Implementing New Approach for Enhancing Performance and Throughput in a Distributed Database

Khaled Maabreh[1] and Alaa Al-Hamami[2]
[1]Faculty of Information Technology and Computer Science, Zarqa University, Jordan
[2]Graduate College of Computer Studies, Amman Arab University for Graduate Studies, Jordan

Abstract: *A distributed database system consists of a number of sites over a network and has a huge amount of data. Besides a high number of users use these data. The lock manager coordinates the use of database resources among distributed transactions. Because a distributed transaction consists of several participants to execute over sites; all participants must guarantee that any change to data will be permanent in order to commit the transaction. Because the number of users is increasingly growing and the data must be available all of the time, this research applied a new method for reducing the size of lockable entities to allow several transactions to access the same database row simultaneously, the other attributes remain available to other users if needed. It is possible to do that by increasing the granularity hierarchy tree one more level down at the attributes. The experimental results proved that using attribute level locking will increase the throughput and enhance the system performance.*

Keywords: *Granularity hierarchy tree, locks, attribute level, concurrency control, data availability.*

Received March 16, 2011; accepted July 28, 2011; published online March 1, 2012

1. Introduction

A distributed database system consists of a number of sites connected via a computer network [10] and a large amount of data items. These items maybe requested by a large number of users and must be available to satisfy the user requirements. Solutions to such problems have been discussed in [2, 4, 15]. All are concentrated on a strategy of dividing the database into units or entities. These database units have variable sizes, it maybe the whole database, entire table or the database row, this maybe done dynamically by the lock manager according to the competition of users to the data items, this competition increased in a distributed database because the number of users is extremely bigger than the centralized one. A distributed transaction is a set of operations, in which two or more network hosts are involved [10]. Each host or computer has a local transaction manager responsible for interacting with other transaction managers in case of a transaction does work at multiple computers [1, 2, 13]. When a transaction needs to lock a data item, it sends a request to the central site, which determines if the lock can be granted. If so, it sends a message to the originated site. Otherwise, it will wait.

In case of reading operations, the transaction perform its action from any site which has a copy of the required data item, whereas in a writing case, all sites owning a copy must participate in this action [5, 11]. The concerning of measuring the attribute level locking approach against system performance and throughput and the simplicity in implementation are two factors considered in choosing the central locking approach.

As the study aims, the locking can be done on the part of the row including the key or the index abreast with the attributes needed by the transaction. This can be done by ensuring that no qualification conflicts will occur among the competing transactions. This procedure is expected to satisfy the following:

1. Increase the concurrency, because the same row may be manipulated by more than one transaction at the same time.
2. Reduce the deadlock problem occurrences, because the competing parts are reduced into some attributes instead of the whole row.
3. Increase performance and system throughput, by increasing the number of transactions executed in the system.

The remainder of this paper is organized as follows: section 2 presents the proposed approach and states the problem with using row level locking as a minimum lockable unit. Section 3 presents the enhanced algorithm for field level locking approach; experiments and discussion are drawn in section 4. Section 5 contains the analysis and conclusion.

2. The Proposed Approach

2.1. Approach Description

Because the number of users is increasingly growing and the data must be always available to fit their requirements, this research aims to increase the granularity hierarchy tree [5, 7, 8, 11] one more level down, to include the attribute level, i.e., locking will be done at the attribute level to allow several transactions to access the same row simultaneously. The suggested level is expected to decrease the user competition for acquiring data items which may increase the throughput and the performance of the database. However, this will increase the overhead on the database.

The proof of the enhanced procedure will be given by building a discrete events simulation program to generate transactions randomly after building the hierarchy tree representing the database with new level added (attributes), and by building a database lock manager [1, 14] responsible for coordination transactions execution, the program was built by using Java technology. Data will be gathered to measure system performance, system throughput, and locking overhead [15]. The distributed database in this research, is composed of three sites, logically correlated as shown in Figure 1, each site consists of one database.

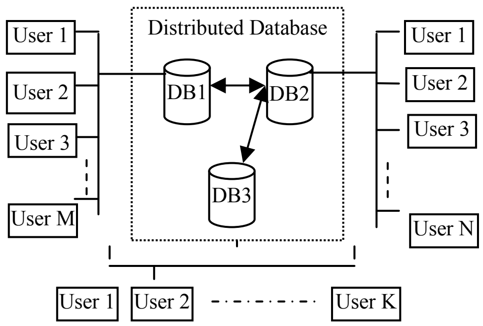

Figure 1. Distributed database architecture for three sites.

Table 1. Simulation parameters.

Parameter	Description	Values
Num-Site	Number of sites	3
DB-Num	Number of databases in each site	1
DB-Obj	Number of database objects	5000
Rep_Deg	Degree of replication	0.2*
Num-Table	Number of tables in a database	15
Num-Trans	Number of transactions in the system	Up to 500
Min-Trans-Size	Minimum number of operation	1
Max-Trans-Size	Maximum number of operation	20
Op-Mod	Operation mode	R,RW,W**
Queue-Length	Maximum queue length	20
Time_Check	Mean time to check a lock	1 ms
Time_Set	Mean time to set a lock	1 ms
Time_Rel	Mean time to release a lock	1 ms
Time_Acc	Mean time to access a data object	20-100ms

*The degree of replication (0.2) is expressed for replication 20% of logical data items over sites [9].
** R, RW and W are shorts for, all the operations of a transaction are Read, mixed of Read and Write or Write, respectively.

According to the system parameters listed in Table 1, there are 15 tables partially replicated over these sites (even in structure), because it is our concern to measure the performance of the system by implementing global transactions (i.e., to make the most of transactions generated by the simulator global). In the sample run for distributed database, the tables distributed over three sites as one dimensional partial replication (some objects to all sites) [9]. The simulation program fills randomly the 15 tables with 5000 database objects (rows), and then it also randomly distributes the tables across the three sites. The parameter named, the degree of replication is considered to replicate the database objects over sites; in this sample, there are 3 out of 15 (0.2*15) tables are replicated as shown in Tables 2 and 3.

Table 2. Distributing database objects into 15 tables.

Table ID	Number of Database Objects
1	500
2	300
3	350
4	420
5	280
6	690
7	280
8	340
9	420
10	220
11	235
12	130
13	275
14	305
15	255

Table 3. Distributing of 15 tables across three sites.

Site 1	Site 2	Site 3
Table 1	Table 1	Table 1
Table 4	Table 2	Table 4
Table 6	Table 3	Table 5
Table 8	Table 4	Table 7
Table 9	Table 10	Table 12
Table 13	Table 11	Table 13
Table 14	Table 13	Table 15

The proposed procedure will execute against the database row as the minimum lockable database unit, and then it will execute to reflect the new added level, comparing between two results will be drawn. The proposed procedure is expected to increase the concurrency, to reduce the deadlock problem occurrences [3], and increase performance and system throughput.

The simulation parameters shown in Table 1 will be used to generate multiple snapshots during progresses of a database, these parameters will vary for each run in order to show the system behaviour. The following assumptions are also considered:

- The time needed for setting and releasing locks is assumed to be 1ms.
- Input output time needed for each operation is assumed to be 1 ms.
- Time needed to complete data processing is randomly selected between 20 to 100 ms.

- Communication delay is assumed to be negligible because the communication performance is not considered to be measured here.

Read and writes sets in a transaction, are assumed to be equals, because of simplifying the analysis and we did not have an actual data that could serve as an indication of what would be a realistic distribution of the size of the read or write sets.

2.2. Deadlock Detection by Timeout

A transaction sets a time out for every lock required, if the lock is not granted within this time, it assumes that the deadlock has occurred. The simplicity and ease of implementation are two reasons for using this method, in addition it does not cause network traffic when detecting deadlock in distributed database, while the timeout must be tuned carefully in order to not detect false deadlocks or to not allow the deadlock to persist in the system for a long time [6].

In this study, the check for an available resource is assumed to take one millisecond, if the lock is not granted immediately, one millisecond is needed before the next trial, when the lock is granted, a random number between 20 and 100milliseconds is chosen as a processing time, (because we don't have real data), so 51 trials for acquiring a lock is sufficient in this study to determine if the resource is blocked or deadlocked. Because if a transaction is granted a lock to a resource and needs 100milliseconds to complete its operation at the resource, then after completion, one millisecond is needed to release a lock, another transaction may try 51 times to get a lock at the same resource with one millisecond between each two successive trials, so it needs 102millisecond which exceeds the total time for the first transaction by one, so in the case of not granted a lock, deadlock has occurred.

3. The Enhanced Algorithm Description for Locking Attributes

The locking could be obtained on the entire database, entire table, page, row or attribute according to compatibility matrix for granularity hierarchy Table 4. The transaction can lock a node in top-down order and unlock in bottom-up order by using the rules mentioned in [11] in addition to:

1. The database row is considered as a node, and can be locked in an intention modes (IS or IX).
2. The key of the row must be locked in a Shared (S) mode, when the transaction does not need the whole row.
3. The locking of attributes as database nodes must be done according to database constraints.
4. Other attributes can be locked in S or X mode.

When a conflict occurs, or when the transaction needs to read or update the whole row, it's locked as in row level locking.

Table 4. Compatibility matrix.

	IS	IX	S	SIX	X
IS	T	T	T	T	F
IX	T	T	F	F	F
S	T	F	T	F	F
SIX	T	F	F	F	F
X	F	F	F	F	F

The abbreviations S, X, IS, IX and SIX are stated for: shared locks (Read), exclusive locks (Write), intention-shared (i.e., explicit locking is being done at lower level of the tree with shared mode locks), intention-exclusive (i.e., the explicit locking will be used at a lower level of the tree with exclusive mode or shared-mode locks) and shared with intention-exclusive (i.e., the sub tree rooted by that node is locked explicitly in shared mode and explicit locking is being done at lower level with exclusive-mode) respectively [5, 11].

Databases are assumed to be well normalized and have a set of assertions to satisfy its correct state [12], for example, if a database has associates with such assertion (Z=X+Y). So, these items must be locked together when using attribute level locking, this is the responsibility of a database lock manager to accomplish this task, in this example case, the transaction must lock both X and Y when it needs to lock Z .

4. Experimental Work

4.1. Performance Evaluation of Row Level Locking

The results shown in Table 5 are presented to appear the behavior of the system during 25 runs, (times are measured in seconds), we can see that, the system begin thrashes when the number of transactions entering the system becomes 150 or higher as shown in Figure 2.

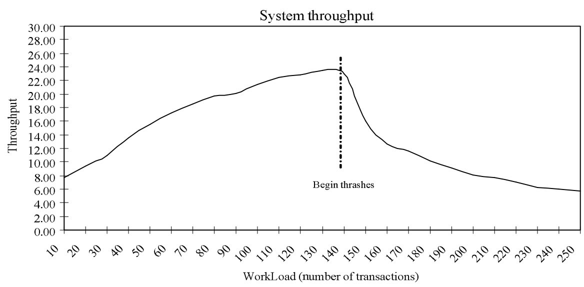

Figure 2. System throughput at row level locking.

Which means that, the system does not complete all transactions entering the system, (i.e., the competition among transactions as well as the probability of conflict becomes high), and this means that the throughput is affected?

Table 5. Results of 25 runs of simulation at row level locking.

Number of Transactions	Completed Transactions	Simulation Time	Mean Service Time	Mean Waiting Time	Mean Number of Operations	Mean Number of Locks	Arrival Rate	Throughput
10	10	1.297	0.699	0	8	25	7.71	7.71
20	20	2.144	0.891	0.1221	10	28	9.33	9.33
30	30	2.75	0.987	0.2981	9	26	10.91	10.91
40	40	2.956	1.022	0.3876	8	29	13.53	13.53
50	50	3.219	1.127	0.4243	8	28	15.53	15.53
60	60	3.485	1.169	0.4548	8	29	17.22	17.22
70	70	3.79	1.212	0.4702	9	27	18.47	18.47
80	80	4.069	1.234	0.5101	8	27	19.66	19.66
90	90	4.469	1.302	0.5231	6	29	20.14	20.14
100	100	4.678	1.359	0.5871	9	31	21.38	21.38
110	110	4.912	1.421	0.6204	7	31	22.39	22.39
120	120	5.247	1.531	0.7299	9	31	22.87	22.87
130	130	5.531	1.591	0.8626	8	32	23.50	23.50
140	140	6.112	1.728	0.9241	8	31	22.91	22.91
150	148	9.202	2.233	1.4324	8	30	16.30	16.08
160	154	12.214	3.691	2.814	8	29	13.10	12.61
170	164	14.203	3.981	3.021	6	28	11.97	11.55
180	170	16.782	4.117	3.394	8	29	10.73	10.13
190	178	19.469	4.4	3.697	7	32	9.76	9.14
200	181	22.563	4.404	3.962	10	31	8.86	8.02
210	186	24.204	4.527	4.178	7	33	8.68	7.68
220	195	27.641	4.806	4.436	8	32	7.96	7.05
230	199	31.719	5.135	4.406	7	31	7.25	6.27
240	207	34.859	5.312	5.222	7	29	6.88	5.94
250	210	36.36	6.24	6.508	7	32	6.88	5.78

Mean service time mean service time and mean waiting time as shown in Figure 3, increased when the number of transactions entering the system increased because the system workload increased.

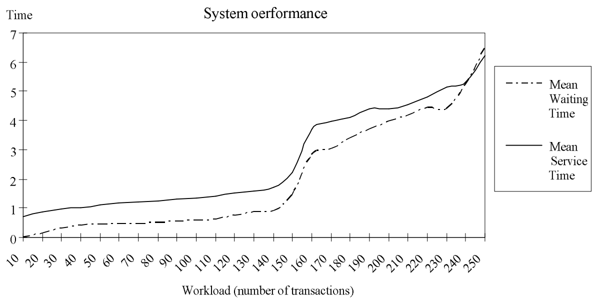

Figure 3. System performance at row level locking.

Figure 4 shows the mean number of locks needed by transactions at row level locking because it depends on the mean number of operations that the transactions need.

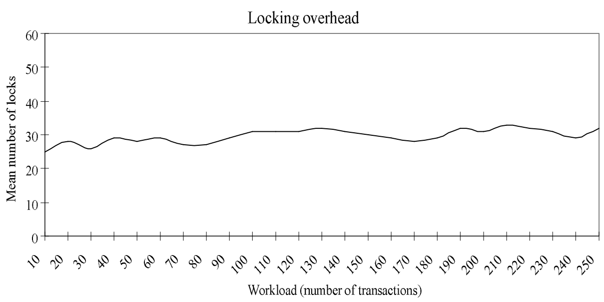

Figure 4. System locking overhead at row level locking.

4.2. Performance Evaluation of Field Level Locking

After modifying the hierarchy tree by adding the attributes level to be locked, simulation is executed 25 times on different workloads to show the system behaviour, the results are presented in Table 6. The new system (alternative two) executes up to 190 transactions successfully without deadlock. When the number of transactions becomes 200 or higher, the system begins thrashes as shown in Figure 5.

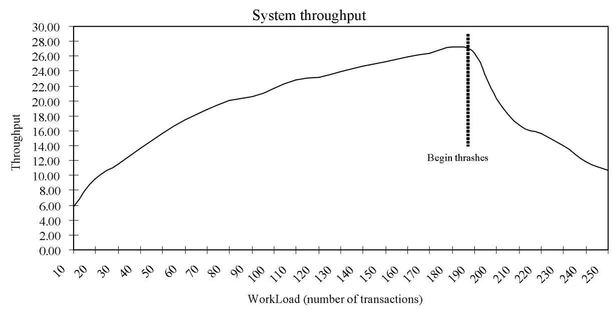

Figure 5. System throughput at field level locking.

The important thing is that 150 transactions are completed successfully on alternative two (at field level locking), while there are two transactions were deadlocked, when using the row as minimum lockable unit mean service time and mean waiting time on alternative two becomes less than those produced when using alternative one. Figure 6 shows this behaviour because the transaction does not need to waits for long time to get its lock. But unfortunately, the mean number of locks increased as shown in Figure 7.

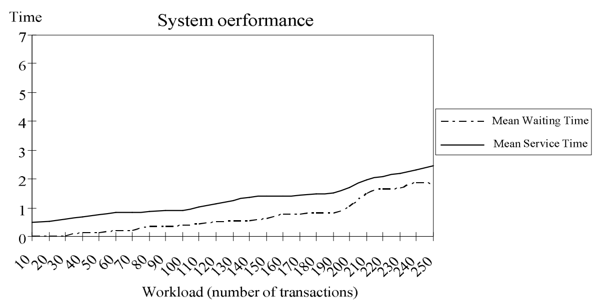

Figure 6. System performance at field level locking.

Table 6. Results of 25 runs of simulation at field level locking.

Number of Transactions	Completed Transactions	Simulation Time	Mean Service Time	Mean Waiting Time	Mean Number of Operations	Mean Number of Locks	Arrival Rate	Throughput
10	10	1.713	0.492	0	8	35	5.84	5.84
20	20	2.105	0.512	0.0031	10	36	9.50	9.50
30	30	2.609	0.613	0.0123	9	38	11.50	11.50
40	40	2.934	0.696	0.0985	8	41	13.63	13.63
50	50	3.202	0.76	0.1223	8	41	15.62	15.62
60	60	3.437	0.816	0.1876	8	42	17.46	17.46
70	70	3.714	0.835	0.2068	9	40	18.85	18.85
80	80	3.991	0.885	0.3482	8	42	20.05	20.05
90	90	4.361	0.893	0.3527	6	41	20.64	20.64
100	100	4.612	0.914	0.3621	9	43	21.68	21.68
110	110	4.835	1.02	0.4102	7	42	22.75	22.75
120	120	5.177	1.125	0.4863	9	44	23.18	23.18
130	130	5.429	1.231	0.5361	8	44	23.95	23.95
140	140	5.678	1.369	0.5422	8	46	24.66	24.66
150	150	5.922	1.387	0.6101	8	45	25.33	25.33
160	160	6.188	1.402	0.7512	8	45	25.86	25.86
170	170	6.429	1.454	0.7723	6	43	26.44	26.44
180	180	6.612	1.491	0.7856	8	44	27.22	27.22
190	190	7.181	1.522	0.7902	7	45	26.46	26.46
200	198	9.736	1.712	1.0125	10	49	20.54	20.34
210	200	11.914	1.979	1.4701	7	49	17.63	16.79
220	206	13.204	2.081	1.6164	8	50	16.66	15.60
230	215	15.345	2.189	1.6731	7	47	14.99	14.01
240	220	18.631	2.295	1.8697	7	48	12.88	11.81
250	227	21.241	2.441	1.8341	7	46	11.77	10.69

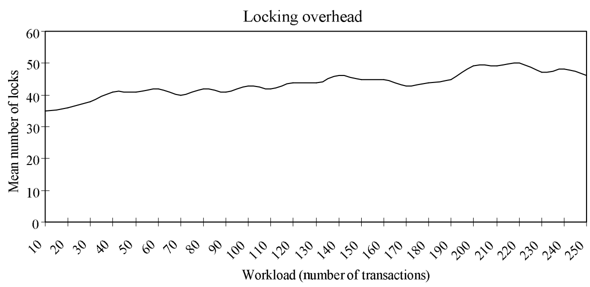

Figure 7. System locking overhead at field level locking.

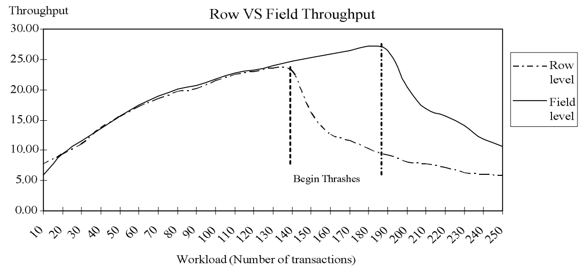

Figure 8. Throughput for the two alternatives.

4.3. Comparing the Two Alternatives

Table 7, shows mean service time, mean waiting time, throughput and the mean number of locks for the two alternatives in order to compare between them.

The throughput for field level locking is higher than for row level locking as shown in Figure 8, because the competitions among transactions becomes less due to increasing in a database size (i.e., the number of transactions that are completed successfully is higher than at an alternative one). Alternative one (row level locking) becomes thrashes before alternative two (field level locking). At the same time, the mean service time and mean waiting time in alternative two becomes less in general as shown in Figures 9 and 10, because transactions can proceed immediately when no conflicts occurs. Figure 11 shows the increasing of locking overhead, because at field level locking approach, the lock manager needs extra work to manage the locks needed, especially when transactions need many attributes in the same row. It can be reduced by returning one level up on the hierarchy tree -at row level- when transactions need many attributes.

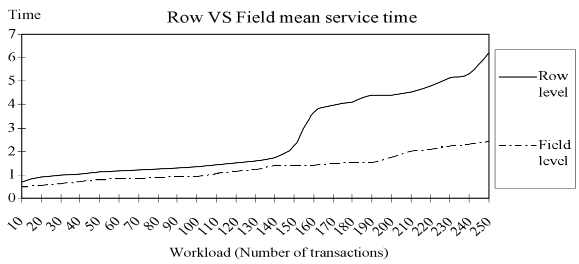

Figure 9. Mean service time for two alternatives.

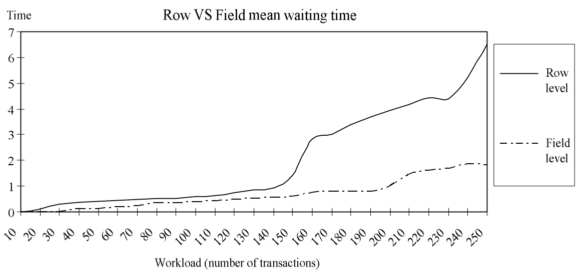

Figure 10. Mean waiting time for two alternatives.

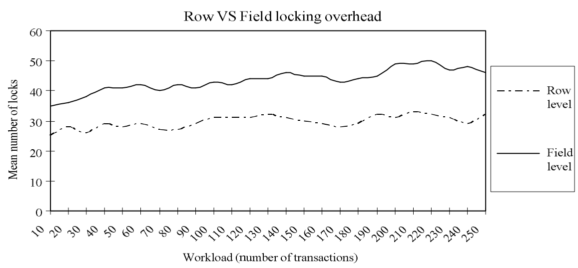

Figure 11. Locking overhead for two alternatives.

Table 7. Row level locking versus field level locking performance.

Number of Transactions	Row level locking				Field level locking			
	Mean Service Time	Mean Waiting Time	Throughput	Mean Number of Locks	Mean Service Time	Mean Waiting Time	Throughput	Mean Number of Locks
10	0.699	0	7.71	25	0.492	0	5.84	35
20	0.891	0.1221	9.33	28	0.512	0.0031	9.50	36
30	0.987	0.2981	10.91	26	0.613	0.0123	11.50	38
40	1.022	0.3876	13.53	29	0.696	0.0985	13.63	41
50	1.127	0.4243	15.53	28	0.76	0.1223	15.62	41
60	1.169	0.4548	17.22	29	0.816	0.1876	17.46	42
70	1.212	0.4702	18.47	27	0.835	0.2068	18.85	40
80	1.234	0.5101	19.66	27	0.885	0.3482	20.05	42
90	1.302	0.5231	20.14	29	0.893	0.3527	20.64	41
100	1.359	0.5871	21.38	31	0.914	0.3621	21.68	43
110	1.421	0.6204	22.39	31	1.02	0.4102	22.75	42
120	1.531	0.7299	22.87	31	1.125	0.4863	23.18	44
130	1.591	0.8626	23.50	32	1.231	0.5361	23.95	44
140	1.728	0.9241	22.91	31	1.369	0.5422	24.66	46
150	2.233	1.4324	16.08	30	1.387	0.6101	25.33	45
160	3.691	2.814	12.61	29	1.402	0.7512	25.86	45
170	3.981	3.021	11.55	28	1.454	0.7723	26.44	43
180	4.117	3.394	10.13	29	1.491	0.7856	27.22	44
190	4.4	3.697	9.14	32	1.522	0.7902	26.46	45
200	4.404	3.962	8.02	31	1.712	1.0125	20.34	49
210	4.527	4.178	7.68	33	1.979	1.4701	16.79	49
220	4.806	4.436	7.05	32	2.081	1.6164	15.60	50
230	5.135	4.406	6.27	31	2.189	1.6731	14.01	47
240	5.312	5.222	5.94	29	2.295	1.8697	11.81	48
250	6.24	6.508	5.78	32	2.441	1.8341	10.69	46

5. Conclusions

Simulation is implemented to prove the idea of obtaining a lock at attributes level on a distributed database. The discussion presented in sections 4.1 through 4.3, shows that the system at field level locking behaves better than at row level locking because multiple transactions can proceed at the same database row simultaneously, which decreases the mean service time as well as the mean waiting time because transactions does not need to wait for a long time to get their locks, which increases the availability of data. Also alternative two executes more transactions than alternative one at a time unit before thrashing occurs, which means that more transactions are completed successfully than alternative one which means higher throughput obtained, it is due to the increasing of database size by attribute level.

The increasing of overhead that occurs in alternative two can be managed by choosing the appropriate granule size for each transaction because the approach implemented here is suitable for the applications that have mixed size of transactions (short and long). It can be reduced by returning one level up on the hierarchy tree to be at the row level when transactions need many attributes.

Acknowledgment

This research is funded by the Deanship of Research and Graduate Sudies in Zarqa University/Jordan.

References

[1] Bernstein P. and Newcomer E., *Principles of Transaction Processing for the Systems Professional*, Bentham Press, 2004.

[2] Chandy K., Misra J., and Hass L., "Distributed Deadlock detection," *ACM Transactions on Computer Systems*, vol. 1, no. 2, pp. 144-156, 1983.

[3] Coffman E., Elphick M., and Shoshani A., "System Deadlocks," *ACM Computing Surveys*, vol. 3, no. 2, PP. 67-78, 1971.

[4] Croker A., "Improvements in Database Concurrency Control with Locking," *Journal of Management Information Systems*, vol. 4, no. 2, pp. 74, 1987.

[5] Elmasri R. and Navathe S., *Fundamentals of Database Systems*, Pearson Addison Wesley, Boston, 2010.

[6] Krivokapi N., Kemper A., and Gudes E., "Deadlock Detection in Distributed Database Systems: A New Algorithm and a Comparative Performance Analysis," *The VLDB Journal*, vol. 8, no. 2, pp. 79-100, 1999.

[7] Maabreh K. and Hamami A., "Increasing Database Concurrency Control Based on Attribute Level Locking," *in Proceedings of International Conference on Electronic Design,* Penang, pp. 1-4, 2008.

[8] Maabreh K. and Hamami A., "Applying Attribute Level Locking to Decrease the Deadlock on Distributed Database," *in Proceedings of the 11th International Arab Conference on Information Technology*, Libya, 2010.

[9] Matthias N. and Matthias J., "Performance Modeling of Distributed and Replicated Databases," *IEEE Transactions on Knowledge*

Data Engineering, vol. 12, no. 4, pp. 645-672, 2000.

[10] Ozsu T. and Valduriez P., *Principles of Distributed Database Systems*, Springer Science and Business, New York, 2011.

[11] Silberschatz A., Korth H., and Sudarshan S., *Database System Concepts*, McGraw-Hill, New York, 2010.

[12] Sinha M. "Constraints: Consistency and Integrity," *ACM SIGMOD Record*, vol. 13, no. 2, pp. 60-63, 1983.

[13] Taibi T., Abid A., Jiann W., Fei Y., and Ting C., "Design and Implementation of a Two-Phase Commit Protocol Simulator," *The International Arab Journal of Information Technology*, vol. 3, no. 1, pp. 20-27, 2006.

[14] Weikum G. and Vossen G., *Transactional Information Systems, Theory, Algorithms and the Practice of Concurrency Control and Recovery*, Morgan Kaufman Publishers, USA, 2002.

[15] Wu H., Chin W., and Jaffar J., "An Efficient Distributed Deadlock Avoidance Algorithm for the AND Model," *IEEE Transactions on Software Engineering*, vol. 28, no. 1, pp. 18-29, 2002.

Khaled Maabreh is a lecturer in computer information system at Zarqa University, Jordan. He holds PhD degree in computer science from Amman Arab University for Graduate Studies in 2008. He has more than 17 years of experience including developing IT-related projects. He also teaches different courses at BSc level in computer science and computer information systems.

Alaa Al-Hamami is senior lecturer in computer science. He holds a BS in physics from Baghdad University, in 1970, MSc in computer science from Loughborough University, England in 1979, and a PhD degree in computer science-database security from the University of East Anglia-England in 1984. He has a membership in many different scientific societies including ACM and IEEE. He is a deanship of Graduate College of Computer Studies-Amman Arab University for Graduate Studies. He has more than 31 years of experience including extensive project management experience in planning and leading a range of IT-related projects in addition to management posts. He has more than 117 published papers in different indexed journals. He supervises more than 50 PhD and MSc students in computer science, information management and integration, security, and knowledge management. He also leads and teaches modules at BSc, MSc and PhD levels in computer science and security.