Cache Based Query Optimization Approach in Distributed Database

Mantu Kumar[1], Neera Batra[2] and Hemant Aggarwal[3]

**[1] Computer Science and Engineering, Maharishi Markandeshwar University
Mullana(Ambala), Haryana-133207, India**

**[2] Computer Science and Engineering, Maharishi Markandeshwar University
Mullana(Ambala), Haryana-133207, India**

**[3] Infosys Ltd.
Bangalore, Karnataka, India**

Abstract

Query optimization in distributed databases is explicitly needed in many aspects of the optimization process, often making it imperative for the optimizer to consult underlying data sources while doing cost based optimization. This not only increases the cost of optimization, but also affects the trade-offs involved in the optimization process significantly. The leading cost in this optimization process is the cost of costing that traditionally has been considered insignificant. The optimizer can only afford a few rounds of messages to the under-lying data sources and hence the optimization techniques in this environment must be geared toward gathering all the required cost information with minimal communication. In this paper, a cache based query optimization model has been proposed which shows better hit ratio even for the initial queries made since local cache has been used instead of global cache. A cache is implanted between the local optimizer and local database. Whenever a query is given to a local optimizer, local optimizer first checks the cache rather than fetching the data directly from the database. In case, if the solution of query can be obtained from the cache, it results in saving a huge amount of computation time as accessing a cache is faster than accessing the database. The proposed cost optimization model works on the basis of four different factors i.e. server distance, server capacity, server load and current queue length to provide optimal node where query should be executed.

Keywords: *distributed database, cost optimizer, query optimization, cache, local cache, server load.*

1. Introduction

In world of universal dependence on information system, people want to access database from different parts of the world. Company also wants to deploy business worldwide. Due to global business policy, distributed database has become very popular and mostly used worldwide. A distributed database is a database in which storage devices are not all attached to a common processing unit such as the C.P.U. It may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers [1].

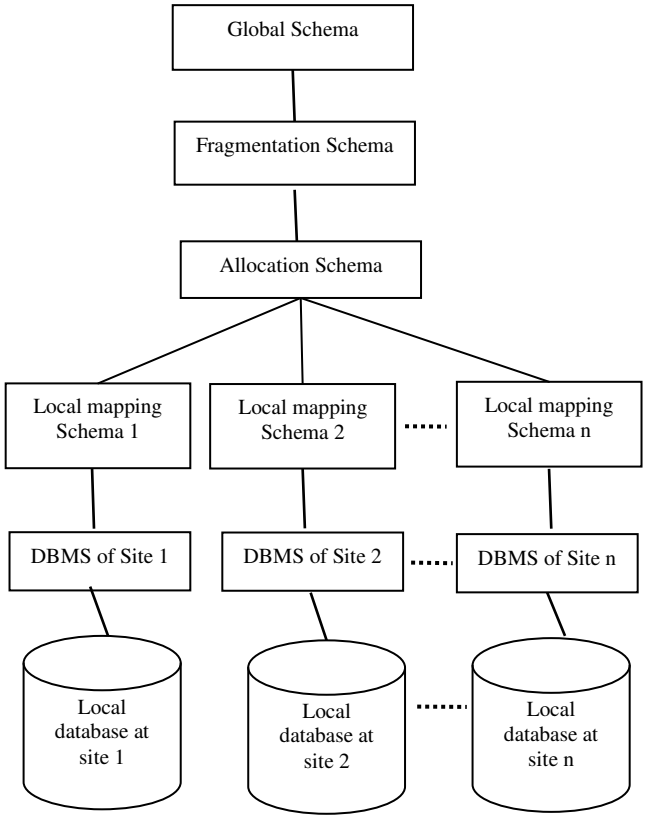

Figure 1. Distributed database system architecture

The management of query processing becomes very complex and its time taking process, so, query processing is a key issue in distributed database system.

Figure 1 shows the collection of data (e.g. in a database) that can be distributed across multiple physical locations. A distributed database can reside on network servers on the Internet, on corporate intranets or extranets or on other company networks. The replication and distribution of databases improves database performance at end-user worksites.

Query optimization is a function of many relational database management systems in which multiple query plans for satisfying a query are examined and a good query plan is identified. This may or may not be the absolute best strategy because there are many ways of doing plans [9][14]. There is a trade-off between the amount of time spent figuring out the best plan and the amount running the plan. Different types of database management systems have different ways of balancing these two factors. Cost based query optimizers evaluate the resource footprint of various query plans and use this as the basis for plan selection [5].

The rest of paper is organized as follows. In section 2, we summarize the literature survey. In section 3, proposed query optimization model is presented followed by cost optimizer in section 4. Finally, in section 5, query optimization algorithm and experimental results are discussed and section 6 concludes the work.

2. Literature Survey

According to Swati Gupta et al.[1] distributed database systems provide an improvement on communication and data processing due to its data distribution throughout different network sites. Not only is data access faster, but a single-point of failure is less likely to occur and it provides local control of data for users [23].

Fan Yuanyuan et al.[2] and Reza Ghaemi et al.[3] emphasized that the search complexity is constantly increasing due to new distributed database applications such as huge deductive database systems and we need better algorithms to speedup traditional database queries. An optimal dynamic programming method for such high dimensional queries has been the big disadvantage of its exponential order and thus we are interested in semi-optimal but faster approaches.

Advantage of using cache in query optimization is to reduce server load for frequently requested contents, reserve server capacity for other non-cacheable request of client/user and capital expenditure as well as operational savings may also be achieved by optimizing server utilization.

D. Kossmann et al.[6] describe how cache investment component can be built and integrated with the query optimizer without changing its basic components such as plan enumeration, search strategy or cost model. S. Adali et al.[4] uses distributed caching which makes network traffic flow less congested and also allows better load sharing along with more fault tolerance. Q. Luo et al. [12] describes DBCaches, which is deployed at application server. When arbitrary SQL statements are generated from the same request by the application that are intended for a backend database server, they can be answered: at the cache, at the backend database server, or at both locations in a distributed manner. The factors that determine the distribution of workload include the SQL statement type, the cache content, the application requirement on data freshness and cost-based optimization at the cache.

3. Proposed Query Optimization Model

To solve the problem of query processing in distributed database systems, a cache based query optimization model has been proposed.

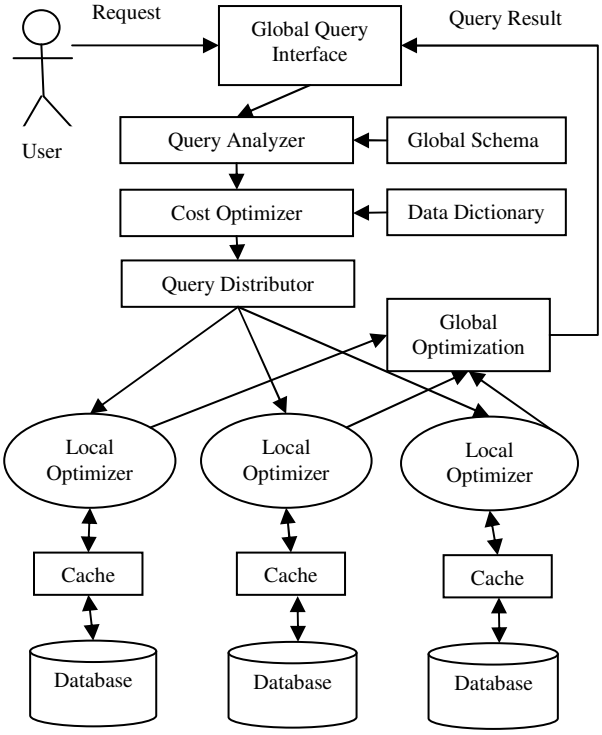

Figure 2. Cache Based Query Optimization in Distributed Database

In this model, cache is implanted between local optimizer and database server where each cache is associated with its corresponding local database server.

Figure 2 shows the proposed cache based query optimization model in distributed database. It has following modules:

- User: A user is an agent, either a human agent (end-user) or software agent who uses a computer or network service. A user often has a user account and is identified by a username (also user name). Other terms for username include login name, screen name (also screen name), nickname (also nick), or handle which is derived from the identical Citizen's Band radio term. Users are also widely characterized as the class of people that use a system without complete technical expertise required to understand the system fully. In hacker-related contexts, such users are also divided into users and power users. In projects in which the actor of the system is another system or a software agent [13], it is quite possible that there is no end-user for the system. In this case, the end-users for the system would be indirect end-users.
- Global Query Interface: Global Query Interface is used to obtain a pointer to another interface, given a Graphical User Interface Database (GUID) that uniquely identifies that interface.
- Query Analyzer: Query analyzer receives the global query from the GUI and consults the Global Schema to analyze the query and decides about the appropriate nodes where requested query result resides.
- Cost Optimizer: One of the hardest problems in query optimization is to accurately estimate the costs of alternative query plans [19][22]. Optimizers cost query plans using a mathematical model of query execution costs that relies heavily on estimates of the cardinality or number of tuples flowing through each edge in a query plan [21]. Cardinality estimation in turn depends on estimates of the selection factor of predicates in the query [15]. Traditionally, database systems estimate selectivity through fairly detailed statistics on the distribution of values in each column, such as histograms.
- Query Distributor: It is responsible for sending the sub queries to appropriate sites so that the actual computation on those sub queries can be carried out [11][16].
- Cache: In computer science, a cache is a component that transparently stores data so that future requests for that data can be served faster [10][24]. The data that is stored within a cache might be values that have been computed earlier or duplicates of original values that are stored elsewhere. If requested data is contained in the cache (cache hit), this request can be served by simply reading the cache, which is comparatively faster. Otherwise (cache miss), the data has to be recomputed or fetched from its original storage location which is comparatively slower. Hence, the greater the number of requests that can be served from the cache, the faster the overall system performance becomes. To be cost efficient and to enable an efficient use of data, caches are relatively small. Nevertheless, caches have proven themselves in many areas of computing because access patterns in typical computer applications have locality of reference. References exhibit temporal locality if data is requested again that has been recently requested. References exhibit spatial locality if data is requested that is physically stored close to data that has been requested already.
- Database: A database is an organized collection of data, today typically in digital form. The data are typically organized to model relevant aspects of reality (for example, the availability of rooms in hotels), in a way that supports processes requiring this information (for example, finding a hotel with vacancies).The term database is correctly applied to the data and their supporting data structures and not to the database management system (DBMS). Users can query the database through the Global User Interface. Queries can be federated queries [25], federated queries are those queries which require access from more than one database. When a Global User Interface, receives a federated query, it first divides the federated query into sub-queries and passes those sub-queries to the appropriate databases. The retrieved result from each database is then integrated and results are shown at Global User Interface.

4. Cost Optimizer

It is the responsibility of cost optimizer to select the appropriate node for processing of the query. Since data is replicated at multiple sites [17], cost optimizer works on the basis of parameters chosen to select the appropriate processing node:

- Current Queue Length: This is the number of processes which are either running or waiting for their turn to get executed on the server.
- Server Distance: Server distance refers to the geographical distance of the server from the requesting client. Nearer will be the server, lesser will be the cost of fetching data from that server.
- Server Capacity: It is the number of processes which can run on a server without hampering the server's normal functioning.

- Load: It is the ratio of actual processes running on a server to the capacity of the server i.e. Load = Current Queue Length / Server Capacity.

The parameters taken into consideration are Queue length, Server Distance and Load. Since queue length is by far the most important factor in determining the waiting time, a process will have to wait for execution in the queue. The proposed model has been designed by prioritizing different parameters taken into consideration for cost optimization. The highest priority has been given to queue length in order to reduce the traffic bottleneck in a replicated environment and subsequently server distance has been given second highest priority in order to reduce the transmission cost incurred. Hence, queue length is divided by factor of 10 whereas server distance divided by factor of 100 in mathematical calculations in order to select the optimal node for computation with the assumptions that queue length will always remain considerably small in size when compared with server distance. So lesser the queue length at a server, less is the cost and subsequently higher will be the response time.

Load refers to the ratio of number of requests served by the server to the total capacity of the requests that can be handled by the server. These factors help to balance the load amongst the servers.

The equations derived for determining the cost of executing a query based upon all the above considered parameters are as follows:

Let there are n number of servers. For each server i and j

$$P_r = P_r + (QL_i - QL_j)/10$$

$$P_r = P_r + (SD_i - SD_j)/100$$

$$L_i = QL_i / SC_i$$

$$L_j = QL_j / SC_j$$

$$P_r = P_r + (L_i - L_j)$$

Where P_r = net priority of server

QL_i - queue length of server i

QL_j - queue length on server j

where $\{0 < i < n\}$ and $\{0 < j < n\}$

SD_i - server distance of server i from requesting node

SD_j -server distance of server j from requesting node

L_i - load on server i

L_j - load on server j

SC_i - Server capacity of server i

SC_j - Server capacity of server j

5. Query Optimization Algorithm

Cache based query optimization approach is based on the fact that there is a cache placed near the database (or server). A request for accessing data is made by the user through User Interface, this request (or query) is analyzed by the query analyzer and the cost of query is optimized by cost optimizer in order to decide about the optimal nodes where sub-queries are to be sent for further processing [18][20]. Query distributor is responsible for further distributing the sub-queries to the appropriate sites. Local optimizer at each site checks whether the results can be returned from cache or the data need be fetched from the database depending upon the fact that a repeated request has been received or a fresh one. If data is fetched from the database then the cache is updated accordingly and results are displayed at User Interface.

5.1 Proposed Algorithm

STEP 1: Input: Query
Output: Optimized Result
STEP 2: site []=sites on which sub query is replicated
Length = site [].length
STEP 3: if (Length>1)
3.1 Start cost optimizer
3.2 Consider the following factors for each site i where sub query result is replicated.
- Current queue length(QL_i)
- Server distance(SD_i)
- Server capacity(SC_i)
- Load(L_i)

3.3 Prioritize all the sites on the basis of value of these parameters using following equations:

Let n be the number of servers for each site i in site []

$$P_r = P_r + (QL_i - QL_j)/10$$

$$P_r = P_r + (SD_i - SD_j)/100$$

$$L_i = QL_i / SC_i$$

$$L_j = QL_j / SC_j$$

$$P_r = P_r + (L_i - L_j)$$

$site_0$ = site with maximum priority
else
$site_0$ = site [0]
end if
STEP 4: send sub query to $site_0$.
STEP 5: check local cache of $site_0$
STEP 6: if (data present in local cache) i.e. at $site_0$
return results from local cache
else
- fetch data from database
- update cache
- return results.

end if
STEP 7: End of Algorithm

5.2 Experimental Results

The experimental evaluation of the proposed model uses sets of PC Memory distributed databases, operating system with windows 7 Server and database management system with MS Access.

Factors like server distance, server queue length and server load have been considered to prioritize the sites for cost estimation. A screenshot of prioritization factors taken into consideration is shown below in Figure 3.

Tabular data

ID	QLength	SerDis	SerCap	Load	Priority
1.0	85.0	100.0	100.0	0.85	0.0
2.0	77.0	200.0	160.0	0.48125	0.16875
3.0	83.0	150.0	130.0	0.6384615384615384	-0.08846153846153842

Figure 3. Prioritization of database site

As shown in the above figure, optimized site is the site with ID 2.0 with best priority shown as 0.16875, so query distributor sends the sub query to site ID 2 for further processing.

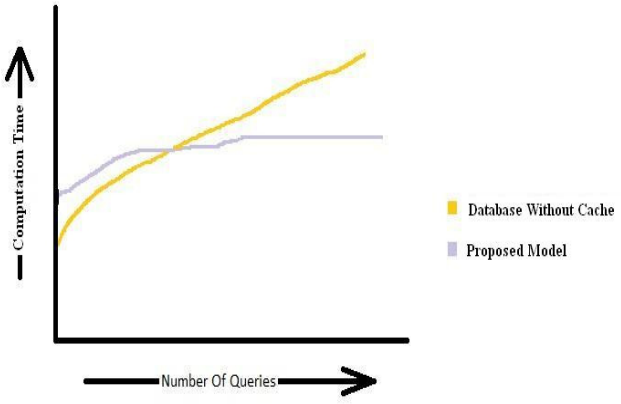

Figure 4. Computation time vs. number of queries

The figure 4 shows how the proposed cache based model is efficient in improving query optimization as compared to existing model which don't use local cache. In the present scenario, replicated database used for evaluating the results of the proposed model when used without cache is considered as an existing model and compared with proposed cache based approach (proposed model).

As figure 4 shows, initially the response time shown by the proposed model is not better than that of when using database system without cache. This is due to the fact that initially cache is empty, data is fetched from the database and also additional time is required to update the cache.

After a certain period of time, as the number of queries increase, computation time shown is almost same in both the cases. This is due to fact that time saved by fetching data from cache is neutralized by the time required to update the cache. Further increase in number of queries doesn't affect the proposed scheme much, since the result sent against most of the requested queries is fetched from the cache and no computation is required. Hence, the proposed technique significantly increases the response time resulting in faster access to data by user.

We have compared the results obtained by us with the Wen-Syan results [24] which are obtained using global cache. Wen-Syan obtained the results based upon two factors: no. of queries and query type. But we take into consideration only no. of queries since query type has little significant when repeated queries are received from users.The comparison of results of our proposed model with that of results obtained by Wen-Syan [2003] has been tabulated in table 1.

TABLE 1 Experimental Results

No. of Query	Cache hit-rate (Wen-Syan[2003])	Cache hit-rate (Proposed Model)
0	0	0
20	0.3	0.6
50	0.5	0.71
60	0.64	0.72
80	0.75	0.75
100	0.8	0.76
120	0.81	0.81
150	0.82	0.83

Table 1 shows the number of queries vs. cache hit rate ratio when the proposed model is implemented and comparison of parametric values obtained is done with the values shown by the cache technique used by Wen-Syan

Li [2003] which is considered as an efficient technique in query optimization.

Figure 5 shows the correlation between the number of queries and the cache hit rates as represented in table 1. As shown in the figure, when 20 queries are selected, the cache hit rate is close to 60%. And, when 100 queries are selected, the cache hit rate is close to 80%.

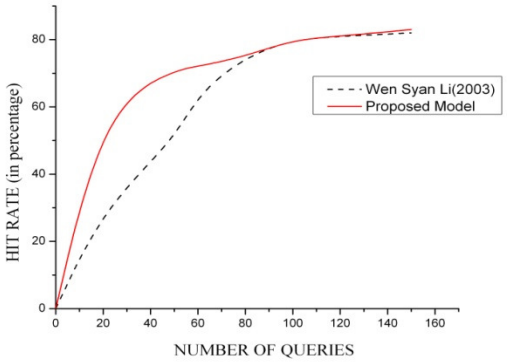

Figure 5. Hit Rate vs. Number of Queries

Wen-Syan Li[2003] has also used the cache concept but it can be concluded from the figure shown that are proposed model has a sharp raise in hit rate as compared to the technique used in Wen-Syan Li[2003] . This can be justified by the fact that, in the proposed technique, database-side cache has been used at each database i.e. cache is local to server while network-based caches are used in Wen-Syan Li [24].

6. Conclusion

Query optimization using cache based approach has proved to be a better option for optimizing queries in homogenous distributed database systems. It can prove to be of great advantage in the fields where access is generally of read only type for in those cases, updating the cache periodically and simultaneously maintaining concurrency with the database are not big issues and also this will increase the computation time.

This model can prove to be useful in situations where users generally access a certain part of database only, which is seen in most of cases i.e. locality of reference. In such cases, cache can save lot of accesses to the database and since accessing the database consumes a lot of time because of seek time and disk latency. Both seek time and disk latency can't be reduced below a certain limit. Hence cache proves to be a great solution as accessing cache is much faster than accessing database.

As for future work, the proposed model can be implemented for heterogeneous distributed database systems. The concept of global caching can also be introduced in the system for better result. The facts like maintaining the concurrency of data in database systems where access is both read as well as write type can also be introduced.

References

[1] Swati Gupta, Kuntal Saroba, Bhawna, "Fundamental Research of Distributed Databse", International Journal of Computer Science and Management Studies, vol. 11, 2011, pp. 138-146.

[2] Fan Yuanyuan, Mi Xifeng, "Distributed Database System Query Optimization Algorithm Research", IEEE international conference on Computer Science and Information Technology, 2011, vol. 8, pp.145-149.

[3] Reza Ghaemi, Amin Milani Fard, Hamid Tabatabee, Mahdi Sadaghizadeh, "Evolutionary Query optimization for Heterogeneous Distributed Database Systems" ,World Academy of Science, Engineering and Technology, vol. 45, 2008, pp. 43-49.

[4] Adali S., Candan K. S., Papakonstantinou Y., Subrahmanian V. S., "Query caching and optimization in distributed mediator systems", ACM SIGMOD, vol. 2, 2006, pp. 45-118.

[5] Yannis. E. Loannidis and Youngkyung Cha Kang, "Randomized Algorithms for optimizing large Join Queries", ACM SIGMOD, 1990, vol. 19, pp. 47-53.

[6] D. kossmann, M. J. Franklin, G. Drasch, "Cache Investment: Integrating Query Optimization and Distributed Data Placement", ACM, vol. 25, 2000, pp. 517-558.

[7] D. Kossman, "The state of the art in distributed query processing" , ACM Computing Surveys, vol. 32, 2000, pp. 422-469.

[8] P. Griffiths, Selinger, M. M. Astrahan, D. D. Chamberlin, R.A. Lorie, T. G. Price, "Access path selection in a rational database management system", ACM SIGMOD, vol. 20, 1979, pp. 23-34.

[9] Stocker, Kossman, Braumandl, Kemper, "Integrating Semi Join Reducers into state of the art query processors", ICDE, 2001, pp. 143-156.

[10] Neera Batra, A . K . Kapil, "Three Tier Cache Based Query Optimization Model in Distributed Database", IJEST, vol. 2, 2010, pp. 3206-3212.

[11] C. Damianos, K. A. Ross, "Partitioned Optimization of Complex Queries", ELSEVIER, vol. 32, 2007, pp. 248-282.

[12] Q. Luo, S. Krishnamurthy, C. Mohan, H. Pirahesh, H. Woo, B. G. Lindsay, J. F. Naughton, "Middle-tier database caching for e-business", ACM SIGMOD, 2002, pp. 600-611.

[13] A. Korzyk, "Towards XML As A Secure Intelligent Agent Communication Language", the 23rd National Information Systems, 2000, pp. 134-138.

[14] Huang, Kuan – Tsae, Davenport, Wilbur B., "Query Processing in Distributed Heterogeneous Systems", MIT Laboratory for information and Decesion Systems, 1981, pp. 1-17.

[15] J. Callan, "Distributed Information Retrieval " , W. B. Croft, Ed. Kluwer Academic Publishers, 2000.

[16] Patricia G. Selinger, Michael E. Adiba, "Access Path Selection in Distributed Database Management Systems", ICOD, 1980, pp. 204-215.

[17] Syam Menon, "Allocating fragments in distributed Database", IEEE Transactions on Parallel and Distributed Systems, vol. 16, 2005, pp. 577-585.

[18] D. Kossmann, K. Stocker, "Iterative Dynamic Programming: A New Class of Query optimization Algorithms", ACM Computing Surveys, vol. 25, 2000, pp. 422-469.

[19] Peng-Yeng Yin, Shiuh-Sheng Yu,Pei-Pei Wang and Ye-Ti Wang, "A Hybrid particle swarm optimization algorithm for optimal task assignment id distributed systems", Computer Standards and Interfaces, vol. 28, 2006, pp. 441-450.

[20] LEE Chiang, CHIH Chi-seng, CHEN Yaw-huei, "Optimizing large join queries using a graph based approach", IEEE Transactions on Knowledge and Data Engineering, vol. 13, 2001, pp. 298-315.

[21] Tsai P S M, CHEN A L P, "Optimizing queries with foreign function in a distributed environment", IEEE Transactions on Knowledge and Data Engineering, vol. 14, 2002, pp.809-824.

[22] S. Pramnaik, D. Vineyard, "Optimization Join Queries in Distributed Database" , IEEE Transaction on Software Engineering, vol. 14, 1998, pp.1319-1326.

[23] Pankti Doshi, Vijay Raisinghami, "Review of Dynamic Optimization Strategies in Distributed Database", IEEE International Conference on Electronics And Computer Technology, 2011, vol. 6, pp. 145-149.

[24] Wen-Syan Li, Oliver Po, Wang-Pin Hsiung, K. Selc_uk Candan, Divyakant Agrawal , "Freshness-driven adaptive caching for dynamic content Web sites" , ELSEVIER, vol. 47, 2003, pp. 269-296.

[25] Wen-Syan Li, Dengfeng Gao, Haifeng Jiang, "Improving parallelism of federated query processing", ELSEVIER, vol. 64, 2008, pp.511-533.

Mantu Kumar is a student of Master of Technology in Computer Science and Engineering, Maharishi Markandeshwar University, Mullana, Ambala, India. He worked as Computer faculty for one year after completion of Master of Computer Application from Madurai Kamaraj University, Mudurai, Tamilnadu, India. He also worked 1.5 years as a software developer and trainer after completion of Bachelor of Computer Application in 2005. He has published one paper in international journal. His research area is query optimization, concurrency control, security in distributed database and network security.

Dr. Neera Batra received PhD in Computer Science & Engineering from Maharishi Markandeshwar University, Mullana, Ambala and Master of Technology in Computer Science & Engineering from Kurukshetra University, Kurukshetra. Dr. Neera Batra is in teaching and Research & Development since 2007. She has supervised several M.Tech thesis. She has published more than 15 research papers in International/National journals and refereed international/national conferences. Her research interests are in concurrency control, query optimization, security, load balancing, check pointing & recovery, access control in distributed database and network security.

Hemant Aggarwal is an employee of Infosys Ltd. Bangalore working as system engineer. He has been working since 2009 after completing B.tech in computer science from Kurukshetra University, Kurukshetra. He has published three papers in international/National journals. His research area is query optimization, security in distributed database and network security.