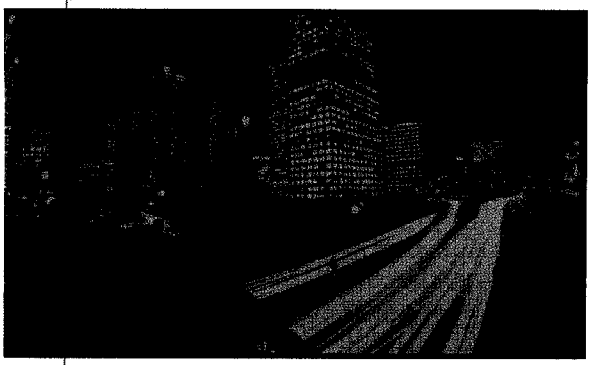
# Nomadic transaction management

## *Examining the ACID test requirements for the roaming computer*

Ravi A. Dirckze and Le Gruenwald

The distinguishing characteristic for nomadic computing is the wireless communication medium. This makes it possible for a fixed computer system to support a mobile user. The wireless communication medium is used to transfer data between the user to the fixed computer system—revolutionizing distributed computing.

In today's technology dominating and communication intensive business environment, wireless computing offers numerous possibilities. For example, a stockbroker can transact with the New York stock exchange while commuting on a commercial airline. An automobile insurance adjuster can provide a cost estimate for repairs from the site of the accident, itself. As such, nomadic computing expands the horizons of the computer realm.





In the static environment, many autonomous databases cooperate with other autonomous systems to provide extended services to users. For example, users can verify airline ticket reservations that involve multiple airlines. The nomadic user wants access to information available on the fixed system from anywhere at anytime. Thus, such cooperating database systems—referred to as multidatabase systems—need to extend their services to mobile users. Existing multidatabase systems will need to be tailored to support nomadic users. These systems are referred to as Nomadic MultiDataBase Systems (NMDBSs).

### The architecture

The general nomadic computing model has two distinct sets of entities: a fixed network system and a continuously changing set of mobile hosts (Fig. 1). The fixed networking system has a collection of static computers connected by a wired network. Some static units are able to communicate with mobile units through a wireless medium. These units are called base stations or Mobile Support Stations (MSS). The area covered by a MSS is called a cell. The (wireless) communication network includes the cellular architecture, the radio transmission over FM, the satellite services and the wireless Local Area Network (LAN).

Although wireless technology is fairly reliable, it is not as robust as the mediums used in the static systems. Thus, the nomadic user will experience frequent disconnections. The user will operate in many modes ranging from highly connected to disconnected. However, a characteristic of these modes of operation is that they are foreseeable.

The mobile hosts are portable computers that vary in size, processing power, memory, and so forth. The typical mobile computer will have limited resources compared to their desktop counterparts. These limitations include battery power, processing power, volatile memory, disk space and network bandwidth.

### The Nomadic Multidatabase environment

The general NMDBS discussed here will be a collection of autonomous databases connected to a fixed network. The respective database management systems retain complete control over their data. Each database may be viewed as an independent site in the network. These databases operate in different environments and may use different data models, data manipulation facilities, transaction management and concurrency control mechanisms, and so forth. Such a composition of connected autonomous database systems is called a multidatabase system. Thus, a NMDBS may be viewed as a multidatabase system that supports mobile users.

Users of the independent databases, called internal users, access these databases through their respective database management systems. The execution of local transactions submitted by these local users will be transparent to any global process. Users accessing more than one database, called external users, submit global transactions to the Nomadic MultiDataBase Management System (NMDBMS).

The NMDBMS is a set of software modules existing on the fixed network that cooperates with each other. Together they project the illusion of a single database to the external user. A global transaction consists of a set of sub-transactions that need to be executed at different sites. The Global Transaction Manager (GTM), a software component of the NMDBMS, manages the executions of the global transactions.

Global transactions are allowed only limited access to the individual databases. For example, external users are allowed to make reservations on a commercial airline database system. However, they are not allowed to execute adhoc queries that could compromise sensitive information. Thus, each database provides a service interface that specifies the operations accepted and the services provided to external users. The GTM cooperates with the respective service interfaces in order to execute global transactions.


0278-6648/98/$10.00 © 1998 IEEE

As the databases are autonomous, each local DataBase Management System (DBMS) is responsible for the correctness of all local and global sub-transactions executed within its domain. For this same reason, we may assume that no integrity constraints exist over data items at different sites as this would violate local autonomy.

## Responsibilities of the GTM

In DBMSs, a transaction is defined as a basic independent unit of consistent and reliable computing. A properly formed transaction has a begin operation followed by a sequence of read/write operations and is terminated by a commit or abort operation. The transaction management process must coordinate these operations' execution in order to guarantee consistent and reliable computing. Thus, the primary responsibility of the transaction management process is to provide consistent and reliable access to data within its domain.

Consistency and reliability can be achieved by enforcing the Atomicity, Consistency, Isolation and Durability (ACID) properties. *Atomicity* requires that either all operations of a transaction execute successfully or none at all. *Consistency* requires that the database be left in a consistent state after the execution of a transaction. *Isolation* requires that the effects of each transaction be isolated from other transactions executing concurrently. *Durability* requires all alterations made by a successful transaction be permanently reflected in the database.

Together, the consistency and isolation properties guarantee a transaction to be a consistent unit of computing; the atomicity and durability properties guarantee a transaction to be a reliable unit of computing. Thus, the ACID properties ensure that the database is in a consistent state under concurrent access even when failures occur.

Each local transaction management process is responsible for the consistency and reliability of all local transactions and global sub-transactions executed at its site. As nomadic users communicate with only the NMDBMS, the GTM provides the additional services required to support them. Thus, only transaction management issues that affect the GTM will be discussed.

### Acronyms

ACID—atomicity, consistency, isolation and durability
GTM—global transaction manager
LAN—local area network
MSS—mobile support stations
NMDBS—nomadic multidatabase system
NMDBMS—nomadic multidatabase management system
DBMS—database management system

## Management issues

There are three primary features that distinguish nomadic users from static users. They are 1) the frequency of disconnections, 2) the relocation of the user and 3) the relatively poor resources compared to their static counterparts. Despite these challenges, nomadic users will expect to execute the same global transactions they perform on static systems.

In the static environment, users establish a connection with the NMDBMS at some site which is maintained until the end of the session. If the connection is broken prior to this, any uncompleted transaction is terminated by the GTM.

The nomadic user cannot maintain a continuous connection with the same GTM process throughout a session. In the new environment, disconnections cannot be treated as failures that result in terminated transactions. Further, when a user migrates to a new location subsequent operations will be submitted from the new site. The execution time is also affected by mobility. A transaction may be interrupted by frequent disconnections followed by long periods of inactivity prior to reconnection. This will prolong the execution time. Next let us look at the effects of these issues upon the GTM with respect to each of the ACID properties.

## Atomicity

For global transactions, atomicity requires the transaction to succeed at all sites or be aborted at all sites. In the multidatabase environment, the sites retain complete control over their databases. Thus, they retain the right to abort a transaction at anytime prior to a successful commit operation. Some literature suggests that the local sites export a prepare-to-commit operation after which they relinquish the right to abort a transaction. This information can be used by the GTM to provide an atomic commit protocol at the global application level.

However, the prepare-to-commit operation may force sites to hold resources (e.g., locks) for an unspecified period of time. One may argue that mechanisms such as time-outs could be used to release such resources. If so, it can be counter argued that the unilateral abort would have been supported in the first place.

Enforcing atomic commits is further complicated by the characteristics of the nomadic environment. First, the GTM needs to ensure that migrating transactions are managed as single units; otherwise, atomicity cannot be guaranteed. Migrating transactions also affect the commit protocol. Knowledge concerning the constituent operations are now distributed among different sites. Thus, the GTM needs to keep track of migrating transactions and the sites at which they are executed.

The limited resources and the frequency of disconnections necessitate a quick response to user requests. Thus, the commit protocol needs to be tailored to meet the new performance requirements.

## Consistency

In the multidatabase environment, the GTM does not have to enforce the consistency property. This is because the databases are autonomous. Therefore, they have no constraints defined on data items residing at different databases. Each local transaction management process ensures that the operations executed at its site do not violate any overall defined constraints. Therefore, all successful global transactions will, by default, meet the consistency property. Thus, the GTM is relieved of this responsibility.

## Isolation

Serializability is the most widely accepted correctness criterion for ensuring the isolation property. Serializability requires that the effects of executing a set of transactions concurrently be equivalent to some serial execution of the same set of transactions. Optimistic, concurrency control algorithms based on serializability will reject consistent transactions whenever a possibility of a conflict exists. Pessimistic concurrency control algorithms based on serializability will prevent consistent transactions from executing whenever a potential conflict exists. Thus, global serializability—serializability of all global and local transactions executed at all sites—does not work well in the multidatabase environment.

This is largely due to two reasons. First, serializability was originally introduced for centralized database environments and, therefore, is centralized by nature. Second, as the databases are

autonomous, the GTM is not aware of local transactions executed by the DBMSs. Thus, suitable algorithms need to be designed to meet the requirements of the multidatabase environment.

Once again, the characteristics of the nomadic environment further complicate the issue. As mentioned before, mobility prolongs the execution time of a transaction. As the execution time increases, the possibility of conflicting with other concurrent transactions increases as well. Therefore, if optimistic concurrency control algorithms are used, the probability of conflicts with other transactions will increase. On the other hand, if pessimistic concurrency control algorithms are used, concurrency will be restricted.

The limited resources will also have an effect on the isolation property. The isolation property needs to be guaranteed before the commit protocol is executed. Thus, for the GTM to provide a timely response, the concurrency control algorithm needs to provide a timely response to the GTM. This is an additional issue that needs to be addressed by the concurrency control algorithm.

Again, the GTM must track the migrating sub-transactions and the sites where they are executed. This information is required to enforce the isolation property.

## Durability

Durability requires that the values changed by a successful transaction must persist in the database. In the multidatabase environment, the durability property cannot be directly implemented by the global application. This is because it cannot change or restore the values of data items within the local databases. Thus, it needs to rely on the durability property of the local DBMSs.

The global application will be relieved of enforcing this property if an atomic commit operation is available. (It then can rely on the durability property of the local DBMSs.) However, as discussed previously, such an operation is not currently available.

In the mobile environment, disconnections cannot always be treated as failures that must result in aborting unfinished transactions. Whenever the user disconnects, the GTM must determine the status of the user connection before taking drastic action. Yet, due to the nature of the environment, the GTM is likely to make erroneous decisions. (Say, a user connection is deemed to have failed and reconnection is not expected.) To minimize the ill-effects of such erroneous decisions, the transaction should not be aborted until its resources are required by another transaction. Thus, the system needs to define a new "suspended" state to address this issue.

Also, due to disconnections, one cannot guarantee that all responses will be delivered to the user. Any response that cannot be delivered prior to a disconnection needs to be logged by the GTM and delivered upon reconnection.

## Concluding remarks

Limited resources make it necessary to look for more efficient concurrency control algorithms and commit protocols. The frequent disconnections and the migration of the user raise additional issues that complicate the atomicity, isolation and durability properties. These issues are compounded by the restrictions that apply to multidatabase systems. That is, constituent database systems cannot be modified.

Finally, can mobility be addressed entirely within the network layers? This would make it transparent to all applications residing on the static network. The advantage is that existing applications would be able to support mobile users without any modifications. However, this approach is not an effective solution for many reasons.

First, disconnections and migration affect global transactions: i.e., prolong its execution time. If mobility is transparent to the NMDBS, such issues cannot be addressed by the GTM. Second, the NMDBMS cannot be sensitive to the resource limitations of the mobile users. Third, to maintain this virtual connection, additional network traffic generated—along with all communications— will need to be forwarded to the initial site. This is required no matter where the data being accessed is located. Thus, mobility needs to be visible to the NMDBS in order to provide an effective solution.

## Read more about it


• Alonso, R. and Korth, H. F., "Database System Issues in Nomadic Computing," *SIGMOD Record*, May 1993.

• Ben-Hassen, S. and Rusinkiewicz, M., "On Serializability of Distributed Nested Transactions," *Proc. 12th International Conference on Distributed Computing Systems*, Japan, 1992.

• Breitbart, Y., Garcia-Molina, H. and Silberschatz, A., "Overview of Multidatabase Transaction Management," TR92-21, University of Texas at Austin, 1992.

• Du, W. and Elmagarmid, A. K., "Quasi Serializability: A Correctness Criterion for Global concurrency control in InterBase," *Proceedings of the 15th International Conference on VLDB*, Amsterdam, The Netherlands, August 1989.

• Dunham, M. H. and Helal, A., "Mobile Computing and Databases: Anything New?" *SIGMOD Record*, Vol. 24, No. 4, December 1995.

• Imielinski, T. and Badrinath, R. B., "Wireless Computing: Challenges in Data Management," *Communications of the ACM*, Vol. 37, No. 10, October 1994.

• Mehrotz, S., Rastogi, R., Silberschatz, A. and Korth H. F., "A Transaction Model for Multidatabase Systems," *Proc. 12th International Conference on Distributed Computing Systems*, Japan, 1992.

• Noble, B. D. and Satyanarayanan, M., "A Research Status Report on Adaptation for Mobile Data Access," *SIGMOD Record*, Vol. 24, No. 4, Dec. 1995.

• Ozsu, M. T. and Valduriez, P., *Principles of Distributed Database Systems*, Prentice Hall, Englewood Cliffs, NJ, '91.

• Pitoura, E. and Bhargava, B., "Dealing with Mobility: Issues and Research Challenges," *Technical Report CSD-TR93-070*, Purdue University, 1993.

• Pitoura, E. and Bhargava, B., "Revising Transaction Concepts for Mobile Computing," *Proceedings of the IEEE Workshop on Mobile Systems and Applications*, Santa Cruz, CA, Dec. 1994.

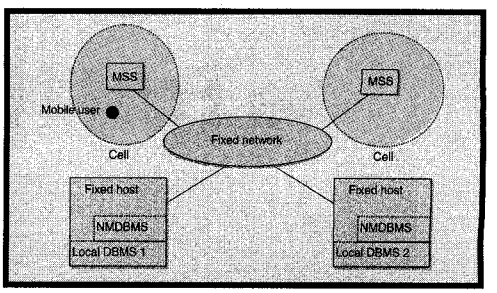




*Fig. 1 Nomadic multidatabase architecture*

## About the authors


Ravi A. Dirckze is a graduate student at the University of Oklahoma working towards his doctoral degree in Computer Science.

Dr. Le Gruenwald is an Associate Professor in the School of Computer Science at the University of Oklahoma. Her research areas include real-time Main Memory databases, Distributed and Mobile databases, Object Oriented databases and Multimedia databases.