Implementing the Update Propagation Strategies on Replicated Database

Su Su Mon, Dr.Khine Khine Oo
University of Computer Studies, Yangon
baby.babybu@gmail.com, k2Khine@gmail.com

Abstract

A distributed database is a database with its data elements stored at several sites of a computer network. Certain data items may be redundantly stored at more than one site for reliability reasons. A replicated database is a distributed database in which multiple copies of the same data items are stored at different sites. Distributed applications frequently use replication as a means to achieve higher level of performance, reliability and availability. Replication is a technique for automatically maintaining the availability of data despite server failures. If data are replicated at two or more failure-independent servers then client software may be able to access data at alternative servers. The basic problem with data replication is update propagation. In this paper, we propose a system that handles replica updates in a distributed database system called Lazy and Eager. This system will be implemented by using C#.NET and Access database.

Keywords: Replication, Update Propagation, Eager Replication, Lazy Replication

1. Introduction

Nowadays, most businesses are organized with distributed pattern. For instance, main office as server site and branch offices as slave sites. For so, data about the organization becomes necessary to store redundantly. Moreover, information about the organization needs for autonomy to all offices. It spends many times for informing changes in main office to all branches and vice versa. So, this proposed system supports for organization to take the most appropriate action informing the changes in main office to all branches and actions in the branch offices to main office within a minimum of time.

Informing formally is often made based on human resource and telephone rather than on the update propagation methods in replication. This practice leads to save time and human resources. The integration of organization actions with computer-based update propagation methods would improve performance, increase availability, reliability and to make it fault tolerance.

The system could assist the business based on replication. This paper is organized as follows: Session 1 introduces the paper, Session 2 will discuss the theoretical background, Session 3 will discuss the two-phase commit protocol, the design and implementation are presented in Session 4, Session 5 discusses the experimental results and Session 6 concludes the paper.

2. Theoretical Background

The background theory of Update Propagation, Replication mechanisms, Eager replication, Lazy replication, Master architecture, Lazy Master, Eager Master and theoretical comparison of Lazy Master and Eager Master are represented in this section.

2.1 Update Propagation

The basic problem with data replication is that an update to any given logical object must be propagated to all stored copies of that object. A difficulty that arises immediately is that some site holding a copy of the object might be unavailable because of a site or network failure at the time of update. The obvious strategy of propagation updates immediately to all copies is thus probably unacceptable, because it implies that the update transaction will fail if any one of those copies is currently unavailable. In fact, data is less available under the strategy than it would be in the non-replicated case**.**

2.2 Replication mechanisms'

Replication is a technique for automatically maintaining availability of data despite server

failures. Replication techniques can be grouped into two criteria:

i. When updates are propagated between replicas. (Lazy or Eager)

ii. Where updates can be applied. (Centralized)

2.2.1 Eager and Lazy Replication

With Eager replication, updates are propagated to and processed by the replicas before the transaction that generated the updates commits. This means that transactions have to wait until updates are installed at every replica. Transaction processing in eager replication has been well suited in several protocols such as Two-Phase commit protocol for serialization. The benefit of this mechanism is that conflicts occurring at replica are detected before the original transaction commits. Therefore there are no inconsistencies.

In Lazy replication, each transaction executes locally and then asynchronously sends its updates to other replicas at some later time (user defined time) after its commits. The replica at each site is updated by a separate transaction. Since each transaction executes locally and independently, the systems do not require multi site commit protocol. But inconsistencies between replicas are not detected within transaction bounds.

2.3 Master Architecture (Primary Copy)

The number of directly updatable replicas can be restricted to one single node in the system. This approach is called primary copy, and avoids concurrent updates on the same object at two different nodes. Each transaction must send its updates first to one dedicated site, the primary copy, which takes care of propagating the updates. The drawback of this technique is that the primary node represents a hot spot in the system.

2.3.1 Lazy Master and Eager Master

Lazy Master scheme allows an update transaction is first done at master site and then propagated to all other slave replicas after the master transaction commits. In this technique does not require the primary copy to wait for the secondary. It can commit or abort its transaction independently. Therefore it may occur only local deadlocks. There are no conflicts across the servers because there is only one server executing the update transactions.

In Eager Master Scheme, an update transaction is first done at master node and then propagated to secondary before master transaction commits. To ensure that both master and slaves install the updates, we use Two-Phase Commit protocol (2PC). In this case, master site initiates 2PC protocol and master site is a coordinator site and the slaves are participant sites. The master site waits for the slave sites until the transaction ends. So it causes long response times. There are no conflicts between servers because there is one server executing update transactions.

2.3.2 Comparison between Lazy Master and Eager Master

The main comparison points between lazy master and eager master are as shown in table 3.1.

Lazy Master	Eager Master
Updates can be done at master site.	Updates are first done at master site.
Updated data is propagated outside the update transaction	Updated data must be propagated inside the update transaction.
This scheme cannot guarantee data consistencies	This scheme can guarantee data consistencies.

Table 2.1 Comparison of Lazy Master scheme and Eager Master Scheme

2.4 Comparison of Propagation strategies with Ownership strategy

The comparison points of propagation strategies with ownership strategies are as shown in table 3.2.

Propagation vs. Ownership	Lazy	Eager
Master	N transactions One object owner	One transaction One object owner

Table 2.2 Comparison of Propagation strategies with Ownership strategy

3. Two-Phase commit protocol (2PC)

Two-phase commit protocol (2PC) is required to guarantee ACID properties in a distributed environment. However, this means that the sites are dependent on one another for completion of an update, and they will give up site autonomy.

In the first phase, the coordinator sends the can Commit to the workers, which pass it on to the other replica managers and collect their replicas before replying to the coordinator.

In the second phase, the coordinator sends the do Commit or do Abort request, which is passed on the members of the groups of replica mangers.

3.1 Nature of Lazy and Eager Scheme

The nature of lazy and eager propagation method is as shown in figure 3.1 and figure 3.2 respectively.

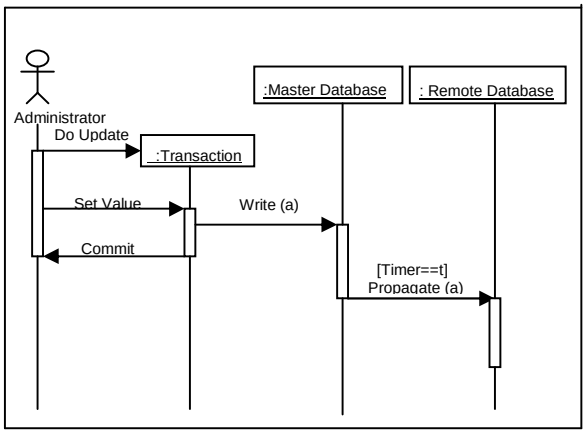

Figure 3.1 Sequence diagram of Lazy replication

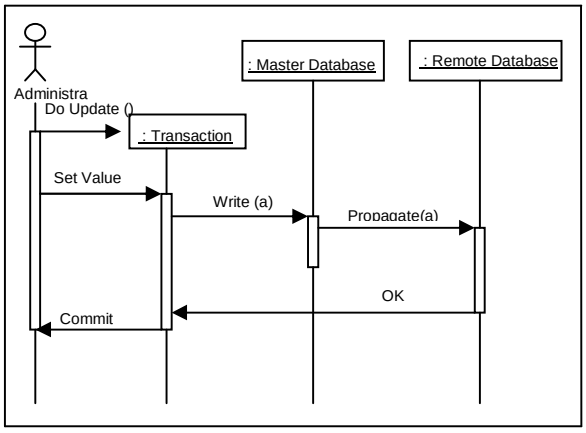

Figure 3.2 Sequence diagram of Eager replication

4. Design and Implementation

This system composed with three databases: one for Main office database; the other two are Branch 1 office and Brach 2 office databases. The overall system design can be see in Figure 4.1.

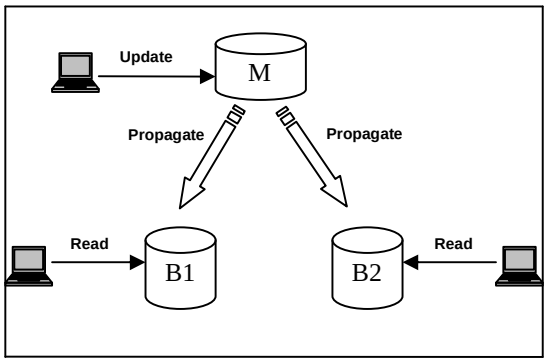

Figure 4.1 Proposed System Architecture

In the Main database, there are six tables. They are products, request, sent, salespersons, customers and orders tables. In the Branch databases (B1 and B1), there are four tables. They are products, salespersons, customers and orders tables. Products table is replicated at each site. And salespersons table is replicated at main office and corresponding branch office. Other tables are not replicated.

All data are handled by main office and then propagate to all other branch offices using lazy or eager method according to the situations.

4.2 Process Flow of Master Site

In the main office (Master site), there is an administrator. At the start of the process, the administrator can choose the desired command (Select, Insert, Update, Delete). If the administrator choose the query (Select), read at the local database and show the result on the display. Otherwise, the administrator choose the other command, set value for the chosen command and write firstly at the local database and then propagate to the remote database using specified propagation strategies.

If the product price data is updated at main office, then the changed price data need to be propagated to all other branch offices immediately. Therefore, in this situation eager method is used. Because of using the eager method, two-phase commit protocol is applied at this portion. And other changed data is propagated using the lazy method. The process flow of master site is represented in figure 4.2.

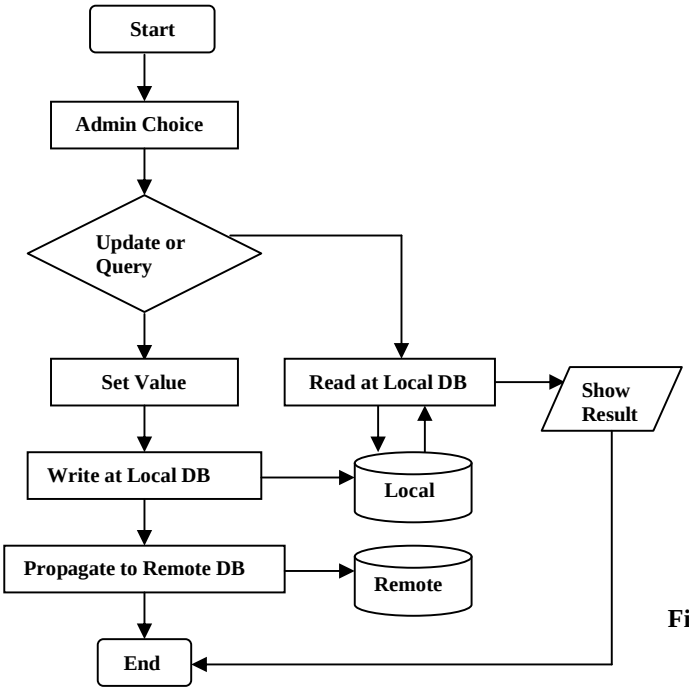

Fig 4.2 Process Flow Diagram of Master Site

4.3 Process Flow of Slave Site

For the slave site point of view, the system divided into three parts. They are order processing for part (a) and request processing for part (b) and view processing for part (c). At the branch offices, user can choose order, request or view commands. If the view request (select statement) is chosen, read at the local database and show the result on the screen. The diagram of view processing is presented in figure 4.3(a).

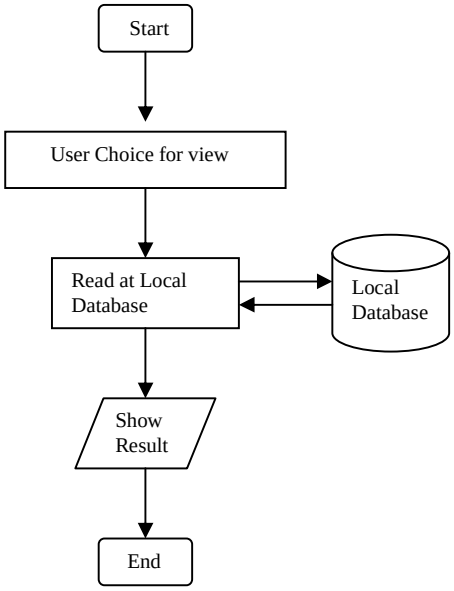

Figure 4.3(a) View Process Flow Diagram of Slave Site

If the order command is chose, sell products to the customer and then update (update statement) the local database. After some time later, the orders are propagated to the remote database (main office database) using the eager method. The diagram of order processing is presented in figure 4.3(b).

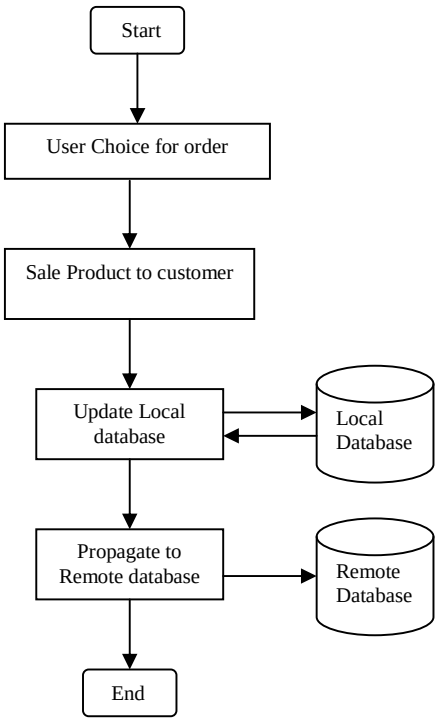

Figure 4.3(b) Order Process Flow Diagram of Slave Site

If the request command is chosen, request product to the master site and check whether or not the requested product is arrived at the branch office. The process flow of request is presented in figure 4.3(c).

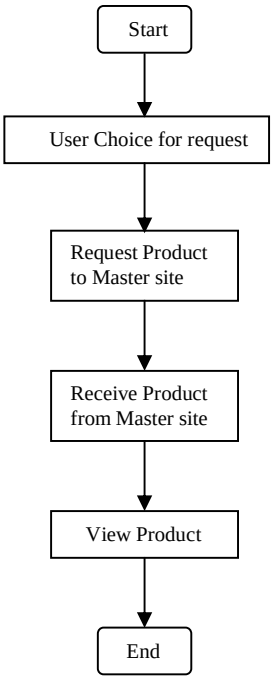

Figure 4.3(c) Request Process Flow Diagram of Slave Site

5. Benefits of proposed system

By implementing update propagation with Eager replication on master architecture, the proposed system can get correct and consistent data. For example, in a global banking system, exchange rates are replicated. It is crucial to have fast access to this replicated data from any bank sites.

By implementing update propagation with Lazy Master scheme, the proposed system ensures that product price information is always available and relatively current and consistent at all sites. So, this scheme is more suitable for data distribution system.

6. Conclusion

With organization supporting diverse hardware and software applications in distributed environments, it becomes necessary to restore data redundantly. Moreover, different applications have different needs for autonomy and data consistency. Replication is a solution for a distributed data environment. The aim of this paper is to study the nature of update propagation strategies in replication environment. Proposed system discusses update propagation on lazy and eager replication with master architecture.

References

[1] C.J Date "An Introduction to Database Systems" International Edition, May 2000.

[2] Nike Simpson "Data Replication for Critical Storage Assets" www.datacore.com

[3] Bettina and Gustavo Alonso "A New Approach to Developing and Implementing Eager Database Replication Protocols."

[4] K. Daudjee, K. Salem University of Waterloo "Lazy Database Replication with Snapshot Isolation"

[5] J.Gray, P.Helland, P.O'Neil, and D.Shasha. "The Danger of Replications and a Solution" SIGMOD, 1996

[6] Nick Blundell. "Replication for Scalability and Fault-Tolerance" Distributed Systems School of Computer Science, University of Birminghan, UK

[7] Y.Breitbast, R.Komondoor, R.Rastigi, S.Seshadri "Update Propagation Protocols for Replicated Database" SIGMOD 1999