ISSN 2229-5518

A Distributed Transaction Model for a Multi-database Management System

Omar Baakeel and Abdulaziz Alrashidi

Abstract— This paper examines the distributed transaction issues that are present in multi-database management systems (DBMSs) and how the distributed transaction in database technology differs from other distributed processing systems. Some common issues that arise with the distribution of transactions over a multi-DBMS while dealing with concurrency control and recovery are highlighted, such as site failure, network failure, and time failure. This study proposes an exceptional distributed transaction model for processing transaction queries in multi-DBMS based on the idea that a complete database can be constructed at any stage.

Index Terms— Database management system (DBMS), distributed transaction, transaction management, transaction network

1 INTRODUCTION

GENERALLY, a *distributed transaction* is defined as a transaction that updates data on two or more networked computer systems. Different conceptual structures for processing transaction elements are required to meet certain needs of applications that must update distributed data in terms of peer to peer connection. Developing a transaction model is difficult because most of these techniques are customized for a single transaction point, and many types of failures, such as client failure, server failure, and network connection failure, can occur. Furthermore, connecting different distributed points requires the presence of advanced network security technology to detect and recover from network connection failure [1].

Recently developed transaction applications have optimized various security integrations so as to secure the transaction between client and server by detecting and managing the change action, such as a database update, before the transaction can occur. A transaction processing network is considered a complicated process because of the multiple security issues that exist in these networks. The performance of a transaction between a single database management system (DBMS) and a multi-DBMS requires coordination with other network parts for the components involved. A distributed transaction coordinator carries out this coordination and functions as a transaction manager for each computer that manages transactions.

Algorithms have been developed to serve specific needs in various areas including banking, investment environments, and enterprise applications [1]. When computers communicate with each other, a transaction manager sends, prepares, commits, and aborts messages to all of its subordinate transaction managers [2]. The steps for performing a distributed transaction are as follows:

1. The client begins the transaction by calling the transaction manager. This application allows for the transaction to be deployed.
2. While the transaction manager processes the client request, an update is saved so that the deployed transaction can carry out the process based on the client details.
3. After processing the client request, the transaction manager keeps a sequential transaction log so that its decisions to commit or abort will be durable. These decisions are based on whether all components are prepared, whether any component cannot be prepared, and whether a component remains prepared but is not committed or aborted.

2 TRANSACTION PROBLEMS

Previous research has addressed problems that arise during the distribution of transactions over a multi-DBMS while dealing with concurrency control and recovery. Most of these problems occur in the distributed databases [3]. These problems can be classified as follows:

1. Site failure: This action happens when one or more sites in a distributed database management system (DDBMS) fail. When a site failure occurs, it is the responsibility of the transaction manager to restore the transaction updates.
2. Network problems: This action occurs when the communication network fails because of a unclear transaction. This type of network failure indicates that one or more sites are cut off from the rest of the sites in the distributed database.
3. Data duplication: This problem occurs when the same multiple copies of data in the database cannot be adequately tracked so as to maintain consistency.
4. Distributed transactions: This problem arises when a distributed transaction happens in multiple sites.
5. Distributed deadlocks: During a transaction, a deadlock may occur in a single site or multiple sites.

- *Omar Baakeel is currently a Ph.D candidate at the College of Business and Public Management, University of La Verne, California, USA. E-mail: obaakeel@yahoo.com*
- *Abdulaziz Alrashidi is currently a Ph.D candidate at the College of Business and Public Administraation, University of La Verne, California, USA. E-mail: aalrashidi99@yahoo.com*

Ensuring the consistency of data also becomes a problem in the transaction management of a distributed environment when site/network failures occur. Various transaction protocols have been developed so as to provide a more stable transaction performance in a network. Moreover, different problems have been addressed in securing the transaction elements in a network.

3 Related Works

Because of the importance of database transactions, new transaction technologies must be developed and adopted. In their 2006 study, [4] developed a new transaction mechanism based on the use of agent systems to provide improved transfer reliability among different distributed algorithms. They developed a transaction algorithm for solving distributed constraint satisfaction problems (DCSPs) by dividing agents in the system into two groups, Variables' Agents and Controller Agents, thus allowing the reformulation of different inter-agent communication algorithms in their framework. The proposed model can also be used to treat non-binary constraints and to manage the transaction according to its original destination. The instantiation of these variables can be carried out by negotiation so as to separate the sub-problems into totally independent ones. Fig. 1 presents the proposed classification based on the agent systems [4].

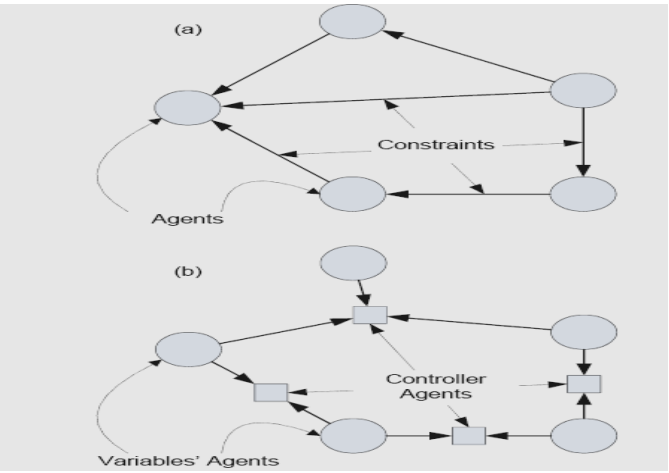

Fig. 1. Constraint network (a) with or (b) without controller agents [4].

In their 2007 study, Qiming and Umesh [5] described transaction issues that exist in e-commerce applications. They customized the computing environment for distributing a limit transaction through DBMSs among a large number of autonomous service requesters. These authors also predicted that the increasing automation of e-commerce applications will lead to the use of transaction agents that cooperate to perform business transaction activities. From the notion that peer-to-peer protocols based on inter-agent communication are required for cooperative transactions, they developed a cooperative multi-agent transaction model that includes peer-to-peer protocols for commit control and failure recovery. This model could help facilitate the agent tasks required for transferring data elements through channel queries. Fig. 2 shows the proposed model structure with the database components [5].

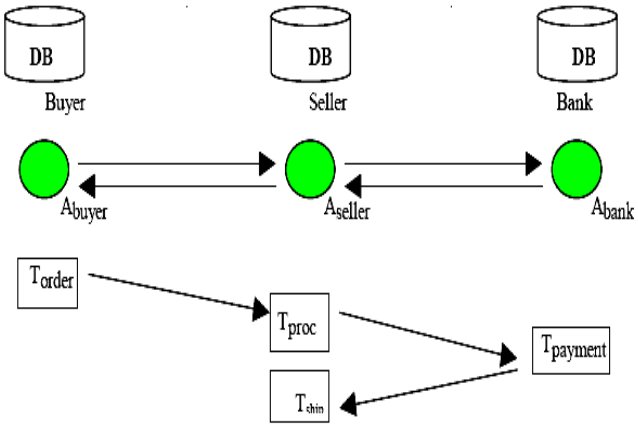

Fig. 2. The proposed model structure with the database components [5].

4 SQL Distributed Transaction Example

Current developments in distributed computing have generated new techniques for retrieving and representing data elements in a grid. Structured query language (SQL) is one such technique. It can perform a high level processing of client queries for a set of processes that seek to achieve some form of cooperation in the database. Moreover, executing a different conceptual representation can simplify the processes of a distributed system. This simplification occurs when some processes stop operating, for instance, as a result of crashing or being disconnected.

SQL server provides sustainable database processing. The first SQL statement begins the transaction. A transaction begins with the execution of one of two SQL statements: COMMIT statement or ROLLBACK statement. This transaction carries out a different meaning for the grid contents when it is committed or rolled back.

The next step is to extract the transaction concept by considering a certain procedure for carrying out a successful transaction along with the database elements. For example, when one or many customers transfer a certain amount of money to an account, it is directed to the savings account or the checking account. There are several important operations involved in carrying out that transaction, such as decreasing the amount of money in the savings account, adding other elements to the checking account, and recording the steps of the transaction by saving its actions.

SQL databases allow a huge number of transactions to be performed when all the SQL operations can be applied to maintain the accounts in proper balance. However, if there is a problem with the transaction (e.g., an invalid transaction number, software/hardware failure) that prevents the completion of one or two of the statements in the transaction, the transaction cannot be carried out. In this situation, the transaction is rolled back so that the correct balance is obtained in each account.

Fig. 3 describes the transaction mode in the SQL database, while Fig.4 presents the SQL transaction statement performance.

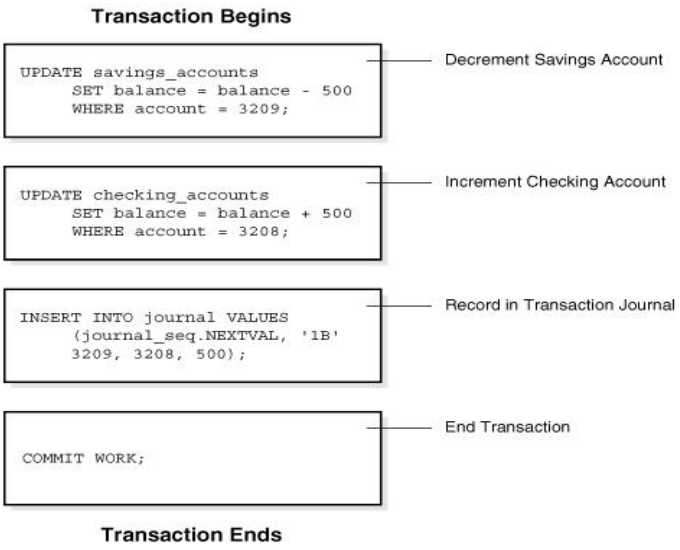

Fig. 3. The transaction mode in the SQL database.

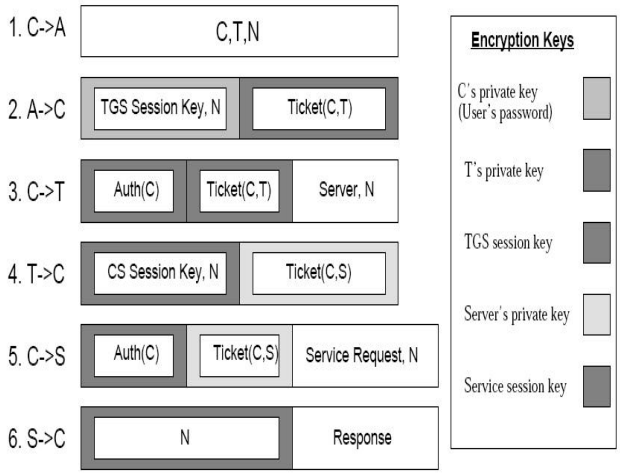

Fig. 4. SQL transaction statement performance.

Furthermore, the enhancement of SQL components, which enables the rapid processing of different statements among different transactions, has generated a lot of interest in the transfer of distributed systems to a wide area network (WAN) environment. This kind of transfer can be provided by Internet services or wireless technology.

It is important to develop new mechanisms to facilitate the distributed transaction process over DBMSs with specific requirements. Some recommendations for achieving a successful transaction over SQL database components include the following:

1. To allow a stable transaction for certain elements over SQL statements, SQL transaction statements must initially agree on the client request by classifying the transaction details into several points with common formats for reprocessing the transactions.
2. A successful data exchange can be easily achieved by using several alternative methods of analysis to determine the required actions for the unknown transactions in the grid.
3. Through the establishment of a collaborative transaction, a new distributed transaction based on the SQL statements for the cooperative processes can be started.
4. A collaborative transaction may require not only agreement on the requested actions but also execution of the transaction queries.

5 Proposed Technique

Based on the problems associated with distributed transaction applications that have been discussed in this paper so far, and based on previous findings of related works, we propose an exceptional distributed transaction model for processing transaction queries in multi-DBMSs. This model provides sustainable communication between the client and the server so as to perform the client queries in a more secure and reliable way. In addition, the proposed model might reduce the number of errors that occur during the transaction process, thus optimizing multiple transactions across servers. By adopting advanced network communication, this model reduces network failure, which occurs during transactions. Fig. 5 shows that distributed communication indirectly depends on the database query, which grants high privileges in the transactions.

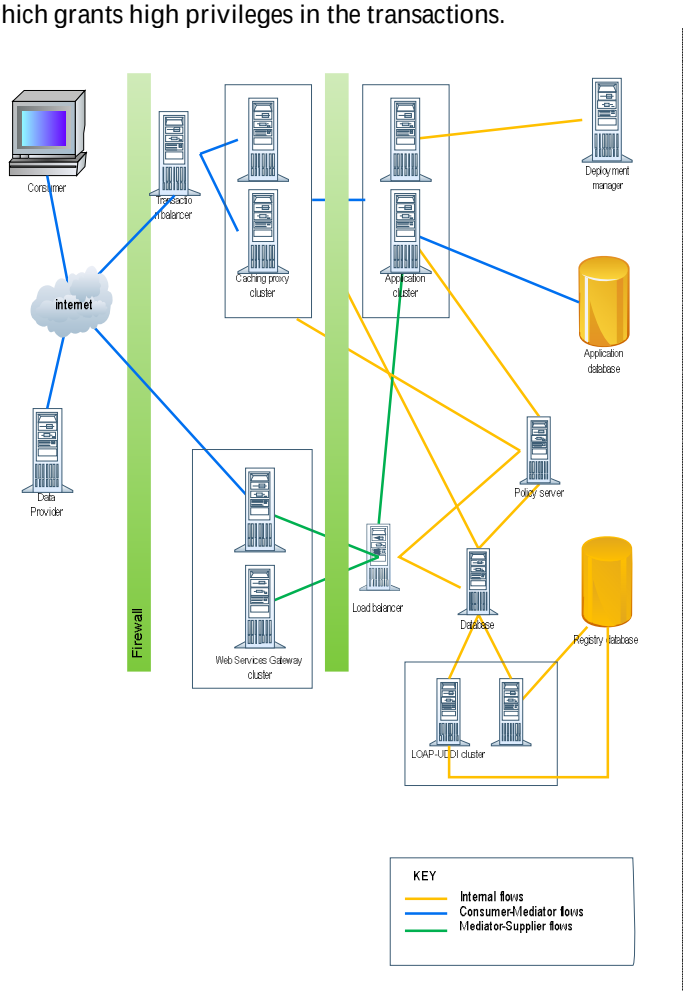

Fig. 5. Proposed distributed database model.

6 Expected Results

The proposed model is expected to produce the following re-

sults: 1) directly address transaction problems by looking into the history of transactions, 2) discover dependencies among services in historical transactions, 3) determine the execution time period data for multiple transactions among DBMSs, and 4) monitor instrumentation over multi-DBMSs.

7 CONCLUSION

This paper examines various issues that exist in database transactions among multi-DBMSs as well as the techniques that help in distributing a database in different fragments. Moreover, this study adopts a new way of resolving site and network failure problems that occur during a database transaction in a multi-DBMS. This paper proposed a new distributed transaction model for performing and managing transaction details more efficiently. In developing the proposed model, we have taken into consideration previous findings of related studies.

REFERENCES

[1] M. Gupta, A. Neogi, M. K. Agarwal, and G. Kar, "Discovering dynamic dependencies in enterprise environments for problem determination," In Proceedings of the 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, volume 2867 of Lecture Notes in Computer Science, pages 221–232. Springer, October 2003.

[2] C. Bessiere and I. Brito, "Asynchronous Backtracking without Adding Links: A New Member in the ABT Family," pp. 7-24, 2005.

[3] S. Al-Maqtari, H. Abdulrab, and A. Nosary, "Constraint Programming and Multi-Agent System mixing approach for agricultural Decision Support System," in Emergent Properties in Natural and Artificial Dynamical Systems, pp. 199-213, 2006.

[4] M. Sami, and A. Habib, "Controller-Agent based approach for Solving Distributed Constraint Problem," LITIS Lab. – INSA of Rouen – France, 2006.

[5] C. Qiming, and D. Umesh, "Multi-Agent Cooperative Transactions for E-Commerce," USA. Vol 2, pp, 33-37, 2007.