Data Consistency of Distributed Transaction for Order Management

Phyoe Su Su Win, Thet Su Mon
Computer University, Taung-Ngu
sulattphyo@gmail.com, thetsumon18@gmail.com

Abstract

The general elements of transaction processing are data capture and validation, transaction-dependent processing steps and database maintenance. Database Management Systems (DBMS) are among the most complicated applications. While DBMS maintains all information in the database, applications can access this information through statements made in Structured Query Language (SQL), a language for specifying high-level operations. This paper intends to build a distributed transaction management system for store ordering with recovery control to guarantee the consistency of database. When the server crashes or connection fails, the uncommitted transaction is saved at the client store as the recoverable objects and will undo at the next time for database consistency.

1. Introduction

In today's electronic world, most companies have implemented database systems and track the transactions of their daily operations. Many forward-looking companies have implemented transaction management systems to empower their staff to make sensible business decisions. In today's competitive global economy, managers are increasingly aware of the fact that their data resources are necessary for their organization. The growth of distributed system has made it possible to impact the transaction processing in a large scale. One of the effective usage of distributed transaction processing system is in distributed store management system. Nowadays, commerce is larger and larger and it affects to open new branch offices in different locations. Therefore it is important to handle the order of each branch by the head office. This proposed system intends to provide the orders of client branches to many different servers or product suppliers.

In this system, there are one or more head offices (servers) and stores (clients) located at the different locations. The clients will made requests to server's services to order sale items and can get the reply from the server. The server will accept the requests of order from different clients and check that whether the amount of items requested and the balance of this items is validate or not. If it is validate, the server will make the transaction process in itself and reply "order transaction is valid" message and then the client may know that the transaction is valid. Otherwise the server will return one of the "Invalid" messages to client and the client can make some changes in the transaction. This system will provide simplification of using the system and interact effectively to the users' requests.

2. Related Works

In [5], the author proposed the Transaction Management Architecture and Business Plan for implementation of Transaction Management System (TMS). They provided access to their transmission system in order to promote competition in wholesale power market. Their TMS is envisioned to automate and integrate the market interface and to coordinate security processes. Their system is fast and reliable enough to handle the hourly market transactions.

In [6], the author proposed a formula to provide the design of bank communication system. They presented the key issues of high availability, traffic prioritization and security. And they also presented authentication mechanisms on personal identification, digital certificate, and biometric.

3. Background Theory

A database system is basically a computerized record keeping system. The main purpose is to store information and to allow users retrieve and update that information on demand. [3]

3.1 Database Management

Users of the system can perform a variety of operations on such files-

- Adding new, empty files to the database;
- Inserting data into existing files;
- Retrieving data from existing files;
- Changing data in existing files;
- Deleting data from existing files;

3.2 Database Management System

Database management system (DBMS) is computer software designed for the purpose of

managing databases based on a variety of data models. A DBMS is a complex set of software programs that controls the organization, storage management, and retrieval of data in a database. DBMS are categorized according to their data structures or types, sometimes DBMS is also known as Database Manager. It is a set of prewritten program that are used to store, update and retrieve a database. A DBMS includes [3]:

- A modeling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.
- Data structure optimized to deal with very large amounts of data stored on a permanent data storage device.
- A database query language and report writer to allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data.
- A transaction mechanism that ideally would guarantee the ACID properties, in order to ensure data integrity.

4. Distributed Transaction Management

The goal of transaction is to ensure that all of the objects managed by a server remain in a consistent state in the presence of server crashes. The server must guarantee that both transactions are carried out and the results recorded in permanent storage, or in the case of crashes, the effects are completely erased. The object that can be recovered after the server crashes is called recoverable object.

A client transaction becomes distributed if it invokes operations in several different servers. There are two different types of distributed transactions:

- Flat transaction
- Nested transaction [2]

In a flat transaction, a client makes requests to more than one server. For example, in figure 1, transaction T is a flat transaction that invokes operations on objects in servers X, Y and Z. A flat client transaction completes each of its requests before going on to the next one. Therefore each transaction accesses servers' objects sequentially. [2]

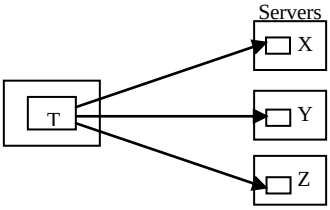

Figure 1. Flat Distributed Transaction

In a nested transaction, the top level transaction can open subtransactions and each subtransaction can open further subtransactions down to any depth of nesting. [2]

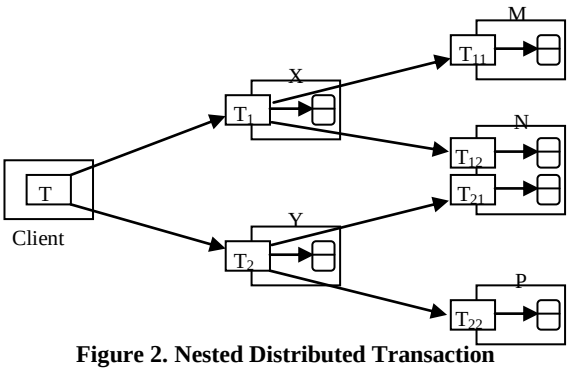

Figure 2. Nested Distributed Transaction

Figure 2 shows a client's transaction T that opens two subtransactions T1 and T2, which access objects at server X and Y. The subtransactions T1 and T2 open further subtransactions T11, T12, T21 and T22, which access objects at servers M, N and P. In the nested case, subtransactions at the same level can run concurrently, so T1 and T2 are concurrent, and, they can run in parallel. The four subtransactions T11, T12, T21 and T22 also run concurrently. [2]

This proposed system uses flat architecture of distributed transaction.

4.1 Client/ Server Network Environment

The client/server method of network communications is the most popular today. Its ease of implementation and scalability make it a good choice in many different kinds of networking environments. A client is a computer that requests access to shared network resources from a server, a computer that provides shared resources in response to client request. Client/server computing generally refers to a network structure in which the client computer and the server computer share the processing requirement. A server is best described as a machine whose only function is to respond to client requests. [4]

Server-based networks provide centralized control over network resources, primarily by instituting network security and control thorough the server's own configuration and setup. In most cases, servers are dedicated to handling network requests from the client communities. Server-based networks also provide centralized verification of user accounts and passwords so that one or more specialized servers act as sentries, guarding access to the network [4].

Server-based networks also typically require only a single log on to the network itself; User needs not to remember numerous passwords for individual resources [4]

5. Proposed System

The overview of the proposed system is shown in figure 3. In this system, there are two main parts

such as clients and the servers. The clients are the operational stores and the servers are the product suppliers. The client stores can make their orders to many different suppliers or servers with only one transaction. If the transaction in all servers is committed, this transaction is committed and never undone. If the transaction in one of the servers is not completed, the client saves the transaction with this server in its local database to be undone.

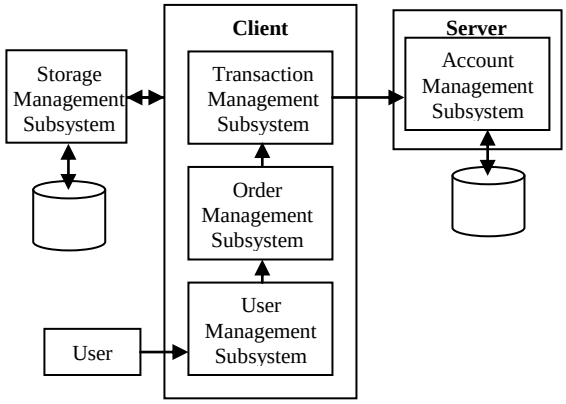

Figure 3. Overview of the Proposed System

The client has its own local database and DBMS for daily transaction and the server also has its own local database and DBMS. By using distributed transaction for ordering, the client requests the server DBMS. The server DBMS accepts the client's order requests and processes and then returns the appropriate response to client. By using this response, the client can know this transaction is commit or rollback.

This system intends to use five subsystems namely, user management, order management, transaction management, storage management and account management subsystem.

User Management Subsystem

This subsystem manages the different users of the system by using their login information. This system will use user names and passwords as login information. There are three types of users in this system such as operators, store managers and managers at server sites.

Order Management Subsystem

This system will manage the order from the client stores. The items ordered by store are grouped into their respective servers by using database. Then these items groups are sent to transaction management subsystem to make distributed transactions.

Transaction Management Subsystem

This subsystem is responsible for the management of distributed transactions. The ordered item groups are sent to the respective servers as a transaction. If the whole transaction completes successfully, this subsystem will commit. Otherwise, the abort transaction is saved to rollback for recovery.

Storage Management Subsystem

This subsystem will manage DBMS-specific calls for information storage and retrieval. It acts as a common interface for other subsystem to manage data.

Account Management Subsystem

It will store the account balance of each client. The server manager can query the account balance with clients from this system.

As shown in figure 4, this system will request the login information from the user to make an order. If the login information is success, the system will request the order list and then the server lists is also acquired from the database. After getting the server address list, the ordered items are grouped by their respective servers. Then the transaction to first server is sent and the client will wait to be returned from the server. If the transaction commits, successful message will be returned. Otherwise, unsuccessful message will be returned. This process will iterate until the transactions to all of the servers hires. After transaction to all servers is completed, the final result will be displayed.

At the client, the items are stored by their respective servers' names. Example of such item table is shown in table 1.

Item	Name	Server
Ballpen	Uni	Server 1
Pencil	Tokiwa	Server 2
Book	Top Choice	Server 3
Bag	Tri	Server 2

Table 1. Item Table

When the client makes the order, order management subsystem uses this item table to group the items with their respective servers. In this example, pencil and bag orders are grouped because their servers are the same. Then the order group list is sent to transaction management subsystem. This subsystem uses server table to get the ip list of the servers. Example of the server table is shown in table 2.

Server	ip
Server 1	192.168.1.1
Server 2	192.168.3.2
Server 3	192.168.2.4

Table 2. Server Table

After getting the server ip list, the transactions are sent to each server, one after another.

The system flow diagram is shown in figure 4. As shown in this figure, this system is limited only for the administrators to order the items to the

servers. To verify the users or administrators, this system gets log in information from the user. If the login information is correct, the administrator can make order processing. And otherwise, the user is requested the login information until this information is correct.

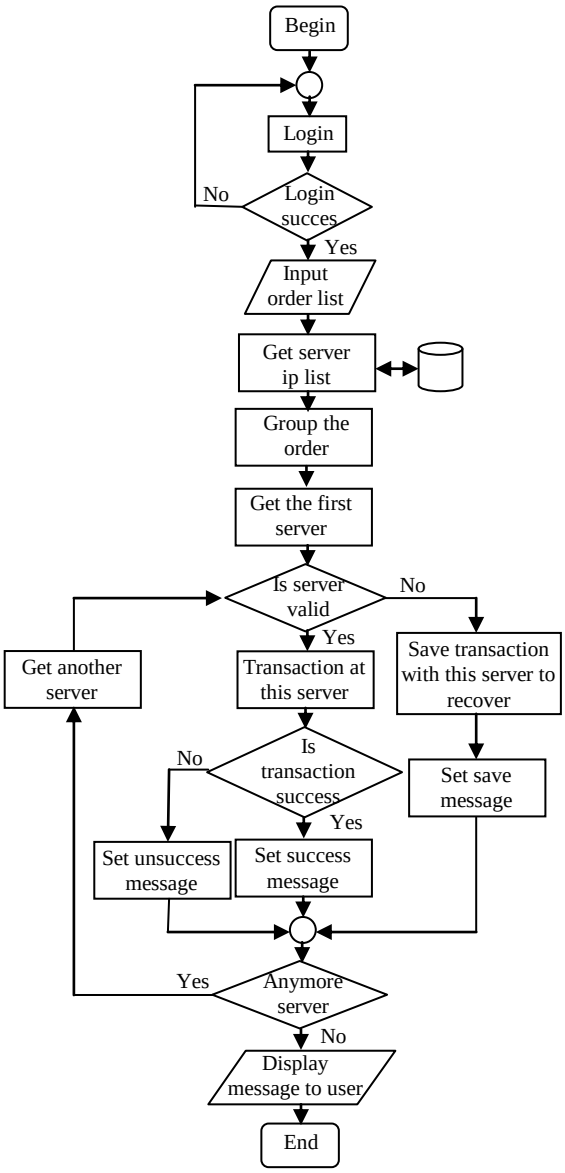

Figure 4. System Flow Diagram of the Proposed System

When the supplier server accepts the item list, the account management subsystem in the server creates the SQL statements with this list to query that the item ordered (requested) has sufficient amount in this database. In our example, if the client transaction to server 1 is Uniball ballpen and the amount is 5000, then the SQL query statement may be:

*Select * from Server1 where Item = 'Ballpen' and Name= 'Uniball' and amount-5000>=0*

If any value returns from this SQL statement, the account management subsystem knows that the server has sufficient amount for this order request. Then it creates SQL update statement like that:

Update Server1 set amount = amount-5000 where Item = 'Ballpen' and Name= 'Uniball'

After this update is successfully completed, the server 1 returns "Transaction is ok" message to the client. Otherwise the returned message may be one of the rollback messages.

For the rollback message, the client saves the order list to this server in its own client database for recovery process to be ensured the database consistency.

The components interaction is shown in sequence diagram of figure 5.

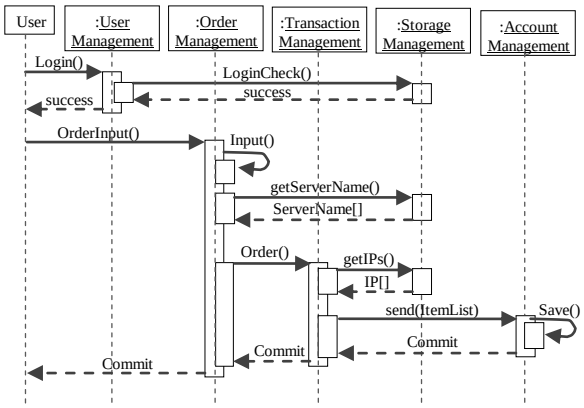

Figure 5. Sequence Diagram of the Proposed System

6. Conclusion

Nowadays, the business processes are wider and wider and it affects to use distributed transaction processing in many organization. The objective of this paper is also to present the use of distributed transaction in real world, to explain the problem raised by using distributed transaction and to maintain the consistency of each database. This proposed system can be extended for security checking and bank cheques validation of the client. Reconciliation is very important for transaction between enterprises. Reconciliation of order items between the client and server should be extended in this system to guarantee the verification.

7. References

[1] A. McFadden, Fred and Hoofer, A. Jeffrey, " Modern Database Management (Fourth Edition)", The Benjamin/Comings Publishing Company, Inc., 1993.

[2] C. George, D. Jean, and K. Tim, "Distributed Systems – Concepts and Design (Third Edition)", Peterson Education Ltd, Edinburgh Gate, Harlow, Essex CM20 2JE, England

[3] C. J. Date, “An Introduction to Database Systems (Seventh Edition)”, The System Programming Series, Addison-Wesley Publishing, Company, Inc., 1994.
[4] Mellon, Carnegie, “Client/Server Software Architecture and Overview”, Software Engineering Institute, 1997.
[5] A.Y. Palevy, et al. “Schema Mediation in Peer Data Management Systems” in Proceedings of the19th International Conference on Data Engineering (ICDE'03) 2003.
[6] DBGlobe. [cited; Available from: http://softsys.cs .uoi. gr/ dbglobe /index1.html.]