Replication: SQL Server 2000 - Part 1

(2003-11-10) - Contributed by Mahesh Kodli

In this first installment of a 2-part series, Mahesh describes and explains what Replication is, its features and benefits, and how you can put it to use.

Database management systems are among the most important software systems driving the information age. In many Internet applications, a large number of users who are geographically dispersed may routinely query and update the same database. In this environment, the location of the data can have a significant impact on application response time and availability. A centralized approach manages only one copy of the database. The centralized approach suffers from two major drawbacks:

- Performance problems due to high server load or high communication latency for remote clients.
- Availability problems caused by server downtime or lack of connectivity. Clients, in portions of the network that are temporarily disconnected from the server, cannot be serviced.

These issues would be effectively answered by Replication.

Replication is the process of sharing data between databases in different locations. Using replication, you create copies of the Database and share the copy with different users so that they can make changes to their local copy of the database and later synchronize the changes to the source database.{mospagebreak title=Replication Benefits&toc=1}

- Users working in different geographic locations can work with their local copy of data thus allowing greater autonomy.
- Database replication can also supplement your disaster-recovery plans by duplicating the data from a local database server to a remote database server. If the primary server fails, your applications can switch to the replicated copy of the data and continue operations.
- You can automatically back up a database by keeping a replica on a different computer. Unlike traditional backup methods that prevent users from getting access to a database during backup, replication allows you to continue making changes online.
- You can replicate a database on additional network servers and reassign users to balance the loads across those servers. You can also give users who need constant access to a database their own replica, thereby reducing the total network traffic.
- Database-replication logs the selected database transactions to a set of internal replication-management tables, which can then be synchronized to the source database. Database replication is different from file replication, which essentially copies files.{mospagebreak title=SQL Server Platform for Replication&toc=1} Microsoft SQL server uses publishing industry model to represent the components and processes in replication architecture. Publishing industry publishes Magazines/Books; there are Distributors and Agents who carry these publications to the Subscribers. Subscribers of the magazine obtain copies of the publication and read the articles of interest to them; this is how the SQL Server Replication model works. Figure 1 depicts the typical Publishing industry flow.

Figure 1

Based on the above model we can identify the following Entities for the SQL Server replication model.

- Publisher
- Distributor
- Agent
- Subscriber
- Articles
- Publications
- Subscriptions

Let us further explore each of these Entities.{mospagebreak title=Entities for the SQL Server Replication Model&toc=1}

Publisher

Publisher is a server that makes the data available for subscription to other servers. In addition to making data available for replication, a publisher also identifies what data has changed at the subscriber during the synchronizing process.

Depending on the type of replication, changed data is identified at different instances. We will learn more about Replication types in the Replication Types section.

Distributor

Distributor maintains the Distribution Database. The role of the distributor varies depending on the type of replication. Two types of Distributors are identified: Remote distributor and Local distributor. Remote distributor is separate from publisher and is configured as a distributor for replication. Local distributor is a server that is configured as a publisher and a distributor.

Agents

Agents are the processes that are responsible for copying and distributing data between Publisher and subscriber. There are different types of Agents supporting different replication types.

Subscriber

Subscriber is a server that receives and maintains the published data. Modifications to the data at the subscriber-level can be propagated back to the publisher; in some cases Subscriber may re-publish the data to the other subscribers.

Articles

An article can be any database object, viz. Tables (Column filtered or Row filtered), Views, Indexed views, Stored Procedures, User defined functions.

Publication

Publication is collection of articles.

Subscriptions

Subscription is a request for copy of data or database objects to be replicated.{mospagebreak title=Entities Further Explained...&toc=1}

Subscription Types

Changes to the subscriptions at the publisher can be replicated to subscribers via PUSH subscription or PULL subscription.

With Push subscription the publisher is responsible for synchronizing all the changes to the subscriber without the subscriber asking for those changes.

With Pull subscription the subscriber initiates the replication instead of the publisher.

Replication Types

Microsoft SQL Server supports the following types of replication:

- Snapshot Replication
- Transactional Replication
- Merge Replication

Snapshot Replication

Snapshot replication is also known as static replication. Snapshot replication copies and distributes data and database objects exactly as they appear at the current moment in time.

Characteristics of Snapshot Replication

- The changes to data at the subscriber are not updated to the subscriber continuously
- Subscribers are updated with complete modified data and not by individual transactions
- Propagating the changes to the subscribers takes more time as it is a one time process or scheduled process.

When Do I use Snapshot Replication?

Following are some of the scenarios where snapshot replication fits in ideally:

- Data/Db objects are static or do not change frequently
- Replicate Look Up tables that do not change frequently
- The amount of data to be replicated is small
- Users often work in disconnected mode, and are not always interested in the latest data.

Transactional Replication

Transactional replication is also known as dynamic replication. In transactional replication, modifications to the publication at the publisher are propagated to the subscriber incrementally.

Characteristics of Transactional Replication

- Publisher and the subscriber are always in synchronization.
- Transaction boundaries are preserved; i.e. if there are modifications to 5 rows of data, either all the 5 modified rows are propagated to the subscriber or none are propagated.
- The publisher and the subscriber should always be connected.

When do I use Transactional replication?

- Replicating Database with rollup information, Database with regional, central sales or inventory database that is updated and replicated to different sites.
- Subscribers always need the latest data for processing.

Merge Replication

Merge replication provides advantages of both Snapshot replication and Transactional replication. The initial snapshot applied to the subscribers and then SQL server tracks changes to the data at publisher and subscriber levels. The data is synchronized on a scheduled basis or on demand. Since data modifications are made independently at publisher and subscriber levels, conflicts are likely to occur during synchronization.

Characteristics of Merge Replication:

- Updates to the data are made independently at more than one server.
- Data is merged on a scheduled basis or on demand.

- Allows users to work online/offline and synchronize the publisher and subscriber on a scheduled basis or on demand.

When Do I use Merge Replication?

- Site autonomy is very critical.
- Multiple subscribers need to update the data either at the same time or at different times and propagate the changes to the publisher{mospagebreak title=Implementing Replication&toc=1}

With the above basic knowledge we can now proceed to understand the implementation of replication. There are different ways by which you can implement and monitor replication based on different replication types. But in general replication has the following general steps:

- Configuring replication
- Generating and applying initial snapshot
- Modifying replicated data
- Synchronizing and propagating data

Configuring Replication:

Configuring replication involves the following steps:

- Configure the publisher and distributor. Distributor can be on the same server or even on a different server
- Create publications based on data, sub sets of data and database objects
- Determine the type of replication to use, the subscriber database and location of the snapshot file
- Configure when the synchronization will occur and options that will be used with publications
- Create push and/or pull subscriptions at either the publisher or the subscriber and configure your replication schedule and options

Generating and Applying Initial Snapshot:

SQL server 2000 creates a snapshot of data and schema and saves it in the snapshot file location. After the subscription is created, the snapshot is applied, and is based on a configured schedule. Creating a publication or a snapshot can be applied manually. The snapshot agent is responsible for creating the snapshot file and stores it in the snapshot file location.

Modifying Replicated Data:

Depending on the type of replication and replication options, the subscriber will be able to modify the data after the snapshot has been applied and propagate the changes back to the publisher or other subscribers.

Synchronizing and Propagating Data Changes:

Synchronization refers to the propagation of data changes between subscriber and publisher. How the data is synchronized is dependent on the type of replication used.

- Incase of snapshot replication, a snapshot file is reapplied at the subscriber
- Incase of transactional replication, all data modification through Insert/Update and Delete are distributed between publisher and subscriber
- Incase of merge replication, data modification at various servers are merged. Conflicts, if any, are detected and resolved.{mospagebreak title=Implementing Replication, Cont'd&toc=1}

Data Considerations for Replication

Special consideration should be taken for some of the data types and properties during replication. These Data types and properties are:

- Identity range management
- Unique identifier and Timestamp data types
- NOT FOR REPLICATION option

Identity Range Management

Value of a column marked as Identity is incremented automatically, when new rows are added to the column table. In replication, where publication contains identity columns, the following configurations can be used to manage Identity columns.

- Use Auto Identity range management of SQL Server 2000

For example, you can set the Identity range of 1 to 500 for Publication ‘A’ at Publisher and 501 to 1000 for the same publication at Subscriber, with a threshold of 80%.

In this case, a newly inserted row at publisher will have Identity from 1 to 500 and a newly inserted row at subscriber will have identity from 501 to 1000.

When the threshold has reached 80%, a new identity range is used for the next inserts. In this case, if the identity value reaches 400 at the Publisher any new inserts after that will use the new identity range from 1001 to 1500. Similarly, if the Subscriber threshold reaches 800, any new inserts after that will have Identity range from 1501 to 2000.

The threshold value should be set carefully by evaluating the frequency of updates at the subscriber and synchronization schedule. Setting the threshold to a lower value will result in many unused Identity values.

The following system-stored procedures can also be used to set Identity range explicitly:

- Sp_adjustpublisheridentityrange
- Sp_addmergearticle
- Use NOT FOR REPLICATION option when defining Identity columns

Identity ranges can also be managed by defining check constraint and the NOT FOR REPLCIATION option on Identity column. When an identity column is specified as NOT FOR REPLCATION, then its range should be provided programmatically. When this option is set, SQL server retains the original values set by the replication agent but continues to increment the value of the Identity column in a normal value, i.e. without resetting the Identity value.