Database Replication in DDBMS

Meena Talele[#1],Ashok Navale[#2], Vidya Harpude[#3] , Neha Mulchandani[#4]
[#1]*Computer Technology Dept..,*[#2]*Electronics and Telecommunication Technology Dept.,*[#3]*Instrumentation Dept.,*[#4]*Electronics and Communication Dept*
[#1234]*Lecturer,VESPolytechnic,Chembur*
[#1]meena.talele@ves.ac.in

Abstract-Database replication is the creation and maintenance of multiple copies of the same database. In most implementations of database replication, one database server maintains the master copy of the database and additional database servers maintain slave copies of the database. Database writes are sent to the master database server and are then replicated by the slave database servers. Database reads are divided among all of the database servers, which results in a large performance advantage due to load sharing. In addition, database replication can also improve availability because the slave database servers can be configured to take over the master role if the master database server becomes unavailable.

Keywords-*Database Replication, Snapshot replication, Merge replication, Transactional replication*

*****

1. INTRODUCTION

For someone who has worked in an environment in which the same database is used for data entry and reporting, or perhaps managed a single database server that was utilized by too many users, the advantages brought by data replication are clear. The main purpose of this paper is to emphasize those advantages as well as presenting the different types of Database Replication and the cases in which their use is recommended.

Imagine a scenario in which you have to develop an application that all the company's staff will use to perform different tasks. Each person has a laptop and will be connected to the company's network. This type of application can be developed in two different ways. One of those is the traditional approach of separating the tables from the other objects in the database so that the data can reside in a back-end database on a network server, or on the Internet or an intranet, while the queries, forms, reports, macros, and modules reside in a separate front-end database on the user's computer. The objects in the front-end database are based on tables that are linked to the back-end database. When users will retrieve or update information in the database, they use the front-end database. The second way enables you to take a new approach to building this solution by creating a single database that contains both the data and objects. Using Database replication, you can then make replicas of the database for each user and synchronize each replica with the Design Master on a network server. In this scenario, you can choose to replicate only a portion of the data in the Design Master, and you can replicate different portions for different users by creating partial replicas. By using partial replicas, you can duplicate only the data that each user actually needs. A complete set of data is still contained in the Design Master, but each replica handles only a subset of that data. The Design Master is the first member in a replica set and it is used in the creation of the first replica in a replica set. You can make changes to the database structure only with the Design Master. Replicas in the same replica set can take turns being the Design Master, but there can be only one Design Master at a time in each replica set.[1]

1 DDBMS:

The end result is a distributed database management system (DDBMS) that confers upon the organization a number of benefits, including:

- Sharing the same data across networks and users through network synchronization.
- Improved data reliability, consistency, and efficiency.
- Increased availability in the event of a site failure or other data disaster.
- Faster and more efficient queries through reduced network load.
- Improved availability in the event of a system failure.
- Less interference from users when searching for specific data.

2 THE CONCEPT OF REPLICATION

To better understand the method behind Database Replication we start with the term "Replication" which represents the process of sharing information to ensure consistency between redundant resources, such as software or hardware components,

to improve reliability, fault-tolerance, or accessibility. It could be data replication if the same data is stored on multiple storage devices, or computation replication if the same computing task is executed many times.

The access to a replicated entity is typically uniform with access to a single, non-replicated entity. The replication itself should be transparent to an externaluser. In addition, in a failure scenario, a failover of replicas is hidden as much as possible. In systems that replicate data the replication itself is either active or passive.

We talk about an active replication when the same request is processed at every replicated instance and about passive replication when each request is processed on a single replica and then its state is transferred to the other replicas.

If at any time one master replica is designated to process all the requests, then we are talking about the primary-backup scheme (master-slave scheme) predominant in high-availability clusters.

On the other side, if any replica processes a request and then distributes a new state, then this is a multi-primary scheme (called multi-master in the database field). [2] Even thought the process of Data Replication it's used to create instances of the same or parts of the same data, we must not confuse it with the process of backup since replicas are frequently updated and quickly lose any historical state. Backup on the other hand saves a copy of data unchanged for a long period of time.

3 WHAT IS DATABASE REPLICATION

Database replication is the process of creating and maintaining multiple instances of the same database and the process of sharing data or database design changes between databases in different locations without having to copy the entire database. In most implementations of database replication, one database server maintains the master copy of the database and the additional database servers maintain slave copies of the database. The two or more copies of a single database remain synchronized. [3] The original database is called a Design Master and each copy of the database is called a replica. Together, the Design Master and the replicas make up a replica set. There is only one Design Master in a replica set. Synchronization is the process of ensuring that every copy of the database contains the same objects and data. When you synchronize the replicas in a replica set, only the data that has changed is updated.

You can also synchronize changes made to the design of the objects in the Design Master. [1] Database writes are sent to the master database server and are then replicated by the slave database servers. Database reads are divided among all of the database servers, which results in a large performance advantage due to load sharing. In addition, database replication can also improve availability because the slave database servers can be configured to take over the master role if the master database server becomes unavailable. [3]

4 When to choose Database Replication

Implementing and maintaining replication might not be a simple proposition. If you have numerous database servers that need to be involved in various types of replication, a simple task can quickly become complex.

Implementing replication can also be complicated by the application architecture. However, there are numerous scenarios in which replication can be utilized.[4] Database replication is well suited to business solutions that need to:

• **Share data among remote offices.** You can use database replication to create copies of a corporate database to send to each satellite office across a wide area network (WAN). Each location enters data in its replica, and all remote replicas are synchronized with the replica at corporate headquarters. Individual replicas can maintain local tables that contain information not included in the other replicas in the set.

• **Share data among dispersed users**. New information that is entered in the database while users are out of the office can be synchronized any time the users establish an electronic link with the corporate network. As part of their workday routine, users can dial in to the network, synchronize the replica, and work on the most current version of the database. Because only the incremental changes are transmitted during synchronization, the time and expense of keeping up-to-date information are minimized. By using partial replicas, you can synchronize only specified parts of the data.

• **Make server data more accessible.** If your solution does not need to have immediate updates to data, you can use database replication to reduce the network load on your primary server. Introducing a second server with its own copy of the database improves response time. You determine the schedule for synchronizing the replicas, and you can adjust that schedule to meet the changing needs of your users. Replication requires less centralized administration of the database while offering greater access to centralized data.

• **Distribute solution updates**. When you replicate your solution, you automatically replicate not only the data in your tables, but also your solution's objects. If you make changes to

the design of the database, the changes are transmitted during the next synchronization; you don't have to distribute complete new versions of the software.

• **Back up data.** At first glance, database replication might appear to be very similar to copying a database. However, while replication initially makes a complete copy of the database, thereafter it simply synchronizes that replica's objects with the original objects at regular intervals. This copy can be used to recover data if the original database is destroyed. Furthermore, users at any replica can continue to access the database during the entire backup process.

• **Provide Internet or intranet replication.** You can configure an Internet or intranet server to be used as a hub for propagating changes to participating replicas.[1]

5 DATABASE REPLICATION USAGE

When Database Replication should not be used Although database replication has many benefits and can solve many problems in distributed-database processing, we should recognize the fact that in some situations replication is less then ideal. Database Replication is not recommended if:

• There are frequent updates of existing records at multiple replicas. Solutions that have a large number of record updates in different replicas are likely to have more record conflicts than solutions that simply insert new records in a database. If changes are made to the same record by different users and at the same time then record conflicts will definitely appear. This can be real time consuming because the conflicts must be resolved manually.

• Data consistency is critical at all times. Solutions that rely on information being correct at all times, such as funds transfers, airline reservations, and the tracking of package shipments, usually use a transaction method. Although transactions can be processed within a replica, there is no support for processing transactions across replicas. The information exchanged between replicas during synchronization is the result of the transaction, not the transaction itself.

6 METHODS OF PERFORMING DATABASE REPLICATION

Database replication can be performed in at least three different ways:

• **Snapshot replication:** Data on one database server is plainly copied to another database server, or to another database on the same server.

• **Merging replication:** Data from two or more databases is combined into a single database.

• **Transactional replication:** Users obtain complete initial copies of the database and then obtain periodic updates as data changes.

6.1 Snapshot replication This type of Database Replication is one of the simplest method to set up, and perhaps the easiest to understand. The snapshot replication method functions by periodically sending data in bulk format. Usually it is used when the subscribing servers can function in readonly environment, and also when the subscribing server can function for some time without updated data. Functioning without updated data for a period of time is referred to as latency.

For example, a retail store uses replication as a means of maintaining an accurate inventory throughout the district. Since the inventory can be managed on a weekly or even monthly basis, the retail stores can function without updating the central server for days at a time. This scenario has a high degree of latency and is a perfect candidate for snapshot replication.

Additional reasons to use this type of replication include scenarios with lowbandwidth connections. Since the subscriber can last for a while without an update, this provides a solution that is lower in cost than other methods while still handling the requirements.

Snapshot replication also has the added benefit of being the only replication type in which the replicated tables are not required to have a primary key. Snapshot replication works by reading the published database and creating files in the working folder on the distributor. These files are called snapshot files and contain the data from the published database as well as some additional information that will help create the initial copy on the subscription server.[5]

Snapshot replication is often used when needing to browse data such as price lists, online catalogs, or data for decision support, where the most current data is not essential and the data is used as read-only.

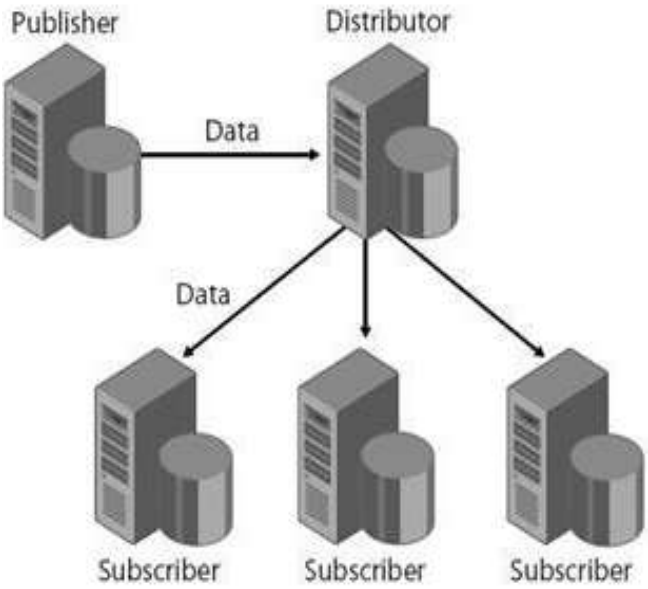

Fig.1 Snapshot Replication

Snapshot replication is helpful when: • Data is mostly static and does not change often. • It is acceptable to have copies of data that are out of date for a period of time. • Replicating small volumes of data in which an entire refresh of the data is reasonable.

6.2 Merging replication Merge replication is the process of distributing data from Publisher to Subscribers, allowing the Publisher and Subscribers to make updates while connected or disconnected, and then merging the updates between sites when they are connected.

Merge replication allows various sites to work autonomously and at a later time merge updates into a single, uniform result.

The initial snapshot is applied to Subscribers, and then changes are tracked to published data at the Publisher and at the Subscribers. The data is synchronized between servers continuously, at a scheduled time, or on demand. Because updates are made at more than one server, the same data may have been updated by the Publisher or by more than one Subscriber. Therefore, conflicts can occur when updates are merged.

Merge replication includes default and custom choices for conflict resolution that you can define as you configure a merge publication. When a conflict occurs, a resolver is invoked by the Merge Agent and determines which data will be accepted and propagated to other sites. Merge Replication is helpful when:

• Multiple Subscribers need to update data at various times and propagate those changes to the Publisher and to other Subscribers.

• Subscribers need to receive data, make changes offline, and later synchronize changes with the Publisher and other Subscribers.

• You do not expect many conflicts when data is updated at multiple sites (because the data is filtered into partitions and then published to different Subscribers or because of the uses of your application). However, if conflicts do occur, violations of ACID properties are acceptable.[1]

6.3 Transactional replication In what could be considered the opposite of snapshot replication, transactional replication works by sending changes to the subscriber as they happen. As an example, SQL Server processes all actions within the database using Transact-SQL statements. Each completed statement is called a transaction.

In transactional replication, each committed transaction is replicated to the subscriber as it occurs. You can control the replication process so that it will accumulate transactions and send them at timed intervals, or transmit all changes as they occur. You use this type of replication in environments having a lower degree of latency and higher bandwidth connections.

Transactional replication requires a continuous and reliable connection, because the Transaction Log will grow quickly if the server is unable to connect for replication and might become unmanageable. Transactional replication begins with a snapshot that sets up the initial copy. That copy is then later updated by the copied transactions. You can choose how often to update the snapshot, or choose not to update the snapshot after the first copy.

Once the initial snapshot has been copied, transactional replication uses the Log Reader agent to read the Transaction Log of the published database and stores new transaction in the distribution Database. The Distribution agent then transfers the transactions from the publisher to the subscriber.

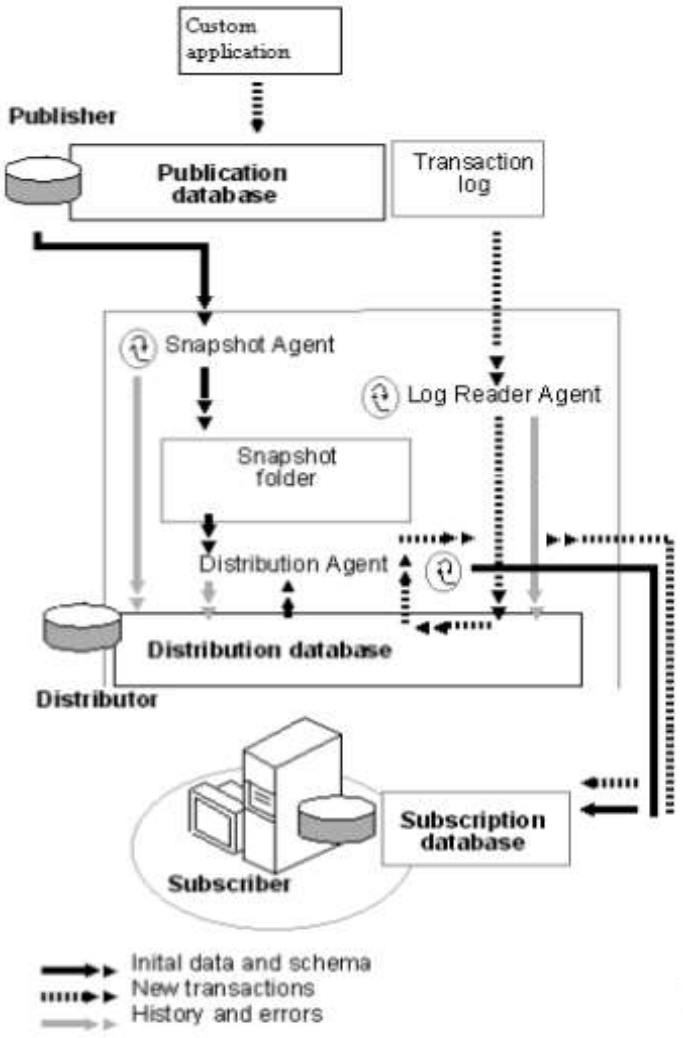

Fig. 2 How it Works (Transaction Replication)

Transactional replication with updating subscribers

An offshoot of standard transactional replication, this method of replication basically works the same way, but adds to subscribers the ability to update data. When a subscriber makes a change to data locally, SQL Server uses the Microsoft Distributed Transaction Coordinator (MSDTC), a component included with SQL Server 2000, to execute the same transaction on the publisher.

This process allows for replication scenarios in which the published data is considered read-only most of the time, but can be changed at the subscriber on occasion if needed. Transactional replication with updating subscribers requires a permanent and reliable connection of medium to high bandwidth. [5] Transactional replication is helpful when:

• You want incremental changes to be propagated to Subscribers as they occur.

• You need transactions to adhere to ACID properties.

• Subscribers are reliably and/or frequently connected to the Publisher.[6]

7. CONCLUSION

It is obvious that Database Replication it's not a very simple process but if applied in the right circumstances it can be an extraordinary solution for developing better applications, for improving performance and for better experience for users.

These advantages do not come without a cost. Data replication obviously requires more storage, and updating replicated data can take more processing time than updating a single object. In the same time, Database Replication can turn out to be complicated when it increases in size and magnitude but used properly, replication can improve considerably your data infrastructure.

Clients at the site to which the data is replicated experience improved performance because those clients can access data locally rather than connecting to a remote database server over a network and clients at all sites experience improved availability of replicated data. If the local copy of the replicated data is unavailable, clients can still access the remote copy of the data.

In a few words replication improves with availability and being highly distributed. Consider companies where users are disconnected during the day but need to update orders/inventories and other information automatically after normal working hours. Database Replication provides an easy solution when data must be highly distributed.

REFERENCES

[1] Microsoft MSDN Library - http://msdn.microsoft.com

[2] WikiPedia - http://www.wikipedia.com

[3] Community for sharing technology information - http://www.tech-faq.com/

[4] IT information, Introduction to Database Replication- http://www.informit.com

[5] Mark A. Linsenbardt, Shane Stigler "McGraw-Hill/Osborne Media book SQL Server 2000 Administration" - Chapter 10, 'Replication'

[6] Sql Server Library TechNet – Microsoft - http://technet.microsoft.com

[7] D. Agrawal, G. Alonso, A. El Abbadi, and I. Stanoi. Exploiting atomic broadcast in replicated databases. In Proceedings of EuroPar (EuroPar'97), Passau (Germany), 1997.

[8] P. Bernstein, V. Hadzilacos, and N. Goodman. Concurrency Control and Recovery in Database Systems. Addison-Wesley, 1987.

[9] P.A. Bernstein, D.W. Shipman, and J.B. Rothnie. Concurrency control in a system for distributed databases (sdd-1). ACM Transactions on Database Systems, 5(1):18–51, March 1980.

[10] A. Bhide, A. Goyal, H. Hsiao, and A. Jhingran. An efficient scheme for providing high availability. In Proceedings of 1992SIGMOD International Conference on Management of Data, pages 236–245, May 1992.

[11] A. W. Fu and D. W. Cheung. A transaction replication scheme for a replicated database with node autonomy. In Proceedings of the International Conference on Very Large Databases, Santiago, Chile, 199

[12] Daudjee, Khuzaima, and K. Salem. Lazy database replication with snapshot isolation. In International Conference on Very Large Data Bases (VLDB), pages 715–726, 2006.

[13] Elnikety, S. G. Dropsho, and F. Pedone. Tashkent: Uniting durability with transaction ordering for high-performance scalable database replication. In ACMSIGOPS/EuroSys European Conference on Computer Systems, pages 117–130, 2006.

[14] S. Elnikety, F. Pedone, and W. Zwaenepoel. Database replication using generalized snapshot isolation. In Symposium on Reliable Distributed Systems (SRDS), pages 73–84. IEEE, 2005.

[15] J. Gray, P. Helland, P. E. O'Neil, and D. Shasha. The Dangers of Replication and a Solution. In ACM