iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** I **Month of publication:** January 2018

DOI: http://doi.org/10.22214/ijraset.2018.1194

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor :6.887
Volume 6 Issue I, January 2018- Available at www.ijraset.com

An overview of Concurrency Control Techniques in Distributed Database

Rajdeep Singh solanki[1]
[1]*Research scholar*

Abstract: ***Today's business environment has an increasing need for distributed database and client/server applications as the desire for reliable, scalable and accessible information is steadily rising. 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 also a single-point of failure is less likely to occur, and it provides local control of data for users. However, there is some complexity when attempting to manage and control distributed database systems.***
Concurrency control is an integral part of a database system. Devising a concurrency control technique that has a low lost opportunity cost and a low restart cost is a hard problem. The interconnection network in a distributed database system can act as a powerful coordination mechanism by providing certain useful properties. We identify several such useful network properties, and present a new family of concurrency control techniques that are built on top of these properties. Concurrency control techniques use network properties to keep the lost opportunity cost and restart cost low. Our thesis is that network properties can be exploited to achieve efficient concurrency control of transactions.

I. INTRODUCTION

In the past, implementation of distributed database systems was deemed impractical because
network technology was either too unreliable or immature to be used and because computers were too expensive to be implemented in large numbers. However, as networks have become more reliable and computers have become much cheaper, there has been a large interest to use distributed database systems. There are five big reasons for using a distributed database system:
Many organizations are distributed in nature.
Multiple databases can be accessed transparently.
Database can be expanded incrementally – as needs arise, additional computers can be connected to the distributed database system.
Reliability and availability is increased – distributed database can replicate data among several sites. So even if one site fails, redundancy in data will lead to increased availability and reliability of the data as a whole.
Performance will increase – query processing can be performed at multiple sites and as such distributed database systems can mimic parallel database systems in a high-performance network.
Even with these benefits, distributed database systems have not been widely used because of many problems in designing distributed database management system (DDBMS). These problems arise when designed distributed DBMS as designing traditional DBMS: distributed DBMS also have to consider how to do query optimization and concurrency control but distributed DBMS requires different solutions because of its nature. Distributed database systems (DDBS) are systems that have their data distributed and replicated over several locations; unlike the centralized data base system (CDBS), where one copy of the data is stored. Data may be replicated over a network using horizontal and vertical fragmentation similar to projection and selection operations in Structured Query Language (SQL). Both types of database share the same problems of access control and transaction management, such as user concurrent access control and deadlock detection and resolution.

A. Advantages of Distributed DBS

Since organizations tend to be geographically dispersed, a DDBS fits the organizational structure better than traditional centralized DBS. Each location will have its local data as well as the ability to get needed data from other locations via a communication network. Moreover, the failure of one of the servers at one site won't render the distributed database system inaccessible. The affected site will be the only one directly involved with that failed server. In addition, if any data is required from a site exhibiting a failure, such data may be retrieved from other locations containing the replicated data.
The performance of the system will improve, since several machines take care of distributing the load of the CPU and the I/O. Also, the expansion of the distributed system is relatively easy, since adding a new location doesn't affect the existing ones.

©IJRASET (UGC Approved Journal): All Rights are Reserved

B. Disadvantages of Distributed DBS

On the other hand, DDBS has several disadvantages. A distributed system usually exhibits more complexity and cost more than a centralized one. This is true because the hardware and software involved need to maintain a reliable and an efficient system. All the replication and data retrieval from all sites should be transparent to the user. The cost of maintaining the system is considerable since technicians and experts are required at every site.

Another main disadvantage of distributed database systems is the issue of security. Handling security across several locatios is more complicated. In addition, the communication between sites maybe tapped to. Concurrency control (CC) is an integral part of a database system, and is the activity of coordinating the actions of transactions that operate in parallel, access shared data, and potentially interfere with one another. Concurrency control has been actively investigated for the past several years, and the problem for nondistributed DBMSs is well understood. A broad mathematical theory has been developed to analyze the problem, and one approach called Two- phase locking (2PL), has been accepted as a standard solution. Dynamic two-phase locking (2PL) is the CC technique that current databases use almost exclusively.

A dynamic 2PL system thrashes at high data contention levels, restricting performance to levels inconsistent with available resources.

Today's business environment has an increasing need for distributed database and client/server applications as the desire for reliable, scalable and accessible information is steadily rising. 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. However, there is some complexity when attempting to manage and control distributed database systems. We describe distributed database system and their issues. A major issue of distributed database is concurrency control problem. So we are also describing concurrency control problem and different concurrency control technique.

C. Transaction Concept

In the database, all operations are completed by transactions [13]. In this section, we will see the transaction concept.

In a traditional way, a transaction is an agreement between a buyer and a seller to exchange an asset for payment. This is the definition in a business sense. In a database viewpoint, it can be understand as a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions.

The transaction in a database should have two purposes. First, is to provide reliable units of work that allow a correct recovery from failure. Second, is to provide isolation between programs to help them access database concurrently.

A transaction in a database must have ACID properties to run the program correctly. In a distributed database, transactions are implemented over multiple applications and hosts. Moreover, distributed transactions also enforce the ACID properties over multiple data stores.

A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network. It is necessary to ensure the data in a distributed database should be integrity so that all users who access the database can get the correct data and run their program without any error. To address this issue, we can use ACID properties for distributed database transaction. The definition of ACID properties for distributed database is that it is a set of properties that guarantee the reliability of database transactions. ACID defines properties that traditional transaction must display; they are Atomicity,Consistency,Isolation, and Durability.

The DDBMS must ensure 4 important properties of transactions

Transactions should be atomic. Either they happen or they don't happen at all. Each transaction, run by itself, alone, should preserve the consistency of the database. The DBMS assumes that consistency holds for each transaction.

1) Isolation: Transactions are isolated from the effect of other transactions that might be executed concurrently

2) Durability: Once the user is notified that the transaction was successful, its effects should persist even if the system crashes.

It is really important for database to have the ACID properties to perform Atomicity, Consistency, Isolation and Durability in transactions that are proposed by. It ensures all data in scientific research and business field to be correct and valid statues, without them, database will be in a mess.

D. Concurrency control

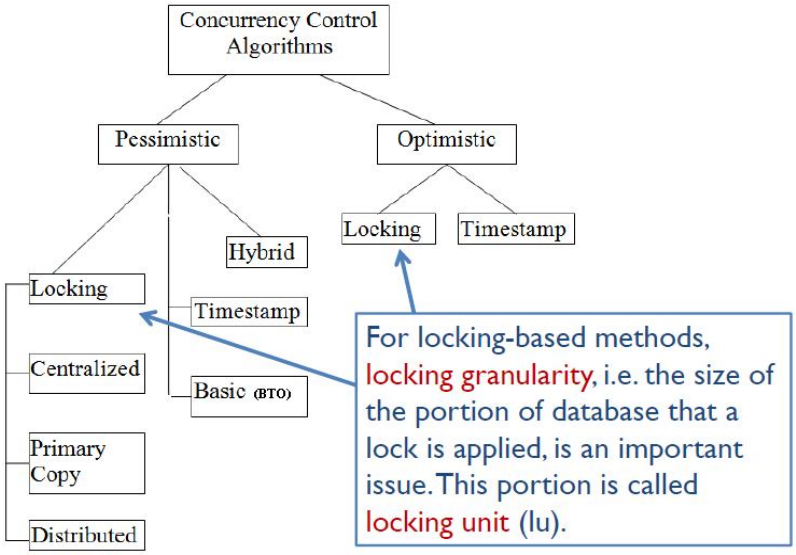
Concurrency control is another issue among database systems. "Concurrency control is the activity of coordinating concurrent accesses to a database in a multi-user database management system (DBMS)." There are a number of methods that provide

©IJRASET (UGC Approved Journal): All Rights are Reserved

concurrency control such as: Two phase locking, Time stamping, optimistic and Hybrid mechanisms. Some methods provide better concurrency control than others depending on the system.

E. Concurrency Control Techniques

Traditionally, Concurrency Control techniques have been classified into four categories.
Two Phase Locking
Timestamp- ordering
Optimistic
Hybrid



F. Locking

The most popular concurrency control technique is locking-based. In such schemes, a lock, in either shared or exclusive mode, is placed on some unit of storage whenever a transaction attempts to access it.
These locks are placed according to lock compatibility rules such that read-write, write-read, and write-write conflicts are avoided.

Lock Type	Read Lock	Write Lock
Read Lock		X
Write Lock	X	X

Table: Lock compatibility Table

X indicates incompatibility, means a case when a lock of the first type (in left column) on an object blocks a lock of the second type (in top row) from being acquired on the same object (by another transaction). It is a well known theorem that if lock actions on behalf of concurrent transactions obey a simple rule, then it is possible to ensure the serializability of these transactions: “No lock on behalf of a transaction should be set once a lock previously held by the transaction is released.” This is known as two-phase locking, since transactions go through a growing phase when they obtain locks and a shrinking phase when they release locks. In general, releasing of locks prior to the end of a transaction is problematic. Thus, most of the locking-based concurrency control algorithms are strict in that they hold on to their locks until the end of the transaction.
In two-phase locking, every transaction obtains locks in a two-phase manner. During the growing phase, the transaction obtains locks without releasing any locks. During the shrinking phase, the transaction releases locks without obtaining any locks. A basic 2PL scheduler follows the following three rules. When the 2PL scheduler receives a lock request, it tests whether the requested lock conflicts with another lock that is already set. If so, it queues the lock request. If not, it responds to the lock request by setting the lock.Once the 2PL scheduler has set a lock on a data item, it cannot release the lock until the DM has completed processing of the

©IJRASET (UGC Approved Journal): All Rights are Reserved

lock's corresponding operation. Once the 2PL scheduler has released a lock for a transaction, it may not subsequently obtain any locks for the same transaction.

G. Timestamp Ordering

Lock method maintains serializability by mutual exclusion and in Timestamp method maintains serializability by assigning a unique timestamp to every transaction and executing transactions accordingly. In timestamp ordering methods, the TM assigns a unique timestamp to each transaction it executes, and attaches the transaction's timestamp to every operation issued by the transaction. A timestamp ordering (TO) scheduler orders conflicting operations according to their timestamps.

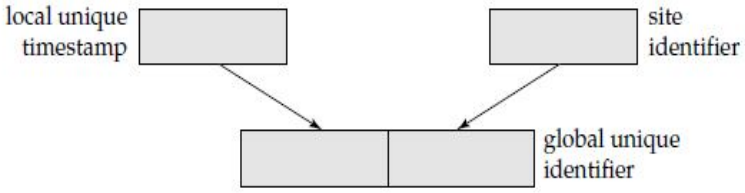


Fig Generation of unique timestamps.

In the distributed scheme, each site generates a unique local timestamp by using either a logical counter or the local clock. We obtain the unique global timestamp by concatenating the unique local timestamp with the site identifier, which also must be unique. The aggressive form of TO-based concurrency control is basic timestamp ordering (BTO). BTO associates timestamps with all recently accessed data items and requires that conflicting data accesses by transactions be performed in timestamp order. A BTO scheduler executes an operation immediately if it is possible to do so. The scheduler rejects an operation if it has already executed a conflicting operation with a later timestamp. When an operation is rejected, the transaction that issued it must abort and restart with a later timestamp. If a BTO scheduler receives operations in an order widely different from their timestamp order, it may reject too many operations, thereby causing too many transactions to abort. Starvation of transactions can occur in a BTO system because of continuous conflicts and aborts.

For replicated data, the "read any, write all" approach is used, so a read request may be sent to any copy while a write request must be sent to (and approved by) all copies. Integration of the algorithm with two-phase commit is accomplished as follows: Writers keep their updates in a private workspace until commit time.

H. Advantage and disadvantage of Timestamp ordering Concurrency Control

In 2PL, and other locking techniques as well, the deadlock prevention or detection in a distributed environment, which is much more complex and costly. Timestamp ordering techniques (TO) avoid deadlocks entirely. There are many advantages and disadvantage of BTO define as following:

1) *Advantage:* No Deadlock: Deadlocks cannot occur in BTO, because transaction operations are executed in timestamp order, and transaction timestamps are drawn from a totally ordered domain. Better Performance: Basic TO (BTO) usually shows better overall performance in a distributed environment.
2) *Disadvantage:* Additional cost: For better performance BTO required Timestamp management. Hence increase the additional cost for Timestamp management.

I. Distributed Optimistic Concurrency Control

Optimistic Concurrency Control (OCC) is based on the assumption that a conflict is rare, and that it is more efficient to allow transactions to proceed without delays to ensure serializability. When a transaction wishes to commit, a check is performed to determine whether a conflict has occurred. There are three phases to an optimistic concurrency control protocol:

J. Read phase

The transaction reads the values of all data items it needs from the database and stores them in local variables. Updates are applied to a local copy of the data and announced to the database system by an operation named pre-write.

K. Validation phase

The validation phase ensures that all the committed transactions have executed in a serializable fashion. For a read-only transaction, this consists of checking that the data values read are still the current values for the corresponding data items. For a transaction that

©IJRASET (UGC Approved Journal): All Rights are Reserved

contains updates, validation consists of determining whether the current transaction leaves the database in a consistent state, with serializability maintained.

L. Write phase

This follows the successful validation phase for update transactions. During the write phase, all changes made by the transaction are permanently stored into the database. The idea behind optimistic concurrency control is to do all the checks at once; hence, transaction execution proceeds with a minimum of overhead until the validation phase is reached. If there is little interference among transactions, most will be validated successfully. However, if there is much interference, many transactions that execute to completion will have their results discarded and must be restarted later. Under these circumstances, optimistic techniques do not work well. The techniques are called "optimistic" because they assume that little interference will occur and hence that there is no need to do checking during transaction execution.

M. Advantage and disadvantage of Distributed Optimistic Concurrency Control

There are many advantage and disadvantage when using optimistic concurrency methods in distributed systems. It is essential that the validation and the write phases be in one critical section. These operations do not need to be executed in one phase. It is sufficient to guarantee that no other validating transaction uses the same data items before an earlier validated transaction has written them.

1) Advantage:

a) Non-blocking: It is Simple to implementation on blocking

b) Deadlock-free: Non-block data-access operations. No mutual blocking between transactions results in a deadlock free.

2) Disadvantage: Preserving the atomicity of validating and write phases. One has to find a mechanism to guarantee that the validate-write critical section is atomic for global transactions.

The validation of sub transactions is made purely on a local basis. In the global validation phase, we are interested only in the order between global transactions. Conflicts that is not detectable at the validation phase. Transaction may be non-existent in the system, active or validated. A conflict is always detected between two active transactions. Unnecessary restart. They have the problem of unnecessary restarts and heavy restart overhead. This is due to the late conflict detection that increases the restart overhead since some near-to-complete transactions have to be restarted.

N. Hybrid

The methods mentioned above can be combined in various ways in order to make use of either advantage. For example, a distributed optimistic 2PL scheduler might attempt an optimistic execution first, and only if this one fails, it undergoes the second attempt by means of 2PL.Bernstein and Goodman enumerate several concurrency control methods that combine 2PL and TO. A distributed optimistic 2PL scheduler executes transactions optimistically, but if a transaction is aborted, the scheduler uses 2PL to execute the transaction a second time. Other hybrid approaches are optimistic with dummy locks, hybrid optimistic concurrency control and broadcast optimistic concurrency control.

1) Advantage

a) Deadlock-free: Non-block data-access operations. No mutual blocking between transactions results in a deadlock free.

b) Reduce unnecessary restart: They have reduced the problem of unnecessary restarts and heavy restart overhead.

c) Batter performance: The performance of Hybrid concurrency control is significantly better than BTO, 2PL and OCC.

2) *Disadvantage*

a) Additional costs: They have additional cost for deadlock detection in 2PL and Timestamp management in BTO. So increase the cost.

b) Unnecessary restart: They have the problem of unnecessary restarts and heavy restart overhead.

c) Low network latency: Very low perform when network latency is low.

O. Comparative Study

In this section the effectiveness of various techniques studied above has been compared.

©IJRASET (UGC Approved Journal): All Rights are Reserved

Techniques	Lock based Technique	Time stamp based Technique	Optimistic Technique
Waiting Time	More	Less	Less than locking but more than TSO
Delay	Delay occurs	Delay occurs	No delay
Occurrence of Deadlock	Deadlock occurs	Deadlock-free	Deadlock-free
Serialization Order	Decided dynamically	Decided statically	Decided dynamically

II. CONCLUSION

Different techniques show that there are many other ways to control concurrency in databases. Locking is the simple technique that prevents concurrency but it is have a problemof deadlock. Timestamp based concurrency control is a deadlock-free technique. Optimistic technique is better to other techniquesas it assumes that not too many transactions will conflicts with each other. It is also a deadlock-free and allows maximum parallelism. Some studies are being done for object-oriented systems while others are dealing with semantics of transactions and weaker form of consistency.

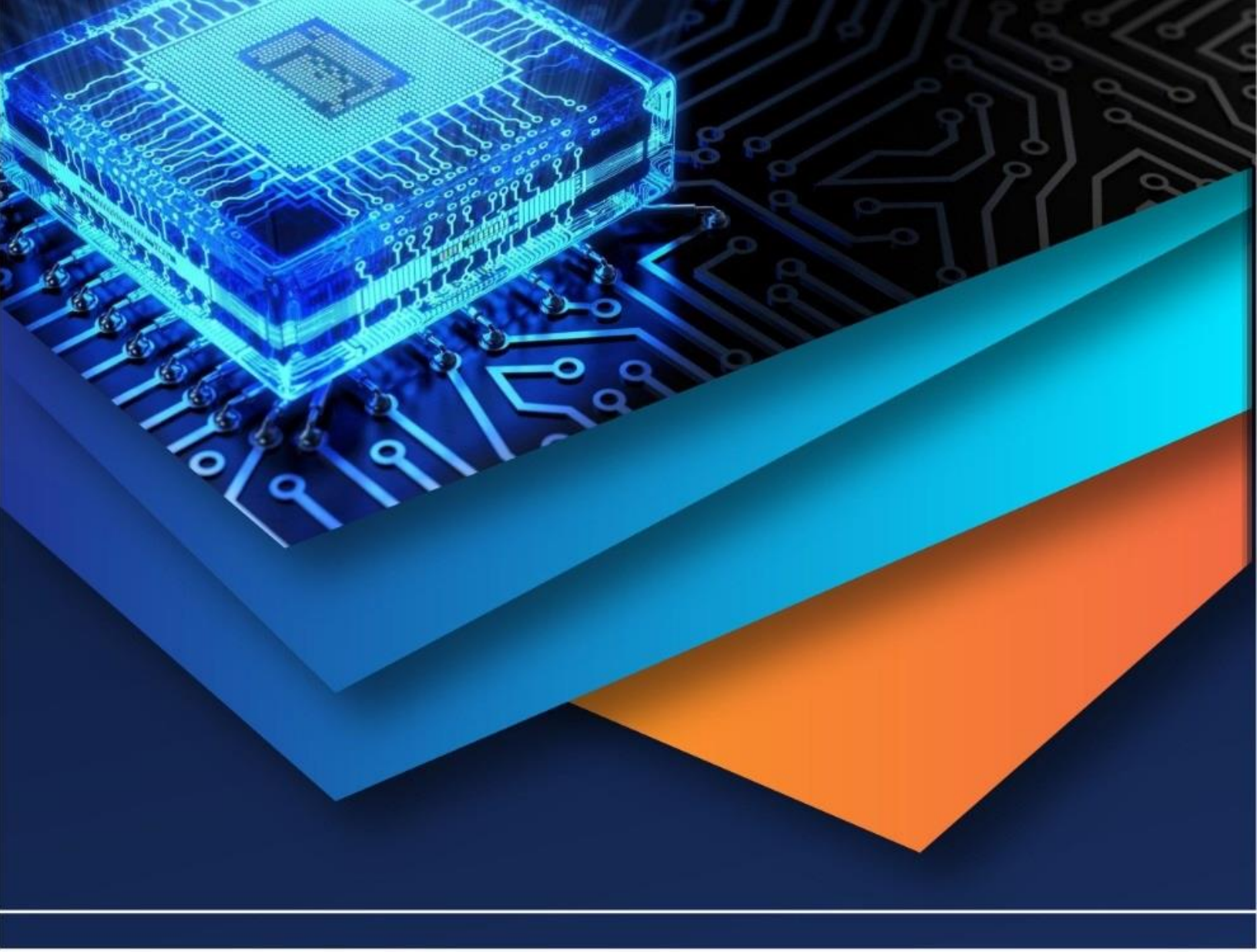
REFERENCES

[1] Philip A. Bernstein, VassosHadzilacos, and Nathan Goodman. Concurrency Control and Recovery in Database Systems. Addison-Wesley, 1987.
[2] Lance Hammond, Brian D. Carlstrom, Vicky Wong, Ben Hertzberg, Mike Chen, Christos Kozyrakis, and KunleOlukotun. Programming with transactiona coherence and consistency (tcc). In ASPLOS-XI: Proceedings of the 11th international conference on Architectural support for programming languages and operating systems, pages 1{13. ACM Press, Oct 2004.
[3] Christos Kotselidis, Mohammad Ansari, Kim Jarvis, MikelLuj_an, Chris Kirkham, and Ian Watson. Designing a distributed software transactional memory system. In ACACES '07: 3rd International Summer School on Advanced Computer Architecture and Compilation for Embedded Systems, July 2007.
[4] M. Tamer Ozsu and Patrick Valduriez. Principles of Distributed Database Systems, Second Edition. Prentice-Hall, 1999.
[5] Concurrency Control in Distributed Database Systems By PHILIP A. BERNSTEIN AND NATHAN GOODMAN.
[6] Badal D. Z., Correctness of Concurrency Control and Implications in Distributed Databases, Proceedings of COMPSAC 79 Conference, Nov 1979.
[7] Distributed Transaction Processing on an Ordering Network By RashmiSrinivasa, Craig Williams, Paul F. Reynolds
[8] Transaction Processing in Distributed Databases. By ManpreetKaur
[9] Database Concepts by ElmasriNavathe By Pearson Education
[10] Data base Concepts by Colloly. By Pearson Education
[11] BrahimMedjahed, MouradOuzzani, Ahmed K. Elmagarmid. Generalization of ACID Properties
[12] "Distributed databases Principals & Systems", Stefano Ceri, GinseppePelagatti, McGrawHill Book Company, 1984.
[13] "ORACLE 8: A Beginner's Guide", Michael Abbey, Michael J. Corey, Tata McGraw -Hill Publishing Company Limited, 1997.
[14] "ORACLE The Complete Reference Third Edition" George Koch, Kevin Loney, Osborne McGraw -Hill, 1995.
[15] Tanenbaum, van Steen: Distributed Systems, Principles and Paradigms; Prentice Hall 2002
[16] Coulouris, Dollimore, Kindberg: Distributed Systems, Concepts and Design; Addison-Wesley 2005
[17] World of DBMS By Jitendrasheetlani, Dhirajgupta
[18] WWW.Citeseer.com
[19] "The InfoSeek Search Service", http://www.infoseek.com
[20] Bell, David and Jane Grisom, Distributed Database Systems. Workinham, England: Addison Wesley,1992.
[21] Bertino, Elisa, "Data Hiding and Security in Object-Oriented Databases," In proceedings EighthInternational Conference on Data Engineering, 338-347, February 1992.

©IJRASET (UGC Approved Journal): All Rights are Reserved

[22] Denning, Dorothy E. et al., "Views for Multilevel Database Security," In IEEE Transactions on Software Engineering, vSE-13 n2, pp. 129-139, February 1987.
[23] Denning, Dorothy. E. et al., "A Multilevel Relational Data Model". In Proceedings IEEE Symposium on Security and Privacy, pp. 220-234,1987.
[24] Haigh, J. T. et al., "The LDV Secure Relational DBMS Model," In Database Security, IV: Status and Prospects, S. Jajodia and C.E. Landwehr eds., pp. 265-269, North Holland: Elsevier, 1991.
[25] Herbert, Andrew, "Distributing Objects," In Distributed Open Systems, F.M.T. Brazier and D. Johansen eds., pp. 123-132, Los Alamitos: IEEE Computer Press, 1994.
[26] "Illustra Object Relational Database Management System," Informix white paper from the Illustra Document Database, 1996.
[27] Jajodia, Sushil and Ravi Sandhu, "Polyinstantiation Integrity in Multilevel Relations," In Proceedings IEEE Symposium on Research in Security and Privacy, pp. 104-115, 1990.
[28] " Concurrency Control in Distributed Transaction Process" by JitendraSheetlani and manojjangde.
29 Analysis of Effectiveness of Concurrency Control Techniques in Databases by Rachna Behl and Ruchi Dagar.

©IJRASET (UGC Approved Journal): All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)