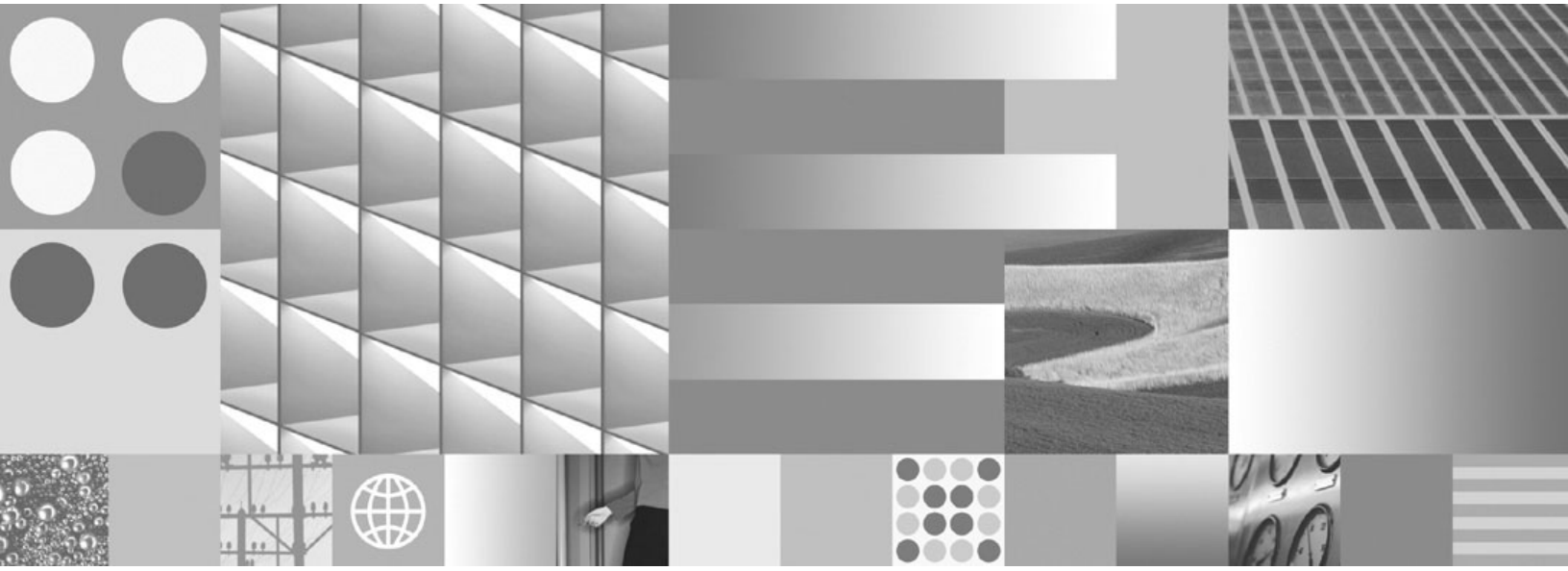
IBM Information Integration

Version 9.5

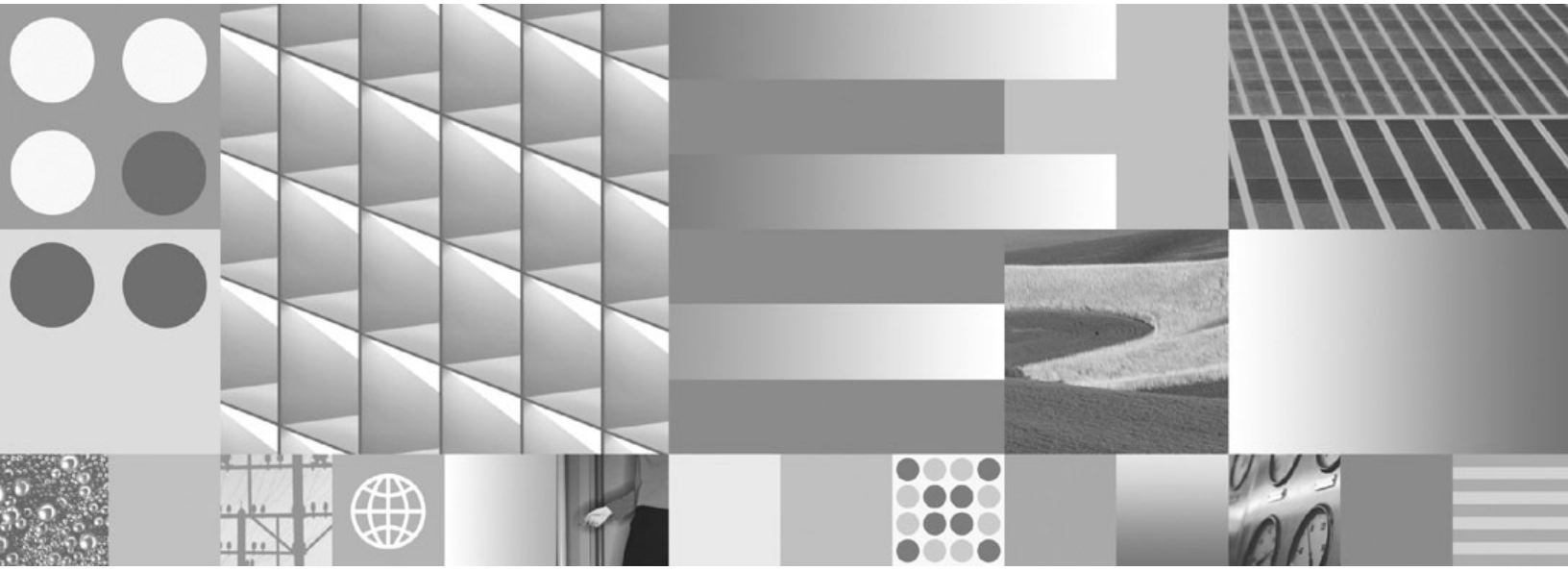


Introduction to Replication and Event Publishing

GC19-1028-01

IBM Information Integration

Version 9.5



Introduction to Replication and Event Publishing

GC19-1028-01

Note
Before using this information and the product that it supports, read the information in "Notices" on page 55.

Contents

Chapter 1. Introduction to SQL replication

SQL replication captures changes to sources and uses staging tables to store committed transactional data. The changes are then read from the staging tables and replicated to corresponding target tables. With staging tables, data can be captured and staged once for delivery to multiple targets, in different formats, and at different delivery intervals.

You can use SQL replication for a variety of purposes that require replicated data, including capacity relief, feeding data warehouses and data marts, and auditing change history.

You can replicate continuously, at intervals, or for one time only. Replicating continuously can be useful if your applications need data in near-real-time, such as applications for making airline reservations. Replicating at intervals can be useful for replicating large batches of data during off-peak hours. You can also trigger replication through database events.

Sources and targets can be either in DB2® databases or in non-DB2 relational databases. You can replicate from sources and to targets in the following relational database management systems: DB2 on Linux®, UNIX®, Windows®, z/OS®, and iSeries™; Informix®; Microsoft® SQL Server; Oracle; Sybase; and Teradata (target only). If you plan to replicate to or from a non-DB2 relational database, you will need the federated server function of DB2 Information Integrator. See the *IBM WebSphere Information Integration Federated Systems Guide* for more information about federated servers.

You have considerable flexibility with the data that you want to replicate. For example, you can choose to have all rows and columns replicated or you can choose just a subset of these. By subsetting rows and columns, you transport across your network only the data that you want.

You can also clean, aggregate, or otherwise manipulate data. If you manipulate data, you can do so centrally at the source and then distribute the manipulated data, or you can manipulate the data when you replicate it, so that some targets get manipulated data and others do not.

Chapter 2. Introduction to Q replication

Q replication is a replication solution that can replicate large volumes of data at low levels of latency. Q replication captures changes to source tables and converts committed transactional data to messages.

The data is not staged in tables. As soon as the data is committed at the source and read by Q replication, the data is sent. The messages are sent to the target location through WebSphere® MQ message queues. At the target location, the messages are read from the queues and converted back into transactional data. The transactions are then applied to your target tables with a highly parallelized method that preserves the integrity of your data.

You can use Q replication for a variety of purposes that require replicated data, including failover, capacity relief, geographically distributed applications, and data availability during rolling upgrades or other planned outages.

Sources can be relational tables in DB2 servers on Linux, UNIX, Windows, or z/OS, and targets can be relational tables in DB2 servers on Linux, UNIX, Windows, or z/OS, or relational tables in supported non-DB2 relational databases. You can replicate from a DB2 source on any of these platforms to a DB2, Informix, Microsoft SQL Server, Oracle, or Sybase target on the same platform or any of the other supported platforms. To use WebSphere MQ message queues for transporting data, you need WebSphere MQ on your source and target systems.

You can also use Q replication to replicate data from non-relational tables by using WebSphere Classic Replication Server for z/OS, WebSphere MQ, and WebSphere Replication Server. Classic replication uses its own mechanism for capturing changes. The Classic capture components converts transactions to WebSphere MQ messages that are picked up by the Q Apply program on the WebSphere Replication Server.

You can replicate a subset of columns and rows from the source tables. All subsetting occurs at the source location so that only the data that you want is transported across the network. If you want to perform data transformations, you can use the SQL column expressions feature in Q Apply or pass replicated data to your own stored procedures.

Chapter 3. Introduction to event publishing

Event publishing captures changes to source tables and converts committed transactional data to messages in an Extensible Markup Language (XML) format or delimited format. Each message can contain an entire transaction or only a row-level change. These messages are put on WebSphere MQ message queues and read by a message broker or other applications. You can publish subsets of columns and rows from source tables so that you publish only the data that you need.

You can use event publishing for a variety of purposes that require published data, including feeding central information brokers and Web applications, and triggering actions based on updates, inserts, or deletes to source tables.

Source tables can be on DB2 servers on Linux, UNIX, Windows, or z/OS. To make use of WebSphere MQ message queues for transporting data, have WebSphere MQ available on your source and on the system where messages are received.

Event publishing can be very useful in a variety of different applications. Consider a scenario in which changing prices and inventory are published to potential buyers. For example, a food wholesaler procures perishable food products such as bananas from world markets in bulk and sells them to grocery food retailers and distributors.

The value of bananas decreases the longer that they are in the warehouse. The wholesaler wants to inform its potential buyers of the changing price and inventory data and can set up event publishing to do just that. Each time the price changes, an XML or delimited message can be sent to potential buyers, informing them of the ″price change event.″

Each buyer (retailer or distributor) wants to maximize profit. These organizations can determine when to buy the bananas based upon price, age (or time to spoil), and historical knowledge regarding how quickly they can sell a certain quantity. If they purchase too early, they will pay a higher price for the bananas and will not achieve the maximum profit. Buying too late will likely result in spoilage and again profit will not be maximized. Profit maximization can be achieved by timing the purchase appropriately. Applications can receive the event notification messages and generate purchase orders automatically at the right time to maximize profit.

Chapter 4. Administrative interface for replication and publishing

The Replication Center is a graphical user interface that you can use to define, operate, and monitor your replication and publishing environments. It comes with the DB2 Administration Client and runs on Linux, UNIX, and Windows systems. The Replication Center provides a single interface for administering your replication environments on different platforms across multiple systems.

The following list describes some of the Replication Center's most useful features:

- Launchpads that show you step by step how to configure basic replication and publishing environments.
- Wizards that help you set up simple to highly customized replication and publishing configurations.
- Profiles that you can customize that let you create replication objects with schemas, names, and other attributes that conform to your own conventions and storage requirements.
- Reports that show you how your replication and publishing environments are performing.

If you prefer, you can define and modify your replication and publishing environments through a command-line tool, the ASN command-line processor (ASNCLP). You can then operate the programs for replication and publishing through system commands.

Chapter 5. Utilities for replication and publishing

Three utilities that come with the replication and publishing technologies help you monitor your replication and publishing environments and keep sources and targets synchronized.

Replication Alert Monitor

The Replication Alert Monitor is a utility that checks the health of programs that are part of the replication and event publishing solutions. The monitor checks for situations in which a program terminates, issues a warning or error message, reaches a threshold for a specified value, or performs a certain action. You tell the Replication Alert Monitor which situations to watch for. If any of them occur, the Replication Alert Monitor sends an e-mail message to the person or group of persons that are designated as the appropriate contacts for such a situation.

For example, you might have the Replication Alert Monitor notify you when a replication program is not running or has reached the maximum amount of memory that you expect it to use.

Reconciliation utilities

Source and target tables can lose synchronization when, for example, a target table is changed by an application. The asntdiff and asntrepair utilities can detect and repair differences between source and target tables in Q replication and SQL replication. By using them, you can avoid comparing tables manually and having to reload targets to resynchronize them with sources.

The asntdiff utility generates a relational table of the differences between a source and target table. After you run the asntdiff utility, you can run the asntrepair utility, which uses this table of differences to determine which rows to insert, update, or delete at the target table to synchronize it with the source table.

Chapter 6. SQL replication—Overview

This section describes the basic concepts involved in SQL replication.

The following figure shows a simple configuration in SQL replication. This topics in this section describe the different parts of this figure.

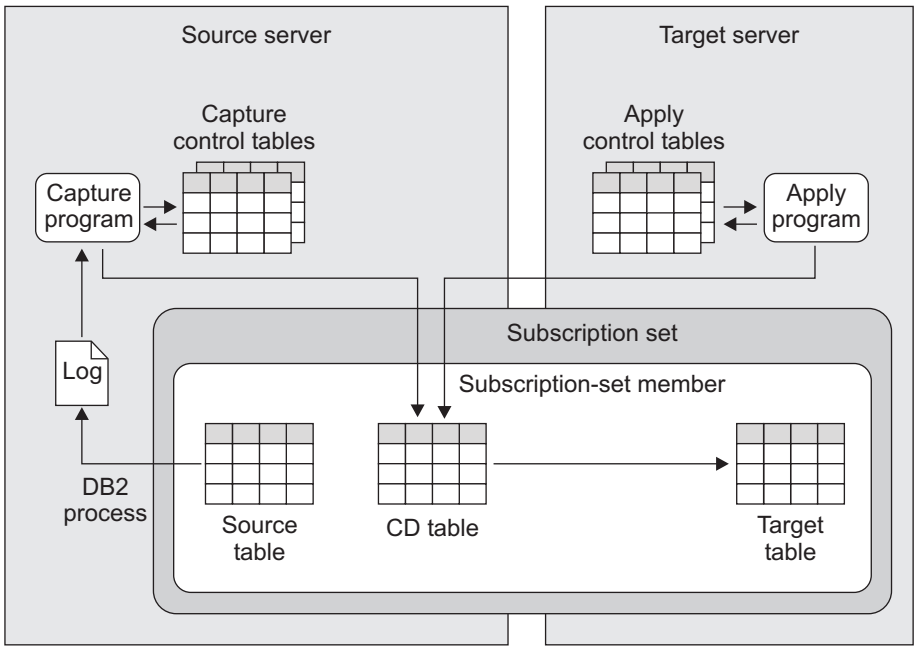


Figure 1. A simple configuration in SQL replication

Infrastructure for an SQL replication environment

SQL replication allows you to replicate data from DB2 sources to targets by using two programs: Capture and Apply.

The Capture program runs on the source system. The Capture program reads DB2 recovery logs for changed source data and saves the committed changed data to staging tables. The Apply program typically runs on the target system. The Apply program retrieves captured data from staging tables and delivers the data to targets. Both programs use a set of DB2 tables to track the information that they require to do their tasks and to store information that they generate themselves, such as information that you can use to find out how well they are performing. You create these tables before you tell SQL replication what your replication sources and targets are.

The Capture program uses a set of DB2 tables called *Capture control tables*. These tables contain information about replication sources and the current position of the Capture program in the DB2 recovery log. In most cases, the control tables for a Capture program need to be on the same DB2 server as the sources associated with the program.

You can run multiple Capture programs on the same DB2 server. Each Capture program uses its own set of Capture control tables. The schema associated with a set of Capture control tables identifies the Capture program that uses those control tables. This schema is called a *Capture schema*.

If your sources are located on a non-DB2 relational database, triggers are used to capture changes to them, though a set of control tables is still required.

The Apply program uses a set of DB2 tables called *Apply control tables*. These tables contain information about your targets and where their corresponding sources are located. The control tables for the Apply program usually reside on the system where the Apply program runs. Unlike with the Capture program, you can create multiple Apply programs that use the same set of control tables. Each Apply program is identified in these control tables by a name called an *Apply qualifier*.

For each supported operating system or platform that is supported by DB2 and for each non-DB2 relational database management system that is supported by SQL replication, you can set your own default specifications for the table spaces that will be used by control tables.

The following figure shows the infrastructure for a simple configuration in SQL replication.

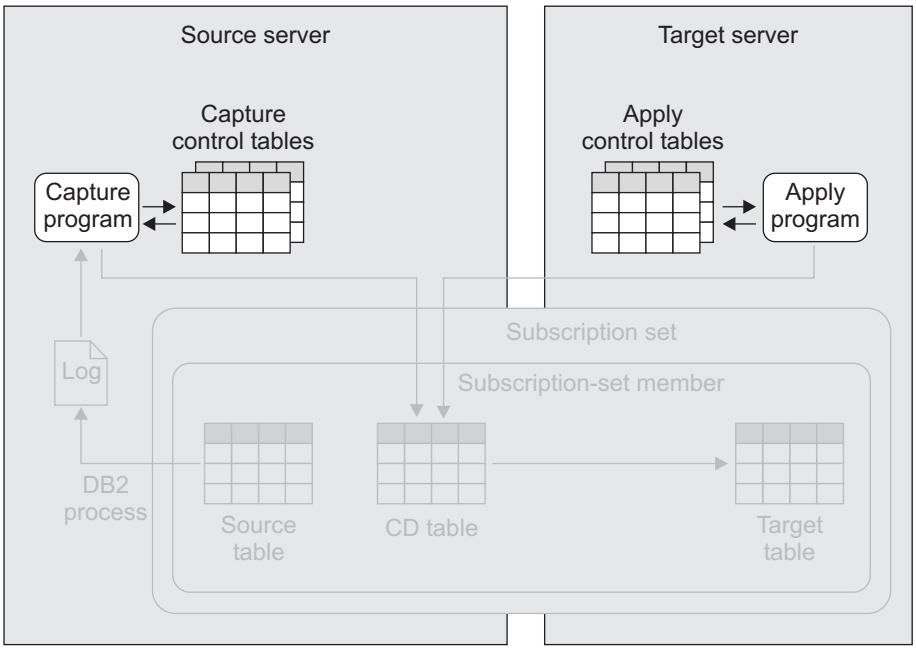


Figure 2. Infrastructure for a simple configuration in SQL replication. You create a set of DB2 relational tables on the source server called Capture control tables. Information about sources is written to these tables. The Capture program, which runs on the source server, uses this information to know what data it is supposed to capture. Information about targets will go in the Apply control tables, which are typically on your target server. The Apply program, which is also typically on your target server, uses this information to know which targets it is supposed to write data to.

Registration of sources in SQL replication

When you know what you want your replication sources to be, you tell SQL replication about them by *registering* them. You can register sources that are DB2 tables and views, or tables on non-DB2 relational databases.

Note: When you register source tables on non-DB2 relational databases, you use SQL replication together with DB2 Relational Connect. You map your source database to a federated database and you create nicknames for each source table.

You can specify one of three methods for replicating a table. The first is called *full-refresh replication*, and the second and third are types of *change-capture replication*:

Replicate by refreshing the target table. Do not capture changes.
At intervals that you can specify later, the Apply program invokes one or more DB2 utilities to delete the content of the corresponding targets for the registered table and load the targets with content derived from the registered table. When you register the table, you need to provide only its schema and name. You can choose to replicate this way if a target is small or if you want to replicate infrequently. For this method of replication, no Capture program or Capture triggers are involved.

Capture a row whenever the value changes in one or more columns that you want to replicate.
This method, the most common way of using SQL replication, reduces the amount of data that is replicated when changes frequently affect only unregistered columns. For example, assume that you have 100 columns in your table and you register 50 of those columns for replication. Whenever a change is made to one of the 50 registered columns in your table, SQL replication captures the values of those 50 columns for the row in which the change occurred. If a row is changed, but the changes are limited to other 50 columns, no data is captured for replication.

Capture a row whenever the value changes in any column of the table.
This method is often chosen when changes in the table almost always occur in registered columns only, or when information is replicated for auditing purposes. For example, you want to know only which rows are changed in a table. Assume that you have 100 columns in your table and you register the primary key columns for replication. Any time a change is made to any of the columns in your table, SQL replication captures the values of the primary key columns for the row in which the change occurred.

When you register a table for either type of change-capture replication, you provide SQL replication with the same information and SQL replication creates the same objects:

- **If your source is a DB2 table:**

  You tell SQL replication the schema and name of the source table, as well as the columns that you want to replicate from it. The columns that you choose are called *registered* columns. They are also referred to as *after-image* columns because they contain values resulting from changes made to the source table. For example, if in one row of a registered table the value in registered column A is changed from 25 to 30, 30 is the after-image.

  You can also choose to replicate the *before-images* of your registered columns, which are the values that existed in the columns before changes were made to the source table. In the example given in the preceding paragraph, 25 is the before-image. Before-images are useful for a number of purposes, including enabling the applying program to find rows in the target when key values have changed.

  SQL replication creates a change-data (CD) table in which to record committed changes to the table. For every column that you register in the source table,

there is an identical column in the CD table. If you choose to replicate before-images for a column, a column is created in the CD table to record those before-images.

- **If your source is a non-DB2 table:**

  You tell SQL replication the schema and name of the nickname for the source table, as well as the columns that you want to replicate from the source table. As is the case when you register DB2 tables, the columns that you choose are called registered columns, and you can replicate after-images and before-images. SQL replication creates three triggers, called *Capture triggers*, directly on the source table in the non-DB2 relational database: one trigger for inserts, another trigger for updates, and one more trigger for deletes. SQL replication also creates a staging table for the source table. The staging table is called a consistent change-data (CCD) table and is populated by the triggers. For example, when an update occurs in the source table, the update trigger on that table stores the update as a record in the corresponding CCD table. Only committed transactional data is stored in the CCD table. A nickname for the CCD tables is created on the federated database.

For every source server, you can set your own default naming conventions for CD and CCD tables. You can also set your own defaults for the table spaces used for CD and CCD tables.

For more information about registering views, see the *IBM WebSphere Information Integration SQL Replication Guide and Reference*.

The following figure shows the objects that are involved when you register a source table.

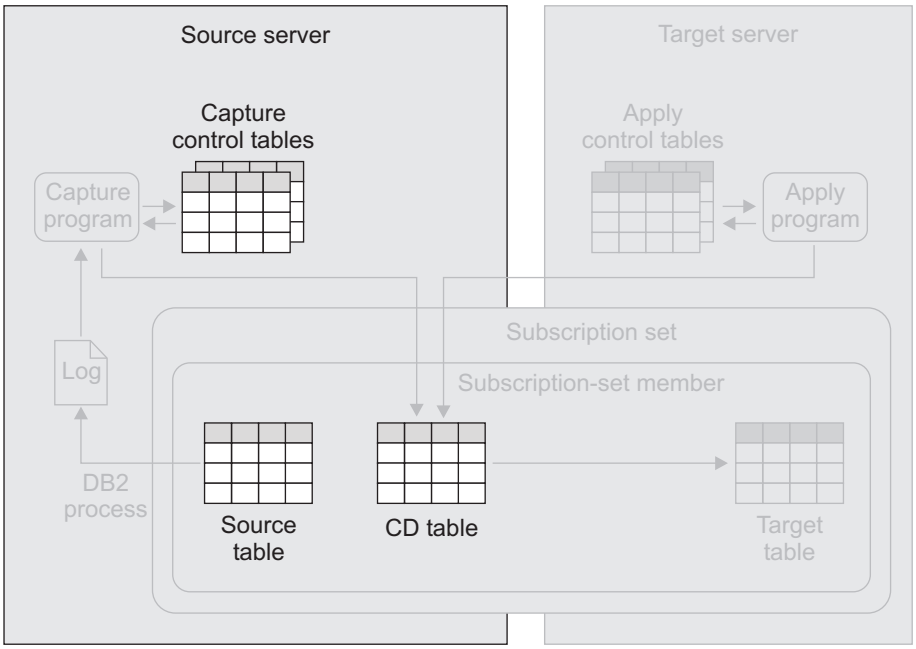


Figure 3. Registering a source table. When you register a source table for replication, SQL replication stores the registration information in the Capture control tables. SQL replication also creates a CD table that the Capture program will use to record changes to the source table.

Subscription sets in SQL replication

After you register your sources, you create *subscription sets*, in which you pair your sources with targets. Each source-target pair is referred to as a member of the subscription set in which it is created.

You can use subscription sets to schedule the replication of data in one or more source-target pairs from one source server to one target server. The Apply program coordinates this replication so that it applies data to targets in a manner consistent with the original transactions on the source server.

For each subscription set that you create, you tell SQL replication the following details:

Which Apply program to use for processing the subscription set
When you create a subscription set, you associate it with an Apply program by selecting an Apply qualifier. Each Apply program can process more than one subscription set.

Where the source tables or views are located
The sources in the members of a subscription set are all located on the same server.

Where the target tables are located
The targets in the members of a subscription set are all located on the same server.

How often to replicate data from the sources to the targets
You can choose to replicate data in a subscription set at regular intervals, continuously, or when an event occurs.

Interval timing
This method, also known as relative timing, is the simplest method of scheduling your subscription set. The interval is approximate, which depends on the workload or system resource available at the time. The interval pertains to all the subscription-set members within the subscription set. Therefore, all the tables will replicate within the subscription set at the each interval.

Continuous replication
This method lets the Apply program replicate data in a subscription set as frequently as it is able, depending on its workload and available resources.

Event timing
This method lets an application or a user determine when to start replicating. The application or user inserts a row into an Apply control table called the events table. When the Apply program sees that row, it begins replicating.

Whether to use data blocking
Data blocking allows you to specify how many minutes worth of change data the Apply program can replicate at a time. The number of minutes that you specify determines the size of the data block. If there is a backlog of change data that is greater than the size of the data block, the Apply program processes the subscription set by replicating a block of data at a time until all of the current change data is replicated. One benefit of data blocking is that by retrieving smaller sets of data, the Apply program can lessen both the network load and the temporary space required for the

retrieved data. Another benefit is that if there is an error, the Apply program has to rollback only the current block of data rather than the entire set of change data.

For example, if you use SQL replication to populate the tables of a data warehouse every 24 hours. Rather than replicate in one block the entire set of changes made to your source tables every 24 hours, the Apply program can replicate 20 minutes worth of changes at a time until the entire 24-hour block of data is replicated. You can adjust the size of the block to match your own requirements and restrictions.

Whether to issue one COMMIT for all applied data or to issue interim commits
The Apply program can apply data in table mode or in transaction mode. In table mode, the Apply program processes the fetched changes for each table separately, and then issues a single commit after all data is applied. In transaction mode, the Apply program applies the fetched changes to all of the target tables, in the order that the changes occurred in their corresponding transactions on the source server. The Apply program commits these transactions at transaction boundaries. You specify how many transactions to apply before each commit.

Whether to transform data in the subscription set with SQL scripts or stored procedures
When data in the source tables of the subscription set is replicated to corresponding targets, you can use SQL scripts or stored procedures to transform that data.

For each source-target pair that you create within a subscription set, you specify whether to create a new target table or use an existing one. If you create a new target table, you can specify one of the following types:

User copy
This read-only target type is the most common for basic data replication. The structure of this type of target table can be the same as the source table, or only a subset of the source columns. Before–images or after–images and calculated columns can also be defined. This target type requires replicated columns that uniquely describe each row of the target table. These columns can be either a unique index or primary key.

Point-in-time
The structure of this read-only type of target table is similar to that of a user copy target table. A timestamp column is added so that the Capture program can indicate when data was committed in the source table. You can select this target type if you want to keep track of the time when the changes were applied to the target table.

Aggregate
This is a read–only target table that summarizes the entire contents of a source table or its changed data. The target columns are defined from the SQL column functions such as SUM, COUNT, MIN, MAX, and AVG. These columns contain the computed value of the SQL function instead of the actual source data. A timestamp is included to indicate when the Apply program performed the aggregation. The two types of aggregation target tables are:

- **Base aggregate.** Summarizes the entire contents of the source table every time the source table is replicated. For example, you could use this target type if you want to keep track of year-to-date summaries or average sales by salesperson or region.

- **Change aggregate.** Summarizes the changes that occurred at the source table since the source table was last replicated. For example, you could use this target type to track monthly sales totals by salesperson, region, or customer.

CCD This read-only target type contains committed changes that occurred at the source, with replication control fields to determine the type operation from the source table (insert, update, and delete). There are different types of CCD tables and the type that you should use depends on your goals and your replication requirement. With a CCD table, you can track the history of source changes in various ways, depending on how you define the CCD table. For example, you can track before and after comparisons of the data, when changes occurred, and which user ID made the update to the source table.

You can also set up a multi-tier configuration, where your target CCD table acts as a source to other target tables. Your source table is the first tier, your CCD the second tier, and the targets mapped to your CCD the third tier. One reason to set up a multi-tier replication environment is to provide stable sources for the third-tier targets. Because you can collect changes from tier 1 in CCD tables at tier 2, you can reduce the number of changes replicated to the targets at tier 3. You can also avoid many of the database connections to your source system, thus moving the connection cost to the second tier.

Replica

This target type is used in update-anywhere replication. The other target types are read-only, but this target type allows writes which can be replicated back to the source table, which acts as a master table. If any conflicts occur with the data in the master table, the change replicated from the replica will be rejected. Replica target tables cannot be defined at non-DB2 relational databases.

For each source-target pair, you also specify:

- **Which source columns to replicate to the target.** Although you can subset columns when you register your source, you can subset further when you create a subscription-set member.
- **How to map columns from the source to columns in the target.** If your target table already exists in the target location, you tell SQL replication how the registered columns in your source table or view map to the columns in the target table. If you want SQL replication to create the target table for you, the registered source columns are automatically mapped to the columns in the target table.

  In both cases, you can transform data by mapping a source column to a target column with a corresponding data type or by mapping a source column to a calculated column.
- **A predicate for replicating a subset of rows to the target.** You can subset the rows that you replicate from the source.
- **A method for loading the target table.** In most cases, the data from the source table is loaded into the target table so that the target table is identical to the source table before replication begins. SQL replication allows for two methods of loading a target table: automatic loads and manual loads.

  For automatic loads, you can tell the Apply program to call one utility or a pair of utilities. The Apply program can call the LOAD FROM CURSOR option of the LOAD utility, the EXPORT and LOAD utilities, or the EXPORT and IMPORT

utilities, depending on the platform on which the Apply program is running. You can also tell the Apply program to choose the option that is most appropriate for the target table.

If you decide to load the target table manually, you can choose any method that you prefer. After performing the load, you notify the Apply program that the target table is ready and replication begins.

When you create a subscription-set member, you can also decide whether to replicate only a subset of the registered source columns and whether to replicate only a subset of the rows from the source.

For every target server, you can set your own default naming conventions for target tables. You can also set your own defaults for the table spaces that are used for your target tables.

The following figure shows a subscription set in a simple configuration in SQL replication.

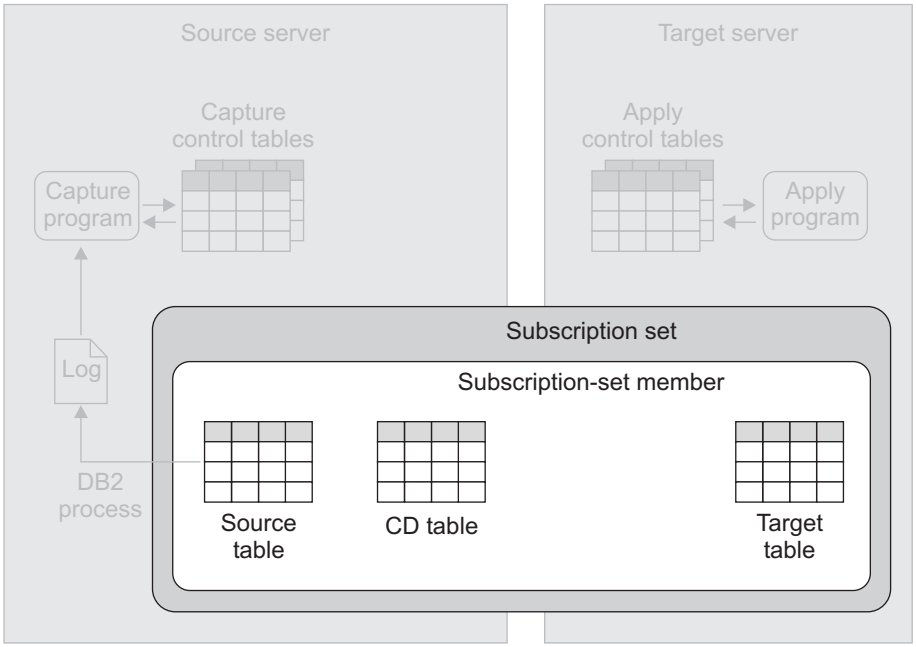


Figure 4. Creating a subscription set. You map sources to targets as part of creating subscription sets. A subscription set groups together one or more source-target pairs, also called subscription-set members. In the figure, a source table is mapped to a target table. The data replicated to the target is first staged in the CD table for the source.

Capture of data from DB2 sources in SQL replication

After you register your DB2 replication sources and create subscription sets to pair your sources with targets, you are ready to begin capturing the changes made to your sources. You use a program called the Capture program to do this.

After you start the Capture program and that program receives signals from the Apply program to indicate that the sources and targets are synchronized, the Capture program reads the DB2 log sequentially for changes to the source tables in which you are interested. If it reads a change to one of your source tables, the Capture program adds the change to the corresponding database transaction that it is retaining in memory. Transactions in memory are potentially subsets of the corresponding transactions in the log; they contain only changes to your source

tables. The Capture program collects changes in memory until it reads either a ROLLBACK or a COMMIT statement for the transaction in which those changes are made. Upon reading a ROLLBACK statement, the Capture program erases from memory the changes associated with the rolled-back transaction. Upon reading a COMMIT, the Capture program stores the changes associated with the committed transaction.

For example, if you registered Table A and Table B as replication sources. For each of these tables, SQL replication creates a CD table as part of the registration process. After you start the Capture program and after that program receives signals from the Apply program to indicate that the target tables are synchronized with the source tables, the Capture program reads the DB2 logs for changes to those source tables. Application 1 makes a series of changes to Table A. Each change is recorded in the DB2 logs. The Capture program collects these changes in memory. Application 1 issues a ROLLBACK statement. When the Capture program reads this statement, it erases from memory the changes that are associated with that transaction.

Application 2 makes a series of changes to Table B. As before, the Capture program collects theses changes in memory. Application 2 then issues a COMMIT statement. When the Capture program reads this statement, it appends a copy of each change to the CD table for Table B.

At least one Capture program must run on each server where replication sources are located. However, you might want to have more than one Capture program on a single server for the following reasons:

You want to increase parallelism
: You can use multiple Capture programs to parallelize traffic. Multiple Capture programs, which could also be helpful in large sysplex, can improve performance and achieve higher throughput. The trade-off is additional CPU overhead associated with multiple log readers. Using multiple Capture programs also requires more DB2 connections.

You need to meet different replication requirements
: You can create multiple Capture programs to direct the flow of source changes toward different uses. For example, if you need to replicate a large source table with very low latency and your current Capture program is already capturing data for a large number of source tables, you could use another Capture program to capture changes made only to that table.

  For example, if you need to replicate one set of source tables with very low latency, and you need to replicate another set of source tables to feed a data warehouse on a daily basis, the performance and tuning requirements might be very different. You could use a Capture program for each of these separate sets of source tables, and tune each Capture program to the needs of the respective applications.

You need to use separate encoding schemes
: On z/OS systems, you can use multiple Capture programs to work with UNICODE or EBCDIC encoding schemes separately.

The following figure shows the Capture program capturing data in a simple configuration in SQL replication.

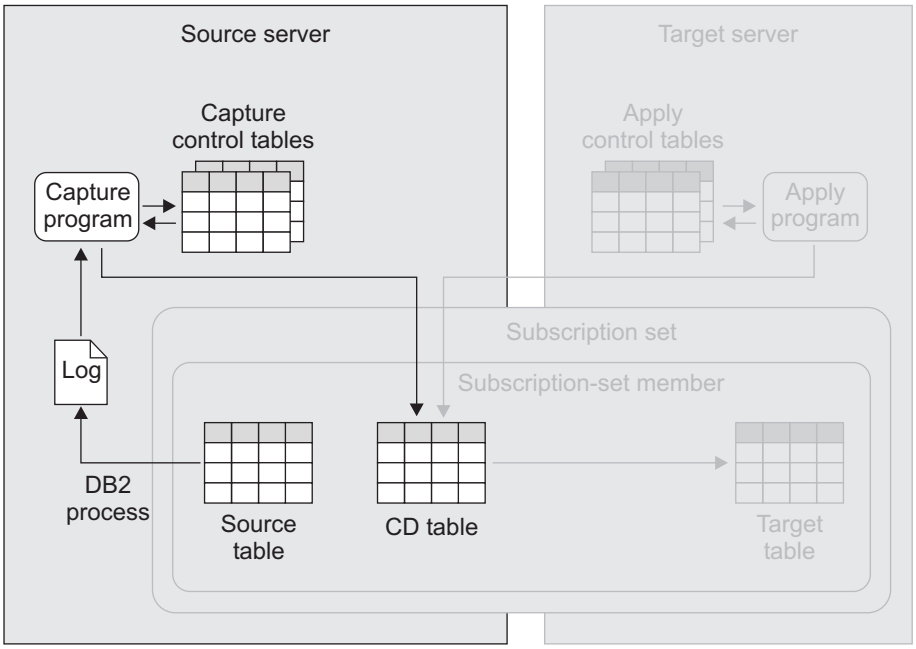


Figure 5. Capturing data from a DB2 source table. The Capture program reads the DB2 recovery log for changes made to your sources and then stages those changes in relational tables. In the figure, the Capture program reads the DB2 log for changes to the source table and stages committed transactional data in the source's change-data (CD) table. The Capture program continuously updates its control tables with information that you can use to monitor its progress.

Application of data to DB2 targets in SQL replication

To start replicating data from sources to targets, you start the Apply program. After you do so, the Apply program begins processing the subscription sets that you assigned to it. The Apply program processes any active subscription sets one at a time, according to the scheduling or event criteria that you specified when you created the subscription sets.

For example, if you chose to replicate subscription set Set_One every 60 minutes, the Apply program will replicate that subscription set as close to every 60 minutes as possible. If the Apply program has other work to finish when the 60–minute interval elapses, Set_One will be replicated as soon as the Apply program finishes its other work.

The following list describes how the Apply program processes each member of a subscription set.

For sources that you registered for change-capture replication

The first time that the Apply program processes the corresponding subscription set, the Apply program can populate the targets with the content of the sources. You can tell the Apply program to call one or more utilities, such as the EXPORT and LOAD utilities, to populate the targets. The utilities that the Apply program chooses depends on the platform on which the Apply program is running.

Then, at the time intervals that you specified when you created your subscription sets or whenever an event occurs, the Apply program reads the CD tables of your sources for rows that were inserted into the CD table

since the Apply program last looked. The rows in the CD tables indicate whether they are records of deletes, updates, or inserts to the corresponding sources.

The Apply program uses the data from the CD table to insert, update, and delete rows in the target. Predicates are used to identify the rows to be updated or deleted.

For sources that you registered for full-refresh replication

At intervals that you specify the Apply program populates the targets with the content of the sources. You can tell the Apply program to call one or more utilities, such as the EXPORT and LOAD utilities, to refresh your target tables. The utilities that the Apply program chooses depends on the platform on which the Apply program is running.

The following figure shows the Apply program applying data in a simple configuration in SQL replication.

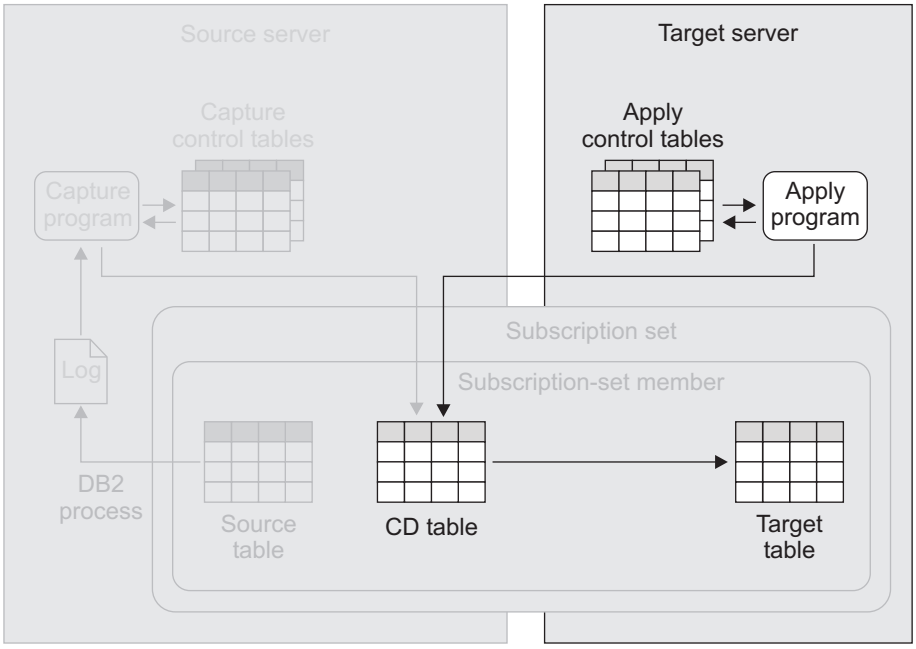


Figure 6. Applying data to DB2 targets. The Apply program reads data from staging tables and makes the appropriate changes to targets. In the figure, the Apply program reads data from the source table's CD table and then makes the appropriate changes to the target table. The Apply program continuously updates its control tables with information that you can use to monitor its progress.

Capture of data from non-DB2 sources in SQL replication

The Capture program is not involved in capturing changes made to tables on non-DB2 relational databases. Instead, when you register such a table, SQL replication creates three triggers (called *Capture triggers*) on the table that run after the event (an INSERT trigger, a DELETE trigger, and an UPDATE trigger) and also creates a consistent change-data (CCD) table, which is similar to a CD table because it stores records of changes made to the source table. SQL replication also creates corresponding nicknames for the CCD tables on the same federated server where the nicknames for the source tables are located. The Apply program accesses the CCD tables through these nicknames.

When one of these triggers is activated by a change to the source table, the trigger records the change in the CCD table. If the application transaction is rolled back, the new records in the staging table are rolled back, too. Only changes that are part of committed transactions are replicated.

SQL replication also creates a fourth trigger on a non-DB2 relational database that is used as a source database. This is the pruning trigger that deletes replicated records from the CCD tables after the Apply program successfully replicates changes to target tables.

The following figure shows how changes to a source table on a non-DB2 relational database are replicated to a DB2 target table.

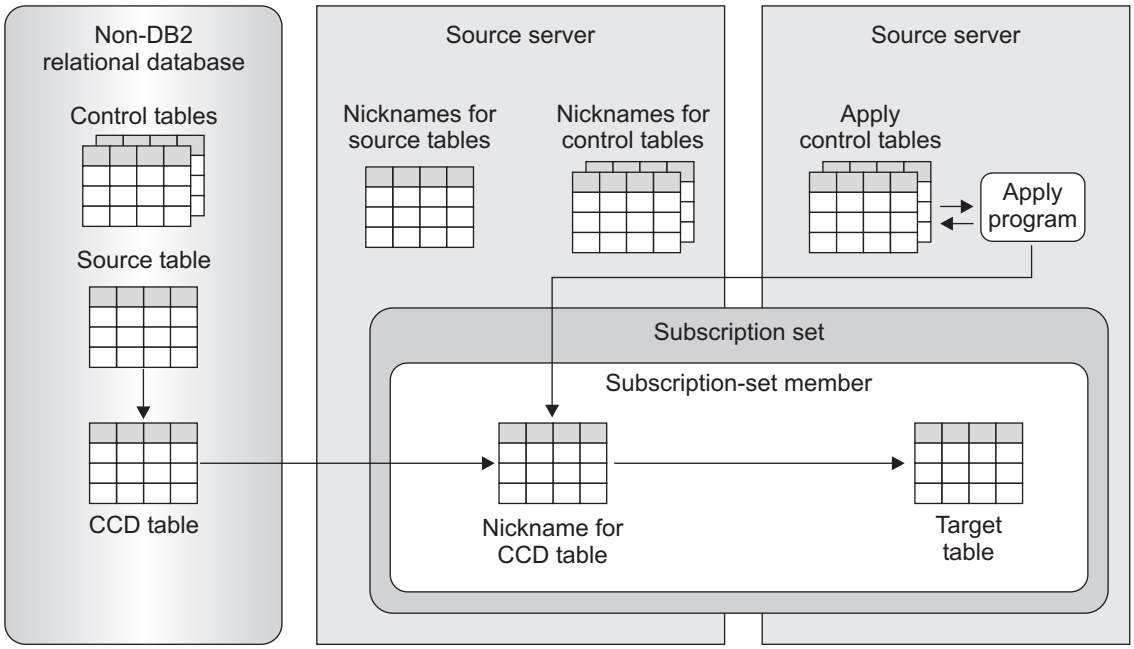


Figure 7. Replicating data from non-DB2 sources to DB2 targets.. Triggers populate a staging table called a consistent-change-data (CCD) table with records of changes made to a source table. The Apply program reads the CCD table through that table's nickname on the federated database. The Apply program then makes the appropriate changes to the target table.

Application of data to non-DB2 targets in SQL replication

As when replicating to DB2 targets, the Apply program can replicate to non-DB2 targets in two ways: through change-capture replication and through full-refresh replication.

Change-capture replication for targets on a non-DB2 relational database
If the source is on a DB2 server, the Apply program reads the change-data (CD) table for the source. The Apply program then applies the changes to the target by accessing the target table's nickname on a federated database.

If the source is on a non-DB2 relational database, the Apply program reads the CCD table for the source by accessing the CCD table's nickname. It then applies the changes to the target table by accessing the target table's nickname. The nickname for the CCD table and the nickname for the target table can be in the same DB2 federated database or in different ones.

Full-refresh replication for targets on a non-DB2 relational database
If the source is on a DB2 server, the Apply program reads the source

directly, deletes the rows of the target table by accessing the target table's nickname, and inserts the source rows into the target table by accessing the target table's nickname.

If the source is on a non-DB2 relational database, the Apply program reads the rows of the source by accessing the source's nickname. It then deletes the rows of the target by accessing the nickname, and inserts the source rows into the target table by accessing the target table's nickname. The nickname for the source table and the nickname for the target table can be in the same DB2 federated database or in different ones.

The following figure shows how changes to a DB2 source table are replicated to a table on a non-DB2 relational database.

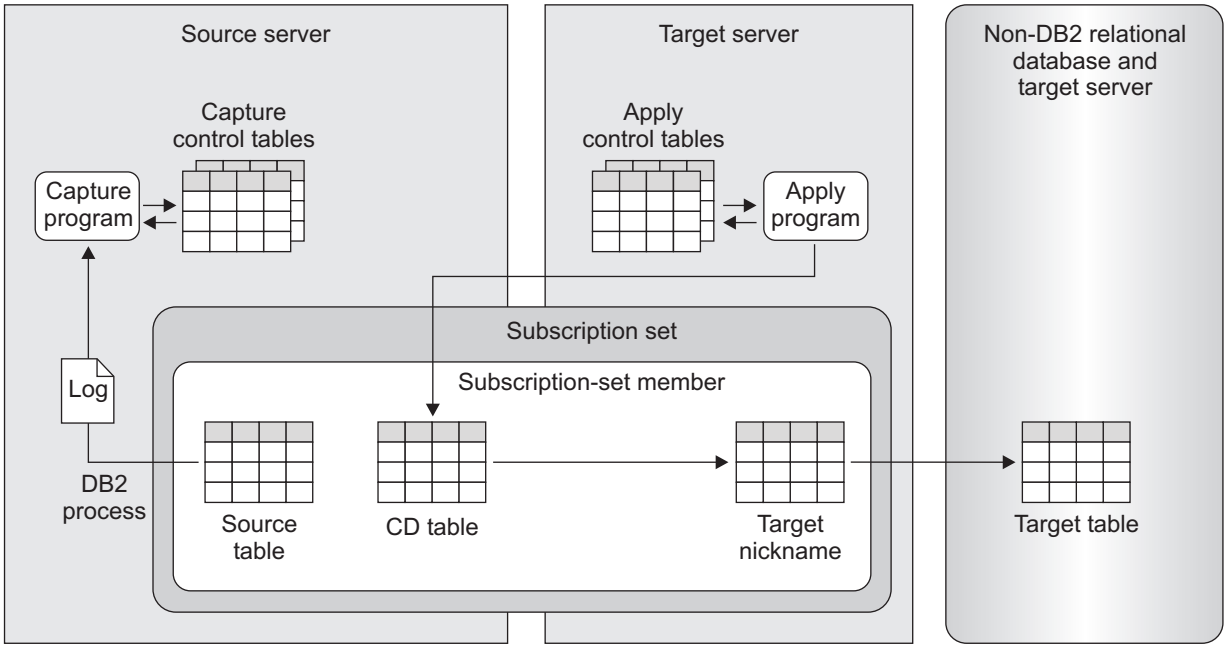


Figure 8. Replicating from DB2 sources to non-DB2 targets. The Capture program populates a CD table with the changes made to the corresponding source table. The Apply program reads the data in the CD table and applies the changes to the target table through the target's nickname on the DB2 Information Integrator federated database.

Chapter 7. Q replication—Overview

This section describes the basic concepts involved in Q replication.

The following figure shows a simple configuration in Q replication. The topics in this section describe different parts of this figure.

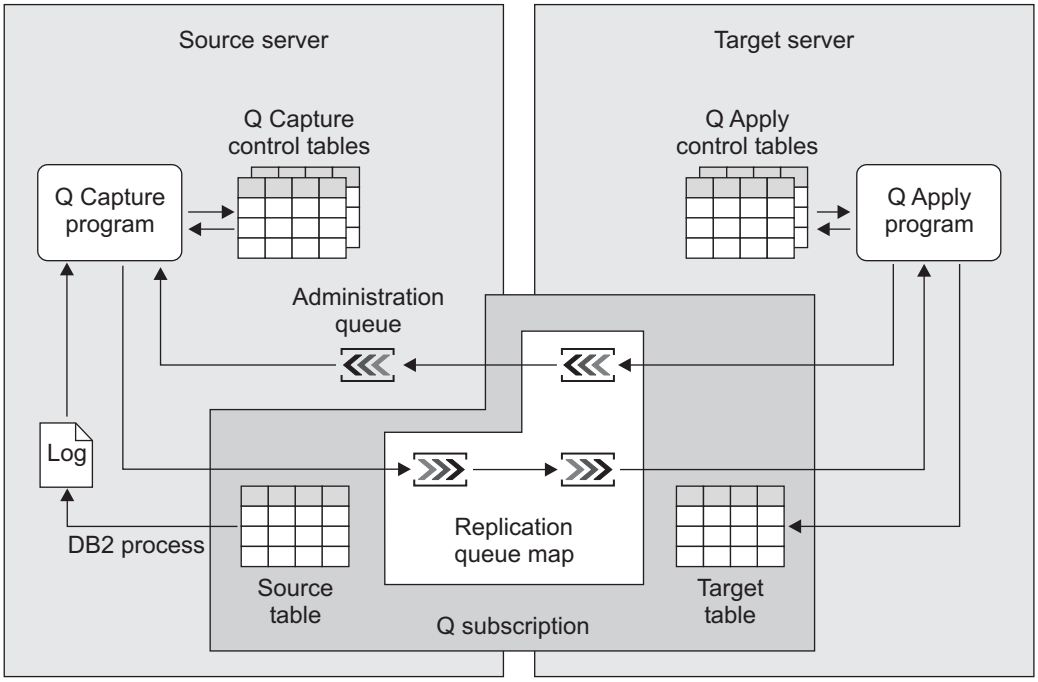


Figure 9. A simple configuration in Q replication

Infrastructure for a Q replication environment

With Q replication you replicate committed transactional data from DB2 sources to DB2, Informix, Microsoft SQL Server, Oracle, and Sybase targets by using two programs: Q Capture and Q Apply.

Q Capture program
The Q Capture program runs on the source system. The Q Capture program reads DB2 recovery logs for changed source data and writes the changes to WebSphere MQ queues.

Q Apply program
The Q Apply program runs on the target system. The Q Apply program retrieves captured changes from queues and writes the changes to targets.

The Q Capture and the Q Apply programs use a set of DB2 tables to track the information that they require to do their tasks and to store information that they generate themselves, such as information that you can use to find out how well they are performing. You create these tables before you tell Q replication what your replication sources and targets are.

The Q Capture program uses a set of control tables called *Q Capture control tables*. These tables contain information about your replication sources, the targets that correspond to them, and which WebSphere MQ queue manager and queues are being used by the Q Capture program. These tables also contain data that you can use to check and monitor the Q Capture program's performance, such as data about the Q Capture program's current position in the DB2 recovery log.

You can run multiple Q Capture programs on the same DB2 server. Each Q Capture program uses its own set of control tables. The schema that is associated with a set of Q Capture control tables identifies the Q Capture program that uses those control tables. This schema is called a *Q Capture schema*.

The Q Apply program uses a set of control tables called *Q Apply control tables*. These tables contain information about targets and where their corresponding sources are located, and information about which WebSphere MQ queue manager and queues are being used by the Q Apply program. Like the Q Capture control tables, these tables also contain data that you can use to check and monitor the Q Apply program's performance.

Like Q Capture programs, you can run multiple Q Apply programs on the same server. Each Q Apply program uses its own set of control tables. The schema associated with a set of Q Apply control tables identifies the Q Apply program that uses those control tables. This schema is called a *Q Apply schema*.

The following figure shows the infrastructure for a simple configuration in Q replication.

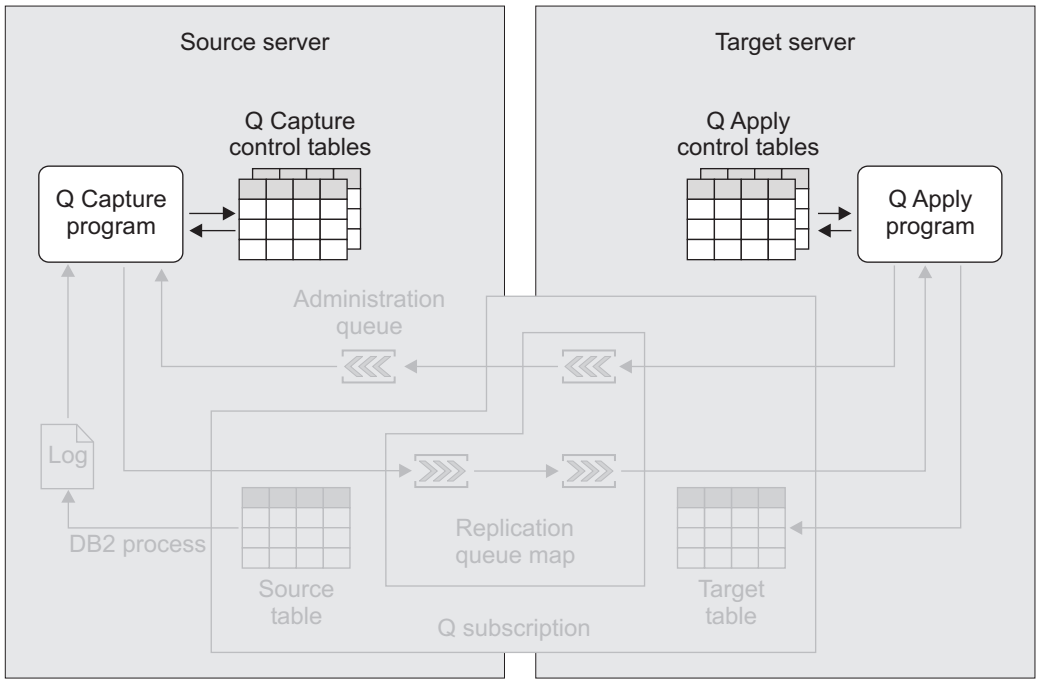


Figure 10. Infrastructure for a simple configuration in Q replication. You create a set of DB2 relational tables on the source server called Q Capture control tables. Information about your sources and targets is written to these tables. The Q Capture program, which runs on the source server, uses this information to determine the data to capture and send to the Q Apply program. Information about your source and target tables goes in the Q Apply control tables, which are on your target server. The Q Apply program, which runs on the target server, uses this information to determine which target tables to write data to.

Sources and targets in Q replication

You pair source tables with targets by defining *Q subscriptions*.

When you create a Q subscription, you specify the following primary attributes:

The source server and source table
The source server can be a DB2 server that contains at least one set of Q Capture control tables, or the source server can be a WebSphere Classic Replication Server for z/OS. You can use a DB2 relational table on a source server as your source table. For Classic replication, you can use tables or views on the nonrelational database.

The target server and target table or stored procedure

You can replicate data to DB2 tables, federated tables, or a stored procedure on a DB2 server. The target server must contain the Q Apply control tables.

If you need to transform replicated data, you can use a stored procedure as a target or you can use SQL column expressions that are run by the Q Apply program. If you want to keep a history of what changed at the source, you can use a consistent-change-data (CCD) table as a target.

The "Comparison of sources and targets in SQL replication and Q replication" on page 33 topic provides further details about the sources and targets for SQL replication and Q replication.

Which source columns to replicate to the target table or stored procedure
You can subset the columns that you replicate from the source.

How to map the source columns to the target columns or parameters in a stored procedure
If your target table exists in the target location, you tell Q replication how the selected columns in your source table correspond to the columns in the target table. If you want Q replication to create the target table for you, the selected source columns are automatically mapped to the columns in the target table.

If you are replicating to a stored procedure, you tell Q replication how the selected columns in your source table correspond to the parameters in your stored procedure.

A predicate for replicating a subset of rows to the target table
You can subset the rows that you replicate from the source by using a predicate. A predicate limits the rows that are returned, like a WHERE clause in a SQL statement. In Classic replication, you must define views on tables to accomplish row filtering.

The method of loading the target table
In most cases, the data from the source table is loaded into the target table so that the target table is identical to the source table before replication begins. Q replication allows for two methods of loading a target table after you start a Q subscription: automatic loads and manual loads.

For automatic loads, the Q Apply program manages the loading of target tables. The Q Apply program calls one utility or a pair of utilities to perform the load. You can tell the Q Apply program to call the LOAD FROM CURSOR option of the LOAD utility, the EXPORT and LOAD utilities, or the EXPORT and IMPORT utilities, depending on the platform

on which the Q Apply program is running. You can also tell the Q Apply program to determine which of these options is most appropriate for the Q subscription.

For manual loads, you handle the loading of target tables, and then signal the replication programs when loading is done.

The WebSphere MQ message queues to use for transporting messages
The Q Capture program puts messages on a queue called a *send queue*. The Q Apply program receives the messages on a queue called a *receive queue*. (These names are used only in Q replication, not in WebSphere MQ.) You choose a *replication queue map* to tell Q replication which send queue and receive queue you want to use for the messages for your Q subscription.

The following figure shows the parts of a Q subscription in a simple configuration in Q replication.

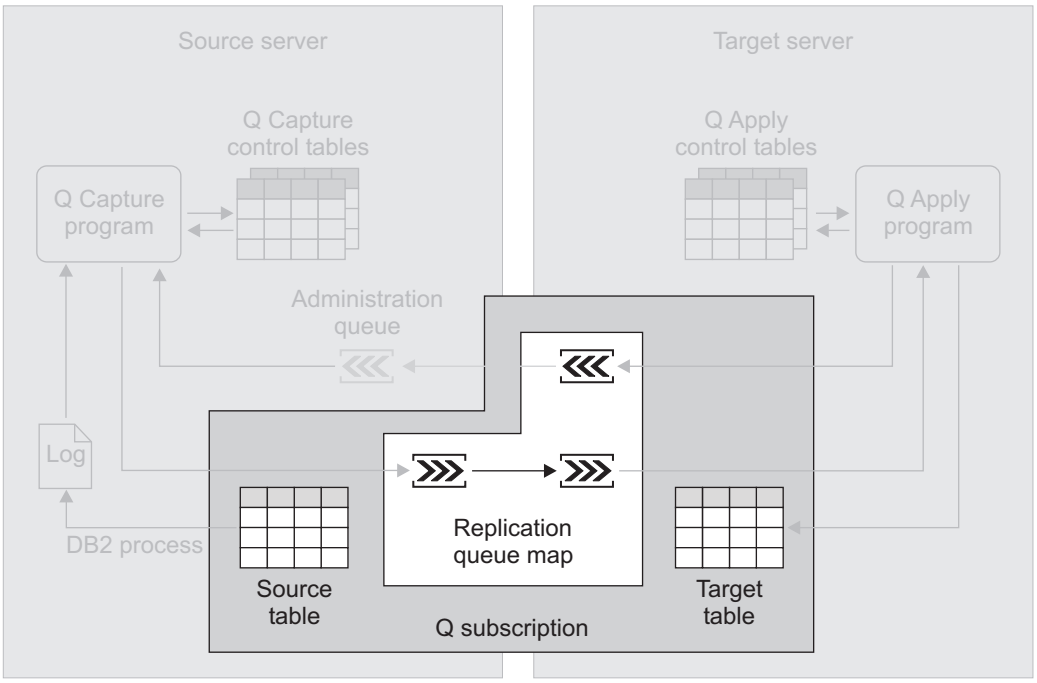


Figure 11. Creating a Q subscription. You map sources to targets by creating Q subscriptions. One Q subscription maps one source table to one target table. Data for the Q subscription is replicated across a send queue and a receive queue, both of which are part of a replication queue map.

Capture of data in Q replication

The Q Capture program reads the DB2 recovery log sequentially for changes to source tables. If the capture program reads a change to one of your source tables, the Q Capture program adds the change to the corresponding database transaction that is retained in memory.

Transactions in memory are therefore potentially subsets of the corresponding transactions in the log because they contain only changes to the source tables in your Q subscriptions. When the Q Capture program reads the COMMIT statement for a transaction, it converts the transaction into a message and puts the messages on a send queue.

For example, you create two Q subscriptions for two different source tables on the same source server: QSUB1 and QSUB2. Both Q subscriptions use the same replication queue map. The Q Capture program reads a COMMIT for a database transaction that involves changes to the source tables in both Q subscriptions. The Q Capture program converts the changes into a message and writes the message to the send queue that is part of the replication queue map.

You can run more than one Q Capture program on the same source server. Although one Q Capture program can capture changes made to many sources and send those changes to many target servers, in some situations you might benefit from having more than one Q Capture program. For example, You can use multiple Q Capture programs to parallelize traffic. Multiple Q Capture programs, which could also be helpful in large sysplexes, can improve performance and achieve higher throughput. The trade-off is additional CPU overhead associated with multiple log readers. Using multiple Q Capture programs also requires more DB2 connections.

The following figure shows the Q Capture program capturing data in a simple configuration in Q replication.

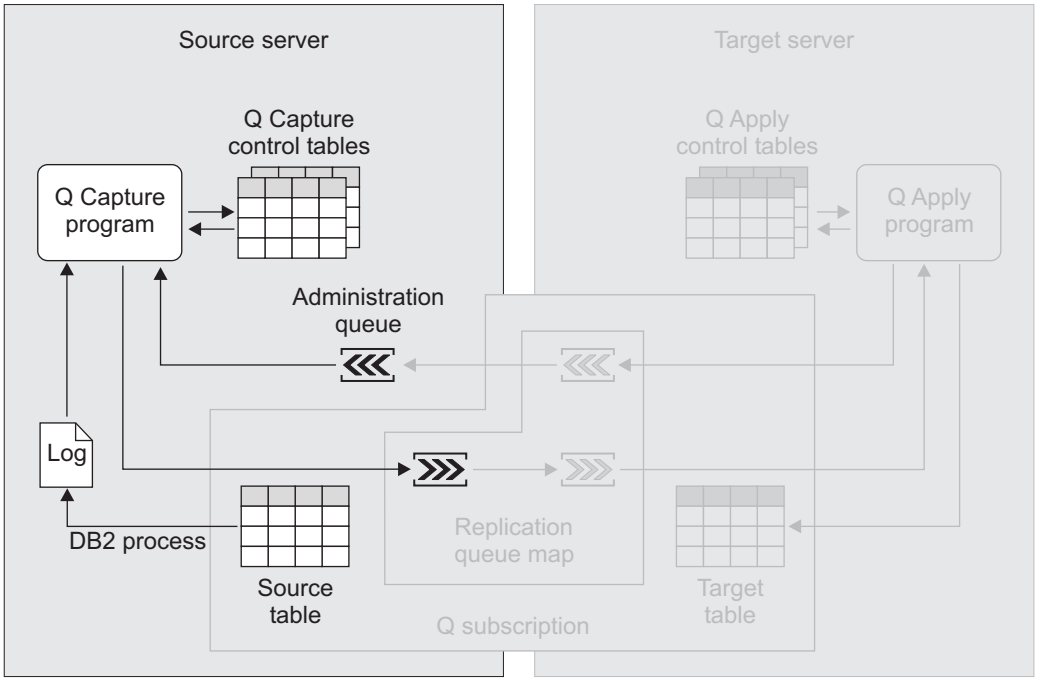


Figure 12. Capturing data in Q replication. The Q Capture program reads the DB2 recovery log for changes made to your sources and converts committed transactional data to messages, which it puts on a send queue. The Q Capture program continuously updates its control tables with information that you can use to monitor its progress.

Application of data to targets in Q replication

The Q Apply program reads the messages that contain committed transactional data for Q subscriptions. These messages arrive at the target server on receive queues. The Q Apply program converts the messages to SQL and applies the transactions to the relevant target tables.

The Q Apply program is multithreaded and can apply several transactions concurrently, whenever those transactions are not dependent on one another. If

transactions are dependent on one another, the Q Apply program applies them in the order in which they were committed at the source server.

The Q Apply program can receive messages for a large number of Q subscriptions on a single receive queue and apply the transactions at very high rates of speed. In most cases, you can use a single replication queue map between one source server and one target server and not experience noticeable latency.

A single pair of Q Capture and Q Apply programs can be configured with multiple replication queue maps. If multiple applications are running at the source server that update independent sets of tables, then you should consider defining multiple replication queue maps to allow for parallel delivery and application of data for each independent set of tables. A Q Apply program will create a multithreaded process for each receive queue.

In most cases, one Q Apply program per server is sufficient, even for replication configurations with a high volume of transactions being replicated to a large number of tables. A single Q Apply program can be configured to process one or more receive queues. You can also run multiple Q Apply programs on a server if, for example, you require the data on one or more receive queues to be processed differently than the data on other receive queues.

The following figure shows the Q Apply program applying data in a simple configuration in Q replication.

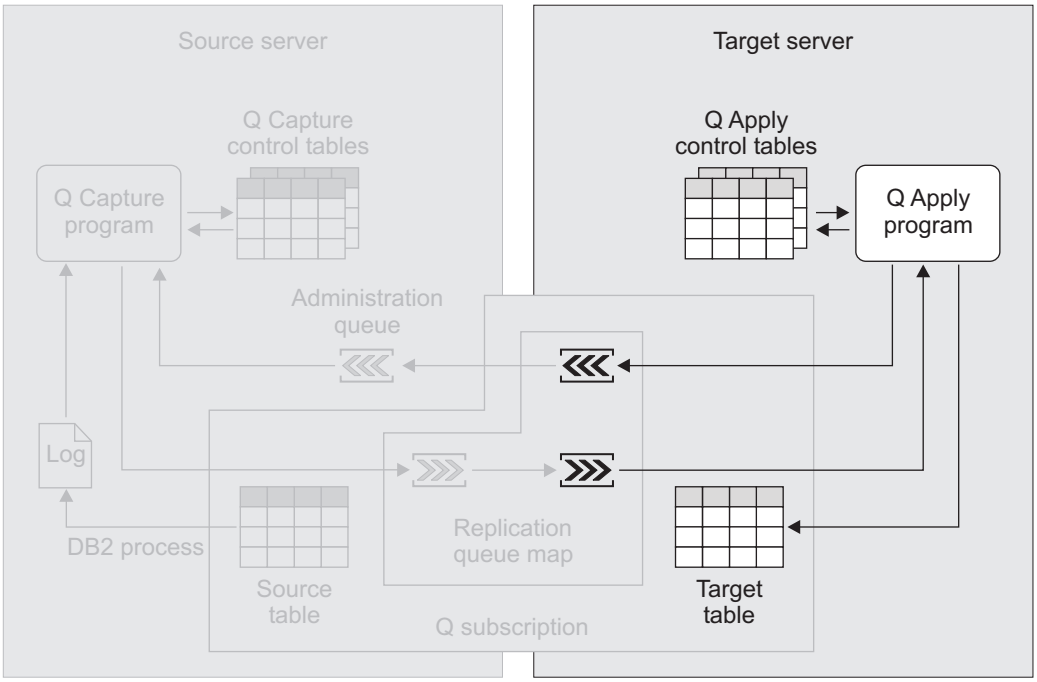


Figure 13. Applying data in Q replication. The Q Apply program reads messages from a receive queue, converts them to SQL, and applies data to target tables. The Q Apply program continuously updates its control tables with information that you can use to monitor its progress.

Types of replication in Q replication

With Q replication, you can configure three different types of replication.

Unidirectional replication

In unidirectional replication, changes that occur to a source are replicated to a target. Target tables are typically used exclusively by read-only applications. You can replicate all of the rows and columns from a source, or you can choose to replicate column and row subsets. No replication takes place from the target back to the source.

You can have one source replicating to one target, one source replicating to multiple targets, multiple sources replicating to one target, or multiple sources replicating to multiple targets. For each source and its corresponding target, there is one Q subscription. Q subscriptions should use the same replication queue map if their targets are logically related; by grouping Q subscriptions this way, you ensure that the data applied to the targets is consistent with the original transactions on the source server.

At each source server, you create at least one set of Q Capture control tables. At each target server, you create at least one set of Q Apply control tables.

For Q subscriptions in unidirectional replication, you can have the Q Apply program call a stored procedure and pass the source data as input parameters to the stored procedure. Instead of the source columns mapping to target columns, the source columns map to parameters in the stored procedure. By mapping source columns directly to parameters in a stored procedure, you avoid the need to parse the incoming data and have a clean, simple programming model. The Q Apply program calls a stored procedure for each row operation instead of inserting the rows into a table. The stored procedure is responsible for getting the source data to its final destination.

Unidirectional replication is the only configuration that is available for Classic replication.

Bidirectional replication

In bidirectional replication, replication occurs between tables on two servers. Tables on one server that are involved in replication are identical in structure to the corresponding tables on the other server. Two corresponding tables have the same number of columns, the same column names, and compatible data types, though they can have different schemas and names. You cannot replicate subsets of rows. Changes made to a table on either server are replicated to the corresponding table on the other server.

Applications on one server can make changes to a table at the same time that applications on the other server make changes to the corresponding table. If conflicts occur in the data that is replicated between corresponding tables, you can choose which of the two tables takes precedence. Conflicts are found by comparing old values to current values. This method of detection might not find all conflicts in your data, but requires less overhead than other methods.

For every two corresponding tables, there are two Q subscriptions. For example, if you are replicating between Table_One on server Red and Table_Two on server Blue, there are these two Q subscriptions:

- One Q subscription replicating from Table_One to Table_Two
- One Q subscription replicating from Table_Two to Table_One

When you create the Q subscriptions, if you want an initial load performed automatically, you can choose which table contains the data that you want

to start with. In this example, if you specify that Table_One contains the data that you want to start with, Table_Two will be loaded with the data from Table_One.

A Q Capture program and a Q Apply program run at both servers. The Q Capture program and Q Apply program on each server have matching schemas. For example:

- If on one server the schema of the Q Capture program is Green, the schema of the Q Apply program on that server is also Green.
- If on the other server the schema of the Q Capture program is Yellow, the schema of the Q Apply program on that server is also Yellow.

Peer-to-peer replication

In peer-to-peer replication, replication occurs between tables on two or more servers. Tables on one server that are involved in replication are identical in structure to the corresponding tables on the other servers. All corresponding tables have the same number of columns, the same column names, and compatible data types, though they can have different schemas and names. You cannot replicate subsets of rows. Changes made to a table on any server are replicated to the corresponding tables on the other servers.

Applications on one server can make changes to a table at the same time that applications on the other servers make changes to the corresponding tables. Conflicts are detected and resolved with the help of version columns and triggers that are added to the tables when you create the Q subscriptions. Convergence is possible with peer-to-peer replication, which means that if changes to replicated tables are stopped and all changes are propagated, corresponding tables will be identical.

For each pair of corresponding tables, there are two Q subscriptions. For example, if you are replicating between Table_One on server Red, Table_Two on server Blue, and Table_Three on server Green, there are three pairs of corresponding tables:

- Table_One and Table_Two
- Table_Two and Table_Three
- Table_One and Table_Three

For each of these pairs, there are two Q subscriptions. For example, between Table_One and Table_Two, there are these two Q subscriptions:

- One Q subscription replicating from Table_One to Table_Two
- One Q subscription replicating from Table_Two to Table_One

A Q Capture program and a Q Apply program run at each server. The Q Capture program and Q Apply program on each server have matching schemas. For example, if you are replicating between two servers:

- If on one server the schema of the Q Capture program is Blue, the schema of the Q Apply program on the same server is also Blue.
- If on the other server the schema of the Q Capture program is Red, the schema of the Q Apply program on the same server is also Red.

Chapter 8. Comparison of SQL replication and Q replication—Overview

This section describes the similarities and differences between SQL replication and Q replication.

Comparison of the infrastructure of SQL replication and Q replication

The location of the control tables and the number of Capture and Apply programs per server differ between SQL replication and Q replication.

Table 1 compares the infrastructure of SQL replication and Q replication.

Table 1. Comparison of the infrastructure in SQL replication and Q replication

Points of comparison	SQL replication	Q replication
Location of the control tables for the capturing program	You must create the control tables for a Capture program on the DB2 server where the Capture program runs. In most cases, this server is the same DB2 server where the sources associated with the program are located.	You must create the control tables for a Q Capture program on the DB2 server where the Q Capture program runs. This server is the same DB2 server where the source tables associated with that program are located.
Number of capturing programs possible per DB2 server	Multiple, each with its own set of control tables and identified by the schema of those control tables.	Multiple, each with its own set of control tables and identified by the schema of those control tables.
Location of the control tables for applying program	You can create the control tables for the Apply program on any server that can connect to the source server and the target server. In general, the control tables should be located on the server where the Apply program runs.	You must create the control tables for a Q Apply program on the server (DB2 or non-DB2) where the target tables associated with that program are located.
Number of applying programs possible per DB2 server	Multiple, each sharing one set of control tables and identified by a string called an Apply qualifier.	Multiple, each with its own set of control tables and identified by the schema of those control tables.

Comparison of sources and targets in SQL replication and Q replication

Sources and targets in SQL replication differ from sources and targets in Q replication. In addition, the way that you interact with these objects differs for the two types of replication.

The following table compares the sources and targets in SQL and Q replication

Table 2. Comparison of sources and targets in SQL replication and Q replication

Points of comparison	SQL replication	Q replication
Source and target platforms	Sources and targets can be on the following relational database management systems: • DB2 for Linux, UNIX, and Windows • DB2 UDB for z/OS • DB2 UDB for iSeries • Informix • Microsoft SQL Server • Oracle • Sybase • Teradata (for targets only)	Sources and targets can be on the following relational database management systems: • DB2 for Linux, UNIX, and Windows • DB2 UDB for z/OS • Informix (target only) • Microsoft SQL Server (target only) • Oracle (target only) • Sybase (target only) Classic replication sources can be on the following nonrelational data sources: • Adabas databases • CA-IDMS databases • IMS™ databases • CICS® VSAM files • VSAM files
Database objects that can be targets	• DB2 tables and views. • Tables on non-DB2 relational databases.	• DB2 tables and stored procedures. • Tables on non-DB2 relational databases.
Pairing of sources and targets	You register a source. This registration information is stored in the Capture control tables. Then, you create one or more subscription-set members to map this registered source to targets. Information about these subscription-set members is stored in Apply control tables.	You create a Q subscription to map a source to a single target. No registration of the source is required. A source can be replicated to multiple targets by creating one Q subscription for each target. Information about the Q subscription is stored in Q Capture control tables and the Q Apply control tables.
Grouping of sources-target pairs	You group source–target pairs into subscription sets. Each source–target pair is referred to as a subscription-set member.	You can group Q subscriptions by replication queue map. No subscription set object exists in Q replication.
Subsetting of source columns and rows allowed?	Yes.	Yes.
Transforming data	You can transform data by using stored procedures or SQL statements that are run by the Apply program. On the source server, the stored procedures or SQL statements can transform staged data. On the target server, the stored procedures or SQL statements can transform data in targets. The stored procedure interface limits transformations. The Apply program passes no parameters to a stored procedure and runs it once for an entire subscription set rather than for each subscription–set member. You can also use triggers to transform data that the Capture program writes to staging tables.	You can transform data by using stored procedures or SQL statements that are run by the Q Apply program. The Q Apply program calls the stored procedures and passes the changed data to them as parameters. This method of using stored procedures allows considerable versatility for data transformation. Q replication supports all DB2 SQL expressions for computed columns.

Comparison of data capturing and applying in SQL replication and Q replication

The location of the Capture and Apply programs and the method of transporting the data differ between SQL replication and Q replication.

Table 3 compares data capturing and applying in SQL replication and Q replication.

Table 3. Comparison of data capturing and applying in SQL replication and Q replication

Points of comparison	SQL replication	Q replication
Location of the capturing program	The Capture program runs on the DB2 server where its control tables are located. In most cases, the source tables are also on the same server.	The Q Capture program runs on the DB2 server where its control tables are located. The source tables are also on the same server. In Classic replication, the Classic capture components consist of agents and services that run on the data server.
How changed data is captured	The Capture program reads the DB2 recovery log and stores committed transactional data in staging tables.	The Q Capture program reads the DB2 recovery log and converts committed transactional data to messages. In Classic replication, the Classic capture components read the log and pass the changed data through multiple services that convert the data to a relational format.
How changed data is transported	The Apply program fetches data from staging tables and applies the data to targets with DB2 SQL.	The Q Capture program or Classic capture components put data as messages in queues. WebSphere MQ moves the messages to the target system. The Q Apply program gets these messages from queues and applies data to the targets using DB2 SQL.
Location of the applying program	Apply programs can run on any DB2 server in your network, provided that they can connect to the servers that contain the source, target, and Apply control tables.	Q Apply programs run on target servers.
How data is applied to DB2 targets	The Apply program can process a subscription set in table mode or transaction mode. In table mode, the Apply program processes the fetched changes for each table separately, and then issues a single commit after all data is applied. In transaction mode, the Apply program applies the fetched changes to all of the target tables, in the order that the changes occurred in their corresponding transactions on the source server. The Apply program commits these transactions at transaction boundaries. You specify how many transactions to apply before each commit.	The Q Apply program can apply transactions concurrently whenever they do not involve dependent changes to the target tables. The Q Apply program uses various methods of conflict handling and resolution to make bidirectional and multidirectional replication possible.

Replication solutions for common scenarios

Choose the replication solution that works best for your scenario.

Table 4 describes the most common scenarios that are possible with SQL replication and Q replication and recommends which replication technology to use for each scenario. The recommendations are general and might not apply for your particular needs.

Table 4. Recommended replication solutions for common replication scenarios

Configuration	Description of configuration	Recommended replication solution
Data consolidation	You can replicate data from many sources to a central repository. Data consolidation configurations are most often used to build data warehouses or consolidate data from multiple points of sale.	You can use either SQL replication or Q replication successfully in this scenario.
Data distribution	You can replicate data from a source to one or more targets that reside anywhere in a distributed network, for example, distributing a price list to multiple points of sale. Applications use the local target tables so that they do not overload the network or central server. You can choose to have a subset of columns and rows replicated so that each site sees only the data that it needs to see.	You can use SQL replication, Q replication, or a combination of both. SQL replication has better data distribution capabilities because you can stage the data once and replicate to many targets. SQL replication provides better two-tier distribution to many targets because the captured data is written once to the changed data tables. Q replication offers lower latency and throughput. A three-tier distribution that uses both Q replication and SQL replication takes advantage of the strengths of each. Q replication can replicate data from the source server faster and with less overhead to a CCD table. One or more SQL Apply programs are used to then distribute the data from the CCD table to many targets.
Update-anywhere replication	You can replicate data between a master data source and one or more replicas of the source. Changes made to the master source are propagated to the replicas, and changes made to the replicas are also propagated to the master source. Whenever a conflict occurs between the data sent from the master source and data sent from a replica, the data from the master source takes precedence.	In a typical hub-and-spoke configuration with more than one replica, SQL replication is required. Such a configuration replicates from one master data source to multiple replicas. For a scenario with a single master and a single replica, Q replication is recommended.
Hot standby	You can replicate data from a production server to a backup server. If the production server goes down, applications can instantly switch to using data on the backup server. When the production server is back online, committed changes made at the backup server can be replicated to the production server and applications can then use the data on the production server again.	Q replication is recommended. Either bidirectional or peer-to-peer replication can be used in this configuration, depending on the needs of your applications.

Table 4. Recommended replication solutions for common replication scenarios (continued)

Configuration	Description of configuration	Recommended replication solution
Peer-to-peer replication	You can replicate data between table copies on two or more servers. Committed changes made to one table copy are replicated to all other corresponding table copies. This type of replication can support geographically-distributed applications. For example, an online business might have three servers on three continents. Each server contains the same product and ordering information. Customers typically access the product data and provide ordering information at the server on their home continents. However, if the server on one continent goes down, those customers can be rerouted to a server on another continent. Later, their ordering information can be replicated back to the server for their continent so that local financial and shipping offices have access to the information.	Q replication is recommended because of its strengths in conflict detection and resolution. Peer-to-peer replication is easy to set up in Q replication through the Replication Center.
Auditing or change history	You can maintain a history of all table changes for auditing or for feeding an Extract-Transform-Load (ETL) tool. You can maintain a history of changes in tables called consistent-change-data (CCD) tables to audit or track a history of source table changes. The target CCD tables contain columns that provide details about the changes that occurred. These columns include the log sequence number, the type of operation (Insert, Update, Delete), the commit time, transaction ID, user ID, and the values that were modified.	You can use either SQL replication or Q replication successfully in this scenario. Depending on your performance requirements and other considerations, use Q replication for high throughput and lower latency.
Query offloading	When an online transaction processing (OLTP) server is overloaded, you can improve the performance of the OLTP application on that server by offloading the query workload to another server. For example, a cable television company might have a large number of technicians handling customer tickets during each working day. The tables used to keep track of these tickets receive a large number of updates, inserts, and deletes per day. These tables are on a production system that handles other aspects of the business at the same time. The company could replicate relevant tables to another system and run a reporting tool on that system.	Q replication is recommended. You can also use SQL replication.

Table 4. Recommended replication solutions for common replication scenarios (continued)

Configuration	Description of configuration	Recommended replication solution
Replicate data from nonrelational database	You can replicate data from VSAM, IMS, Computer Associates CA-IDMS, and Software AG Adabas data sources to DB2 or federated DB2 targets such as Informix, Microsoft SQL Server, Oracle, and Sybase.	Q replication is recommended. Q replication offers lower latency, higher throughput and less CPU usage at the source. This configuration requires WebSphere Classic Replication Server for z/OS at the nonrelational data source and WebSphere Replication Server at the target.

Chapter 9. Event publishing—Overview

This section describes the concepts involved in event publishing.

The following figure shows a simple configuration in event publishing. The topics listed above each highlight and describe different sections of this figure.

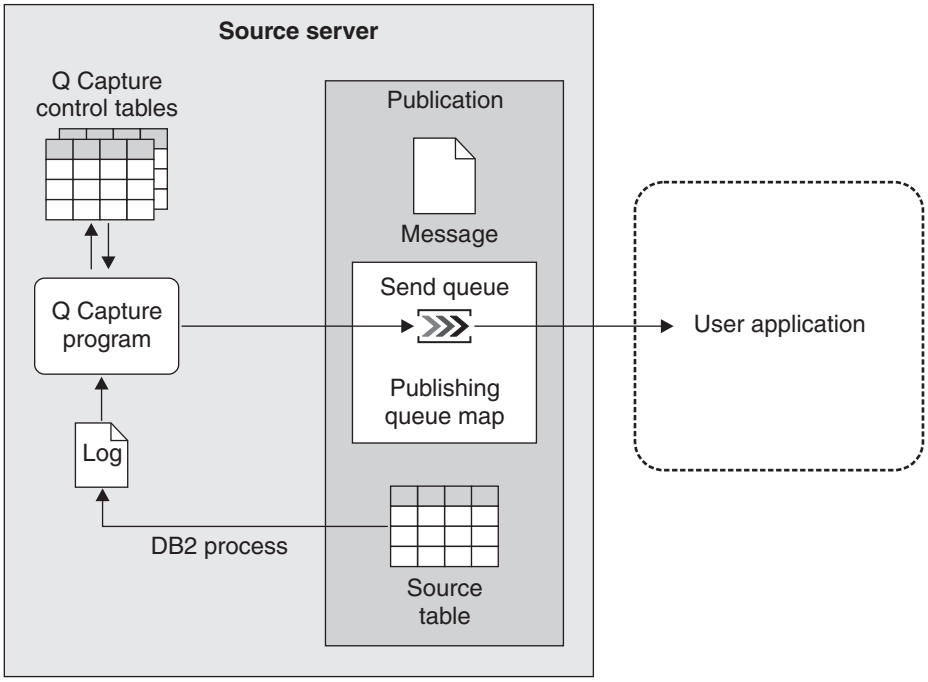


Figure 14. A simple configuration in event publishing

Infrastructure for an event publishing environment

Event publishing allows you to publish committed transactional or row-level data from DB2 tables as messages in WebSphere MQ message queues. The messages can be read and interpreted directly by user applications, or they can first be interpreted by a message broker such as WebSphere Business Integration Message Broker or a DB2 MQ listener daemon.

Event publishing captures data by using a program called the *Q Capture program*. This is the same Q Capture program that is used in Q replication. In fact, it is possible for one Q Capture program to be involved in Q replication and event publishing at the same time. The Q Capture program uses a set of control tables, called *Q Capture control tables*, to store the information that the Q Capture program requires to perform its tasks (such as information about what its sources are and what to publish from its sources) and to store information that it generates itself (such as information about how well it is performing).

You can run multiple Q Capture programs on the same DB2 server. Each Q Capture program uses its own set of control tables. The schema associated with a set of Q Capture control tables identifies the Q Capture program that uses those control tables. This schema is called a *Q Capture schema*.

The following figure shows the infrastructure in a simple configuration in event publishing.

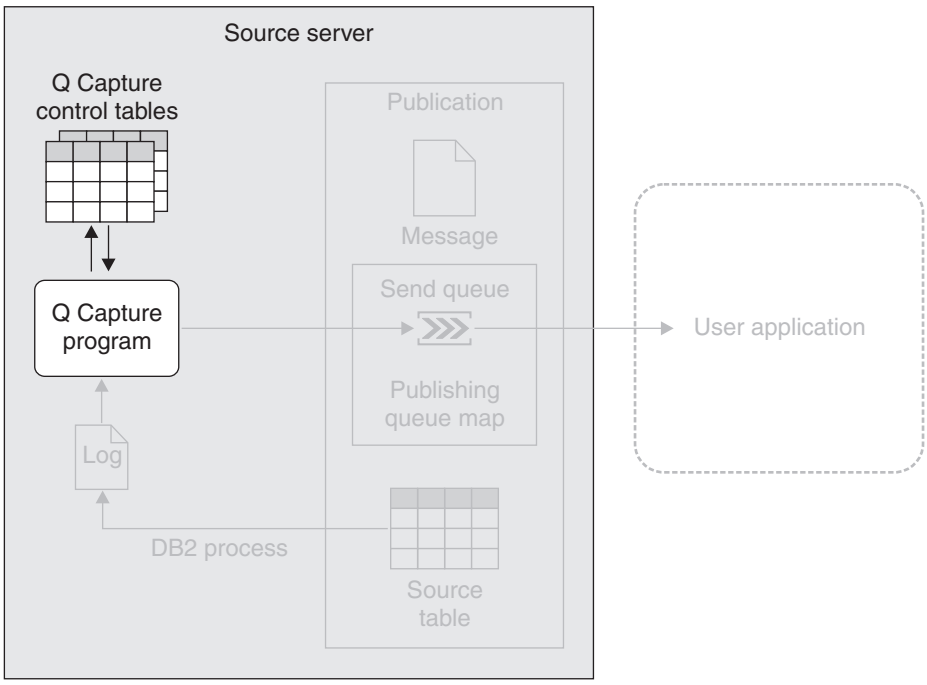


Figure 15. Infrastructure for a simple configuration in event publishing. You create a set of DB2 relational tables on the source server called Q Capture control tables. Information about your sources is written to these tables. The Q Capture program, which runs on the source server, uses this information to know what data to capture and publish.

Sources in event publishing

With event publishing, you create objects called *publications* to define how changes from a single source table are published to a WebSphere MQ message queue. You can then have an application retrieve and use those messages.

When you create a publication, you specify the following attributes:

The source server and source table to publish data from
A source servers is a DB2 server that contains at least one set of Q Capture control tables. You can use a DB2 relational table located on a source server as your source table.

The columns that contain data that you want to publish
You can subset the columns that you publish from the source.

A predicate for publishing a subset of rows
You can subset the rows that you publish from the source.

Message format
You can choose to publish the messages in an XML format or in a delimited format such as comma separated values (CSV). Your choice of format likely depends on the application that consumes the messages. For example, you could publish data to a Web application by using an XML format or you could publish data to WebSphere DataStage™ by using a comma separated values format.

Which publishing queue map to use
The Q Capture program writes data to a WebSphere MQ object that is referred to in event publishing as a *send queue*. The send queue passes the

data to another queue that a user application can read the data from. Event publishing conveniently lets you group a send queue and other options for sending data messages into an object called a *publishing queue map*. For every publication, you select a publishing queue map to use. Multiple publications can use the same publishing queue map.

The publishing queue map that you select determines whether each message that the Q Capture program sends contains committed data for all source rows changed in a transaction or for only a single source row changed in a transaction.

For example, you might want each message to contain committed data for all source rows involved in a transaction if your application handles purchase orders. In this case, you would want a collection of data published together, such as product type, price, shipping address, and billing information, so that with one message you could start processes in systems for billing, shipping, and inventory tracking.

You might want each message to contain committed data for one source row involved in a transaction if you are providing streaming information to a Web application that provides stock quotes. In this case, each change to a stock price is unassociated with any other data.

Whether values in unchanged non-key columns should be published together with updates to other non-key columns

By default, if there are updates to one or more of the columns that you selected, the updates are published but the values in the unchanged columns are not. For example, from table T1 you want to publish columns A1 (the primary key), A2, and A3. If a transaction updates a value in column A2 and then commits, the Q Capture program will publish a message that contains only the new value in column A2, as well as the value in the key column A1.

You can choose to have the messages contain updated and unchanged values in non-key columns. If column A2 is updated, the Q Capture program will publish a message that contains the new value in column A2 and the unchanged value in column A3, in addition to the value in the key column A1.

You might want to choose this option if it is easier to write your application to always expect a value for each column. You also might want to choose this option if you are writing an application to audit changes to your table and that application needs a complete snapshot of each row.

Whether to include new and old values for the data in updated non-key columns, or only the new values

When there are updates to one or more of the columns that you selected, the message containing those updates will provide only the new values by default. For example, from table T2 you want to publish columns B1 (the primary key) and B2. If a transaction updates column B2 from 5 to 6, the message that contains that update will provide only the value 6.

You can choose to have the messages provide the old values together with the new ones. In this case, the message that contains the update to column B2 would provide both the value 5 and the value 6.

Choose to send old values if the application that receives the published changes requires both old and new values. For example, your application might be aggregating information and finding the difference between the

old and new values. Also, if you are publishing significant price changes to a Web application and that application requires the old prices, sending the old prices along with the new prices saves the application from having to look up the old prices elsewhere.

The following figure shows a publication in a simple configuration in event publishing.

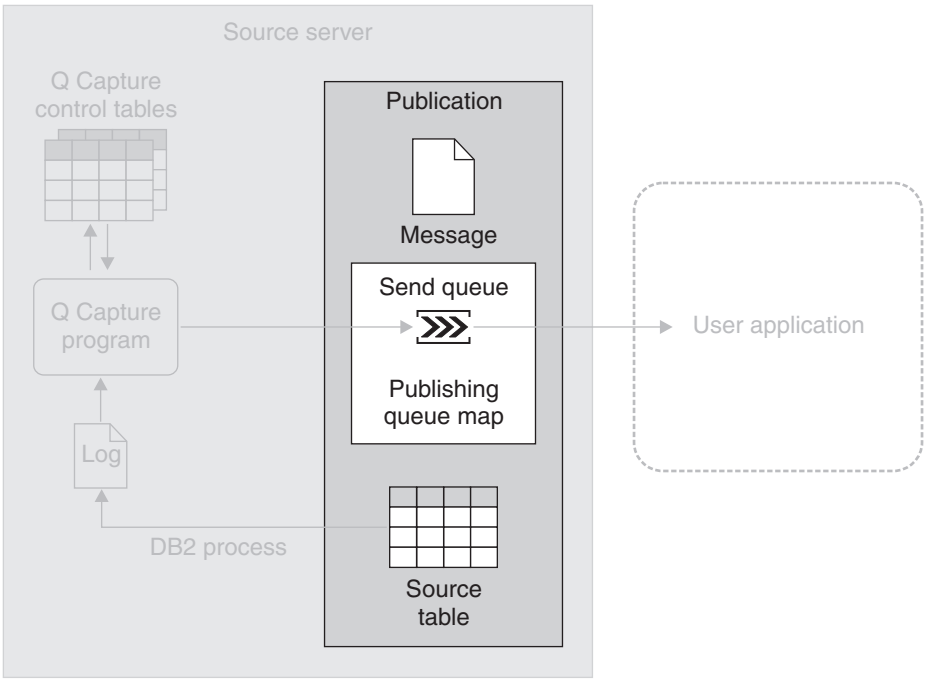


Figure 16. A publication. A publication tells the Q Capture program which changes to capture for a source table and which send queue to use for transporting messages.

Capture of data in event publishing

The Q Capture program reads the DB2 log sequentially for changes to the source tables. If it reads a change to one of your source tables, the Q Capture program adds the change to the corresponding database transaction that it is retaining in memory.

Transactions in memory are therefore potentially subsets of the corresponding transactions in the log; they contain only changes to your source tables. When the Q Capture program reads the COMMIT statement for a transaction, it converts the transaction into one or more messages, depending on whether you want to send row-level messages or transaction-level messages and depending on how many send queues are being used by the source tables that are involved in the transaction. The Q Capture program then puts the message on the corresponding send queues.

For example, you create three publications: PUB1, PUB2, and PUB3. PUB1 and PUB2 use the same publishing queue map (which uses SENDQ1), and you want to use these publications to publish messages that contain committed row-level changes. For PUB3, you choose a different publishing queue map (which uses SENDQ2), and you want to use this publication to publish messages that contain

committed transaction-level changes. A publishing queue map can send either row-level messages or transaction-level messages, but it cannot send both types of messages.

The Q Capture program reads a COMMIT for a database transaction that involves changes to the source tables in all three publications. The Q Capture program converts the parts of the transaction that involve the source tables in PUB1 and PUB2 to row–level messages that it writes to SENDQ1. The Q Capture program also converts the parts of the transaction that involve the source table in PUB3 to a transaction-level message that it writes to SENDQ2.

You can run more than one Q Capture program on the same source server. Although one Q Capture program can capture changes made to many sources and send those changes to many user applications, in some situations you might benefit from running more than one Q Capture program. For example, multiple Q Capture programs, which could also be helpful in large sysplexes, can improve performance and achieve higher throughput. The trade-off is additional CPU overhead associated with multiple log readers. Using multiple Q Capture programs also requires more DB2 connections.

The following figure shows the Q Capture program capturing data in a simple configuration in event publishing.

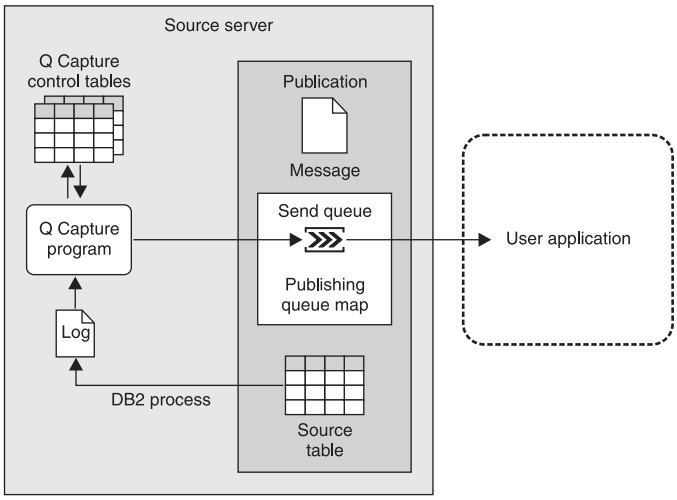


Figure 17. Capturing data in event publishing. The Q Capture program reads the DB2 recovery log for changes made to sources and converts committed transactional data to messages, which the Q Capture program puts on a send queue. The send queue transports data to any number of possible applications that can read the messages. The Q Capture program continuously updates its control tables with information that you can use to monitor its progress.

Chapter 10. Comparison of SQL replication, Q replication, and event publishing

SQL replication, Q replication, and event publishing differ in their uses. You can compare these differences to determine which solution best works for your environment.

The following table compares the main features of SQL replication, Q replication, and event publishing.

Table 5. Comparison of the main features in SQL replication, Q replication, and event publishing

Points of comparison	SQL replication	Q replication	Event publishing
Uses	Multiple, including capacity relief, feeding data warehouses and data marts, auditing change history.	Multiple, including failover, capacity relief, supporting geographically distributed applications, data availability for planned or rolling upgrades or outages.	Multiple, including feeding central information brokers and Web applications, and triggering actions based on updates, inserts, or deletes to source tables.
How data is replicated or published	Committed transactional data is captured and stored in staging tables by a capturing program. An applying program reads the information from the staging tables and applies the transactions to target tables.	Committed transactional data is captured and put on WebSphere MQ message queues by a capturing program. An applying program reads the information from message queues and applies the transactions to target tables.	Committed transactional data is captured and put on WebSphere MQ message queues by a capturing program.
Sources	Tables (DB2 or other supported relational database management systems), views	DB2 tables	DB2 tables
Targets	Tables (DB2 or other supported relational database management systems), views	Tables (DB2 or other supported relational database management systems), stored procedures	User application
Filtering possible?	Column and row filtering.	Column and row filtering.	Column and row filtering.
Where the filtering is done?	Source, target, or both.	Source	Source
Data transformation possible?	Data cleansing, data aggregation, and calculated columns in target tables. Manipulate data once and replicate to many targets. Manipulate data and replicate to selected targets.	Using user-developed stored procedures.	Any required data transformation is performed by the user application.

Table 5. Comparison of the main features in SQL replication, Q replication, and event publishing (continued)

Points of comparison	SQL replication	Q replication	Event publishing
Utilities supported for automatic loads of target tables	EXPORT/IMPORT, EXPORT/LOAD, LOAD FROM CURSOR option of the LOAD utility, depending on the target platform.	• DB2 targets: EXPORT/IMPORT, EXPORT/LOAD, LOAD FROM CURSOR option of the LOAD utility, depending on the target platform. • Non-DB2 targets: EXPORT and IMPORT utilities	No automatic loads because there are no target tables.
Supported operating systems	Linux, UNIX, Windows, z/OS, iSeries	Linux, UNIX, Windows, z/OS	Linux, UNIX, Windows, z/OS
Supported relational database management systems	DB2, Informix, Microsoft SQL Server, Oracle, Sybase, Teradata	DB2, Informix, Microsoft SQL Server, Oracle, Sybase	DB2
Administrative interfaces	Replication Center or command line	Replication Center or command line	Replication Center or command line

Chapter 11. Comparison of Q replication to high availability disaster recovery (HADR)

DB2 high availability disaster recovery (HADR) provides a high-availability technology to help you recover from complete site failures, as well as to support applications that demand ultrafast failover for partial site failures.

HADR is also useful for rolling upgrades and other kinds of planned outages. HADR ships database log records for an entire database from a source copy of the database (called the primary) to a target copy of the database (called the standby). The standby database cannot be accessed by applications. The standby is initialized using a restore or a split mirror, which is an identical and independent copy of disk volumes that can be attached to a different system and can be used in various ways.

When you start HADR, it retrieves log records and replays those records on the standby until the standby catches up to the in-memory log set of the primary. The primary then writes log data to local disk and sends them to the standby. Updates to the standby database occur by rolling forward log data that is generated on the primary database and shipped to the standby database.

You can also use Q replication to protect your data from complete site failures and to support application failover for partial site failures, as well as for rolling upgrades and other planned outages. Table 6 summarizes the differences between HADR and Q replication when used for the same purposes as HADR.

Table 6. Comparison of HADR with Q replication

Points of comparison	High availability disaster recovery	Q replication
What is the scope of the setup procedure?	Entire DB2 database.	Tables within a DB2 database or subsystem.
How is data propagated to the standby?	Log operations are shipped to the standby database and replayed continuously through forward recovery.	Committed transactional data are captured from the DB2 recovery log and applied to target tables.
Is synchronous replication possible?	Yes.	No.
Is asynchronous replication possible?	Yes.	Yes.
Are client applications automatically rerouted to the standby?	Yes.	Yes.
Which operating systems are supported?	Linux, UNIX, and Windows.	Linux, UNIX, Windows, z/OS.
Can applications read data from the standby?	No.	Yes.
Can applications write data to the standby?	No.	Yes.
Is SQL Data Definition Language (DDL) automatically replicated in addition to SQL Data Manipulation Language (DML)?	Yes.	No.

Table 6. Comparison of HADR with Q replication (continued)

Points of comparison	High availability disaster recovery	Q replication
On what hardware, operating system, and version of DB2 can the standby run?	Hardware, operating system, and version of DB2 must be identical to those where the source database is located.	Hardware, operating system, and version of DB2 can be different from those where the source database is located.
Are there tools for monitoring performance?	Yes.	Yes.
Is network compression and encryption built in?	No.	Yes.
Can the source and standby databases be partitioned?	No.	Yes.

For more information about HADR, see the *Data Recovery and High Availability Guide and Reference*.

Chapter 12. Comparison of event publishing to DB2 MQ user-defined functions

You can use WebSphere MQ functions that are available in DB2 as user-defined functions in scenarios similar to those scenarios that you can set up with event publishing. These functions allow users to use SQL to access WebSphere MQ message queues from DB2.

These functions can be used by applications written in any of the programming languages supported by DB2 in a wide variety of scenarios:

Data collection
The application receives information in the form of messages from one or more sources. An information source can be any application. The application receives the data from queues and stores the data in database tables for additional processing.

Workload distribution
The application posts work requests to a queue that is shared by multiple instances of the same application. When an application instance is ready to perform some work, it receives a message that contains a work request from the head of the queue. Multiple instances of the application can share the workload that is represented by a single queue of pooled requests.

Application signaling
In a situation where several processes collaborate, you can use messages to coordinate their efforts. These messages might contain commands or requests to perform work.

Event publishing differs from the DB2 WebSphere MQ functions in that you can publish events as messages without having to modify the applications that generate these events. If you want to use the DB2 WebSphere MQ functions, you have to code your applications to work with them.

Accessing information about the product

IBM® has several methods for you to learn about products and services.

You can find the latest information on the Web:
http://www.ibm.com/software/data/sw-bycategory/subcategory/SWB50.html

To access product documentation, go to publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/.

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order.
- To order publications by telephone in the United States, call 1-800-879-2755.

To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide.

Providing comments on the documentation

Please send any comments that you have about this information or other documentation.

Your feedback helps IBM to provide quality information. You can use any of the following methods to provide comments:

- Send your comments using the online readers' comment form at www.ibm.com/software/awdtools/rcf/.
- Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and part number of the information (if applicable). If you are commenting on specific text, please include the location of the text (for example, a title, a table number, or a page number).

Accessible documentation

Documentation is provided in XHTML format, which is viewable in most Web browsers.

XHTML allows you to view documentation according to the display preferences that you set in your browser. It also allows you to use screen readers and other assistive technologies.

Syntax diagrams are provided in dotted decimal format. This format is available only if you are accessing the online documentation using a screen reader.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM trademarks and certain non-IBM trademarks are marked at their first occurrence in this document.

See www.ibm.com/legal/copytrade.shtml for information about IBM trademarks.

The following terms are trademarks or registered trademarks of other companies:

Adobe®, the Adobe logo, PostScript®, the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine™ is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel®, Intel logo, Intel Inside® logo, Intel Centrino®, Intel Centrino logo, Celeron®, Intel Xeon®, Intel SpeedStep®, Itanium® and Pentium® are trademarks of Intel Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT® and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL® is a registered trademark and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library® is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Other company, product or service names may be trademarks or service marks of others.

Index

T

U

X

Printed in USA

GC19-1028-01