Microsoft SQL Server Data Replication Techniques

Reasons to Replicate Your SQL Data

SQL Server replication allows database administrators to distribute data to various servers throughout an organization. You may wish to implement replication in your organization for a number of reasons, such as:

**Load balancing.** Replication allows you to disseminate your data to a number of servers and then distribute the query load among those servers.

**Offline processing.** You may wish to manipulate data from your database on a machine that is not always connected to the network.

**Redundancy.** Replication allows you to build a fail-over database server that’s ready to pick up the processing load at a moment’s notice.

In any replication scenario, there are three main components. In SQL Server documentation, this is called the PDS metaphor.

**Publishers** have data to offer to other servers.

**Distributors** receive the published information, stores it, and may forward it to subscribers.

**Subscribers** are database servers that wish to receive updates from the Publisher when data is modified.

Data Latency Dictates Pace of Replication

**Latency** is the time required for data updates on the source server to be reflected on the secondary server or servers. Some systems can tolerate high levels of latency, such as traveling sales recording software that may be synchronized with the central office on a nightly or weekly basis.
At the other extreme, some financial transaction systems require central office updates to be available instantaneously to the regional and branch levels. The increasing demand for near-real-time data should continue to drive acceptable data latency levels lower and lower.

“Information anytime, anywhere” will be the clarion call of the successful business in the future. In addition to data latency, site autonomy is an important criterion when formulating a replication strategy.

Site Autonomy Correlates with Latency

Site autonomy is the degree of independence between the source server and any secondary servers. Some forms of replication require a tight coupling between source and secondary systems. The Publisher is the server that makes data available to other servers participating in the replication activity. Subscribers receive and apply the data updates provided by the Publisher through a Distributor.

Other replication methods can endure significant logical separation. With merge replication, a Subscriber can be entirely disconnected and miss one or more data updates, but receive the updates when it is back online. With snapshot replication, entire publications are sent over the wire at each replication cycle. Therefore, the next cycle will update disconnected Subscribers after they have rejoined the replication process.

Note that data latency and site autonomy requirements and characteristics move in unison. That is, systems that require low latency levels also require low autonomy; likewise, systems that can endure high latency will also be more autonomous. Before exploring replication further, consider a related topic: distributed transactions.
Distributed Transactions Are Similar to Replication

Distributed transactions (DTs) are often confused with replication. Both approaches provide distribution of data across multiple servers to achieve high availability of the data. Of all the methods for making data available at many sites, DTs provide the lowest data latency and require the least site autonomy.

The near-real-time financial transaction example mentioned earlier would very likely be accomplished with DTs, when it "absolutely, positively has to be there...in an instant."

An important distinction between DTs and replication is that DTs require the use of the `begin distributed transaction`, `commit tran`, and `rollback tran` SQL commands. DTs also require the Microsoft Distributed Transaction Coordinator (MSDTC), which ships with SQL Server 6.5 and later. MSDTC provides transaction objects for complete transaction management in a distributed environment. DTs, through the use of custom SQL statements and the MSDTC, offer near-real-time data updates when the highest level of concurrency is required.

Immediate Updating Subscribers (IUS) Replication Methods

The IUS replication method also uses the MSDTC to perform simultaneous updates, although the fast update is applied to the publication database only. In IUS, updates to the Subscribers are applied via traditional replication methods. Furthermore, the use of IUS replication incurs higher data latency and provides higher site autonomy than DTs.

There’s nothing preventing a single system from acting in both of these capacities. In fact, this is often done in large-scale distributed database systems.
Microsoft SQL Server supports three types of database replication. They are:

**Snapshot replication** acts in the manner its name implies. The publisher simply takes a snapshot of the entire replicated database and shares it with the subscribers. Of course, this is a very time and resource-intensive process. For this reason, most administrators don’t use snapshot replication on a recurring basis for databases that change frequently. There are two scenarios where snapshot replication is commonly used. First, it is used for databases that rarely change. Second, it is used to set a baseline to establish replication between systems while future updates are propagated using transactional or merge replication.

**Transactional replication** offers a more flexible solution for databases that change on a regular basis. With transactional replication, the replication agent monitors the publisher for changes to the database and transmits those changes to the subscribers. This transmission can take place immediately or on a periodic basis.

**Merge replication** allows the publisher and subscriber to independently make changes to the database. Both entities can work without an active network connection. When they are reconnected, the merge replication agent checks for changes on both sets of data and modifies each database accordingly. If changes conflict with each other, it uses a predefined conflict resolution algorithm to determine the appropriate data. Merge replication is commonly used by laptop users and others who can not be constantly connected to the publisher.

Each one of these replication techniques serves a useful purpose and is well-suited to particular database scenarios.
Snapshot Replication Updates Periodically with Bulk Transfers

Snapshot replication is the easiest form of replication to set up and maintain. It consists of the periodic bulk transfer of an entire publication from the publication server, through the distribution server, to one or more subscription servers. It incurs a high degree of latency because the publications are only refreshed periodically.

The Publisher is responsible for producing publications that are distributed by agents on the Distributor to one or more Subscribers. Publications consist of one or more articles. Articles are the lowest level data construct in replication. The DBA can customize an article to provide various cuts, or views, of the data. Tables used in an article can be vertically or horizontally partitioned to provide tailored replication data to different Subscribers.

Subscribers come in two forms: push and pull. Push Subscribers, the default and most common method, have the replication updates pushed to them by the Distributor. Push Subscribers are defined and managed on the publishing database. The two substantial benefits of push subscriptions are that they are more secure than pulls and they enable centralized management. The DBA can create many push subscriptions at one time.

Pull subscriptions, on the other hand, are initiated and managed at each subscriber site, which makes them more flexible. This gain in flexibility is offset by reduced security, since control is relinquished to each subscription server. Now that the groundwork is in place, consider the four replication methods.