Boosting the Availability of Information Systems by Data Replication

J. E. Armendáriz[1], H. Decker[2] and F. D. Muñoz-Escoí[2],

[1] Dpto. de Matemática e Informática, Univ. Pública de Navarra, 31006 Pamplona, Spain
[2] Instituto Tecnológico de Informática, Campus de Vera, 46022 Valencia, Spain
Email: *enrique.armendariz@unavarra.es,* {*hendrik, fmunyoz*}*@iti.es*

Abstract. High availability of information in medium and large-sized enterprises may naturally benefit from a wide-area replication of data. We discuss recent developments of replication architectures that boost the availability of business data in distributed information systems.

1 Introduction

The business of medium and large-sized enterprises is statically structured by branches settled in distributed locations. It is structured dynamically by activities taking place in geographically dispersed areas. Both the network of branches and the work and negotiations of field staff and customers typically range over widespread areas, possibly across countries and continents. High availability of information is key to a successful business. Therefore, it seems natural to deploy a networked IT infrastructure in such enterprises which matches this distributed business structure.

Apart from the obvious analogy between the distributed wide-area network structures of the business world and IT, the main motivation to invest in replication is to boost the availability of information. Business services, in particular those offered via web interfaces or other IT network infrastructure (e.g., remote database access), are confronted with the need to raise their availability, performance and failure-resilience, primarily for competitive reasons and for improving customer satisfaction. We believe that data replication is a very promising means of boosting the availability, performance and failure resilience of business information systems. Here, replication does not just consist of redundant hardware or backup copies. Rather, it involves a fully transparent on-line distribution of copies of databases or parts thereof, for sites that are dispersed over possibly very wide areas. In particular, it includes a wide range of protocols for replication and failure recovery as well as policies to trade off requirements of availability and consistency of replica in transaction-intensive systems such as OLTP databases.

Two major obstacles need to be overcome when introducing replication for improving the availability, performance and failure resilience of services with, within and between diverse business units. The first is the error-prone complexity of developing protocols for replication and failure recovery, as well as the overhead produced by such protocols for maintaining the consistency of replicated data [1]. For business-critical applications, this can easily amount to a severe impediment. The second hurdle which may prevent the use of replication for business information systems is that different users use different services and have different requirements on the consistency and availability of business data. For instance, for the strategic planning of an airline enterprise, the statistics delivered by a data warehouse typically do not rely on most recent updates to business data, nor do they need them all the time. Rather, business report services usually refer to selected database states reached cyclically at well-defined breakpoints, without taking into account currently ongoing transactions or intermediate states. On the other hand, information services for flight schedules and reservations need to keep track of up-to-date database states. In general, different classes of business users and services typically have different requirements on the accuracy, replication consistency, timeliness and availability of the underlying data. However, in practice, at most just a single manner of replication, a fixed consistency maintenance scheme and a uniform policy for availability and failover management is supported. A more flexible replication architecture which can be adapted to the changing needs of different business services and users is therefore desirable.

In this paper, we sketch a new middleware architecture for enhancing the availability, fault tolerance and consistency of business information systems in the framework of a distributed infrastructure, as part of the solicited project CONFIA. It is destined to overcome the two obstacles mentioned above. With regard to the first, standard SQL constructs (views, triggers etc) and ready-made SQL functionality (schema definition, trigger firing etc) are used for implementing major parts of the meta data handling and the network communication of the protocols. That way, the protocols themselves become much less cluttered and thus much easier to develop and implement. With regard to the second, CONFIA simultaneously maintains meta data for several protocols, so that the replication strategy can be configured and re-configured seamlessly. Suitable protocols can be chosen, plugged in and exchanged on the fly in order to adapt to the actual needs of given situations. In the remainder, we are going to describe the basic idea of the CONFIA architecture, emphasising protocol exchangeability.

2 Architecture

Similar to its predecessor MADIS [4], the CONFIA architecture is two-layered, as shown in fig. 1. It makes consistency management independent of any DBMS particularities. CONFIA takes advantage of ready-made database resources so that protocol overhead is kept to a minimum, as described in [7]. The replication strategy is configurable on the fly, i.e. the architecture allows for plugging in suitable protocols that fit given situations, i.e., the particular business needs of different users, use cases and applications, possibly depending on the current network load, device-specific bottlenecks and other operational parameters) best.

The upper layer of the architecture consists of replication management functions, while the lower one of database schema extensions for storing and processing transaction data meta data of protocols for replication and failure recovery. The upper layer handles transaction requests from users or applications and uses meta tables on the lower layer for transparent replication management. Hence, the upper layer can be implemented in any programming language with a SQL interface.

Some of the meta data tables on the lower layer account for transactions in the local node (as symbolically shown in fig. 1), and are updated and maintained within the

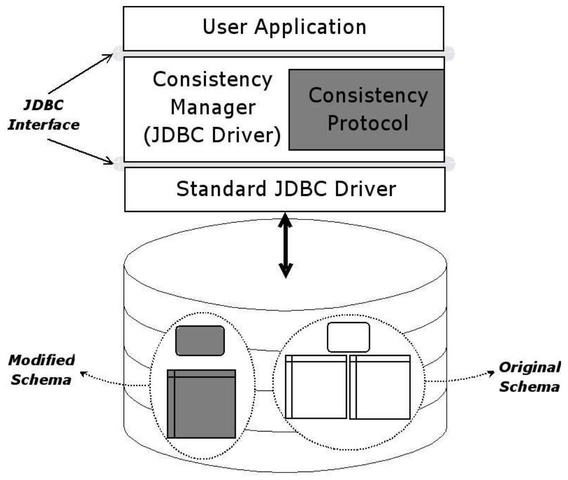

Fig. 1. Layered CONFIA architecture.

transactions accounted for. Communication between database replicas is controlled by the CM which is local to each network node.

The schema extension also includes some stored procedures which serve to hide some table extension technicalities to the upper layer. The latter is sandwiched between client applications and database, acting as a database mediator. Accesses to the database as well as commit/rollback requests are intercepted, such that the replication protocol can work with full transparency. The protocol accesses the report tables to obtain information about transactions, in order to cater for required consistency guarantees. The protocol may also manipulate the extended schema using stored procedures.

Of course, the performance of such a middleware will always tend to be somewhat worse than that of a core-based solution, which integrates replication functionality into an existing database kernel implementation such as Postgres-R [8]. But the advantage of our middleware solution is to be independent of the underlying database and to be easily portable to other DBMSs.

As already emphasised, implementation of the consistency manager CM, i.e., the core of CONFIA, is independent of the underlying database. In [7], we have described a Java implementation with a common JDBC driver interface. Its functionality is fully transparent to users and applications. The CM handles transaction requests, including multiple sequential transactions in different JDBC consistency modes, and communicates with database replicas. It facilitates the plugging and swapping of replication protocols chosen according to given needs and requirements. Protocol swapping, even at runtime, is seamless and fast, since the meta data for each protocol in the CONFIA repertoire (e.g., protocols with eager or lazy update propagation, optimistic or pessimistic concurrency control, etc) is readily at hand at plug-in time. Characteristics of different kinds of protocols are surveyed in more detail in [3].

3 Related Work

COPLA and MADIS [4] are two predecessors of CONFIA with likeminded objectives. The first provides support for different protocol strategies, but is configurable only off-line. The second provides on-line support for protocol swapping, though not for simultaneous users and applications. CONFIA strives to overcome these shortcomings.

In general, replication approaches can be classified as *middleware-based* (such as CONFIA), *trigger-based*, *shadow-table-based* and *timestamp-based* Pros and cons are discussed in [11,12]. Most solutions other than middleware-based ones, e.g. [9,13], depend on the underlying DBMS, which entails problems of maintenance, migration and interoperability. Middleware architecture such as [10] do not support protocol swapping, nor do replication support systems offered on the market, such as Progress DataXtend RE (a.k.a. PeerDirect) [11] ORION Integrator [5] and C-JDBC [2].

4 Conclusion

Requirements of high availability of information in medium and large-sized enterprises may naturally benefit from a distributed replication of data. We have discussed a recent development of a replication architecture for boosting the high availability of distributed business data. Thus, it may serve as an infrastructure for effectively aligning business and information system structures. In general, different business services and users require different kinds of replication strategies. Hence, an adequate choice of appropriate protocols is due. Hence, a middleware which provides flexible support for choosing, plugging in, operating and exchanging suitable protocols is desirable for many applications. This innovative kind of pluggability is being realized in CONFIA.

References

1. J. Gray, P. Helland, P. O'Neil, and D. Shasha: The dangers of replication and a solution. *SIGMOD*, 173-182, 1996.
2. *http://c-jdbc.objectweb.org*, as of 5 May 2006.
3. F. Muñoz, L. Irún, H. Decker: Database Replication Protocols. *Encyclopedia of Database Technologies and Applications*, 153-157, Idea Group, 2005.
4. *http://www.iti.upv.es/madis/*, as of 5 May 2006.
5. *http://www.orionintegrator.com*, as of 5 May 2006.
6. L. Irún, J. Armendáriz, H. Decker, J. González de Mendívil, F. Muñoz: Replication Tools in the MADIS Middleware. *Proc. VLDB'05 Workshop DIDDR*, 25-32, 2005.
7. L. Irún, H. Decker, R. de Juan, F. Castro, J. Armendáriz, F. Muñoz: MADIS: A Slim Middleware for Database Replication. *Proc. 11th Euro-Par*, LNCS 3648, 349-359, 2005.
8. B. Kemme: *Database Replication for Clusters of Workstations*. PhD, ETH Zurich, 2000.
9. B. Kemme, G. Alonso: A Suite of Database Replication Protocols based on Group Communication Primitives. *Proc. Distributed Computing Systems*, 156-163, 1988.
10. Y. Lin, B. Kemme, M. Patiño, R. Jiménez. Middleware Based Data Replication Providing Snapshot Isolation *Proc. SIGMOD*, 419-430, 2005.
11. Overview & Comparison of Data Replication Architectures. *Peer Direct*, 2002.
12. Replication Strategies: Data Migration, Distribution and Synchronization. *Sybase*, 2003.
13. S. Wu, B. Kemme: Postgres-R(SI): Combining Replica Control with Concurrency Control based on Snapshot Isolation. *Proc. IEEE ICDE*, Tokio, Japan, Apr. 2005.