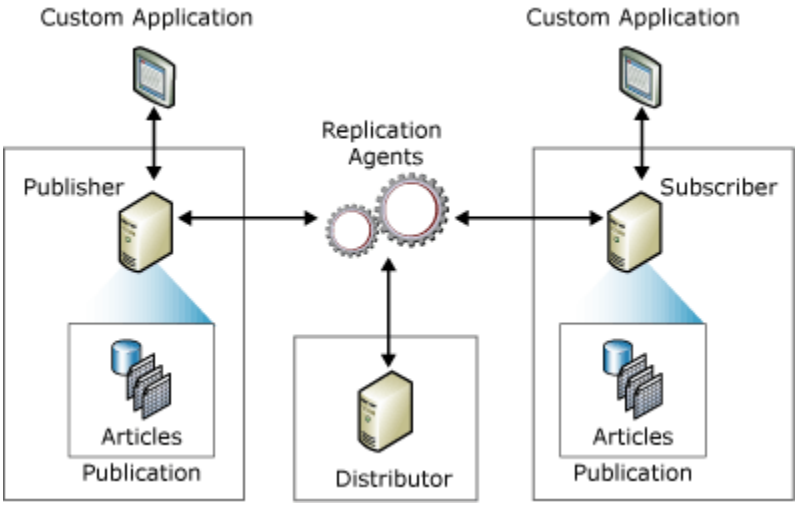
Replication Publishing Model Overview

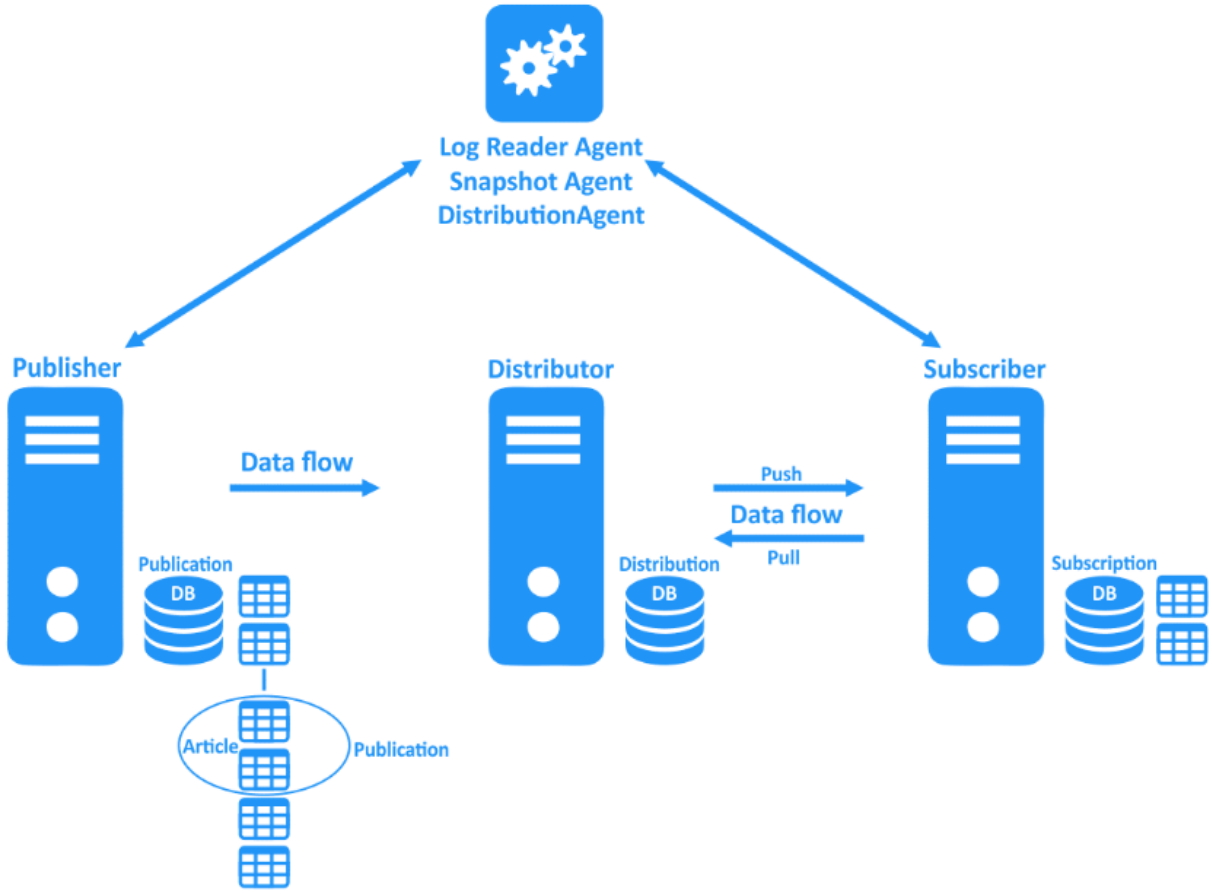
Replication uses a publishing industry metaphor to represent the components in a replication topology, which include Publisher, Distributor, Subscribers, publications, articles, and subscriptions. It is helpful to think of Microsoft SQL Server replication in terms of a magazine:

- A magazine publisher produces one or more publications
- A publication contains articles
- The publisher either distributes the magazine directly or uses a distributor
- Subscribers receive publications to which they have subscribed

Although the magazine metaphor is useful for understanding replication, it is important to note that SQL Server replication includes functionality that is not represented in this metaphor, particularly the ability for a Subscriber to make updates and for a Publisher to send out incremental changes to the articles in a publication.

A *replication topology* defines the relationship between servers and copies of data and clarifies the logic that determines how data flows between servers. There are several replication processes (referred to as *agents*) that are responsible for copying and moving data between the Publisher and Subscribers. The following illustration is an overview of the components and processes involved in replication.





Publisher

The Publisher is a database instance that makes data available to other locations through replication. The Publisher can have one or more publications, each defining a logically related set of objects and data to replicate.

Distributor

The Distributor is a database instance that acts as a store for replication specific data associated with one or more Publishers. Each Publisher is associated with a single database (known as a distribution database) at the Distributor. The distribution database stores replication status data, metadata about the publication, and, in some cases, acts as a queue for data moving from the Publisher to the Subscribers. In many cases, a single database server instance acts as both the Publisher and the Distributor. This is known as a *local Distributor*. When the Publisher and the Distributor are configured on separate database server instances, the Distributor is known as a *remote Distributor*.

Subscribers

A Subscriber is a database instance that receives replicated data. A Subscriber can receive data from multiple Publishers and publications. Depending on the type of replication chosen, the Subscriber can also pass data changes back to the Publisher or republish the data to other Subscribers.

Article

An article identifies a database object that is included in a publication. A publication can contain different types of articles, including tables, views, stored procedures, and other objects. When tables are published as articles, filters can be used to restrict the columns and rows of the data sent to Subscribers.

Publication

A publication is a collection of one or more articles from one database. The grouping of multiple articles into a publication makes it easier to specify a logically related set of database objects and data that are replicated as a unit.

Subscription

A subscription is a request for a copy of a publication to be delivered to a Subscriber. The subscription defines what publication will be received, where, and when. There are two types of subscriptions: push and *pull*.

Replication Agents Overview

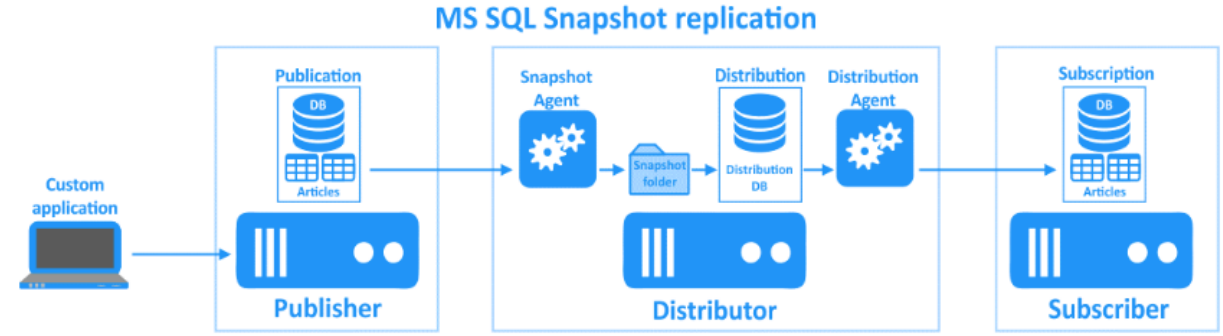
SQL Server Agent

SQL Server Agent hosts and schedules the agents used in replication and provides an easy way to run replication agents. SQL Server Agent also controls and monitors operations outside of replication.

Snapshot Agent

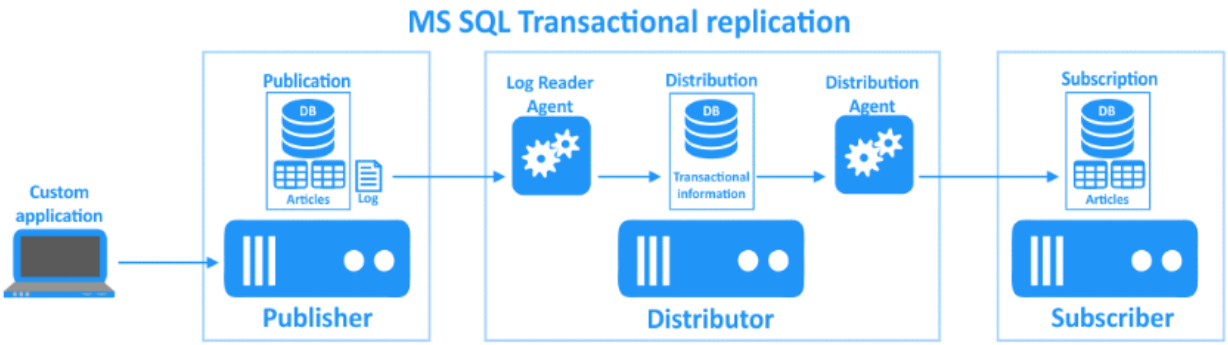
The Snapshot Agent is typically used with all types of replication. It prepares schema and initial data files of published tables and other objects, stores the snapshot files, and records information about synchronization in the distribution database. The Snapshot

Agent runs at the Distributor.



Log Reader Agent

The Log Reader Agent is used with transactional replication. It moves transactions marked for replication from the transaction log on the Publisher to the distribution database. Each database published using transactional replication has its own Log Reader Agent that runs on the Distributor and connects to the Publisher (the Distributor can be on the same computer as the Publisher).



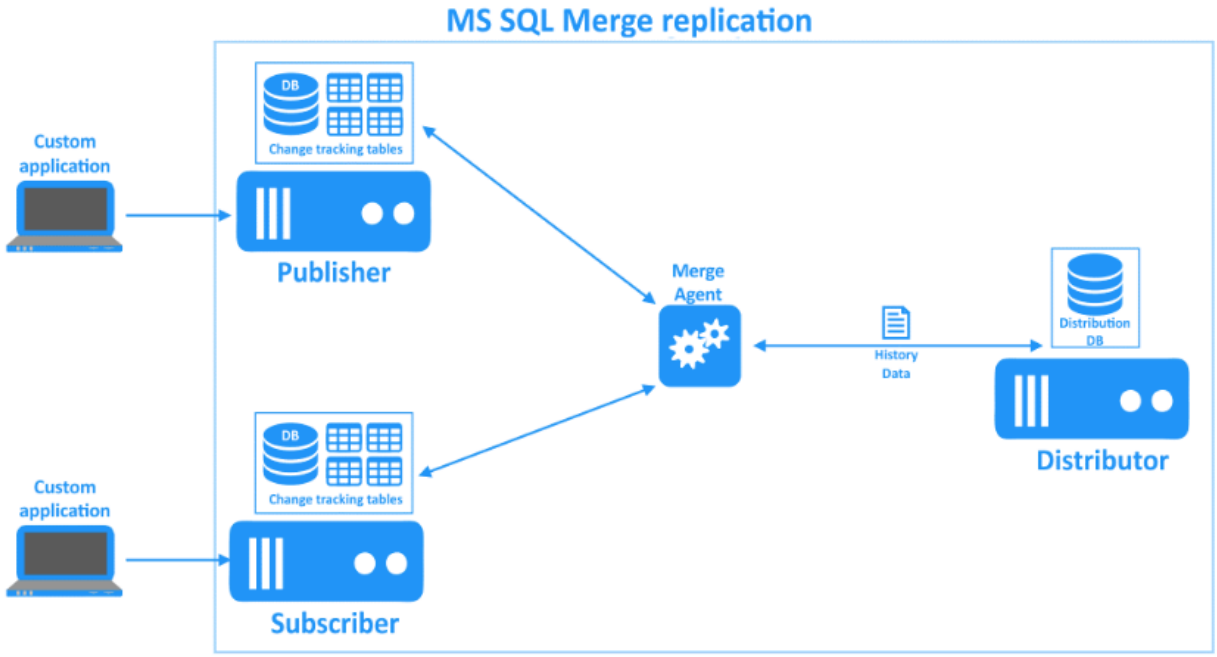
Distribution Agent

The Distribution Agent is used with snapshot replication and transactional replication. It applies the initial snapshot to the Subscriber and moves transactions held in the distribution database to Subscribers. The Distribution Agent runs at either the Distributor for push subscriptions or at the Subscriber for pull subscriptions.

Merge Agent

The Merge Agent is used with merge replication. It applies the initial snapshot to the Subscriber and moves and reconciles incremental data changes that occur. Each merge subscription has its own Merge Agent that connects to both the Publisher and the Subscriber and updates both. The Merge Agent runs at either the Distributor for push

subscriptions or the Subscriber for pull subscriptions. By default, the Merge Agent uploads changes from the Subscriber to the Publisher and then downloads changes from the Publisher to the Subscriber. For more information.



Queue Reader Agent

The Queue Reader Agent is used with transactional replication with the queued updating option. The agent runs at the Distributor and moves changes made at the Subscriber back to the Publisher. Unlike the Distribution Agent and the Merge Agent, only one instance of the Queue Reader Agent exists to service all Publishers and publications for a given distribution database.

Replication Maintenance Jobs

Replication has a number of maintenance jobs that perform scheduled and on-demand maintenance.

Merge Replication

https://learn.microsoft.com/en-us/sql/relational-databases/replication/merge/merge-replication?view=sql-server-ver16

Merge replication, like transactional replication,

typically starts with a snapshot of the publication database objects and data. Subsequent data changes and schema modifications made at the Publisher and Subscribers are tracked with triggers.

The Subscriber synchronizes with the Publisher when connected to the network and exchanges all rows that have changed between the Publisher and Subscriber since the last time synchronization occurred.

Merge replication is typically used in server-to-client environments. Merge replication is appropriate in any of the following situations:

- Multiple Subscribers might update the same data at various times and propagate those changes to the Publisher and to other Subscribers.
- Subscribers need to receive data, make changes offline, and later synchronize changes with the Publisher and other Subscribers.
- Each Subscriber requires a different partition of data.
- Conflicts might occur and, when they do, you need the ability to detect and resolve them.
- The application requires net data change rather than access to intermediate data states. For example, if a row changes five times at a Subscriber before it synchronizes with a Publisher, the row will change only once at the Publisher to reflect the net data change (that is, the fifth value).

Merge replication allows various sites to work autonomously and

Later merge updates into a single, uniform result.

Because updates are made at more than one node, the same data may have been updated by the Publisher and by more than one Subscriber. Therefore, conflicts can occur when updates are merged and merge replication provides a number of ways to handle conflicts.

Merge replication is implemented by the SQL Server Snapshot Agent and Merge Agent.

If the publication is unfiltered or uses static filters, the Snapshot Agent creates a single snapshot.

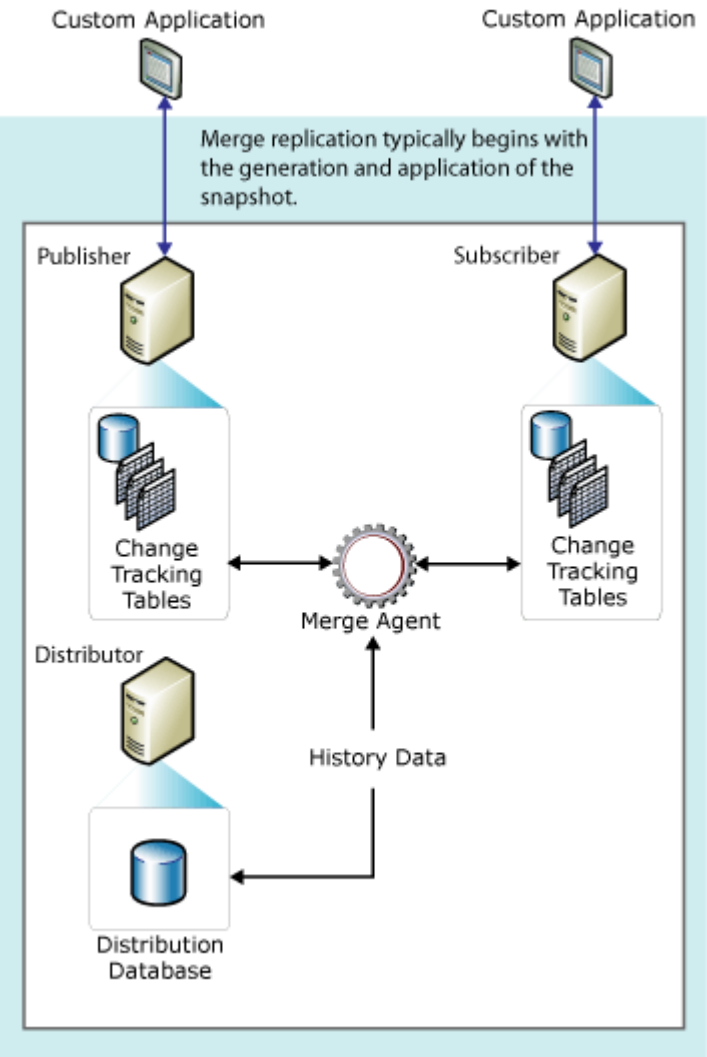
If the publication uses parameterized filters, the Snapshot Agent creates a snapshot for each partition of data.

The Merge Agent applies the initial snapshots to the Subscribers.

It also merges incremental data changes that occurred at the Publisher or Subscribers after the initial snapshot was created, and detects and resolves any conflicts according to rules you configure.

To track changes, merge replication (and transactional replication with queued updating subscriptions) must be able to uniquely identify every row in every published table. To accomplish this merge replication adds the column rowguid to every table, unless the table already has a column of data type uniqueidentifier with the ROWGUIDCOL property set (in which case this column is used). If the table is dropped from the publication, the rowguid column is removed; if an existing column was used for tracking, the column is not removed. A filter must not include the rowguidcol used by replication to identify rows. The newid() function is provided as a default for the rowguid column, however customers can provide a guid for each row if needed. However, do not provide value 00000000-0000-0000-0000-000000000000.

The following diagram shows the components used in merge replication.



Transactional Replication

https://learn.microsoft.com/en-us/sql/relational-databases/replication/transactional/transactional-replication?view=sql-server-ver16

Transactional replication typically

starts with a snapshot of the publication database objects and data.

As soon as the initial snapshot is taken, subsequent data changes and schema modifications made at the Publisher are usually delivered to the Subscriber as they occur (in near real time).

The data changes are applied to the Subscriber in the same order and within the same transaction boundaries as they occurred at the Publisher; therefore, within a publication, transactional consistency is guaranteed.

Transactional replication is typically used in server-to-server environments and is appropriate in each of the following cases:

- You want incremental changes to be propagated to Subscribers as they occur.
- The application requires low latency between the time changes are made at the Publisher and the changes arrive at the Subscriber.
- The application requires access to intermediate data states. For example, if a row changes five times, transactional replication allows an application to respond to each change (such as firing a trigger), not simply the net data change to the row.
- The Publisher has a very high volume of insert, update, and delete activity.
- The Publisher or Subscriber is a non-SQL Server database, such as Oracle.

By default, Subscribers to transactional publications should be treated as read-only, because changes are not propagated back to the Publisher. However, transactional replication does offer options that allow updates at the Subscriber.

How Transactional Replication Works

Transactional replication is implemented by the SQL Server Snapshot Agent, Log Reader Agent, and Distribution Agent.

The Snapshot Agent prepares snapshot files containing schema and data of published tables and database objects, stores the files in the snapshot folder, and records synchronization jobs in the distribution database on the Distributor.

The Log Reader Agent monitors the transaction log of each database configured for transactional replication and copies the transactions marked for replication from the transaction log into the distribution database, which acts as a reliable store-and-forward queue.

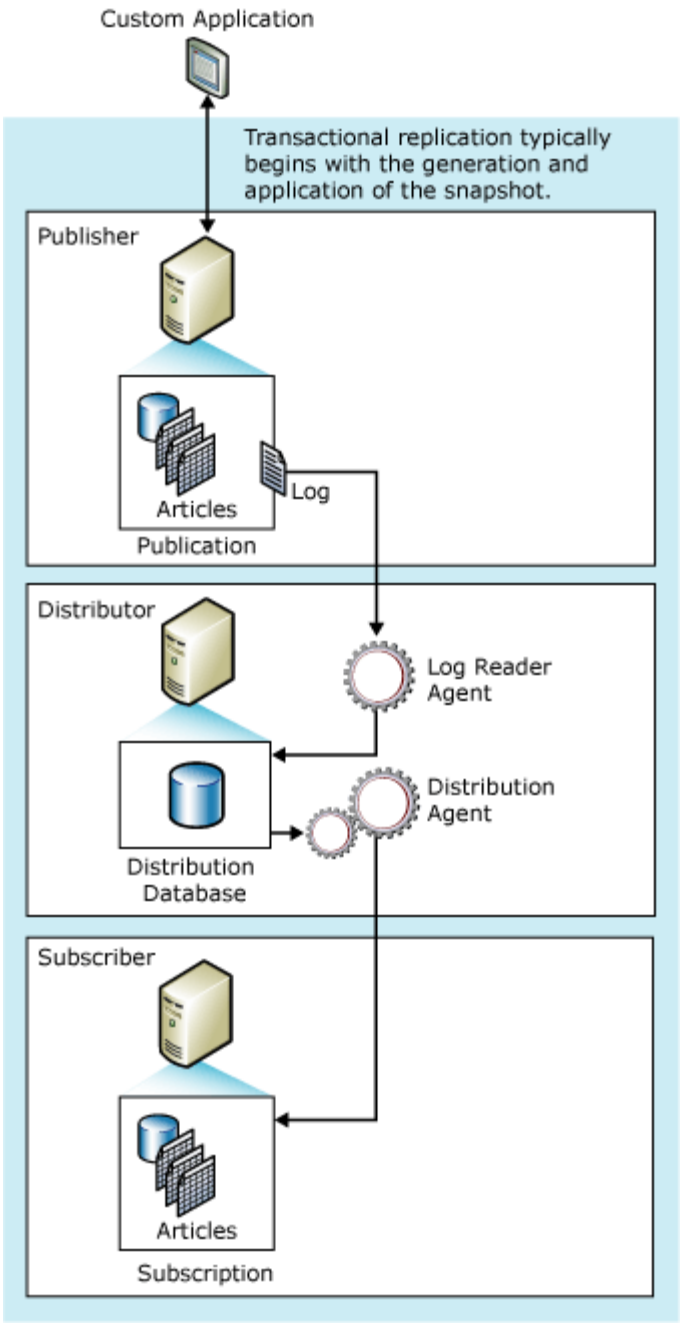
The Distribution Agent copies the initial snapshot files from the snapshot folder and the transactions held in the distribution database tables to Subscribers.

Incremental changes made at the Publisher flow to Subscribers according to the schedule of the Distribution Agent, which can run continuously for minimal latency, or at scheduled intervals.

Because changes to the data must be made at the Publisher (when transactional replication is used without immediate updating or queued updating options), update conflicts are avoided. Ultimately, all Subscribers will achieve the same values as the Publisher.

If immediate updating or queued updating options are used with transactional replication, updates can be made at the Subscriber, and with queued updating, conflicts might occur.

The following illustration shows the principal components of transactional replication.



Initial Dataset

Before a new transactional replication Subscriber can receive incremental changes from a Publisher, the Subscriber must contain tables with the same schema and data as the tables at the Publisher. The initial dataset is typically a snapshot that is created by the Snapshot Agent and distributed and applied by the Distribution Agent. The initial dataset can also be supplied through a backup or other means, such as SQL Server Integration Services.

When snapshots are distributed and applied to Subscribers, only those Subscribers waiting for initial snapshots are affected. Other Subscribers to that publication (those that have already been initialized) are unaffected.

Concurrent Snapshot Processing

Snapshot replication places shared locks on all tables published as part of replication for the duration of snapshot generation. This can prevent updates from being made on the publishing tables. Concurrent snapshot processing, the default with transactional replication, does not hold the share locks in place during the entire snapshot generation, which allows users to continue working uninterrupted while replication creates initial snapshot files.

Snapshot Agent

The procedures by which the Snapshot Agent implements the initial snapshot in transactional replication are the same procedures used in snapshot replication (except as outlined above with regard to concurrent snapshot processing).

After the snapshot files have been generated, you can view them in the snapshot folder using Microsoft Windows Explorer.

Modifying Data and the Log Reader Agent

The Log Reader Agent runs at the Distributor; it typically runs continuously, but can also run according to a schedule you establish. When executing, the Log Reader Agent first reads the publication transaction log (the same database log used for transaction tracking and recovery during regular SQL Server Database Engine operations) and identifies any INSERT, UPDATE, and DELETE statements, or other modifications made to the data in transactions that have been marked for replication. Next, the agent copies those transactions in batches to the distribution database at the Distributor. The Log Reader Agent uses the internal stored procedure sp_replcmds to get the next set of commands marked for replication from the log. The distribution database then becomes the store-and-forward queue from which changes are sent to Subscribers. Only committed transactions are sent to the distribution database.

After the entire batch of transactions has been written successfully to the distribution database, it is committed. Following the commit of each batch of commands to the Distributor, the Log Reader Agent calls sp_repldone to mark where replication was last completed. Finally, the agent marks the rows in the transaction log that are ready to be purged. Rows still waiting to be replicated are not purged.

Transaction commands are stored in the distribution database until they are propagated to all Subscribers or until the maximum distribution retention period has been reached. Subscribers receive transactions in the same order in which they were applied at the Publisher.

Distribution Agent

The Distribution Agent runs at the Distributor for push subscriptions and at the Subscriber for pull subscriptions. The agent moves transactions from the distribution database to the Subscriber. If a subscription is marked for validation, the Distribution Agent also checks whether data at the Publisher and Subscriber match.

Publication types

Transactional replication offers four publication types:

Publication Type	Description
Standard transactional publication	Appropriate for topologies in which all data at the Subscriber is read-only (transactional replication does not enforce this at the Subscriber). Standard transactional publications are created by default when using Transact-SQL or Replication Management Objects (RMO). When using the New Publication Wizard, they are created by selecting Transactional publication on the Publication Type page. For more information about creating publications, see Publish Data and Database Objects.
Transactional publication with updatable subscriptions	The characteristics of this publication type are: -Each location has identical data, with one Publisher and one Subscriber. -It is possible to update rows at the Subscriber -This topology is best suited for server environments requiring high availability and read scalability. For more information, see Updatable Subscriptions.
Peer-to-peer topology	The characteristics of this publication type are: - Each location has identical data and acts as both a Publisher and Subscriber. - The same row can be changed in only one location at a time. - Supports conflict detection - This topology is best suited for server environments requiring high availability and read scalability.

Publication Type	Description
	For more information, see Peer-to-Peer Transactional Replication.
Bidirectional transactional replication	The characteristics of this publication type are: Bidirectional replication is similar to Peer-to-Peer replication, however, it does not provide conflict resolution. Additionally, bidirectional replication is limited to 2 servers. For more information, see Bidirectional Transactional Replication