Handle distributed transactions with federated two-phase commit in WebSphere Federation Server Version 9.1

New feature enables multi-site updates in distributed transaction environments

Amy Chang
Simon Harris
Frankie Sun

November 30, 2006

Introduction

A federated system is a distributed database management system that enables users to access data in a diverse data source environment, as if all data resides in a single database. A federated transaction is a transaction that includes read or write access to one or more federated data sources within a federated system. This article introduces federated two-phase commit, a new feature in WebSphere Federation Server Version 9.1 that provides multi-site update capability in a federated transaction.

Why federated two-phase commit?

Mergers, acquisitions, legacy products, and high availability requirements often mean that data resides on multiple databases, in disparate systems within a given organization. Consequently, applications may need to read and update data in multiple databases in a single transaction. Two-phase commit ensures transaction integrity for this type of multi-site update database access.

Take a fund transfer activity, for example. Let's assume that someone wants to transfer money from her checking account to her savings account and receive a transaction history. If the checking account database resides on an Oracle server, the savings account database resides on a DB2® for z/OS server, and the transaction history table resides on a DB2 for Linux, UNIX®, and
Windows® server, a multi-site update is necessary to preserve transaction atomicity. In other words, the application must debit the checking account, credit the savings account, and update the transaction history table within a single transaction. If any of these updates fail, then the whole transaction must be backed out. Otherwise, the data will be inconsistent.

Many organizations have mainframe applications on legacy systems. How do you read or write data that resides on one or more data sources from these mainframe applications? Assume that you have a CICS application on the mainframe that needs to read or update a table object on an Oracle server, among others tasks. Previously, the WebSphere® Federation Server enabled you to read Oracle data from the CICS application on the mainframe. With the federated two-phase commit feature, mainframe applications can also update both DB2 and non-DB2 data sources through the WebSphere Federation Server. In a WebSphere Federation Server environment, applications can update one or more homogeneous or heterogeneous data sources within a unit of work.

Federated two-phase commit design

In the pure DB2 environment (with the FEDERATED database manager configuration parameter set to no), the DB2 for Linux, UNIX, and Windows server supports the XA two-phase commit protocol. It participates as an XA resource manager in a global transaction managed by an external XA-compliant transaction manager. The DB2 for Linux, UNIX, and Windows server also supports the Distributed Relational Database Architecture (DRDA) two-phase commit protocol. It acts as either the DB2 transaction manager or the DB2 resource manager in a DB2-coordinated transaction.

In the federated environment (with the FEDERATED database manager configuration parameter set to yes), the WebSphere Federation Server extends two-phase commit support to the federated data sources for both global XA transactions and DB2-coordinated transactions. In addition, the WebSphere Federation Server also supports implicit two-phase commit to the federated data sources under CONNECT Type 1 inbound connection.

Federated two-phase commit provides two-phase commit functionality for federated transactions that access one or more federated data sources. This feature uses the industry standard, X/Open XA protocol, to manage and coordinate distributed transaction processing among federated two-phase commit data sources. The WebSphere Federation Server acts as an XA transaction manager for these federated data sources.

The two-phase commit protocol is comprised of the prepare phase and the commit phase. In the prepare phase, the WebSphere Federation Server sends out PREPARE requests to the federated two-phase commit data sources participating in the current transaction, and then waits for replies. If everything goes well, in the commit phase, the WebSphere Federation Server sends out COMMIT messages to the federated two-phase commit data sources awaiting the transaction outcome.

This process is similar to the one described in the Two-phase commit section of the DB2 Database for Linux, UNIX, and Windows Information Center.
In a distributed environment, there can be tens or hundreds of data sources configured on the federated system. To minimize recovery overhead as well as held resources and locks to all data sources in the event of failures, federated two-phase commit design adopts the presumed nothing protocol with additional optimizations for the federated environment. At the beginning of the federated two-phase commit process, a Forced log record that identifies all the federated two-phase commit participants of the current transaction is written to the disk. A Forced log record mandates that the record be written to disk before the transaction can proceed. At the end of the second phase, a single Forget log record is written to the disk to signal the completion of a transaction.

**Figure 1** illustrates the normal two-phase commit flow for a federated transaction with a CONNECT Type 1 inbound connection involving updates of two federated two-phase commit data sources, RM1 and RM2. Three transactional log records are written at the federated database in this scenario: the FedPrepare log record signals the beginning of the federated two-phase commit process; the Commit log record signals that the transaction outcome of the prepare phase is commit; and the Forget log record signals the completion of the current transaction. Indoubt transactions and resynchronization are discussed in later sections of this article.

**Figure 1: Federated two-phase commit flow with a one-phase commit inbound connection**

In the prepare phase, if all the replies from the federated two-phase commit data sources indicate read-only, and there is no update performed on the federated server itself, then the federated transaction is a read-only transaction. Two transactional log records are written at the federated server in this scenario: the FedPrepare log record and Forget log record. The Commit log record is skipped in this case.
Configurations and transactional roles

Federated two-phase commit is supported for all three types of configurations with the WebSphere Federation Server:

- Applications that connect to the WebSphere Federation Server through one-phase commit connections
- Applications that connect to the WebSphere Federation Server through DB2-coordinated two-phase commit connections
- Applications that connect to the WebSphere Federation Server through XA connections

With a one-phase commit connection, also known as CONNECT (Type 1) in the DB2 family, an application can only be connected to and access one DB2 server within a unit of work. In the context of this article, this DB2 server refers to the federated server. In this configuration, WebSphere Federation Server plays the role of the internal transaction manager that coordinates two-phase commit operations between the federated server and the federated two-phase commit data sources. Applications can perform a multi-site update against federated two-phase commit data sources as well as the federated server. Figure 2 shows a DB2 Linux, UNIX, Windows client updating a local table on the WebSphere Federation Server, joining a DB2 Linux, UNIX, Windows nickname on Data Source 1 and an Oracle nickname on Data Source 3, and updating an Oracle nickname on Data Source 2:

Figure 2: Federated configuration in a one-phase commit connection environment

With a DB2-coordinated two-phase commit connection, also known as CONNECT (Type 2) in the DB2 family, an application can connect to and access multiple DB2 servers within a unit of work. This is also called DB2-coordinated two-phase commit. In the context of this article, one of these DB2 servers refers to the WebSphere Federation Server. In this configuration, the WebSphere Federation Server plays two roles: on one end, it is the sub-transaction manager to the federated two-phase commit data sources; on the other end, it is either the DB2 transaction manager to other DB2 servers or the DB2 resource manager to the DB2 transaction manager. Figure 3 shows an application precompiled with CONNECT (Type 2) accessing an Oracle nickname on Data...
Source 1, a DB2 z/OS nickname on Data source 2, and an Informix nickname on Data Source 3 through the WebSphere Federation Server:

**Figure 3: Federated configuration in a DB2-coordinated two-phase commit connection environment**

In the XA distributed transaction processing (DTP) environment, the WebSphere Federation Server can participate as an XA resource manager in a global transaction that is coordinated by an XA-compliant transaction manager. In this configuration, the WebSphere Federation Server plays two roles: on one end, it is the sub-transaction manager to the federated two-phase commit data sources; on the other end, it is the resource manager to the external XA-compliant transaction manager. **Figure 4** shows an MQ application accessing Informix nicknames on Data Source 1, Oracle nicknames on Data Source 2, and DB2 Linux, UNIX, Windows nicknames on Data Source 3. In this example, WebSphere MQ acts as the external XA transaction manager, while the WebSphere Federation Server acts as the XA resource manager to WebSphere MQ and also acts as the sub-transaction manager to Data Source 1, Data Source 2, and Data Source 3.
Enable federated two-phase commit

After setting up the data source to work in an XA two-phase commit environment, you can enable the federated two-phase commit feature for that data source at the federated database using the DB2_TWOPHASE_COMMIT server option. This server option is disabled by default. To permanently enable the federated two-phase commit feature for a data source for all federated users, you can activate the DB2_TWOPHASE_COMMIT server option through either the CREATE SERVER statement or the ALTER SERVER statement. To enable the feature for a data source temporarily for the duration of a user or application connection to the federated database, you can activate the server option by way of the SET SERVER OPTION statement.

The XA_OPEN_STRING_OPTIONS server option is also provided to enable the submission of additional information to the federated xa_open call for some data sources.

Read Enabling two-phase commit for federated transactions for more details.

Transaction error recovery and resynchronization

First-phase errors

In a federated environment, a first-phase error can occur in these scenarios:

- The WebSphere Federation Server encounters an error after writing the FedPrepare log and before sending out the PREPARE request.
- One or more federated data sources encounters an error either before or after writing the FedPrepare log record at the data source side.
- There is a communication error.
- The WebSphere Federation Server hits an error before determining the transaction outcome.

During the first phase, if any data source encounters an error, the WebSphere Federation Server stops sending the PREPARE request to the rest of the participating data sources because the
transaction outcome is already known. Any first-phase error results in a ROLLBACK outcome. Consequently, the WebSphere Federation Server moves on to send the ROLLBACK transaction request to all participating data sources in the current transaction.

Second-phase errors

In a federated environment, a second-phase error can occur in these scenarios:

- The WebSphere Federation Server encounters an error after writing the Commit log record and before delivering the commit outcome.
- One or more data sources encounters an error either before or after writing the Commit log record.
- There is a communication error.
- The federation encounters an error before writing the Forget log record.

During the second phase, if any data source encounters an error, the WebSphere Federation Server remembers the failure and continues to deliver the transaction outcome to the rest of the participating data sources until the end of the second phase. At that time, the WebSphere Federation Server indicates the need for resynchronization. With this design, as long as the WebSphere Federation Server stays up, the RESYNCHRONIZATION request is only sent over to the data sources that encountered an error. It is not sent to all the data sources that participated in the transaction, so as to keep the number of affected data sources to a minimum.

Handling errors

Federated transaction processing handles each of these errors in the following ways:

- First-phase errors: If the WebSphere Federation Server or any participating data source cannot successfully prepare the transaction, the transaction outcome becomes ROLLBACK and WebSphere Federation Server sends a ROLLBACK request to all participating data sources awaiting a transaction outcome.
- Second-phase errors: Once the transaction outcome is determined and logged at the WebSphere Federation Server, any error occurring subsequently does not change the transaction outcome.
- If one or more participating data source fails to commit (when the transaction outcome is COMMIT) or fails to rollback (when the transaction outcome is ROLLBACK), the WebSphere Federation Server redelivers the transaction outcome to the failed data source through a process called resynchronization. The WebSphere Federation Server does not issue an xa_recover call to the federated data sources. An xa_recover call obtains a list of prepared transaction branches from a resource manager. Instead, the WebSphere Federation Server syncs up with the federated data sources through a process called resynchronization.

Resynchronization

An indoubt transaction is a global transaction that was left in an indoubt state. A federated transaction can become indoubt at the federated database site in any one of the following scenarios:
• In a global XA transaction, when either the external XA transaction manager or the WebSphere Federation Server becomes unavailable after successfully completing the first phase, or the PREPARE phase.
• In a DB2-coordinated transaction, when either the DB2 transaction manager or the DB2 resource manager becomes unavailable after successfully completing the first phase. As stated earlier, the WebSphere Federation may act as either the DB2 transaction manager or the DB2 resource manager in this environment.
• In any one of the three inbound connection environments, when either the WebSphere Federation Server or at least one of the federated two-phase commit data sources becomes unavailable after successfully completing the first phase.

The resynchronization process attempts to complete all indoubt transactions, transactions that have completed the first phase but have not completed the second phase, and will either commit or rollback. In this process, the WebSphere Federation Server connects to the data sources involved in each indoubt transaction and resends the transaction outcome. After all data sources complete the transaction, the WebSphere Federation Server marks this indoubt transaction complete. If any data source cannot complete the transaction, the WebSphere Federation Server retries the resynchronization process during the next time interval. The default interval is 180 seconds. The database manager RESYNC_INTERVAL configuration parameter controls how long the WebSphere Federation Server waits between attempts to commit or rollback these indoubt transactions.

Find more details on the RESYNC_INTERVAL configuration parameter in the resync_interval - Transaction resync interval configuration parameter section of DB2 Version 9 for Linux, UNIX, and Windows.

**Note:** If the transaction is coordinated by an XA-compliant transaction manager, it is up to the transaction manager to initiate the resynchronization.

**Manually recover indoubt transactions**

The WebSphere Federation Server supports the manual recovery of indoubt transactions if you cannot wait for the resynchronization process to automatically resolve them. This process is sometimes referred to as heuristic processing. Compared to a DB2 indoubt transaction, a federated indoubt transaction can fall into two additional transaction states:

• Missing federated COMMIT acknowledgement state: The federated transaction outcome is known (COMMIT). The WebSphere Federation Server has not received COMMIT acknowledgement from one or more federated data sources.
• Missing federated ROLLBACK acknowledgement state: The federated transaction outcome is known (ROLLBACK). The WebSphere Federation Server has not received ROLLBACK acknowledgement from one or more federated data sources.

You can heuristically commit a transaction whose state is missing federated COMMIT acknowledgement, or you can heuristically rollback a transaction whose state is missing federated ROLLBACK acknowledgement. But, you cannot heuristically commit a transaction whose state is
missing federated ROLLBACK acknowledgement, or perform the reverse, because the transaction outcome is already determined.

There are three ways to perform heuristic processing to a federated indoubt transaction:

- With the `list indoubt transactions` command
- Through the Indoubt Transaction Manager GUI
- Through heuristic APIs, where a set of APIs is provided to query, commit, and rollback indoubt transactions, and to cancel transactions that were heuristically committed or rolled back

**Performance considerations**

Enabling a remote source for federated two-phase commit incurs this additional logging and communication overhead as compared to a source that is not enabled:

- Additional database log writes at the WebSphere Federation Server (all transactions, including read-only transactions and those that only involve one source)
- Additional database log writes to the data source (update transactions, even if they involve only one source)
- Additional communication between the WebSphere Federation Server and the data source (all transactions)

It is important to realize that enabling federated two-phase commit for a remote source causes all transactions involving that source to experience the above processing, I/O, and network overheads. This does not just apply to those transactions that require two-phase commit. You should only temporarily enable federated two-phase commit for a data source within an application requiring two-phase commit semantics. Federated two-phase commit enabled in this way is active for the duration of the application's connection to the federated database. Enabling federated two-phase commit for a data source permanently through a persistent server option means that all federated applications using that source incur the potential overhead of federated two-phase commit, even if they have no need to support multi-site updates.

For transactions that do not require federated two-phase commit, the overhead -- an increase in elapsed time and resource use -- associated with federated two-phase commit is constant, and is unrelated to the nature or duration of the underlying transaction. Thus, the overhead is more noticeable for transactions that are short, and less significant for transactions that are longer to begin with. Again, you should minimize any unnecessary overhead by only enabling federated two-phase commit within applications that actually require the feature.

A very important factor in minimizing federated two-phase commit overhead is the selection of a log device on the federated server. Since federated two-phase commit involves additional writes to the log, there is a considerable advantage to placing the federated server log files on high-performance disks with fast-write caches. A fast-write cache provides an area of memory that allows I/O to be deferred and batched. Of course, the write cache must guarantee the persistence of writes made to the cache, meaning they must eventually be written to durable disk media.
Performance testing shows improvements on the order of 40% in the aggregate throughput of transactions involving federated two-phase commit-enabled sources when log disks with fast-write cache are used on the federated server rather than simple SCSI devices with no write cache.

**High Availability Disaster Recovery support**

DB2 for Linux, UNIX, and Windows High Availability Disaster Recovery (HADR) is a database replication feature that provides a high availability solution for site failures. HADR protects against data loss by replicating data changes from a source database, called the primary, to a target database, called the standby. Regardless of the inbound connection type, the federated server provides Automatic Client Reroute support for a DB2 for Linux, UNIX, and Windows Version 8.2 and later data sources where the HADR feature is enabled.

Automatic Client Reroute is a DB2 for Linux, UNIX, and Windows feature that allows client applications to recover from a loss of communication with the server, so that the application can continue its work with minimal interruption. Figure 5 illustrates a WebSphere Federation Server enabled with the Automatic Client Reroute feature accessing a federated data source configured with the HADR feature.

**Figure 5: Federated two-phase commit and HADR**

Troubleshooting tips

**Supported data sources and wrapper modes in WebSphere Federation Server Version 9.1**

Federated two-phase commit is supported in these data sources using trusted wrappers in WebSphere Federation Server Version 9.1:

- DB2 for z/OS, DB2 for iSeries, DB2 for Linux, UNIX, and Windows
- Informix
- Oracle
- Sybase
- Microsoft (MS) SQL Server
For Sybase and MS SQL Server support, the WebSphere Federation Server must be installed on the Windows platform.

**Considerations for DB2 for z/OS and DB2 for iSeries data sources**

When a DB2 for Linux, UNIX, and Windows instance is created, the DB2 sync point manager (SPM) is automatically configured with default settings. SPM is a component of DB2 Connect functionality that is required for a DB2 data source that does not yet support native XA flows in DRDA but does support DRDA two-phase commit flows. DB2 for z/OS and DB2 for iSeries Version 5.3 or earlier fall into this category. SPM performs a mapping between the XA and DRDA two-phase commit flows. By default, the DB2 instance has pre-defined values for SPM configuration parameters. The most significant parameter is the SPM_NAME database manager configuration parameter. If your federated application encounters a communication protocol error while attempting to access the DB2 for z/OS data in a two-phase commit setting, please verify if the SPM_NAME is properly set.

When federated two-phase commit access is provided through SPM, not all XA semantics are supported due to incompatibilities between federated support and SPM. For example, transactions cannot be nested. All transactions must be committed or rolled back before starting a new transaction.

**Considerations for accessing DB2 for z/OS or DB2 for iSeries data sources with an external XA-compliant transaction manager**

The architecture of some XA-compliant transaction managers can have commit sequences from different processes. The DB2 Connect connection concentrator technology allows applications to stay connected without consuming any resource on the DB2 host server. The DB2 Connect connection concentrator must be enabled to support commit sequences from different processes of the XA-compliant transaction manager (for example, WebSphere MQ) when DB2 for z/OS or DB2 for iSeries Version 5.3 database servers are to be updated. Currently, the connection concentrator and federation features cannot be enabled together. To facilitate such an external XA-compliant transaction manager accessing either DB2 for z/OS or DB2 for iSeries Version 5.3 and lower data sources in a federated two-phase commit environment, a DB2 Connect hop is required (with the connection concentrator activated) between WebSphere Federation Server and DB2 for z/OS and DB2 for iSeries data sources. Figure 6 illustrates such a configuration with WebSphere MQ acting as the external XA transaction manager.

**Figure 6: WebSphere Federation Server and DB2 Connect (concentrator and SPM) configuration**
These are two types of applications where the connection concentrator is used along with federated two-phase commit:

**Application 1:**
xa_open  
xa_start tx A  
   Federated work  
xa_end tx A

**Application 2:**
xa_open  
xa_prepare tx A

The typical use of the concentrator allows multiple inbound connections to reuse an outbound connection. Example 1 illustrates a scenario where more than one inbound connection works on a single transaction as far as the commit processing is concerned. These two inbound connections must use the same outbound connection to the z/OS or iSeries database because they both deal with transaction `tx A`. By enabling the connection concentrator, this becomes feasible. This is particularly useful in the XA DTP environment where the external XA-compliant transaction manager supports commit sequences from different processes.

```plaintext
xa_open  
xa_start tx A  
   Federated work  
xa_end tx A  
xa_start tx B
```

In Example 2, there is one inbound connection but multiple outbound connections, each for a single in-flight transaction. Transaction `tx A` is `xa` ended but not yet committed or rolled back when transaction `tx B` is started. Transaction `tx B` requires a different outbound connection, while the outbound connection for transaction `tx A` is put aside. However, the outbound connection to the z/OS or iSeries database is tied to a single transaction at any point in time, a DRDA two-phase commit requirement. By enabling the connection concentrator, another outbound connection is initiated and used for transaction `tx B`. There are two outbound connections going to the same z/OS data source, one for each transaction.

**Conclusions**

Federated two-phase commit provides location transparency and transaction atomicity for the WebSphere Federation Server. Federated two-phase commit combines updates to multiple data sources in one transaction so that all sources involved are updated, or none are updated. This strategy ensures that the sources remain synchronized. This feature will further speed up the delivery of new applications and assure data integrity in your distributed environment.
Related topics

- **IBM WebSphere Information Integration Administration Guide for Federated Systems**: Refer to this book for basic product documentation about the WebSphere Federation Server.
- **IBM WebSphere Information Integration Configuration Guide for Federated Data Sources**: Explore federated system concepts and fundamental documentation on how to configure a federated system.
- **The DB2 UDB SQL Reference, Volume 1** and **The DB2 UDB SQL Reference, Volume 2**: Learn how to use SQL with DB2 for Linux, UNIX, and Windows in these reference books.
- **IBM DB2 Universal Database Command Reference**: Explore detailed descriptions of DB2 commands.
- Build your next development project with **IBM trial software**, available for download directly from developerWorks.

© Copyright IBM Corporation 2006

Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)