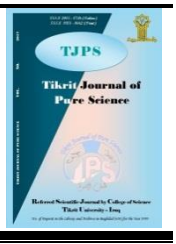
Tikrit Journal of Pure Science 23 (10) 2018

ISSN: 1813 – 1662 (Print)
E-ISSN: 2415 – 1726 (On Line)

TJPS

TIKRIT JOURNAL OF PURE SCIENCE

Journal Homepage: http://main.tu-jo.com/ojs/index.php/TJPS/index



A Case Study of Snapshot Replication and Transfer of Data in Distributed Databases

Saadi Hamad Thalij , Veli Hakkoymaz

Dept. of Computer Eng, Faculty of Electrical & Electronic Engineering, Yildiz Technical University, Istanbul, Turkey

ARTICLE INFO.

Article history:

-Received: 7 / 5 / 2018

-Accepted: 11 / 6 / 2018

-Available online: / / 2018

Keywords: Distributed Database, Replication, Snapshot Replication, Agents

Corresponding Author:

Name: Saadi Hamad Thalij

E-mail:
Sadi_aluhabi@yahoo.com

Tel:

Abstract

The database replication refers to distributing a database among multiple locations. There are three kinds of database replication system: snapshot, transactional and merge. The snapshot replication refers to the fragment of database items and distributing them to multi databases at once. An important goal in this paper is to experiment with a distributed database study the snapshot replication and examine the issues associated with it. In this work, the data from another database is used to increase availability and flexibility as well as provide the information exchange between databases. In this process, the data is infrequently updated at specified periods by copying and changing the data from the original database towards the subscriber database. The work of agents in this technology will do the most of the work to achieve the stated goal. The experimental results show that at both vertical and horizontal fragmentation, the proposed approach of replicating distributed database is efficient and the performance is significantly improved in terms of data transfer time, load sharing and update of database fragmentation. Hence the snapshot replication system is much schedulable and protective replication in business markets.

1. Introduction

Database replication is utilized on various database administration systems generally on a master/slave correlation between the actual and copied databases. Replication is defined as the copy and maintenance process of database objects namely tables to various databases in a distributed system of servers to improve the management process. Replication is generally a distributed technology used to store the data in multiple site locations to avoid data loss and increase the data access for the users. Though it utilizes distributed technology, it is not a distributed database but only called a replicated database. The reason is that the data in distributed databases is available in various locations but the table is positioned only at one location while the table is located in many locations in replication databases. Among the various types of replication, Lazy replication or asynchronous replication generally has better scalability and performance yet the outcome is not satisfactory due to low data consistency which results in failure to satisfy consistency requirement [1]. Replication is used to produce numerous copies of the data that can be distributed to multiple databases throughout an organization. Businesses typically use replication when they have a need to separate their transactional and reporting databases. Another reason for replication usage is that they want to minimize business interruption caused by server failure [2].

Database replication is performed to back up the data in second server in order to efficiently provide data to the requested users [3]. Storage of data can be done in different servers by partitioning the complete database into various parts called fragments and each fragment is placed at a different location. These fragments are the logical data points placed at different locational sites in a distributed system. The storage of data can be done nearer to the site where it is often utilized by clients and those not required by local applications are not needed to be stored at high priority. Concurrently, the fragments are split into distribution units and a transaction is split into sub-queries based on fragments [4]. The partition allocation scheme is considered as vital problem that enhances the performance of the processing applications in the distributed database systems

(DDBs). The queries related to database access the applications while also maintaining its performance. Henceforth the partitions that are accessed through queries are required to be assigned to the respective sites to minimize the cost of communication during the execution of applications while also handling their operations.

Previously, [3], [5], [9] have developed such replication strategies however the open challenge have increased the need for new and efficient models. In this paper, a methodology is presented for clustering the DDBS sites based on their communication cost for determining the allocation of the partitions to specified sites. The distributed database utilized in this work will be detailed in the experimental work section. The main objective of this work is to study and analyze the process and snapshot replication by elaborating the open challenges in clustering the DDBS and enhancing the data provisioning process. The performance will be improved better in terms of data transfer time, load sharing and update of database fragmentation [9].

2. A Brief Description of Replication Types

In general, the following replication types are supported by many distributed database management systems (DBMS): i) snapshot replication, ii) transactional replication, iii) merge replication. *Snapshot replication* is the simplest type of replication in which, all duplicated data (replicas) would be stored from the publishing database to subscribing database(s) at fixed durations. Snapshot replication is utilized for such replication applications that require infrequent variations on data and that the magnitude of replicated data is not very huge. In *transactional replication*, total database variations (insert, update, and delete statements) are captured and deposited in the distribution database. These variations are formerly forwarded to subscribers and utilized in the specified order itself [2]. The transactional replication is utilized mainly for such replication applications that data varies regularly and that the magnitude of replicated records is not small. Automatic variations of the replicated records on the publisher and on subscribers are not needed in transactional replication. *Merge replication* is considered as the utmost challenging replication type. It creates automatic variations to duplicated data on the originator and on the subscribers. Using merge replication, entire incremental data variations are seized in the base and in the destination databases. Clashes can occur while several updates are merged into a single copy in the database. It is defined as the attempt to change the same data records by multiple actors simultaneously. Since updates are made at more than one server, the same data may be updated by the publisher or the subscribers. Conflicts are resolved according to rules in configuration file or utilizing a customized resolver. For such applicators in which automatic variations of replicated data are supported on the publisher and on the subscribers, the merge replication is utilized [3].

3. The Recommend of Snapshot Replication

The snapshot is a point-in-time copy stored in the same volume used to record the status of whole data at the time the data is being taken. Snapshots use only a small amount of additional storage space and have insignificant impact on system performance. With the snapshots, if a user accidentally modifies or deletes data on a shared folder with snapshots, restoring the data back to the previous time at which the snapshot was taken can be done quickly. In addition, it allows users to recover their own deleted or modified files in shared folders without assistance from the administrator.

In snapshot replication, one can take manual (i.e., on-demand) snapshots, enable snapshot schedules, and perform snapshot restore and cloning operations. For scheduled snapshots, one can specify regular time intervals to take snapshot automatically [8].

There are certain conditions that require snapshot replication as opposed to other replication types: If single publisher-single subscriber and one to one business applications are in use, snapshot replication is recommended to use. If single publisher - multiple subscribers are the scenario and application is for distributing the price list, customers list, order list and so on, then transactional replication is recommended. In warehousing applications where, multiple publisher - single subscriber are available, then transactional replication is recommended. For online transactions where multiple publisher- multiple subscribers are deployed, then merge replication is recommended.

4. Data Fragmentation

The concept of data fragmentation permits one to split a single object into multiple partitions where the object includes user's database, system database or even a table. Every partition is positioned at any suitable site in a network and the details about partition storage are updated in the data catalog database design concepts (DDC). DDC can be accessed by the transaction processor (TP) to practice user requests. Totally, there are 3 kinds of data partitioning methods:

• ***Horizontal fragmentation*** is defined as the partition of a relation into smaller bits of tuples. Each partition is saved at different node locations with exclusive tuples. However these exclusive tuples or rows have same columns (attributes) i.e. each partition symbolizes the comparable SELECT statement with WHERE clause on a single column.

• ***Vertical fragmentation*** is defined as the partition of a relation into attributes or columns. Each partition is saved at different node locations with exclusive columns except the key column placed as a common partition.

• ***Mixed fragmentation*** is defined as the mixture of horizontal and vertical policies [6].

5. Data Replication and Allocation

Replication is vital in accessing the data while the most complex case of replication is distributive replication of the whole database at each site to form a complete replicated distributed database. This process can significantly increase the data availability as the users are able to access the data from the system until the last site is functioning. This process also increases the query retrieval performance on a global scale as all sites are able to process the queries locally even in distant locations as the site is connected towards the global server system. It must be noted that full replication has the disadvantage of slowing down the operation update process as even one logical update needs to be done on all replicas of the database to maintain consistency. This becomes more complex if there are multiple copies and it results in high expenses for the concurrency control and recovery methods.

The parallel extreme involves having no replication at all. In this case, each partition will be saved at only the same site in a disjoint manner excluding the repetition of primary keys in vertical or mixed partition strategies. This process is also known as non-redundant allocation [7].

Data allocation labels the practice of determining wherever to trace data. Data provision approaches are listed below:

(i) *centralized data allocation:* Complete database is placed at single site.

(ii) *partitioned data allocation:* Database is partitioned into multiple disjoint fragments and saved at multiple locations.

(iii) *replicated data allocation:* Replicas of one or more database segments are saved at many sites [6].

All replicas must be allocated to specified sites in the distributed system using the procedure called data provision or allocation. The selection of the site locations and the rate of replication is based on the performance and availability objectives of the framework. This procedure is also based on the type and frequencies of the transactions provided at each site. When most transactions end up as retrieval alone, the utilization of full replication is suitable while in case certain transactions accessing specified sections of database are placed at a specified site only, then the resultant set of replicas are set to be placed in that site. The sections of data that are accessed by users at many sites can also be replicated at those sites. In case, there are multiple updates needed to be done, the replication must be limited to reduce expenses and hence selecting an optimal replica number is required in distributed data provision often considered as composite optimization issue [7].

6. Implementation of Snapshot Replication

In order to implement replication: Firstly, 3 types of servers are required:

(i) *publisher*: The server which provides the source data is called as the publisher. The articles which are replicated are called as the *publications*.

(ii) *subscriber*: The server which receives the source data from the publisher is called as the *subscriber*. The articles which are received the replicated data are called as the *subscriptions.*

(iii) *distributor*: This server is responsible for distributing the data from the publisher to the subscriber.

It maintains the history of the replication process.

Snapshot replication is also called as static replication. It is only one-way replication (i.e., from publisher to subscriber). In snapshot publication, initially the snapshot agent will create the snapshot of the articles and initialize on the subscriber. Based on the defined schedule the snapshot agent recreates the snapshot of the articles or is replacing the articles on the subscriber. This is recommended for replicating small quantity of data.

Secondly, a snapshot agent is required: It is responsible for creating the snapshot of the publications. The snapshot will be stored in a specific folder in the distributor. The snapshot is created by using steps shown tn the following figure (Fig 1).

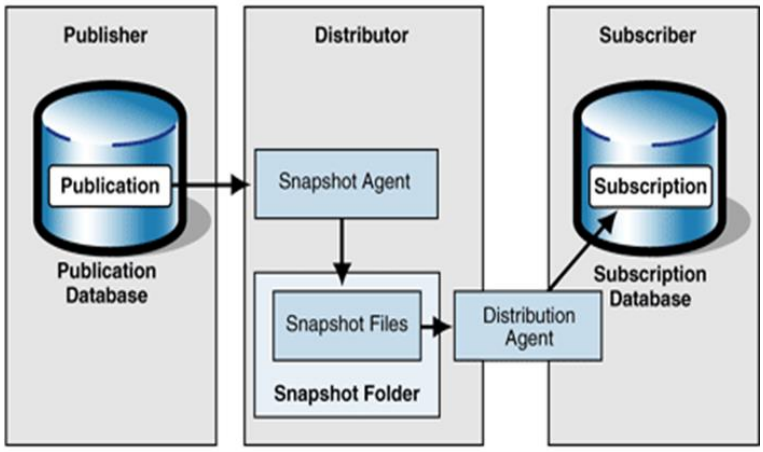


Fig.1. Snapshot replication agents and triggers

7. Experimental work

In this section, the case study for snapshot replication is presented. Firstly, the fragments and allocation database are used, then transfer data among places in the databases, and finally calculate the time of data transfer. Data replicas will be placed in the local server in the system; then fragment database. Afterwards, it is shared to other instance in the possibly same system or other. As the replication data are placed in the same server, updating the replica details is performed using the lock and release strategy. The number of replica is usually based on the number of queries that are able to access the same data components from the same server. Similarly, the computation cost is based on the load balancing, efficient resource allocation and waiting time of the query of the replica in the server. The main objective is balance the load by sharing them using the replica based on the computing resource utilization cost. A first step, a server (DESKTOP-33FQBBL) is used with multi database. In this situation, the measures are checked on two databases as shown in Table1.

Table 1: Calculation of transfer time

Location of Publisher	Location 0f Subscriber	Size of DB	#tables in DB	#Tables	Synchronization Time for Publisher
DESKTOP-33FQBBL	INSTANCE3	1468.00 MB	31	31	00:10,78
DESKTOP-33FQBBL	INSTANCE3	16.00 MB	18	2	00:07,03
DESKTOP-33FQBBL	122.175.53.112	1468.00 MB	31	12	00:13:07

As the second step, the load time is checked when full database is sent from server to single client as shown in Table 2.

Table 2: Load of transfer database

Publisher Name	Subscriber Name	Size of DB	#Tables	Synchronization Time for Publisher
SnapshotPublisherName	Snapshot Rep. Subscriber	1468.00 MB	31	00:09,06
Snap Rep. Pub	Snap Rep. sub.	11.13 MB	19	00:06.01

In the last step, multi rows are updated from publisher to subscriber using horizontal fragmentation and update multi columns from publisher to subscriber using vertical fragmentation as shown in Table 3**.**

Table 3: Update of fragmentation database

Server Name	Client Name	Name of DB	#Rows that Updated	Synchronization Time for Publisher	Type of Fragmentation
DESKTOP-33FQBBL	INSTANCE2	Snapshot Auto Transfer Data Subscriber	143	01:03,04	HORIZONTAL
DESKTOP-33FQBBL	INSTANCE3	Snapshot Horizontal Fragm. Sub,	160	00:23,02	Vertical

8. Conclusion

Snapshot replication is used to update, transfer and synchronize data of databases between two locations. The replication system consists of three components, namely, a publisher, a distributor and one subscriber for each snapshot. Initially, the original data is transferred from one publisher to single subscriber. A bunch of database operations (insert, delete and update) are applied on fragments of the database in the subscribers and ask the replication system to synchronize. Then the performance of the replication system is evaluated in terms of data transfer time, load sharing and update of database fragmentation.

References

[1] Z. Sann, T. T. Soe. (2017) . Agricultural Loan System Using Data Replication Method. Int'l. J. of Scientific Research in Computer Science. Engineering and Information Technology. (IJSRCSEIT). V. **2**. no. **5**. ISSN: **2456-3307**.

[2] ESRI Team. (May 2004). Technical Paper. Using Microsoft SQL Server Snapshot.

[3] Veli Hakkoymaz, Saadi Hamad Thalij. (Dec. 2017). Dynamic Data Distribution for Merge Replication in Databases. IOSR J. of Computer Engineering (IOSR-JCE) e ISSN: **2278-0661**, P-ISSN: **2278-8727**, v **19**, no. **6**, pp **41-46**.

[4] Available Online (Accessed:28May2018). Computer Notes. http://ecomputernotes.com/database-system/adv-database/fragmentation.

[5] Bhuyar P,R. . Gawande A.D. (2012). Distributed Database: Fragmentation and Allocation. J. of Data Mining and Knowledge Discovery ISSN: **2229–6662** & ISSN: **2229–6670**. v. **3**, no.**1**., pp.-**58-64**.

[6] Carlos Coronel, Steven Morris. (2011). Database Systems: Design, Implementation, & Management", 9th edition, ISBN-**13**:**978-0-53874884-1**.

[7] Available Online (Accessed: 19 May 2018), Computer Notes,

https://www.boddunan.com/articles/education/22-computer-science/9391-data-fragmentation-replication-and-allocation.html.
[8] Synology NAS Product. (2014): Data protection with synology snapshot technology. White Paper. www.synology.com.
[9] Lena Wiese. (2014). Clustering - based fragmentation and data replication for flexible query answering in distributed databases", Wiese J. of Cloud Computing: Advances, Systems and Applications, Springer,

دراسة حالة تناسخ اللقطة ونقل البيانات في قواعد البيانات الموزعة

سعدي حمد ثلج ، ولي هاكويماز

قسم هندسة الحاسوب ، كلية الهندسة الكهربائية والإلكترونية ، جامعة يلدز التقنية ، اسطنبول ، تركيا

الملخص

تناسخ قواعد البيانات يوفر توزيعاً لقواعد البيانات عبر عدة مواقع مختلفة.

وهنالك ثلاثة انواع من تناسخ قواعد البيانات وهي تناسخ اللقطة وتناسخ العملية وتناسخ الدمج.

تناسخ اللقطة يوفر لنا عملية تقسيم البيانات في قواعد البيانات وبعدها يقوم بعملية التوزيع البيانات الى مواقع مختلة في قواعد البيانات.

الهدف من هذا البحث هو العمل على دراسة توزيع البيانات على جميع قواعد البيانات في النظام واختبار ذلك التوزيع.

في هذا العمل اختيرت تقنية تناسخ قواعد البيانات اللقطة الذي سوفر لنا تناسخ للبيانات بشكل فوري وذلك اشبه بالتقاط صورة للبيانات ونسخها الى قواعد البيانات الاخرى بشكل مباشر وهذا العمل يوفر لنا السهوله في تناقل البيانات وكذلك يجعل البيانات متوفره في النظام بشكل دائم.

في هذه العملية تتم تجزئة قواعد البيانات الى جزئين افقي وعمودي وهذا العمل يوفر لنا تقليل الضغط الذي يحصل على قاعدة البيانات الرئيسيه ويك ون الاداء عندها منتظماً وتقل الاحمال في تناقل البيانات والبحث في جميع قواعد البيانات وعندها سيقتصر البحث على اماكن محدده فقط.

حيث ان هذه التقنيه تعمل على حماية البيانات من العبث والتلف وتعمل على سرعه في الاستجابه للطلب

ولذلك السبب تكون اكثر التقنيات استخداماً في الاسواق العالمية.