When we were deciding which topics to cover in this book, we came to a point where we wanted to try something a little adventurous. Mark suggested that maybe we should include replication. After I got my breath back, we both agreed that replication would be in the book.

Replication is one of those subjects that most people shy away from because they think it’s hard. Replication should be regarded as a specialty. There are people out there who know replication very well: Microsoft SQL Server MVPs Mike Hotek, Hilary Cotter, and Ron Talmage, to name a few. (You’ll note the omissions of two names from that list: Mark Allison and Allan Mitchell. With all due humility, we’ll stand on the proverbial shoulders of the giants.)

With all this in mind, then, we wanted to present replication without the fire and brimstone, devoid of all the mysticism. Now, we won’t even try to dispel those replication fears (illusions?), but we hope that by looking at replication through the eyes of SQL-DMO you’ll better understand how everything fits together—and how you can exercise more granular control over the process. The application we’ve built is very simple, but it does what it says on the package.

In this chapter, we cover the following topics:

• Types of replication

• Installing merge replication using SQL-DMO and Enterprise Manager

• Generating replication scripts using SQL-DMO

Types of Replication

Although you’re going to be building a merge replication scenario in this chapter, it’s worth looking at the other types of replication available to you.

• Snapshot: This type of replication pushes the whole publication to the subscriber at given intervals. This in general minimizes the number of times you send the data, but it increases the amount of data you send because you have to send all of the data each time.
• **Transactional:** Whenever a change is made to a table that's part of the publication, it's communicated to the subscriber. This, again in general, causes you to send data to the subscriber more often, but you send much less data each time. Transactional replication is designed for well-connected servers—in other words, servers where the network/Internet connection is up the majority of the time.

• **Merge:** In this type of replication, subscribers and publishers make changes to their copies of the publication independent of each other, and when they synchronize, the data is merged. Any conflicts encountered are resolved based on preset parameters. Merge replication is designed for servers and clients that aren't always online—for instance, sales agents who get a portion of their data on their laptop, and then go out in the field and hourly/nightly/weekly synchronize their changes with the publication. Unlike transactional replication, where each bit of data goes across the ether to the subscribers, in merge replication only the net changes go across the wire.

---

**Transactional Replication Example**

The following transactional replication example comes from Microsoft SQL Server MVP Hilary Cotter.

An article showing the Oracle stock price in transactional replication would look like this:

- At 9:00 A.M. Update StockPrice values (“Oracle”, $50.00)
- At 12:00 P.M. Update StockPrice values (“Oracle”, $43.00)
- At 2:30 P.M. Update StockPrice values (“Oracle”, $23.00)
- At 3:30 P.M. Update StockPrice values (“Oracle”, $12.00)

If you were disconnected from the server between 9:00 A.M. and 4:00 P.M., all of these transactions would flow across the ether.

In merge replication, if you were disconnected from 9:00 A.M. to 4:00 P.M., only a single transaction would flow across the ether: the net change showing the price drop of Oracle to $12.00.

- Update StockPrice values (“Oracle”, $12.00)