3]. Beimel, A., Ishai, Y., and Malkin. T. (2004). Reducing the Servers' Computation in Private Information Retrieval: PIR with Preprocessing. Journal of Cryptology, Vol. 17, Issue 2, pp. 125-151.

[4]. Chaudhuri, S. (1998). An Overview of Query Optimization in Relational Systems. In Proceedings of the ACM Symposium on Principles of Database Systems (PODS'98), Seattle, WA, pp. 34-43.

[5]. Chen, M. and Yu, P. (1990). Using Join Operations as Reducers in Distributed Query Processing. In Proceedings of the 2nd Int'l. Symposium on Databases in Parallel and Distributed Systems (PODS'90), ACM, New York, NY, USA, pp. 116-123. doi: 10.1145319057.319074.

[6]. Ganguly, S., Hasan, W., and Krishnamurthy, R. (1992). Query Optimization for Parallel Execution. In Proceedings of the 1992 ACM-SIGMOD Conference on the Management of Data, Vol. 21, Issue2, June 1, 1992, pp. 9-18, San Diego, CA. doi: 10.1145/141484.130291.

[7]. Hellerstein, J. M. and Stonebraker, M. (1993). Predicate Migration: Optimizing Queries with Expensive Predicates. In Proceedings of ACM SIGMOD Conference on Management of Data, Washington, DC, Vol. 22, Issue 2, June 1, 1993, pp. 267-276. doi: 10.1145/170035.170078.

[8]. Graefe, G.. (1993). Query Evaluation Techniques for Large Databases. ACM Computing Surveys (CSUR), Vol. 25, Issue 2, June 1993, pp. 73-169.

[9]. Gupta, S., Saroba, K. and Bhawna. (2011). Fundamental Research of Distributed Database, Int'l Journal of Computer Science and Management Studies, Vol. 11, No. pp.138-146.

[10]. Keim, D. A., kriegel, H.-P., and Miethsam, A. (1993). Object-Oriented Querying of Existing Relational Databases. In Proceedings of the 4th Int'l Conference DEXA'93, Prague, Czech republic, Sept. 6-8, 1993.

[11]. Ali, M. G. (2009). Object-Oriented Approach for Heterogeneous Databases in a Multi-Database System and Local Schemes Modifications Propagation, Int'l Journal of Computer Science and Information Security, Vol. 6, N0. 2, pp. 55-60.

[12]. Damesha, H. S. (2015). Object-Oriented Database Systems – Concepts, Advantages, Limitations and Comparative Study with Relational Management Systems, Vol. 15, Issue 3, Version 1.0, Global Journal of Computer Science and Technology: C Software & Data Engineering.

[13]. Raiparkar, A. and Bemnote, G. R. (2013). Query Processing in Distributed Database Through Data Distribution, Int'l Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 12, February, 2013, pp. 1134-1139.

[14]. Hameurlain, A., and Moryan, F. (2009). Evolution of Query Optimization Methods, Transactions on Large-Scale Data- and Knowledge Centered Systems 1, ISBN: 978-3-642-03721-4. Doi:10.1007/978-3-642-03722-19.

[15]. Dhande, S. S. and Bamnote, G. R. (2014). Query Optimization in OODBMS: Identifying Subquery for Complex Query Management. Computer Science & Information Technology Journal, pp. 161-177. DOI: 10.5121/csit.2014.4414.

[16]. Tamer, M. and Blakeley, J. A. (2010). Query Processing in Object-Oriented database System. In Proceedings of ACM SIGSOFT Software Engineering Notes, Vol. 35, Issue No. 6, pp..

[17]. Satish, L. and Halawani, S. (2011). A Fusion Algorithm for Joins Based on Collections in Object Database for Rapid Application Development (ORDRA). In Int'l Journal of Computer Science Issues, Vol. 8, Issue 4, No. 2, pp. 289-293.

[18]. Dhande, S. S. and Bamnote, G. R. (2013). Query Optimization of OODBMS: Semantic Matching of Cached Queries Using Normalization. In Proceedings of Int'l Conference on Emerging Research in Computing, Information, communication and Applications, ERICA'13, Elsevier Publication.

[19]. Cybula, P. and Subieta, K. (2010). Query Optimization Through Cached Queries for Object-Oriented Query Language (SBOL). Springer Journal Lecture Notes in Computer Science, Vol. 5901, pp. 308-320.

[20]. Blaja, M. S., Sibieta, K., Fac, K. (2009). Optimization of Object-Oriented Queries: Addressing Large and Small Collections. Int'l Multiconference Computer Science and Information Technology, IMCSIT'09.

[21]. Sassi, M. and Grissa-Tonzi, A. (2005). Contribution to the Query Optimization in the Object-Oriented Databases. Journal of World Academy of Science, Engineering and Technology, WASTE Issue No. 1.

[22]. Zaqaibeh, b. and Al Daoud, E. (2008). The Constraints of Object-Oriented Databases. Int'l Journal of Open Problems in Computer Science and Mathematics, IJOPCM, Vol. 1, No. 1.

[23]. Zicari, R. (2009). A New Renaissance for ODBMSs, ICOODB'09 Conference, Zurich, July 2, 2009.

[24]. Narayann, K. R. S. and Jayanthi, T.(2005). Overview of Object-Oriented Databases. In Proceedings of Conference on Recent Advances in Information Technology, READT'05.

[25]. Dhande, S. S. and Bamnote, G. R. (2012). Query Optimization in Object-Oriented Database through Detecting Independent Subqueries. Int'l Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, Issue 2.

[26]. Bernstein, P. A., Goodman, N., Wong, E., Reeve, C. L. and Rothnie, J. B. (1981). Query Processing in a System for Distributed Databases (SDD-1). ACM Transactions on Database Systems(TODS), Vol. 6, Issue 4, Dec. 1981, pp. 602-625. DOI: 10.1145/319628.319650.

[27]. Reza, S., Carlo, Z., Amir, Z. and Jafar, A. (2001). Optimization of Sequence Queries in Database Systems. In Proceedings of the 20th ACM SIGMOD -SIGACT-SIGART Symposium on Principles of Database Systems, pp.71-81. ISBN: 1-58113-361-8. DOI: 10.1145/37551.375563.

[28]. Su, T. and Ozsoyoglu. G. (1991). Controlling FD and MVD Inference in Multilevel Relational Database Systems. IEEE Transactions on Knowledge and Data Engineering, Vol. 3, Issue 4, pp. 474-485. DOI: 10.1109/69.109108.

[29]. Swami, A. N. and Gupta, A. (1988). Optimization of Large Join Queries in Distributed Database. In Proceedings of ACM-SIGMOD Conference on Management of Data, Vol. 17, Issue 3, June 1985, pp. 3-17, ISBN: 0-89791-268-3. DOI: 10.11.1145/50202.

[30]. Kabra, N. and DeWitt, D. .J. (1998). Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans. In Proceedings of the ACM SIGMOD Conference (SIGMOD'98), Seattle, WA, USA, pp. 106-117.

[31]. Wang, C., and Chen, M. (1996). On Complexity of Distributed Query Optimization, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, Issue 4. DOI: 10.1109/69536256.

[32]. Warshaw, L. B. and Miranker, D. P. (1999). Rule-Based Query Optimization Revisited. In Proceedings of the 8th ACM Int'l Conference on Information and Knowledge Management (CIKM'99), pp. 267-275. ISBN: 1-58113-146-1. DOI: 10.1145/319950.320012.

[33]. Aggarwal, M., Bawa, P., Ganesan, H., Garcia, M., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., and Xu, Y. (2005). Two Can Keep a Secret: A Distributed Architecture for Secure Database Services. In Proceedings of the Int'l Conference on Innovative Data Systems Research, 2005.