Transaction Management in Mobile Databases

Ziyad Tariq Abdul-Mehdi
Multimedia University, Malaysia

Ali Bin Mamat
Universiti Putra Malaysia, Malaysia

Hamidah Ibrahim
Universiti Putra Malaysia, Malaysia

Mustafa M. Dirs
College University Technology Tun Hussein Onn, Malaysia

INTRODUCTION

Recent advances in wireless communications and computer technology have provided users the opportunity to access information and services regardless of their physical location or movement behavior. In the context of database applications, these mobile users should have the ability to both query and update public, private, and corporate databases. The main goal of mobile software research is to provide as much functionality of network computing as possible within the limits of the mobile computer's capabilities. Consequently, transaction processing and efficient update techniques for mobile and disconnected operations have been very popular. In this article, we present the main architecture of mobile transactions and the characteristics with a database perspective. Some of the extensive transaction models and transaction processing for mobile computing are discussed with their underlying assumptions. A brief comparison of the models is also included

TRANSACTION MANAGEMENT IN MOBILE DATABASES

A mobile database system is a special multi-database system on a mobile computing environment. It allows mobile hosts to access and manipulate data stored on several per-existing, autonomous, and heterogeneous local database systems located on different parts of the wired network. Transactions in a mobile database system may access data from several local databases at different sites. Management of these transactions requires different approaches in mobile databases than in a multi-database. This is mainly due to the fact that a mobile host is not suitable to manage a global transaction by itself due to the described nature of the mobile computing environment. Usually this management is done by the mobile host's base station or by coordination of it.

Due to the described nature of the mobile computing environments, transaction management has to be reevaluated for mobile databases. The transactions in mobile computing environments are usually long-living transactions, possibly covering one or more disconnected durations. Supporting disconnected operation (i.e., allowing a mobile host to operate autonomously during disconnection) raises issues in consistency. Providing disconnected operation also requires some pre-caching of data that will be required for the necessary operations to be performed during disconnection. The moving behavior of the transactions in mobile computing environments also requires new mechanisms. As a mobile host moves from a cell to another cell, its transactions might need to migrate from one base station to another. In general, transactions in mobile databases require relaxed ACID properties. There are several works on mobile transactions, each addressing some of the issues in mobile transaction management. We will explain some of them in the following sections.

Kangaroo Transactions

Kangaroo transactions (KTs) are introduced in Dunham and Helal (1997). As the name suggests, this model mostly addresses the moving behavior of the mobile transactions. As the transactions hop from one site to another, the management of the transaction also moves.

In addition to the mobile computing environment we have described, these systems introduce a couple of other terminologies. The term *source system* represents a collection of systems that offer information services to mobile users. These systems could be any type of system that exists in the mobile computing environment. One good example is a distributed database system. The term *data access agent (DAA)* represents an agent that is hosted by each base station. Mobile hosts reach data in source systems by sending their transactions to DAAs. When a handoff occurs, the DAA at the new base station receives the transaction information from the DAA in the old base station. A *mobile transaction* is defined as the basic unit of computation in the mobile environment. The management of a mobile transaction might hop through different base stations, which are not known until it completes its execution. DAAs at base stations are responsible for management of the mobile transactions. One part of the DAA responsible for the management of the transactions is called the *mobile transaction manager (MTM)*. The main responsibilities of an MTM are maintaining the status of mobile transactions in execution, logging recovery information, and performing needed check pointing.

In this model it is assumed that a mobile transaction issued by a mobile host to a DAA might include several subtransactions that require access to data at several global database systems (GDBSs) and DBMSs residing at different places of the fixed network. As a result DAAs serve as a mobile transaction manager built on top of GDBSs and DBMSs. DAAs also keep log information about the mobile transaction parts that have executed on them. Remember that a mobile transaction changes its DAA as it moves from one cell to another.

Since mobile transactions are long lived and include possible disconnected durations; the atomicity of a mobile transaction in this model is not always guaranteed. The time between an interruption of a transaction and its resume could be quite long. As a result, it is valuable to commit early on some portions of a mobile transaction, while breaking the atomicity property of the transactions. These early commits enable the release of possible important resources, instead of holding them for a long time.

A mobile transaction in this model, which is called a *kangaroo* transaction, is an extension to global transactions (GTs). Figure 1 shows a global transaction, which consists of subtransactions called local transactions (LTs). Each local transaction is assumed to be issued to a DMBS.

A kangaroo transaction can be composed of both GTs and LTs. The mixture of GTs and LTs are grouped under a transaction type called Joey transactions (JTs) based on the DAA on which they have initiated. Figure 2 shows an example kangaroo transaction.

Each JT represents the unit of execution at one base station. When a mobile host makes a transaction request to the DAA on its associated base station, a KT is formed. In addition to that, a JT is formed for managing subtransactions that originate from the mobile host when the KT is under the control of the first DAA. When a mobile host hops from one cell to another, the control of its KTs changes to the new DAA on the new base station. The new DAA creates a new JT for handling the future subtransactions the mobile host might request to this DAA. The old JT is committed independently from the new one. Note that this breaks atomicity. To enable a KT to be completely undone, previously committed transactions should be compensated.

Kangaroo transactions have two different modes of execution. The first mode is called *compensating mode* where a JT falls and all KT is undone. However, this mode of operation requires compensating transactions for undoing operations of the previous JTs, since they are independently committed building compensating transactions requiring input from the user. As a result this mode is rarely used. Note that this mode tries to preserve atomicity of the KT, which breaks the durability of the subtransactions. The second mode, which is the default mode, is called the *split mode.* In this mode of operation, when a JT fails, no new JTs are created, but the previously committed Ms are also not undone. This mode breaks the atomicity. None of the modes ensures serializability.

Clustered Model

In Pitoura and Bhargava (1995), the sites are grouped in clusters if they are connected by strong network links or they are pre-grouped in clusters. In a cluster, full consistency is always enforced, however over the clusters a bounded inconsistency is permitted. A site can be a member of a cluster or leave one dynamically based oil the network conditions, for example, a mobile host forms a cluster itself when it is disconnected.

Mobile transactions are grouped into two types: strict and weak transactions. Weak transactions access data in the same cluster, and they have two commit points: cluster and global. Global commit can only be made after clusters merge. Strict transaction can only access data that is ac-

Figure 1.

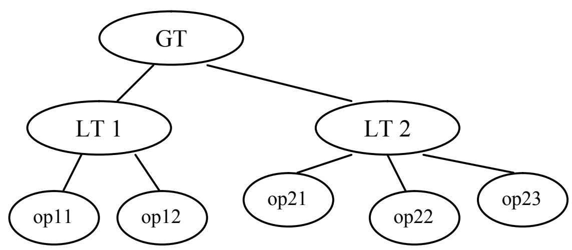

Figure 2.