**Merge Replication**

In the final section of this chapter, we'll explore the second of the two synchronization mechanisms available through SQLCE, merge replication. Like RDA (Remote Database Access), the client agent provides an API that applications use to submit replication requests (add and drop a subscription, reinitialize a subscription, and synchronize) over HTTP to the server agent. As with RDA, Compact Framework developers access the API through the SqlCeReplication class in the SqlServerCe .NET Data Provider.

The subscription includes information about an existing publication created on a SQL Server (referred to as the publisher). The publication consists of individual articles or tables that have been enabled for replication. In short, and as shown in Figure 7-3, when an application creates a subscription using the AddSubcription method, an initial snapshot of the tables (schema and data) is downloaded to the device from a distributor by the server agent, and the tables are created in SQLCE by the client agent. The SQLCE database can then be updated by a Compact Framework application, while not connected as in RDA, thereby making it ideal for occasionally connected scenarios. At some point later, the application reconnects to the network and calls the Synchronize method of the SqlCeReplication class. This method instructs the client agent to extract all the modified rows in the subscription tables and send them to the server agent. On the IIS server, the server agent then creates an input file with all of the INSERT, UPDATE, and DELETE requests from the device. A process on the IIS server called the SQL Server Reconciler then loads a special replication provider for SQLCE,[18] which reads the input file and tells the Reconciler about the changes. The Reconciler then makes the appropriate changes on the publisher and detects and resolves conflicts created by other subscribers updating the same row. This happens according to the rules specified on the publisher.

Technically, SQL Server publications can also include other objects, such as stored procedures, user-defined functions, and views; however, SQLCE supports only tables.

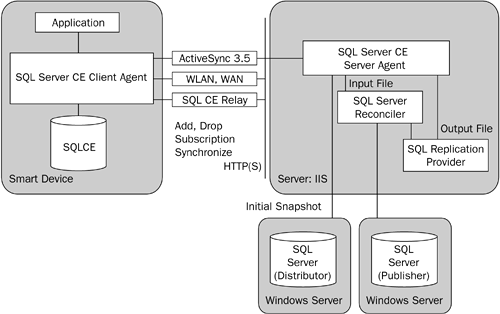
An instance of SQL Server that is used to store the snapshots, history, and statistics. This can be the same server as the one on which the data resides.

Also referred to as the **merge agent.**

This is a pluggable architecture that allows other replication providers, such as the one for SQL Server subscribers, to process synchronizations as well.

There are standard conflict resolvers available on SQL Server that can be configured for the publication to handle simple cases. It is also possible to write custom resolvers to handle more complex logic. For more information, see the SQL Server 2000 Books Online.

Figure 7-3. Merge Replication Architecture. This diagram shows the architecture of merge replication and highlights the components involved.



**TIP**

It should be noted that SQLCE can act only as a subscriber and not as either a publisher or a distributor.

The Reconciler then tells the SQLCE replication provider about the changes that must be made to the subscription database. The provider writes these changes to an output file, which is picked up by the server agent and sent to the client agent on the device. The client agent then applies these changes to the subscription tables.

Obviously, this process is referred to as merge replication because changes made on the device are merged with changes made on other devices and with those made directly on the publisher (for example, through a Web application updating the database). As a result, merge replication is ideal when a database receives continuous updates or is updated both by the client and the server or by multiple clients.

**Features and Scenarios**

Merge replication includes the following advantages that make it attractive for certain types of solutions:

* Efficient resource usage: Merge replication takes advantage of the connectivity architecture discussed previously. In addition to the 8:1 compression that conserves bandwidth and ability to work when the underlying transport is not particularly reliable, this architecture is used to initialize subscriptions and synchronize changes from the server to the device. This allows the client agent to process blocks of data and discard them, rather than requiring storage memory for large snapshots and sets of changes.
* Horizontal and vertical partitioning: Although RDA (Remote Database Access) allows partitioning by formulating queries passed to the Pull method, the various articles within the publications can include row and column filters. This allows administration of the filters to be controlled on the server (and controlled by an administrator), rather than on the client, making administration and development simpler. In addition, merge replication supports dynamic filtering (discussed later in the chapter), whereby the articles contain dynamic logic used to filter based on information provided by the subscriber. In this way the data to be downloaded to the device can be customized by user, group, or geography.
* Multiple tables: Unlike RDA, which supports only pulling and pushing data from one table at a time, individual articles, each representing a table, can be placed in a publication. This allows multiple tables and their data to be created on the device when a subscription is created, thereby making development simpler; however, this flexibility comes with the additional administrative overhead of creating publications on the server.
* Server-based conflict resolution: As described previously, individual articles can be configured with a supplied or custom conflict resolver. These resolvers, which execute on the distributor, encapsulate the rules for determining which subscriber's data should be favored when more than one subscriber updates the same row. This allows the rules that govern conflicts to be centralized on a server rather than existing within the application; however, because of the algorithm required to merge changes between multiple subscribers, SQLCE must track more information for each row. Whereas RDA requires only one column (4 bytes) per row, replication requires three columns (48 bytes) of overhead per row.

There are actually 10 resolvers that ship with SQL Server 2000. See the SQL Server 2000 Books Online for more information about what each does.

* Bidirectional data flow: Perhaps the most significant feature, however, is that replication supports a bidirectional data flow. In other words, while RDA allows tables to be pulled from SQL Server and changes made on the device to be synchronized with the server, it does not support flowing changes made on SQL Server onto the device without repulling the entire table. Merge replication supports this bidirectional flow by default, although it can be limited to synchronize changes originating only on the device. This feature makes merge replication much more efficient for applications that require incremental updates from the server.

The combination of these features means that merge replication is most appropriate in the following scenarios:

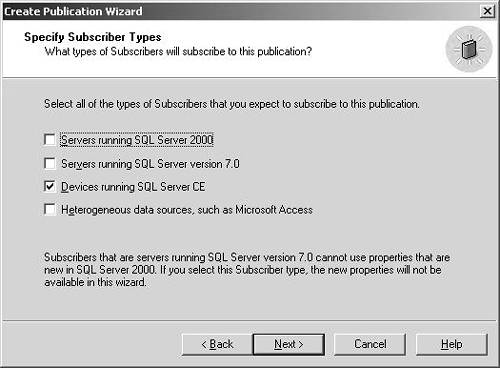
* Modifying shared data: Because merge replication includes server-based conflict detection and resolution that can be customized to any level of complexity, it is well suited for situations where the data is not highly partitioned, so that multiple subscribers may need to update the same row. An example might be the need to view and update customer information that overlaps sales territories.
* Working with dependent data: Because multiple tables (articles) can be grouped into a single publication and made available to the device, merge replication works well when the tables have some dependency between them. For example, an application that requires customer and order history information can subscribe to a publication that includes both as articles, rather than having to make separate calls using RDA.
* Occasionally connected applications: Because merge replication does not support any real-time or connected way of directly querying the SQL Server, it is inherently suited for situations where the data can be updated only through the local SQLCE database on the device. This occasionally connected architecture is appropriate for Compact Framework applications that run on devices where there is unreliable or no wireless connectivity and synchronization occurs only sporadically, for example, through a docking station.
* Working with changing data: Even if the application does not perform updates, merge replication is efficient for pushing out changes made on the server. Because the client agent downloads only incremental changes, there is not the overhead associated with repulling an entire table, as would be the case with RDA. For example, the Multiple Listing Service (MLS) application discussed in the following sidebar is a good example.

**Configuration**

**Configuration**of a system is the arrangement of each of its functional units, according to their nature, number and chief characteristics.

In order to use merge replication, a publication must first be created on the SQL Server. The simplest way to accomplish this is to use the Create Publication Wizard in SQL Server Enterprise Manager. This wizard leads an administrator step by step through the process of selecting the database to publish, the appropriate publication type (merge publication in this case), the types of subscribers expected (shown in Figure 7-4), the articles to publish, and other properties, including filters. If this is the first publication created on the server, it also allows an administrator to identify which server will act as the distributor.

Figure 7-4. Selecting Subscriber Types. This dialog appears in the Create Publication Wizard and is used by the wizard to enable anonymous subscribers and character mode snapshots, if SQL Server CE subscribers is checked.



|  |
| --- |
| SQL Server CE and Real Estate  One of the success stories involving SQL Server CE 1.1 that has gained significant publicity since 2002 has been the Pocket PC MLS application from Offutt Systems, called Pocket InnoVia. This application allows agents to download MLS listings and contact information, including photos of the property, directly to their Pocket PCs for offline viewing.  This system uses merge replication to download the listings and associated data based on a geographic location to the Pocket PC. New listings are then added to the device incrementally when the application performs a synchronization. This system also stores its final data in Oracle and uses Data Transformation Services (DTS) in SQL Server 2000 to refresh the publication database periodically.  Perhaps the most significant aspect of this case study, however, is that because SQLCE includes the built-in replication infrastructure, the development costs were kept under $100,000, while the low administration costs allow Offutt Systems to sell the application for $129 and to charge $50 for a per-user, per-year subscription.  Although the system was originally developed with eVB and SQLCE 1.1, it is being updated to use the Compact Framework and SQLCE 2.0 to take advantage of better query performance, better server performance, and additional features such as parameterized queries. It will also take advantage of the Pocket PC Phone Edition and Short Message Service (SMS), in conjunction with SQL Notification Services, to provide instant notification of new listings and one-touch calling of agents.  For more information on this case study, see the "Related Reading" section at the end of the chapter. |

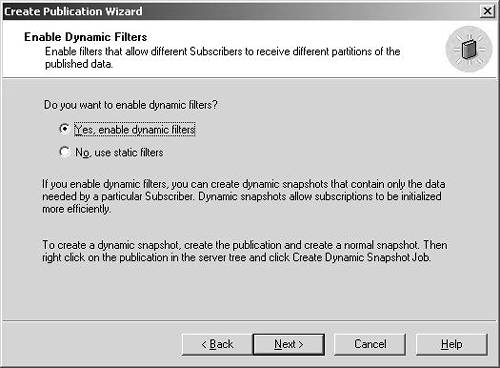
By selecting the Devices running SQL Server CE option in the dialog shown in Figure 7-4, the wizard automatically allows anonymous subscribers for the publication and will create the snapshot data using character mode format. Both of these options are required in order for devices to create subscriptions, and they can also be configured once the publication has been created by right-clicking on it and selecting Properties in the context menu.

In addition, although merge publications support either column- or row-level tracking, only row-level tracking is recognized by the SQLCE replication provider. As a result, if all subscribers will be devices running SQLCE, an administrator should configure the publication accordingly for efficiency. It is also important to configure an explicit snapshot folder on the distributor. This is recommended because the snapshot folder will default to an administrative share (C$) accessible only by administrators. Because the server agent running on IIS needs to access this share, it makes sense to create an explicit nonadministrative share and then configure the distributor to use this share to make the snapshot available.[21]

See the topic "Configuring an Explicit Snapshot Folder" in the SQLCE Books Online.

As mentioned previously, one of the features of merge replication is that row and column filters can be configured on the server and that the row filters can be dynamic in nature. To configure these, the Create Publication Wizard leads the administrator through the process of creating the filters. For example, if row filtering is chosen, the dialog shown in Figure 7-5 allows the administrator to choose further that dynamic filters will be enabled.

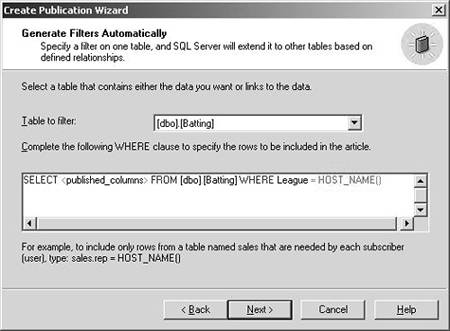
Figure 7-5. Selecting Dynamic Filters. This wizard screen allows an administrator to create dynamic filters on a publication.



The wizard then presents the dialog shown in Figure 7-6 to create the dynamic filter. The important point to note about the filter is that it relies on the HOST\_NAME() Transact-SQL function in the WHERE clause to evaluate which rows are created in the initial snapshot and subsequently updated on the subscriber.[22] This value is populated by the HostName property of the SqlCeReplication class on the device, thereby allowing dynamic views of the article. Typical values with which to populate the HostName property include user name, region, department, and product line.

[22] The suser\_name function is also supported by dynamic filters, but since SQLCE subscribers are always anonymous subscribers, that function will not return a unique value for the user.

Figure 7-6. Creating the Dynamic Filter. This dialog allows the administrator to create the dynamic filter based on the return value of the HOST\_NAME T-SQL function.



When dynamic filters have been created, the default behavior is for dynamic snapshots to be generated as subscriptions are created by the Compact Framework application; however, to speed the initial synchronization time, dynamic snapshots can be precreated. This is accomplished by right-clicking on the publication in SQL Enterprise Manager and selecting Create Dynamic Snapshot Job. This wizard allows an administrator to specify a value for the HOST\_NAME function and then creates the snapshot agent to create, and optionally to schedule, the initial snapshot.

**NOTE**

Although not mentioned previously, one of the other useful features of merge replication is support for auto-ranged identity columns. This feature allows tables with system-assigned primary keys (identity columns) to be published and instructs the server to manage the range of values that can be inserted at the publisher and individual subscribers. A tab in the Article Properties dialog allows the administrator to configure a range size for the publisher and each subscriber along with a threshold at which to assign new ranges. This feature ensures that subscribers do not attempt to insert different rows with the same identity value. An alternate approach would be to use natural keys or the uniqueidentifier data type.

Although configuring merge replication for Compact Framework applications is fairly straightforward, it does entail using a subset of the features available to other subscribers. Some of the limitations that need to be considered follow:

* No alternate synchronization partners: Once a subscription has been created on the device, the Compact Framework application must always synchronize with the original publisher. In other words, alternate synchronization partners are not supported for SQLCE (SQL Server Compact). the CE stands for Community Edition.
* Restricted use of identity columns: Because SQLCE supports only identity columns of type integer, subscribing to an article that includes identity columns of any other data type will fail.
* No computed columns: SQLCE does not support creating computed columns, and so, subscribing to tables that contain computed columns will automatically create a vertical partition that excludes those columns.
* No case sensitivity: As in RDA, it is important to remember that SQLCE does not support case-sensitivity, and so, primary keys and unique indexes based on case-sensitive data in a publication cannot be created on SQLCE.
* No null values in unique constraints: Because SQLCE does not allow NULL values in a column with a UNIQUE constraint, while SQL Server does, subscriptions cannot be created on SQLCE if the publication contains NULL values in a column with a UNIQUE constraint.
* NOT FOR REPLICATION ignored: Because SQLCE does not support this option, any constraints that are marked NOT FOR REPLICATION will still be created on the SQLCE subscriber.

**Security and Connectivity**

Although many of the security settings discussed in the RDA section of this chapter also apply to merge replication, creating a publication also entails configuring several other security-related settings:

* Database access: Because the server agent on IIS invokes the SQL Server Reconciler, the identity of the process on IIS must have permissions on the database in SQL Server. For example, if anonymous authentication is selected in IIS, coupled with Windows authentication in SQL Server, then the IUSR\_computername account on the IIS server must be given permission to access the database. Likewise, if basic or integrated authentication is chosen in IIS and Windows authentication in SQL Server, then the account specified in the InternetLogin property of the SqlCeReplication class must be provided access. Finally, if SQL Server authentication is used, then the account specified by the PublisherLogin property of the SqlCeReplication class must be given access.
* PAL (Phase Alternate Line): Each publication on the SQL Server is also associated with a list of logins that SQL Server uses to restrict access to the publication during synchronization. The account configured in the previous bullet point must also be added to the PAL for publication by right-clicking the publication in the Enterprise Manager and clicking the Publication Access List tab. Optionally, an administrator can also enable the Check Permissions option in the Article Properties dialog to force SQL Server to ensure that the SQL Server Reconciler has permission to perform INSERT, UPDATE, and DELETE statements on the publication database.

In addition to the security settings mentioned in this list, the SqlCeReplication class supports configuring connectivity and security through a series of properties. For example, the PublisherSecurityMode property can be used to specify the authentication mode of the publisher. If the property is set to SecurityType.DBAuthentication, then the PublisherLogin and PublisherPassword properties must be specified so that the SQL Server Reconciler on IIS can use these to login to the publisher. Additionally, the PublisherNetwork and PublisherAddress properties can be used to configure which network protocol and address the Reconciler can use to connect to the publisher if the default protocol configured on IIS should not be used. The only required properties, however, are Publisher and PublisherDatabase, which specify the name of the publishing server and the name of the publication database, respectively.

In the same way the SqlCeReplication class supports a series of distributor properties, none of which is required if the publisher and distributor are on the same computer, but which are used by the Reconciler to connect to the distributor.

As you can imagine, these properties are best configured using the system registry or a configuration file on the database to make them easier to update. As a result, a class that wraps the initialization and creation of SqlCeReplication objects using a simple version of the prototype design pattern[23] would be useful. A skeleton of such a class is shown in Listing 7-6.

[23] Documented in Design Patterns and referenced in the "Related Reading" section at the end of this chapter.

Listing 7-6 Wrapping SqlCeReplication. This class acts as a factory for creating instances of SqlCeReplication and configuring them with the appropriate properties.

public sealed class ReplicationFactory

{

private ReplicationFactory {}

static ReplicationFactory

{

/\*

Read the configuration properties from the

registry or config file.

These would include at a minimum:

Publisher, PublisherDatabase, PublisherSecurityMode

SubscriberConnectionString, Subscriber

InternetLogin, InternetURL

And the following if necessary:

PublisherLogin, PublisherPassword

PublisherAddress, PublisherNetwork

InternetProxy properties if necessary

Distributor properties

LoginTimeout, QueryTimeout

\*/

}

public static SqlCeReplication CreateCeReplication()

{

SqlCeReplication rep = new SqlCeReplication();

\_initClass(ref rep);

return rep;

}

public static SqlCeReplication CreateCeReplication(

string publication)

{

SqlCeReplication rep = new SqlCeReplication();

\_initClass(ref rep);

rep.Publication = publication;

return rep;

}

public static SqlCeReplication CreateCeReplication(

string publication, ExchangeType exchangeType)

{

SqlCeReplication rep = new SqlCeReplication();

\_initClass(ref rep);

rep.Publication = publication;

rep.ExchangeType = exchangeType

return rep;

}

private static void \_initClass(ref SqlCeReplication rep)

{

// set the properties of the rep object based on the values

// read in the constructor

}

}

As you can see, the ReplicationFactory class reads in the default properties from a configuration file or the registry in the static constructor. These could optionally be exposed as static properties that can be set programmatically. It then exposes overloaded CreateCeReplication methods to create the replication objects populated with the default properties, as well as, optionally, the name of the publication and the exchange type (bidirectional or upload only). Obviously, there are a number of permutations of both the configuration properties to set and the arguments to expose in methods such as CreateCeReplication. The class designer's choice of which to set and expose depends on such factors as the security configuration and whether the application supports multiple subscriptions. In this case the class assumes that different publications or exchange types may be configured at runtime.

A client can then use this class as shown in the following snippet, which will return a fully functional SqlCeReplication object that is ready to add the subscription and begin synchronization.

Dim myRep as SqlCeReplication

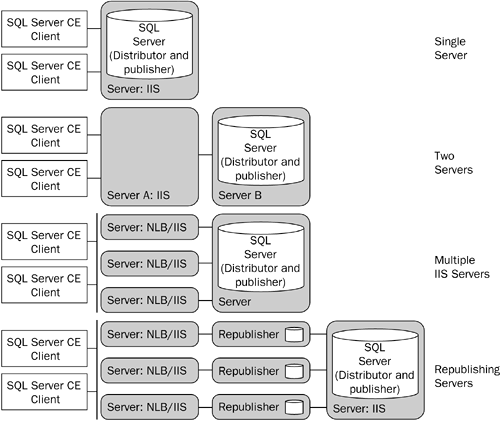
myRep = ReplicationFactory.CreateCeReplication( \_

"CustomerData", ExchangeType.BiDirectional)

**Topologies**

1. Single server: With this option, the publisher, distributor, and server agent with IIS all reside on a single machine. While this configuration certainly puts more load on the loan server and is somewhat less secure, it is easy to configure and can be suitable for applications with few users or when the publisher is used simply to collect data that is later used to update another database via DTS. In this configuration the server can reside behind a firewall.
2. Two servers: In this configuration, the server agent and IIS reside on a server separate from the publisher and distributor (both running on the same SQL Server). With this option, the load on the server is reduced, and the IIS server can reside in a DMZ behind a firewall with another firewall protecting the publisher or distributor. This option enhances security and is used when there are a limited number of concurrent synchronizations.
3. Multiple IIS servers: If the system needs to support many concurrent synchronizations, the two-server solution can be scaled out by implementing a Web farm of IIS servers using load-balancing hardware or software such as the NLB of Windows 2000 Server. NLB can be used because it allows the initial client agent request to be redirected to one of the servers in the cluster and then ensures that the conversation with the server agent on the selected server is preserved (referred to as client affinity, as mentioned previously).
4. Republishing servers: Finally, if additional load balancing is required, a series of republishing servers can be configured. These servers act as subscribers to the original publisher and then republish the data through their own merge publications. This offloads processing from the original publisher; however, this option introduces additional configuration concerns because the SQLCE does not support alternate synchronization partners. In other words, each SQLCE subscriber would need to be configured statically with the name of the publisher, making the load balancing less dynamic.

Figure 7-7. Merge Replication Topologies. This diagram illustrates the various topologies that can be used with merge replication and SQLCE.



**Using Replication**

Once the publication is configured on the server and the connectivity and security properties configured in the Compact Framework application, actually creating the subscription and performing the synchronization are relatively straightforward.

The SqlCeReplication class exposes the AddSubscription, ReinitializeSubscription, and Synchronize methods to create the subscription, reapply the publication's snapshot, and invoke replication, respectively. For example, in the previous code snippet, the ReplicationFactory was used to create a SqlCeReplication object with all of its properties set for the CustomerData publication. To then create the subscription and apply the initial snapshot, the following code can be executed:

Dim myRep as SqlCeReplication

myRep = ReplicationFactory.CreateCeReplication( \_

"CustomerData", ExchangeType.BiDirectional)

Try

myRep.AddSubscription(AddOption.ExistingDatabase)

myRep.Validate = ValidateType.NoValidation

myRep.Synchronize()

Catch ex As SqlCeException

' Wrap the exception and report the error

End Try

**TIP**

Alternatively, the ReplicationFactory class could expose additional overloaded CreateSubscription methods that create and initialize the SqlCeReplication object in addition to calling the AddSubscription and Synchronize methods.

You'll notice that the AddSubscription method can either use the existing database specified in the SubscriberConnectionString property or create a new one. Additionally, as shown in the snippet, the Validate property specifies whether the synchronization process is validated by comparing the row count of the published data on the server with that on the subscriber.[24] This option causes locking to occur on the publisher as row counts are checked, and so, typically this option would be used at some interval to ensure that the subscriber is still in sync with the publisher. If the validation fails, the code can call the ReinitializeSubscription method to reapply the snapshot to SQLCE.

[24] SQL Server also supports validating by calculating a checksum, but SQLCE only supports using the row count value. As a result, the ValidateType enumeration only supports the RowCountOnly value.

**TIP**

ReinitializeSubscription must also be executed when using dynamic filters if the HostName property is changed.

After a synchronization occurs, the SqlCeReplication object will populate the PublisherConflicts, PublisherChanges, and SubscriberChanges properties with row counts that detail the changes that were made. These properties can then be displayed through the UI of the Compact Framework application.

It's important to keep in mind that conflicts detected by the SQL Server Reconciler will be handled by the resolvers configured on the publisher. While there are tracking tables that can be accessed on the server that allow conflicts and their resolutions to be viewed, unlike RDA, SQLCE does not track those conflicts on the device.