An Approach for Designing and Implementing Eager and lazy Data Replication

Mohamed Osman Ali Hegazi
Department of Computer Science, College of Computer Engineering and Sciences, Salman Bin Abdulaziz University, Alkharj, Saudi Arabia
Department of Computer Science, Faculty of Computer Science and Information Technology, Alzaeim Alazhari University, Khartoum North, Sudan

ABSTRACT
Replication can be a success factor in database systems as well as perhaps being one of the needs of proliferation, expansion, and the rapid progress of databases and distributed technology, despite there being a strong belief among database designers that most existing solutions are not feasible due to their complexity, poor performance and lack of scalability. This paper provides an approach that can help designers in implementing eager and lazy replication mechanisms. The proposed approach contains two phases: In the first phase, the database is designed to have indicator fields that can carry the update status, and to consider the replication concepts by classifying, categorizing and determining the kinds and locations of data objects; in the second phase, the updating methodology is provided to make the implementation of eager and lazy replication mechanisms easier and reliable.

General Terms
Computer Science, Database, Distributed Database System.

Keywords
Replication, database, eager replication, lazy replication, distributed database.

1. INTRODUCTION
Replication is the storing of copies of the same data in more than one location (site) and then consistently updating these copies. There are two basic parameters to be selected when designing replication data: when and where. According to GRAY, J., HELLAND, P., O'NEIL, P., AND SHASHA, D. [4] these mechanisms can be categorized according to when updates are propagated and which copies can be updated [5].

Controlling the updating of replicated data is undertaken by two mechanisms, eager and lazy. In eager applications, updating can be carried out within the transaction boundaries, while in lazy replication updating can be undertaken outside the transaction boundaries.

Data replication can be one of the success factors for database systems because it can improve performance by eliminating remote access and it can improve fault tolerance by increasing availability. In addition, data replication is needed to provide information sharing for the continuity of business, organizations, and applications. Data replication can also be one of the factors serving the technology revolution and the explosion in data growth by providing data everywhere.

A large number of existing protocols provide data consistency and fault tolerance in data replication but, according to Kemme and Alonso [5] few of these ideas have ever been used in commercial products. There is a strong belief among database designers that most existing solutions are not feasible due to their complexity, poor performance and lack of scalability. As a result, current products adopt a very pragmatic approach: copies are not kept consistent, updates are often centralized, and solving inconsistencies is left to the user [5].

This paper presents an approach that can help the designer to design and implement a consistent replication database and make the implementation of eager and lazy updating mechanisms easier and more reliable.

2. RELATED WORKS
Much scientific research has dealt with providing solutions for database replication, with most of it dealing with developing solutions to provide consistency, the ability to serialize and/or fault tolerance in replicated data by providing functions or protocols for controlling the process for replicated data, for example [12], [5],[6], [2] and [7]. However, most of these solutions still work on controlling the process at the middleware level; they do not provide a solution at the design and implementation levels. Even those which have tried to obtain a solution at the design level or those that tried to implement the replication technique have not provided a standard solution that can help the designer in designing consistent replication data. Most of the work is concerned with special kinds of systems or environments, specific projects or web applications, for example [11], [8] and [10]. In the context of commercial databases, we found that most of the modern software products, such as PostgreSQL, MySQL and Microsoft SQL Server, use asymmetric replication at the middleware level stage by providing functions for saving the modifications using transactions to carry this out, and then use these amendments as inputs to the rest of the copies (see [3]). Oracle works with the same idea, but through a special protocol (replication protocol) instead of functions in Oracle10g, or through Materialized View and administrative tools in Oracle advance replications Oracle11g [9].

The contribution of this paper is in providing a simple and consistent method to design and implement replication mechanisms. The approach presented provides a solution for the complexity of replication design, and provides an implementation methodology that can lead to obtaining feasible and reliable systems.

3. THE PROPOSED APPROACH
The proposed approach contains two phases. In the first phase, the database is designed to have indicator fields that can carry the update status, and also to consider the replication concepts by classifying, categorizing and determining the kinds and locations of data objects. In the second phase, an updating

methodology is provided to facilitate the implementation of eager and lazy replication mechanisms.

3.1 The Design Phase

The design algorithm:

1- First: Create indicator fields for each replicated data (updated flag and counter).These fields will be used to carry the update situations.

2- Second: Determine the location. The location of the replicated data is determined according to the following (Figure 1):

   a. The location of the author system (or user). The place in which the data are created or initialized.

   b. The locations of the copies. The other places in which the data are replicated.

3- Third: Categorize the data. The replicated data is categorized as follows:

   a. static data(not frequently updated),

   b. data updated in one place and read only in other places,

   c. frequently updated data or data that is updated from many locations.

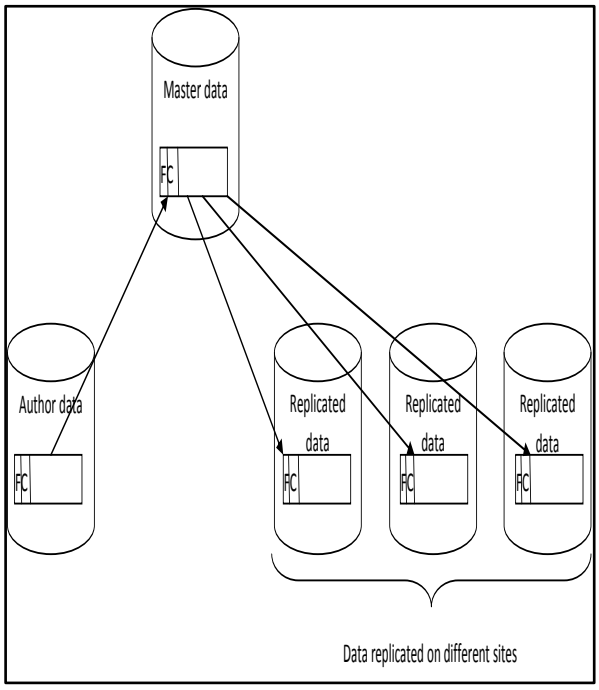

Figure 1: The design concept for the proposed approach

4- Fourth: Apply the data replication constraints. If the replicated data needs to be frequently updated or updated from many locations, then one of the following solutions will be used:

   a. either add additional fields to carry a lock flag, and then keep the master data in a global place.

   b. or use linking techniques to link the data on all sites; Figure 2 shows an example of linking global or remote data inside Microsoft Access.

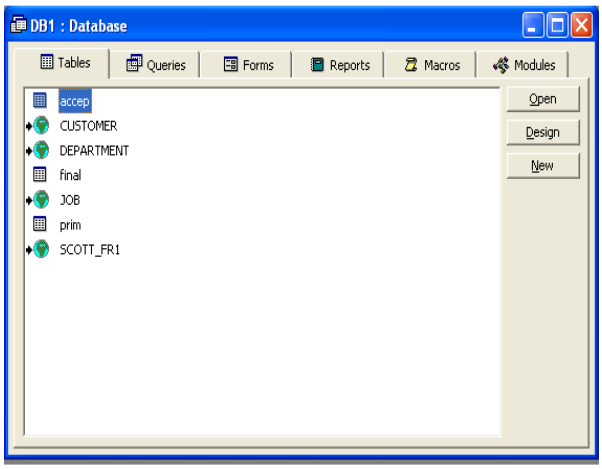

Figure 2: An example of linking global or remote data inside local data.

3.2 The Updating Phase

3.2.1 Eager Replication

This approach provides three ways of implementing eager replication:

3.2.1.1 Eager replication algorithm for infrequently updated data or for data that is updated from only one location

First: Each replicated data has master data in the system's database. Each master data has an indicator field (counter) in which the last update is recorded.

Second: At each location, where these data are replicated, the indicator field (counter field) is then checked by the user transaction and compared with the field of the main system data. If these fields are different then the transaction updates the local data before proceeding. Otherwise, the user transaction goes ahead without any updating to the local data. In both cases the transaction finally updates the master data as seen in Figure 3.

```
if master data IndicatorField> the local data IndicatorField
then overwrite the local data
end if
Do Local transactions & operations
Update master data
Master IndicatorField= Master IndicatorField +1
Local IndicatorField= Master IndicatorField
```

Figure 3: Eager replication algorithm for infrequently updated replicated data

3.2.1.2 Eager replication algorithm for frequently updated data or data that is updated from many locations

First: Each replicated data has master data in the system's database. Each master data has an indicator field to carry the last update counter and a flag field to lock the data.

Second: A copy of the replicated data is stored in a global location (primary copy approach).

Third: At each location, where these data are replicated, the local transaction first checks the lock flag on the global data

(master data); if the flag is on then the transaction waits otherwise the transaction changes the field to on and compares the indicator fields (updated count field); if there is no difference between the local field value and the global field value then the transaction will carry out the update process, otherwise the transaction will update this local data first (by overwriting this local data) and then carry out the updating process, see Figure 4.

```
If LockFlag on then do waiting loop until LockFlag off
end if
LockFlag=on
if master data IndicatorField> the local data IndicatorField
then overwrite the local data
end if
Do Local transactions & operations
Update master data
Master IndicatorField= Master IndicatorField +1
Local IndicatorField= Master IndicatorField
LockFlag=off
```

Figure 4: Eager replication algorithm for frequently updated replicated data or for data that are updated from any replicated site

3.2.1.3 Using the linking approach

Another way of applying eager replication for all kinds of data is by using linking techniques, by which all data will be linked on every site, then all the local systems will work directly on the main system data through the link layer and any updating for these data will be effective on all locations.

Linking can be applied either by using one of the DBMS linking techniques such as import and export (example Figure 2) or by using connection tolls or functions such as programming language code.

3.2.2 Lazy replication

According to our proposed approach, lazy replication can be implemented using the following algorithm:

1- After the master system has finished updating the master copy it then increments the indicator field (the account field).

2- Any process on replicated sites compares the indicator field with the master data field; if the value of its field is less than the value of the master data field it then overwrites its local copy before undertaking any transaction, otherwise it proceeds with its transaction without making any changes to the replica data.

3- Usually in the lazy approach, the replicated data is updated from one site (master system). But if any site needs to do any updating for the replicated data then it must first check if its local copy is up-to-date using step 2 above, after that it can update the global replica data using the following algorithm:

- First update the local data.
- Once the local updating is finished overwrite the master copy.
- Increment the master copy indicator field and let the local field equal that field.

3.2.3 Join replications

According to our proposed approach, we can join the two replication techniques by using the following algorithm:

(1) Once the author's system has finished the transaction on the data, it then updates the main data located in the global area.

(2) Any one of the other subsystems that needs to use this data goes directly to the global area and overwrites its local data (bringing this data onto its machine). There is no need for the queries to check whether or not the data is updated.

The join algorithm in this approach is for the kind of data that does not need to be frequently updated.

4. RESULTS AND DISCUSSIONS

This approach provides a mechanism that can facilitate the implementation of replication databases for the designer; hence it can be applied at the design stage without restrictions on the kind of DBMS, the number of copies, or/and the locations of the replica data. Therefore, it may be suitable for developing large database systems or unifying a heterogonous database system.

The approach supports both eager and lazy replication mechanisms. In addition, it can join these two techniques. This provides more flexibility; hence we can apply the most suitable technique to the particular data. In practice, the lazy mechanism is easily implemented for infrequently updated data, while eager replication may be more suited to data needing frequent updating; in addition, joining these two replication techniques can be easily implemented and also helps in some operations, such as coordination among faraway systems.

The updating techniques presented in this paper provide an approach that deals with one master table and the transaction either, in lazy replication, uses a primary copy approach in which all updates are first performed on a primary copy (the master data) and are then propagated to the secondary copies by overwriting the existing copies, or by just updating the corresponding fields, or, in eager replication, uses the updating everywhere method, in which the user's transaction updates all copies (everywhere) of the data immediately after the updating of the master data is completed.

The approach presented in this paper provides a different concept for applying the replication mechanism, hence most of the scientific work focuses on providing a solution at the middle layer, or providing a solution for a special environment. For example, this approach provides a solution that can help the designer implement replicated data starting from the conceptual design through to the implementation and handling of the updating of the replicated data.

Despite most DBMS providing a mechanism for implementing replication, its tools needs to be used by the system designer, besides most of these tools are functions or protocols working mainly to maintain the consistency of the replica data; for example, Oracle stream replications manage transaction replication and SQL Server is used in transaction replication for the same purpose and both of them have snapshot replication for merging the replication or for capturing the replication process (or Materialized View on Oracle Advance [9]) [1]. The approach presented in this paper, in addition to providing a solution starting from the design level, also provides a mechanism that can make use of these tools in such a way that it is easy for the designer to

implement data replication. Also, this approach can work on integrated data and develop replication of heterogonous database systems which cannot be provided by the tools of one kind of DBMS.

5. REFERENCES

[1] Ashok, G. & Randal, P. S. (2011) SQL Server replication: Providing high availability using database mirroring. © 2011 Microsoft Corporation.

[2] Cecchet, E.,Ailamaki, A., & Candea, G.(2008)Middleware-based database replication: The gaps between theory and practice. SIGMOD '08 Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data.

[3] GORDA, deliverable D1.1 report (2006) State of the art in database replication. GORDA is specific targeted research Project Group for Open Replication of Database. Supported by the European Community under the Sixth European Union Framework Programs for Research and Technological Development Coordinator: University of Minho

[4] GRAY, J., HELLAND, P., O'NEIL, P., & SHASHA, D. (1996). The dangers of replication and a solution. SIGMOD '96 Proceedings of the 1996 ACM SIGMOD international conference on Management of data. Pages 173-182. ACM New York, NY, USA

[5] Kemme, B. & Alonso, G. (2000) A new approach to developing and implementing eager database replication protocols.ACM Transactions on Database Systems, 25(3), p. 333.

[6] Manassiev, K. & Amza, C. (2005) Scalable database replication through dynamic multiversioning. CASCON '05: Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative Research, IBM Press, October 2005.

[7] Milan-Franco1, J. M., Jim´enez-Peris, R.,Pati˜no-Mart´ınez, M., &Kemme, B. (2004) Adaptive middleware for data replication. LNCS 3231, IFIP International Federation for Information Processing, pp. 175–194.

[8] Nithiyalakshmi, P. & Kumar,V. U. (2014)Data consistency for cooperative caching in mobile environments. International Journal of Science and Research (IJSR) 3(1), January 2014.

[9] Oracle(2008) Oracle database advanced replication, 11g Release 1 (11.1) B28326-03, 2008, Oracle. Primary Author: Randy Urbano

[10] Plattner, C.& Alonso, G. (2004) Scalable replication for transactional web applications. In Proc. of Middleware, 2004.

[11] Sivasubramanian. Swaminathan, Alonso. Gustavo , Pierre. Guillaume, and Steen. Maarten van. 2005. GlobeDB: Autonomic Data Replication for Web Applications. International World Wide Web Conference (WWW 2005), May, 2005, Chiba, Japan. ACM

[12] Sleit. Azzam, AlMobaideen. Wesam, Al-Areqi Samih, and Yahya. Abdulaziz. (2007) A dynamic object fragmentation and replication algorithm in distributed database systems. American Journal of Applied Sciences 4 (8): pp.613–618.