An Insight into Concurrency Control Protocols of Distributed Databases

Anjali Ganesh Jivani

***Abstract*—This paper reviews the working of different concurrency control protocols in Distributed Databases. As the development and maturity of the popular centralized database system moves towards the distributed approach, the challenges and roles start becoming more complex and complicated. The discussion revolves around the variety of protocols, their working, their advantages and their disadvantages in a distributed environment. The paper is a comparative between the methods that are popular and accepted.**

***Index Terms*—Concurrency Control, Distributed Database, Fragmentation, Replication.**

I. INTRODUCTION

The term 'Distributed' in the concept of 'Distributed Databases' clearly signifies the nature of the database system that it is not at one place or location but residing at physically independent locations with some interconnection in the logical design. In fact a 'Distributed Database' can be thought of as a collection of multiple, logically interrelated databases distributed over a computer network in such a way the distribution is transparent to the users.

The fact that the database is located at multiple sites, obviously means that it promotes availability of data to multiple users concurrently without losing the integrity and veracity of the database.

Distributed database as compared to the Centralized database differs mainly in the way the data is actually stored and located. Since the data is not available at just one site it signifies that it is located in '*duplicate*' at many sites. This leads to the concept of Replication and Fragmentation of database.

Replication: It clearly means that the same data is duplicated at multiple sites so as to enhance 'availability'.

Fragmentation: It means fragments of the data are available at multiple sites. These fragments could be in turn replicated also. The fragmentation notion in a relational database which is a two-dimensional design allows only two types of breakup – a table divided horizontally or a table divided vertically. A horizontal breakup (tuple level) would need a union to get back the full database and a vertical breakup (column level with primary key) would require a natural join.

Manuscript received October, 2013

Dr. Anjali Ganesh Jivani, Computer Science & Engineering Department, The Maharaja Sayjirao University of Baroda, Vadodara, Gujarat, India.

This scenario where multiple users are accessing the distributed database and maybe the same data at varied locations can create problems which are different from those of a centralized system. There are many protocols which ensure the ACID properties of a transaction in distributed databases also. It is important that replication and fragmentation do not violate the integrity of the database system.

The protocols are discussed below briefly with their pros and cons.

II. SINGLE LOCK MANAGER APPROACH

This approach has the following properties:

1) Can be implemented on replicated as well as fragmented database.
2) Single site is assigned as the lock manager.
3) All lock requests are directed to that site so it works like a centralized database.

Advantages:

a. Simple design as it simulates centralized approach.
b. Deadlock detection and handling is easy.

Disadvantages:

a. The lock manager site can become a bottleneck as all read/write requests and locking is directed to it.
b. If that site fails then no locks can be granted and processing can stop.
c. It is necessary to have backup sites when the lock manager site fails.

III. DISTRIBUTED LOCK MANAGER APPROACH

This approach has the following properties:

1) Can be implemented on non-replicated database only i.e. fragmentation has been done but no duplication.
2) Every site has its own local lock manager which handles lock requests for data stored locally (fragment) e.g. location-wise fragments for bank customers' data.

Advantages:

a. Simple implementation – again it is a centralized approach at each site.
b. Reduces chances of any site becoming a bottleneck.
c. Low overhead.

Disadvantages:

a. Deadlock handling can be complex as inter-site deadlocks are possible. This would require separate deadlock detection techniques which are executed at all the sites and are keeping track of transactions executing at all the different sites.

IV. PRIMARY COPY

This approach has the following properties:

1) Generally used in data replication but can be used for fragmentation also.

2) One site chosen as the primary copy for each data – this primary copy is to be locked whenever any lock is required whether read or write.
3) Each data has a separate primary copy so one site can never become a bottleneck. The primary copy is as shown in Table. I.

Table I. Primary Copy

Sites:	S1	S2	S3	S4	S5
Data:	Q	Q	Q	Q	Q(pc)
	P(pc)	P	P	P	P
	R	R	R(pc)	R	R

As shown above the primary copy for Q is site S5, for P is S1 and for R is S3. By equally distributing the primary copies the chances of one site becoming a bottleneck reduces.
Advantages:
a. Concurrency control similar to that of centralized database.
b. No site becomes a bottleneck.
c. Simple implementation.
Disadvantages:
a. If the primary copy for a data item fails that data becomes unavailable though other sites have that data.
b. Can assign backup site for each primary copy but many backup sites would be needed.
c. Deadlock handling is complicated.

V. MAJORITY PROTOCOL

This approach has the following properties:
1) This protocol can be implemented on replicated as well as fragmented data.
2) As the name suggests, to lock a data item Q, more than half of the sites where Q is replicated should be locked i.e. if n sites have Q, then Q should be locked at n/2 +1 sites at least.
3) A transaction cannot proceed unless majority of the replicas are locked.
4) Writes are performed on all replicas.
Advantages:
a. Locking is not centralized.
b. This protocol can be implemented also if some sites are not available.
Disadvantages:
a. Complicated implementation
b. 2(n/2 + 1) requests to lock and (n/2 +1) requests to unlock
c. Deadlock detection and handling is complicated. Deadlock can occur also when just one data is being locked. E.g. Q is replicated at say four sites. To lock Q, n/2 +1 should be locked i.e. 4/2 + 1 = 3 replicas to be locked.

If transactions T1 and T2 both are trying to lock Q which has been replicated at four sites, as shown in Table II, a deadlock can occur.

Table II. Deadlock in Majority Protocol

Sites	S1	S2	S3	S4
Transactions	T1	T1	T2	T2

Both transactions have locked 2 replicas and are waiting for the third replica to be freed. This can happen with three transactions also. Each of 3 transactions may have locks on 1/3rd of the replicas of a data. This can happen similarly for n transactions.
This can be avoided by fixing the sequence in which sites are to be locked e.g. if locking sequence is S1, S2, S3 and S4, then if T1 has locked S1 and S2, T2 will wait till locks are released on them. T2 cannot lock S3 before locking S1 and S2. In this way T1 will complete first and there will not be a deadlock.

VI. BIASED PROTOCOL

This approach has the following properties:
1) This protocol is used when there has been replication only.
2) It is biased towards read requests. To read a data item any replica containing that data item can be locked in shared mode.
3) A write request however needs all replicas to be locked in exclusive mode.
Advantages:
a. There is less overhead on read operations as only one site is to be locked.
b. A fast and effective protocol if transactions are generally read only.
Disadvantages:
a. Write operations have more overhead as many locks required.
b. Deadlock handling is complicated as locking is extensive.

VII. QUORUM CONSENSUS PROTOCOL

This approach has the following properties:
1) Generally used for replicated databases.
2) Each site is assigned a weight which is an integer.
3) Stable sites are assigned higher weights.
4) Every data item Q has two quorums (weights) associated with it – Qr read quorum and Qw write quorum. These quorums are also integers.
5) If S is the total weight of the sites where Q is replicated, the following two conditions are to be satisfied to implement this protocol,
a. Qr + Qw > S
b. 2* Qw > S i.e. Qw > S/2 (the write quorum is more than half the total weight of sites where Q is replicated).
6) To execute a read or a write operation, enough replicas should be locked so that their total weight is more than or equal to the read quorum or write quorum respectively.
>= Qr or >=Qw
7) The read quorum is generally small so that less number of sites are required to be locked and the write quorums is kept are high to lock more sites.
8) This is a generalization of the majority/biased protocols.
E.g. Q is located at say S1, S2, S3 and S4.
Weights are:
S1 = 50 S2=50 S3=60 S4=70
S = 50+50+60+70 = 230 (total weigth)

Qw = 160 (3 sites to be locked for writing)
Qr = 70 (1 site to be locked for reading)

Qw + Qr = 160 + 70 = 250 > 230 (S)
2*Qw = 2*160 = 320 > 230

VIII. TIMESTAMP BASED PROTOCOL

This approach has the following properties:

1) Each transaction is given a unique timestamp to decide serializability. There are two methods for generating timestamps.
2) In the first method one site is fixed for generating timestamps. For every new transaction starting at any site, the timestamp is first to be obtained from this site. This timestamp site can generate timestamps using a counter or the system clock. This however has a major drawback that the timestamp generating site can become a bottleneck and if the site goes down all transactions will stop processing.
3) In the second method each site generates a unique timestamp using its local counter (LC). The timestamp has two parts – LHS stores the counter value and RHS stores the site identifier e.g.

001 1111 - 1st transaction on site 1111
001 1112 - 1st transaction on site 1112

The counter is stored on the LHS and site identifier on RHS because otherwise a site with a lower address will always generate timestamps which have higher precedence and will always be older transactions compared to others. E.g.
1111 002 would be older as compared to 1112 001 (first transaction of site 1112).

4) Another method of generating LC is, every time a transaction say T1 starts at say site S1, its timestamp would be <001, 1111> (<x, y>), where x is transaction identifier and y is the site identifier.

Assume T1 executes at three sites S1, S2 and S3. The assigning of the timestamps is shown in Table. III. The timestamp once assigned does not change till it completes no matter which site it visits.
Only when a transaction visits a site, if the value of x is more than or equal to its LC then the LC is incremented by x+1.)
Since T1 has not visited S4, when a transaction starts at S4, its timestamp will be <001, 1114>.
When a transaction starts at S2 its timestamp will be <002, 1112>. If this transaction executes at S3, since LC of S3 = 2, it now becomes 3.
In this way the local LCs are incremented every time a transaction with the same or higher x value executes at that site.
System clocks are also used to generate timestamps but all clocks may not synchronize or run at the same speed.

Table III. Assigning Timestamps

Sites	S1(1111)	S2(1112)	S3(1113)	S4(1114)
Initial LC	LC = 0	LC = 0	LC = 0	LC = 0
LC of S1 is 1 as it is the first transaction executing at S1. Timestamp of T1 is <001,1111>	T1 starts LC= 1(<001,1111>)			
LC of S1 incremented by 1 as T1 has visited S2 and T1 had x = 1.		T1 (<x, y>) x = 1, y = 1111 Since x >LC of S2 LC = x+1= 2		
Same as above for site S3			T1 Again x>LC of S2 LC = x+1 = 2	
T2 starts at S4. LC if S4 is 1 as it is the first transaction executing at S4.				T2 starts LC = 1 (<001, 1114)
T3 starts at S2. LC if S2 is 2 as it is the second transaction executing at S2.		T3 starts LC = 2 (<002, 1112>)		

IX. CONCLUSION

The paper contained distributed protocols with discussion on their individual advantages and disadvantages. There are many more protocols like the distributed 2-phase locking protocol, distributed wound-wait protocol, distributed optimistic protocol, etc. Each protocol preserves the ACID properties of transactions. Many more protocols are being designed as per the requirement of present distributed scenario.

ACKNOWLEDGMENT

I would like to thank Silberschatz, Korth and Sudarshan for the wonderful book that they have written on Database Concepts. This book was the sole reason for writing this paper which makes the understanding of the protocols simpler.

REFERENCES

[1] Silberschatz, Korth and Sudarshan , “Database system concepts”, 5th Edition, Mc-graw Hill 2008.
[2] Ricardo, Catherine, “Database Systems: Principles, Design and Implementation”, 1990, MacMillan Publishers, New York.
[3] K. Sugihara, “Concurrency Control Based on Distributed Cycle Detection”, In Proceedings of International Conference on Data Engineering, 1987,pp. 267-274.
[4] Mandeep Kaur, Harpreet Kaur, “Concurrency Control in Distributed Database System”, In Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 7, July 2013.
[5] T.F. Keefe, W.T. Tsai, J. Srivastava, “Database Concurrency Control In Multilevel Secure Database Management Systems”, IEEE Transaction on knowledge and Data Engineering, 1993, 5 (6) 1039-1055.
[6] Ray, L. V. Mancini, S. Jajodia and E. Bertino, ASEP: A Secure and Flexible Commit Protocol for MLS Distributed Database Systems”, 2000, IEEE Transactions on Knowledge and Data Engineering, 12(6): 880 – 899.
[7] S. Mehrotra, R. Rastogi, Y. Breitbart, H. F. Korth, and A. Silberschatz, “The Concurrency Control Problem in Multidatabases: Characteristics and Solutions”, ACM SIGMOD International Conference on Management of Data, San Diego, California, June 2-5, 1992, pp: 288-297.

Dr. Anjali Ganesh Jivani has done her Ph.D. in the field of Text Data Mining from The Maharaja Sayajirao University of Baroda, Gujarat, India. She is presently working as an Associate Professor in the Computer Science & Engineering department of The M. S. University. She has published a number of National and International level papers in conference proceedings as well as international journals related to her field. She has also co-authored a book on ‘SQL and PL/SQL’. She has won prizes at paper presentation competitions and her papers have been cited by many authors. Her main area of interest is Databases, Data and Text Mining, Image Processing and Big Data. She is a life member of ISTE and CSI.