Replication: SQL Server 2000 - Part 2

(2003-11-13) - Contributed by Mahesh Kodli

In this second, and final, installment, Mahesh gives us a step-by-step approach on how to implement replication across a network using SQL Server 2000.

In my last article, I explained what and how Replication can be used, as well as the benefits in using it. The following steps will help to setup Replication using the SQL Server Enterprise Manager. For this example I will consider the standard PUBS database.{mospagebreak title=Getting Started: Step-By-Step&toc=1}

Set Up Proper Login for Replication

To set up replication, you must use a login account that is a member of SQL Server's Process Administrators (or higher authority) server role. I used “sa” and had no problems.

Configuring Server for Publishing, Subscribers and Distributor

Under SQL Server enterprise manager navigate to the Tools/Replication menu, run Configure Publishing and Distribution Wizard:

Select your server as Publisher, Distributor and Subscriber server:

Specify the location where snapshots, from publishers that use this distributor, will be stored:

Configure your server with the default settings or apply custom settings; in this example I would use the default settings. With the above settings you have enabled your server as a distributor server, and have created Publishers and Subscribers. This process will also create a New Distribution Database under the selected server. {mospagebreak title=Getting Started: Step-By-Step, Cont'd.&toc=1}

Creating Publication

- Right click on the PUBS database and click on New/Publication, which will run the Publication wizard.
- Select the PUBS database as the Publication Database

- Chose Merge Replication as the replication type.

- Specify what types of Subscribers will subscribe to this Publication. In this example I chose SQL Server 2000 as the Subscriber types.
- Chose the ‘Object type’ you want to Publish. In this example I chose to publish all types of objects.

- Go to all the table article properties (click on the ellipses) and select the Identity Range tab and check the Automatic Identity Range Manipulation check box and set Publisher and Subscriber range to a desired value. In this example, I will set the Identity range value as 1500. This is a tedious process; hence it is necessary to automate this process. I will use the following system stored procedures to set the identity range explicitly:

- Sp_adjustpublisheridentityrange
- Sp_addmergearticle{mospagebreak title=Code: Stored Procedure&toc=1}

Following is the source code of the stored procedure, which automates the identity range setting process. The Stored procedure takes ‘PublisherName’ and ‘IdentityRange’ as the parameter and internally uses the ‘SP_addmergearticle’ system stored procedure to set the identity range of all the Published Table articles:

```
SET QUOTED_IDENTIFIER ON GOSET ANSI_NULLS ON GO/*'***************************************************'
Purpose :Add Identity range to the user defined Tables under a given publisher' Inputs  :Publisher Name and Identity
Range' Returns         :None' Author       :Mahesh M Kodli'***************************************************
```

```
CREATE PROCEDURE AddMergeArticle

@pPublisherName VARCHAR(255),@pIdentityRange BIGINT

As DECLARE @tRV INT DECLARE @tArticle VARCHAR(255) SET @tRV = 0  DECLARE Merge_Article_Cursor CURSOR
FOR

 --Get all the user defined Tables for which to add Identity Range SELECT TABLE_NAME TableName FROM
INFORMATION_SCHEMA.TABLES  WHERE rtrim(ltrim(table_type))='BASE TABLE'      AND TABLE_NAME Not like
'Conflict%' AND TABLE_NAME Not like 'dtproperties' AND TABLE_NAME Not like 'sys%' AND TABLE_NAME Not like
'MS%'

 OPEN Merge_Article_Cursor

FETCH NEXT FROM Merge_Article_Cursor INTO @tArticle  WHILE @@FETCH_STATUS = 0    BEGIN

--Check if the Table has identity column and accordingly set the Auto --Identity Range to TRUE or FALSE Before adding
Identity Range to the  --article  IF OBJECTPROPERTY ( OBJECT_ID(@tArticle), 'TableHasIdentity') = 1     BEGIN      IF NOT
EXISTS (SELECT * FROM sysmergeextendedarticlesview WHERE name = @tArticle AND pubid IN (select pubid FROM
sysmergepublications WHERE name like @pPublisherName  AND UPPER(publisher)=UPPER(@@servername) and
publisher_db=db_name()))

BEGIN    --Use the System stored procedure add merge article to add Identity range for -- each articleEXECUTE
sp_addmergearticle @publication = @pPublisherName, @article = @tArticle, @source_owner = N'dbo', @source_object
= @tArticle, @type = N'table', @description = null, @column_tracking = N'true', @pre_creation_cmd = N'drop',
@creation_script = null, @schema_option = 0x000000000000CFF1, @article_resolver = null, @subset_filterclause =
null, @vertical_partition = N'false', @destination_owner = N'dbo', @auto_identity_range = N'true', @pub_identity_range
= @pIdentityRange, @identity_range = @pIdentityRange, @threshold = 80, @verify_resolver_signature = 0,
@allow_interactive_resolver = N'false', @fast_multicol_updateproc = N'true', @check_permissions =
0,@force_invalidate_snapshot=1

END    END

ELSE BEGIN

  IF NOT EXISTS (SELECT * FROM sysmergeextendedarticlesview WHERE name = @tArticle AND pubid IN (select
pubid FROM sysmergepublications WHERE name like @pPublisherName  AND
UPPER(publisher)=UPPER(@@servername) and publisher_db=db_name()))

  BEGIN       EXECUTE sp_addmergearticle @publication = @pPublisherName, @article = @tArticle, @source_owner =
N'dbo', @source_object = @tArticle, @type = N'table', @description = null, @column_tracking = N'true',
@pre_creation_cmd = N'drop', @creation_script = null, @schema_option = 0x000000000000CFF1, @article_resolver =
null, @subset_filterclause = null, @vertical_partition = N'false', @destination_owner = N'dbo', @auto_identity_range =
N'False', @pub_identity_range = NULL, @identity_range = NULL, @threshold = NULL, @verify_resolver_signature = 0,
@allow_interactive_resolver = N'false', @fast_multicol_updateproc = N'true', @check_permissions =
0,@force_invalidate_snapshot=1

  END   END     FETCH NEXT FROM Merge_Article_Cursor INTO @tArticle     END    CLOSE Merge_Article_Cursor
 DEALLOCATE Merge_Article_Cursor IF (@@ERROR <> 0)  BEGIN   SELECT @tRV = -95  --UnSuccessful   GOTO XIT
 END  XIT:  RETURN @tRV{mospagebreak title=Getting Started: Step-By-Step, Cont'd.&toc=1}
```

Specify the Publication Name

   -
Accept other settings and create the publication:

With the above steps completed, we have published the data in the ‘Pubs’ database and have also set publication properties.
Create Snap Shot

   - Navigate to <Server>/Replication/Publications/Properties

   - Select the Status tab

   - Click on the “Run Agent Now” button. This will create the snap shot files.

Adding Subscribers

   - Create a new SQL Server Database. In this example I have created a new database named PubsSubscriber.
   - Right click on <Server>/Replication/Publications/Pubs:<Publication>
   - Click “Push New Subscription” to run the New Subscription wizard

   - Choose the newly created database as the Subscriber Database:

   - Set the Merge Agent schedule:

   - Choose Initialize both schema and data option and check Start the Merge agent to initialize the subscription immediately:

Since my Subscriber Database was a new database, I have chosen to initialize both the schema and the data.
Navigate through the wizard, accepting further options and click on finish to complete the wizard steps.
With the above steps completed, we have created a new subscriber database and setup the handshake between Publisher and Subscriber.
Now you can distribute the Subscriber database to its users who can ‘play’ with their own copy of data, and synchronize the data as per the rules of business.
Synchronize

   - To synchronize, Right click on the <Server>/Replication/Publications/Pubs:<Publication>, and click “Start Synchronization”.{mospagebreak title=General Replication Performance Tips&toc=1}

Filtering Published Data

Replication can be tuned to yield better performance, and one of the ways to achieve this is filtering the data before publishing; in other words, give subscribers what they want to see and play with.

Replication facilitates this by splitting it vertically/horizontally. By distributing partitions of data to different Subscribers, you can:

   - Minimize the amount of data sent over the network.
   - Reduce the amount of storage space required at the Subscriber.
   - Customize publications and applications based on individual Subscriber requirements.
   - Reduce conflicts because the different data partitions can be sent to different Subscribers.

In addition, the following are some tips which will help to yield better performance:

- When running SQL Server replication on a dedicated server, consider setting the minimum memory amount for SQL Server to use from the default value of 0 to a value closer to what SQL Server normally uses.
- Don’t publish more data than you need. Try to use Row filter and Column filter options wherever possible as explained above.
- For best performance, avoid replicating columns in your publications that include TEXT, NTEXT or IMAGE data types.
- Avoid creating triggers on tables that contain subscribed data.
- Applications that are updated frequently are not good candidates for database replication.
- Distribute the workload into more than one SQL server using replication.
- Plan for the type of replication to be used before the database design, because the type of replication used will, to a certain extent, guide your database design.{mospagebreak title=Conclusion&toc=1} In a nutshell, replication is the capability to reliably duplicate data from a source database to one or more destination databases. SQL Server 2000 gives you the power for replication design, implementation, monitoring, and administration. This gives you the functionality and flexibility needed for distributing a copy of data and maintaining data consistency among the distributed. You can automatically distribute data from one SQL Server to many different SQL Servers through ODBC (Open Database Connectivity) or OLE DB. SQL Server replication provides update replication capabilities such as Immediate Updating Subscribers and Merge replication. With all the new enhancements to SQL Server replication, the number of possible applications and business scenarios is mind-boggling.