

**The Purpose of the Two-Phase Commit Procedure for
Databases in a Three-Tier Distributed Environment**

Robert Milton Underwood, Jr.

© 2000

The Purpose of the Two-Phase Commit Procedure for Databases in a Three-Tier Distributed Environment

A “distributed system” is just what its name implies- system components are shared, or distributed, among two or more nearby or distant workstations. The users within the system share the resources connected to the system, including, but not limited to, printers, databases¹, communication lines, and servers. In a distributed system, the workstations typically do not share memory, but do share processing ability. In fact, processing is one of the most important features of a distributed system in that access to shared resources speeds up the processing capabilities of transactions². Two other benefits of a distributed system, for which the two-phase commit system is used, is to ensure increased data reliability and improved data availability.

The files of a distributed system may be dispersed among several storage devices or contained within the system. Centralized systems with only one database or data server (see Figure 1) do not have the synchronization issues that a multiple server system has (see Figure 2). Synchronization of processes is critical in multi-server systems, especially when users from remote locations need

¹ According to Warford (1999), databases come in three main varieties: hierarchical systems, network systems, and relational systems. The two-phase commit procedure is most relevant to the network system.

² A transaction is any activity performed by an application or any request made to an application (Monash, 1997). Changing data, deleting data, or adding data are examples of transactions that affect a database. An ad hoc query is a type of transaction that does not affect the database.

Figure 1
Centralized Database System

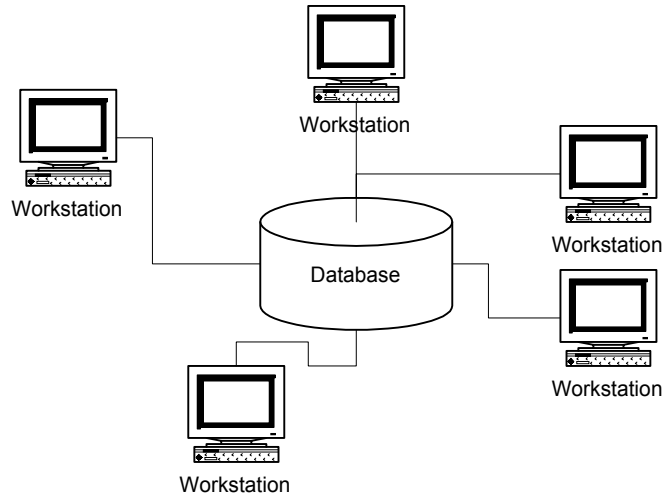
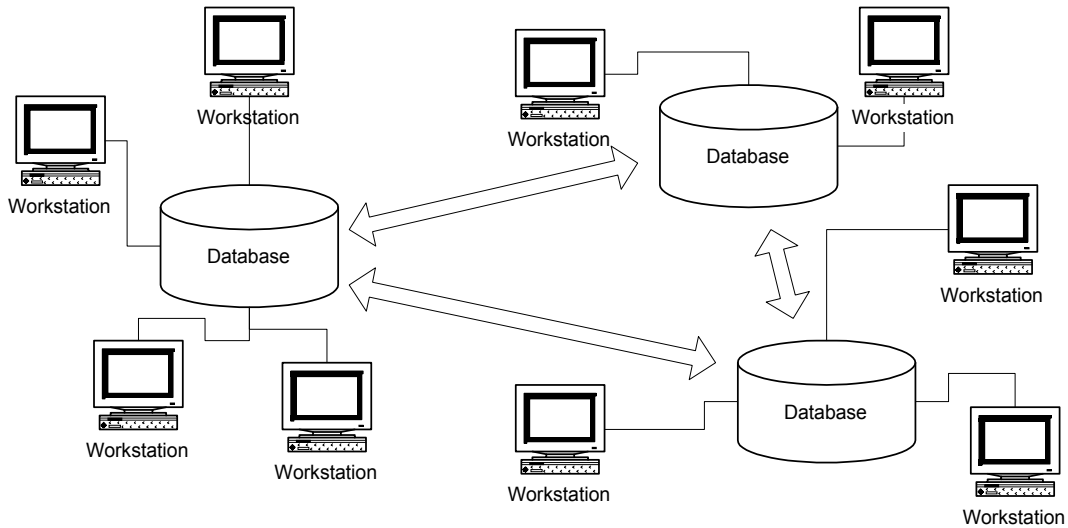


Figure 2
Distributed Database System



to rely on accurate and timely data to perform their jobs. The manner in which communications among servers and databases is configured will help make certain that

there are no bottlenecks in the system or scalability³ issues that could slow or prevent the transmission of data.

The two-phase commit protocol allows the management of transactions in a distributed environment. The “commit” of the data happens in two “footsteps” (Archesis, 2000). In the first footstep, a transaction coordinator (a.k.a., global coordinator; a.k.a., Transaction Manager⁴) is a software routine that sends the message of PREPARE TO COMMIT to all the databases or agents (a.k.a., Resource Managers⁵) interested in the transaction. If responses are positive and occur within the timeout limitations, the coordinator sends the message of COMMIT; if a negative response is received from any station, the message of ROLLBACK is sent. The two-phase commit strategy is designed to ensure that either all the databases are updated, or none of them (internet.com Corporation, 1999).

Detailed discussion of the two-phase procedure introduced in the preceding paragraph will follow after a brief overview of three-tier architecture. The two-phase commit procedure is best understood in the context of its role in a distributed three-tier system.

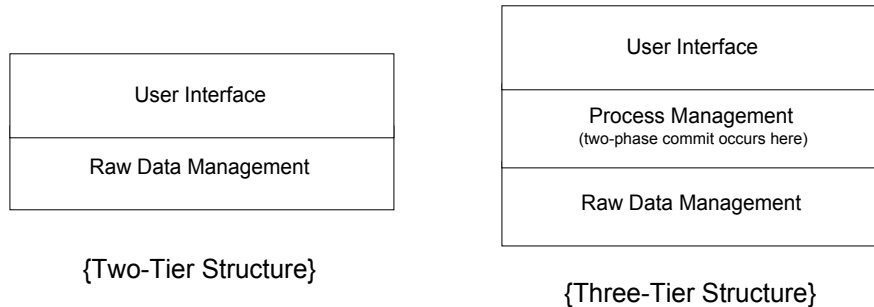
The three-tier, or three layer, software architecture became popular in the past several years to overcome the limitations of the two-tier architecture (see Figure 3). The two-tier architecture is composed of the “client,” which is the user interface, and the “server,” which has the data management component. With three-tier architecture, a middle tier was added to provide process

³ As distributed systems grow larger as companies expand operations or merge with other companies, transaction volume is a major consideration. An application that does not respond well to increases in the volume of transactions is said to lack effective scalability (Monash, 1997).

⁴ Marsh, Donald A., Jr. (1996). *Two-Phase Commit Protocol (2PC)*.

⁵ *ibid*

Figure 3
Two-Tier and Three-Tier Architectures



management where business logic and rules are executed (Sadoski & Comella-Dorda, 2000). Whereas two-tier architecture can accommodate only about 100 workstations, three-tier architecture will allow many times that amount, and provides functions such as application execution, queuing, and database staging (Sadoski & Comella-Dorda, 2000). Many company information technology (IT) experts choose three-tier architecture over two-tier architecture for Internet applications and Internet-based information systems because of its “increased performance, flexibility, maintainability, reusability, and scalability by centralizing process logic” (Sadoski & Comella-Dorda, 2000). In contrast, two-tiered architecture is preferred when the number of users is less than 100, and/or when real-time information processing is not required.

In a three-tier architecture, the top tier includes a user interface where user services such as display management reside. When a user views the monitor at his or her workstation with the application running, what is seen is a result of the user interface. The complexity of the distributed processing is kept hidden from the user. The third tier essentially provides database management functionality and is dedicated to data and various file services. The data management feature ensures that the data on the system is

consistent throughout the distributed environment through the utilization of features such as “data locking, consistency, and replication” (Sadoski & Comella-Dorda, 2000).

The middle tier of a three-tier system provides process management. Process management services include process monitoring, process resourcing, process development, and process enactment. Multiple applications on the system share these process management services. Centralized process logic on the middle tier allows system functionality to be localized, thereby resulting in easier system administration and easier effectuation of and management of changes.

Process management in the middle tier organizes transactions and asynchronous queuing to guarantee reliable completion of transactions, and manages database integrity by the two-phase commit process.

The lowest tier of the three-tier architecture for distributed database technology is represented by the management of data in the database or data server. It represents the physical location where raw data is accumulated and stored, and by which users may obtain results of ad hoc queries.

Atomicity is the essence of the reason behind the need for the two-phase commit procedure. The two-phase commit procedure is the protocol that ensures the atomicity of transactions (Guyon, 1995). The basic concept of atomicity maintains that either all of the operations associated with a system are executed to completion, or none are performed (Silberschatz & Galvin, 1999). It is more difficult to exercise the atomicity property of a transaction in a distributed system than in a centralized system because several sites may be trying to execute a single transaction. If one of these sites fails in its execution of a transaction, then errors and poor information may result.

While atomicity means that all parts of the system do something (e.g., update data) or do not do something (e.g. do not update data), it does not necessarily mean that the activity is done at precisely the same time (Bonilla, 1999). The purpose of the two-phase commit procedure is to ensure that data is updated throughout the distributed system in a synchronistically atomic manner.

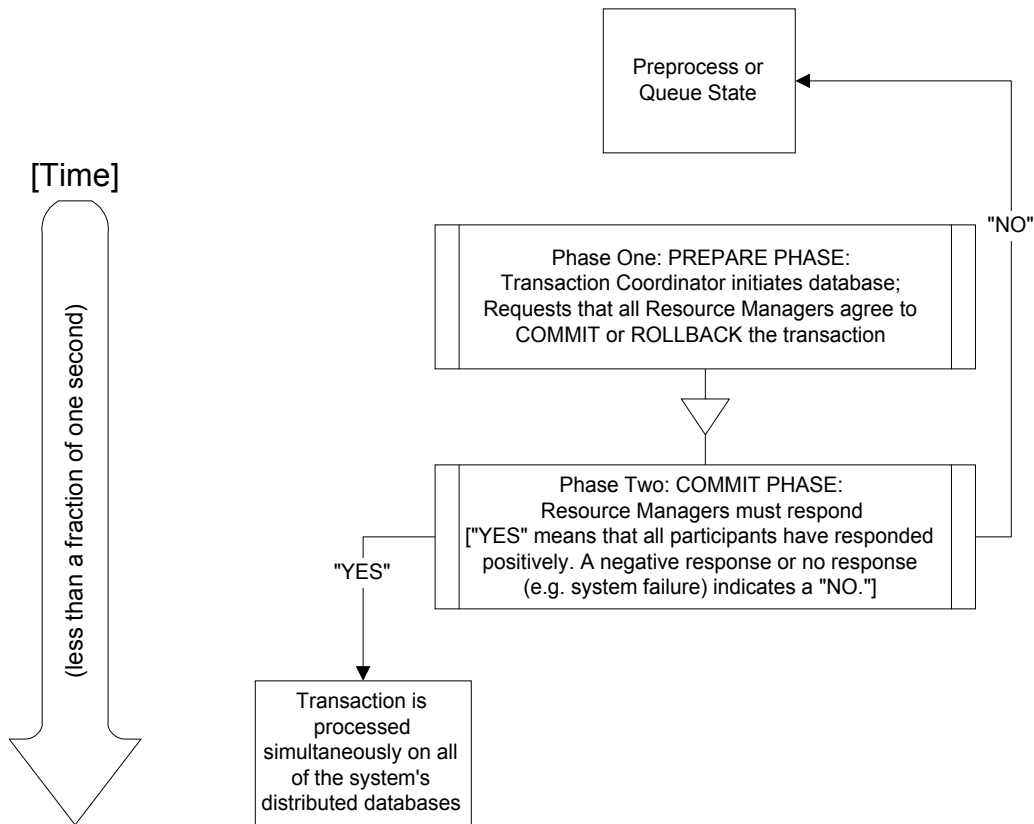
The transaction coordinator in a distributed system makes sure that the execution of the numerous transactions within the system is executed atomically. Each site has a local transaction coordinator that takes care of making sure that (a) the transactions at its site start execution, (b) subsets of transactions are distributed individually to the appropriate sites for execution, and (c) transactions are terminated if necessary, which subsequently means that all sites will experience an abort of transaction.

As has been established, synchronization of data within a distributed system is critical. Centralized database systems only make local updates, as indicated in Figure 1. But in distributed data systems, updates are made globally, as depicted in Figure 2. Synchronization of data ensures the atomicity of operations within the system. While receiving data in a timely manner is important to users, having data synchronized and transmitted globally is even more important because all users will thereby have access to valid data.

Distributed databases using the two-phase commit technique update all databases simultaneously. There are two processes, or “phases,” of the two-phase commit procedure, and the entire procedure occurs in less than a fraction of a second (Figure 4). The first phase is the Prepare Phase (Sadoski, 2000). During this phase, the transaction coordinator initiates the database and commands that all participants (i.e., Resource

Managers; databases) will promise to COMMIT or ROLLBACK the transaction. This “pre-commit” command requires an acknowledgement.

Figure 4
Two-Phase Commit Procedure



The second process of the two-phase commit procedure is the Commit Phase. During this process, all participants indicate to the transaction coordinator that they are prepared. When confirmation is given, the transaction coordinator requests all nodes to commit the transaction. If one node cannot prepare, or if there is a system component failure, the transaction coordinator requests all databases to ROLLBACK the transaction.

If the COMMIT is executed successfully, all databases make the transaction change(s) permanent (internet.com, 1999).

If there is a failure during the two-phase commit procedure due to software, network, or hardware problems, then the two-phase commit protocols will automatically finish the recovery without need for the database administrator to become involved. This recovery process is done through the use of pending transaction tables in each database where data about distributed transactions are kept as they continue on through the two-phase commit procedure (Sadoski, 2000). Data that is in the pending transaction table is used by the recovery process to resolve any transaction that has been rolled back.

According to Koch and Loney (1997), and referring to the capabilities of a database management system using the Oracle database, it is not necessary to do anything special to have applications use a two-phase commit procedure. All sites either commit or rollback together, no matter what errors occur in the network or on the machines connected by the network. Once the database administrator or IT specialist has set up the system, the two-phase commit procedure happens automatically.

Two problems have been recognized with the two-phase commit process (Sadoski, 2000). The first problem is that if one database is down for repair, upgrade, or for any other reason, none of the other databases can get information transferred. This problem can be solved, however, by the database administrator, who has the power to force the COMMIT procedure to the databases that are available and working properly.

The second problem recognized with the two-phase commit process is with regards to networks that grow continually larger, as is the case with companies that merge. While the purpose of the three-tier architecture was to be able to accommodate

hundreds of users, the network architecture itself may be burdened with added resources. A company that previously used a connection speed of 128Kbs, may need to upgrade to T-1 lines to accommodate the added transaction load resulting from their merger with or acquisition of another company.

The two-phase commit procedure is not needed for every transaction handled by large companies. For instance, in Austin, Texas, a real estate leasing company named Superior Views had expanded during 1998 and 1999 to six new cities. Each city has three to five offices. In early 2000 they merged with another leasing company, Leasing Experts, which had numerous offices in eight different cities. After the merger, the company was large enough to need a three-tier architecture, which was implemented over a period of two months. Some processes, such as those related to accounting data, utilized the two-phase commit procedure, because accuracy in financial-related processes was critical as monthly budgets were prepared and payments made through Accounts Payable. But other data, such as specific real estate listings, was not considered necessary for the two-phase commit procedure. The reason is that the real estate listings only needed to be updated locally. Licensed real estate agents with the company only need to specialize in the property inventory of the city in which they work. Thus, an agent working from a San Diego office would not need to access data about property listings in Tucson. So while some transactions at Superior Views were updated globally using the two-phase commit procedure, other transactions were updated only locally.

There are currently no specific industry standards for how the two-phase commit process should be implemented. The reason is that two-phase commit procedures are proprietary by vendors (Sadoski, 2000).

Terminology itself is not even consistent across the industry. For instance, *Transaction Coordinator*, *Global Coordinator*, and *Transaction Manager* are used by different individuals to mean the same thing. Different terminology may refer to essentially the same two-phase commit process, but may vary according to how specialists of a given system use it. For instance, individuals who specialize in client/server programming for the OS/2 system refer to the phrase *syncpoint protection*, which is a stringent level of synchronization. Syncpoint protection addresses the problem of “maintaining the integrity of the network transaction. Its full implementation requires a two-phase commit protocol with rollback and synchronization capabilities (Orfali & Harkey, 1992).”

While there is no industry wide standardization, several vendors have, however, published their two-phase proprietary protocols. Nevertheless, the two-phase commit procedure will continue to be an important protocol for companies that want to ensure the atomicity of transaction processing through a distributed database system.

References

- Archesis. (2000). Il protocollo di Two Phase Commit. [on-line].
- Bonilla, O. (1999, March 26). Two Phase Commit. [on-line].
- Guyon, I. (1995, November 14). Two-Phase Commit and Transaction Control. [on-line].
- internet.com Corporation. (1999). Transaction Processing. [on-line].
- Koch, G. & Loney, K. (1997). Oracle 8: The Complete Reference (p. 1210). Berkeley: Osborne McGraw Hill.
- Marsh, D. (1996). Two Phase Commit Protocol (2PC). [on-line].
- Monash Information Services. (1997). Transaction. [on-line].
- Orfali, R. & Harkey, D. (1992). Client/Server Programming with OS/2 2.0, (2nd Ed.) (p. 490). New York: Van Nostrand Reinhold.
- Sadoski, D. (2000). Database Two Phase Commit. Software Technology Review. [on-line].
- Sadoski, D. & Comella-Dorda, S. (2000, February 16). Three Tier Software Architectures. Software Technology Review. [on-line].
- Silbershatz, A. & Galvin, P. (1999). Operating Systems Concepts (Fifth Edition) (pp. 569-573). New York: John Wiley & Sons, Inc.
- Warford, J. (1999). Computer Systems (p. 22). Boston: Jones and Bartlett Publishers.