Introduction
Ensuring that data is in “the right place at the right time” is increasingly critical as the database has become the linchpin in corporate technology infrastructure driving customer interactions, revenues, and decision making. The customer-facing OLTP servers have real-time data critical to the decision making process supported by the OLAP servers; the challenge is how to make the OLTP data available where and when it is needed. Database replication is a solution for distributing data that has been widely studied for over 20 years providing a mechanism to increase performance and availability as well as to distribute data to support business requirements. However, replication also creates “a large and complex distributed system with intricate dependencies and hidden communications channels” (Konstantinos Krikellas, 2010) requiring experienced DBAs for maintenance and support.

The goal of this paper is to provide a foundation for understanding data replication as well as a discussion of the criteria for selecting an appropriate replication technology.

Background
“Database replication is the process of maintaining multiple copies of data items in different locations called replicas” (Bettina Kemme, 2010). Distributing data to disparate servers has been a research topic for more than two decades during which time multiple approaches to maintaining transactional atomicity, consistency, integrity, and durability (ACID) have been investigated. “Ten years ago, the theoretical basis for database replication revolved around the classic concepts of serializability and locking” (Bettina Kemme, 2010). In these schemes ACID was ensured using a 2-phase-commit protocol where locks were obtained for read operations while write operations used distributed locks. The overhead costs of locks lead some researchers to suggest quorums as a more efficient approach for ensuring the consistency of distributed data; however, many operational issues were identified with these approaches. The seminal paper Dangers of Replication and a Solution (Jim Gray, 1996) suggested that there was an exponential relationship between the number of replicas and “the transaction response times, the conflict probability, and the deadlock rates” (Bettina Kemme, 2010). Research conducted by Jim Gray led to new proposals for replication that eased the restrictions for consistency in order to increase responsiveness.

Early research in replication, as described above, is commonly referred to as “eager” or synchronous replication “which keep the replicas synchronized within transaction boundaries” (Christian Plattner, 2004). “Eager systems conform to 1-copy-serialization:
the resulting schedules are equivalent to a serial schedule on a single database” (Christian Plattner, 2004). Confirming research by Jim Gray, Plattner and Alonso determined that eager replication results in very high communication overhead and a probability of deadlocks “proportional to the third power of the number of replicas” (Christian Plattner, 2004). These issues led to a revised view in the database community “that one could get either performance by sacrificing consistency (lazy replication approaches) or consistency at the cost of performance and scalability (eager replication approaches)” (Bettina Kemme, 2010).

In contrast to eager replication, lazy or asynchronous replication “propagates the updates of a transaction once it has already committed” (F. D. Muñoz-Escoí, 2009). Lazy replication improves performance allowing faster transactional rates; however, this is achieved by sacrificing replica consistency a choice that was not well-received by some in the research community who found it difficult to accept “the fact that it was possible to implement (and commercialize!) a system with such ill-defined consistency guarantees…” (Bettina Kemme, 2010). It was, in fact, the need to commercialize a solution that led to the decision to implement lazy replication as the preferred solution in the real-world notwithstanding the issues of potentially stale data at the replicas as well as the need to implement mechanisms to resolve update conflicts (Christian Plattner, 2004).

**SQL Server Replication**

Microsoft implements the Replication subsystem in SQL Server using an asynchronous (lazy) replication solution that provides a robust conflict resolution mechanism. As with any lazy replication implementation there is a possibility of stale data at the replicas; however, the consistency tradeoff is well-worth the achievable performance of SQL Server Replication. Additionally, the Replication subsystem includes tools to manage data consistency across the replicas allowing for a high degree of confidence in the data.

There are many excellent references available describing the architecture and components of the SQL Server Replication subsystem; this paper will not include redundant discussion of these topics. Instead, the remaining sections of this paper will discuss the criteria for selecting a replication mechanism.

**Replication Scenarios**

Replication is used for “two complimentary features: performance improvement and high availability” (F. D. Muñoz-Escoí, 2009). Performance can be increased with replication because each replicated server can respond to queries as read-only replicas requiring no coordination among the servers; this approach may also provide server redundancy where the failure of any one replica does not impact data accessibility. SQL Server supports this configuration with its Transactional replication mechanism.
Additionally, for replication configurations requiring replicas to update their local data SQL Server provides a Merge replication mechanism with default and customizable conflict resolvers available to meet any business logic need. SQL Server also provides a Snapshot replication mechanism which creates a point-in-time read-only copy of data on the replica. For clarity, the population of decision support systems is encompassed within the performance improvement feature as defined by Muñoz-Escoí.

A major consideration when selecting a replication mechanism is the data modifications and data-type limitations required to support the internal replication mechanisms. Transactional and snapshot replication mechanisms have no intrusive data modification requirements; e.g. the former manages data distribution via the transaction log while the later creates and applies a point-in-time copy of the replicated data. Transactional replication does require that all tables have a primary key defined; additionally, large object types require changes to replication configuration parameters. Merge replication, in contrast, requires data augmentation in order to synchronize data updates across replicas. The major Merge replication requirements and limitations are listed below:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEXT, NTEXT, or IMAGE data types are not replicated.</td>
<td></td>
</tr>
<tr>
<td>Any Foreign Keys in the database must be labeled as NOT FOR REPLICATION</td>
<td>to avoid conflicts.</td>
</tr>
<tr>
<td>Any User Defined Triggers must be labeled as NOT FOR REPLICATION to avoid</td>
<td>conflicts.</td>
</tr>
<tr>
<td>IDENTIFY columns must be labeled as NOT FOR REPLICATION.</td>
<td></td>
</tr>
<tr>
<td>IDENTIFY values must be managed on a site-by-site basis.</td>
<td></td>
</tr>
<tr>
<td>There are limits associated with Merge Replication regarding the amount of</td>
<td>data that may be replicated as measured by row and column size.</td>
</tr>
<tr>
<td>TIMESTAMP values are not replicated they are regenerated.</td>
<td></td>
</tr>
<tr>
<td>Every replicated table must have a UNIQUEIDENTIFIER defined.</td>
<td></td>
</tr>
<tr>
<td>Data inserted using BULK INSERT commands will not be replicated.</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Merge Replication Requirements and Limitations

In my experience there are several scenarios where data replication is commonly utilized: data synchronization to a reporting server; data synchronization to replica(s) in support of business needs (i.e. off-loading data processing, stand-by server); and data synchronization of remote semi-connected replicas. The Replication subsystem defines specific criteria for the implementation of any of the replication mechanisms some requiring schema modifications thus many considerations must be weighed in designing a replication topology.

**Criteria for Selecting a Replication Mechanism**

Distributing data across SQL Server instances in any enterprise environment requires a careful consideration of a myriad of issues. The Replication subsystem provides a range of options as well as customizable configurations to satisfy most data distribution
needs; however, given the available options the selection of a specific replication mechanism may be daunting for the uninitiated. In an attempt to provide a starting point in the selection process I have developed the decision flow charts in Figures 1, 2, and 3 below. Please note that these flow charts represent high-level considerations and should not be interpreted as definitive – they are a starting point for a more thorough analysis.
Data Distribution with SQL Server Replication

Ron Johnson

Figure 1: Merge Replication Selection Criteria

1) What is the data replication latency?
   - Real-time or Near-time data

2) What are the DBMS types?
   - Source and replicas are SQL Server

3) Network connectivity?
   - Replicas constantly connected to source

4) What is the data change frequency?
   - High volume of data changes

5) How much data will be replicated per cycle?
   - Large quantity of data to be distributed

6) Will replicas need to update data?
   - Data updates at replicas

7) Can the data schema be augmented?
   - Schema changes possible

8) Use of resolvers for conflict resolution?
   - Data conflicts acceptable

Merge Replication
1) What is the data replication latency?

Real-time or Near-time data

Source and replicas are SQL Server

Replicas constantly connected to source

High volume of data changes

Large quantity of data to be distributed

Data updates at replicas

Schema changes possible

Data conflicts acceptable

Schema changes possible

Data conflicts acceptable

8) Use of resolvers for conflict resolution?

7) Can the data schema be augmented?

6) Will replicas need to update data?

5) How much data will be replicated per cycle?

4) What is the data change frequency?

3) Network connectivity?

Replicas constantly connected to source

2) What are the DBMS types?

Source and replicas are SQL Server

1) What is the data replication latency?
Figure 3 Transactional Replication Selection Criteria
Data Distribution with SQL Server Replication
Ron Johnson
Summary

Replication is a powerful mechanism supporting the distribution of data throughout an enterprise. Significant research has led the industry to adopt lazy or asynchronous replication implementations in order to support the high-levels of performance necessary in the real-world. The benefits of lazy replication come with a cost - data consistency; however, with replication properly configured and administered the data consistency issues may be mitigated thereby allowing high-performance data distribution. Commercial replication systems have evolved from the research community to satisfy the requirements of business by providing abstractions from the complexities of the underlying replication mechanisms.

Microsoft, in its mission to provide easy-to-use tools for data management, has incorporated the technologies necessary for distributing data within the SQL Server product line. The Replication subsystem included with various feature sets in every edition of SQL Server provides the ability to natively distribute data throughout the enterprise with no additional licensing fees. The Replication subsystem in SQL Server allows for implementations supporting a variety of data distribution needs while minimizing implementation and administrative requirements. Transactional replication supports real-time data synchronization with low overhead for offline processing, standby servers, or reporting servers. Merge replication supports data synchronization across intermittently connected replicas while snapshot replication is most useful for distributing static data representing a point-in-time from source to replicas during off-peak hours.

About the Author

Ron is a Senior DBA who specializes in performance optimization, replication, and security.

Trademark Acknowledgements

Microsoft and SQL Server are trademarks or registered trademarks of Microsoft Corporation, in the United States and/or other countries.
Bibliography


