データベースシステム 116－32
（１９９８．７．９）

A Locking Protocol for Location Databases in Mobile Environment

Budiarto Kaname HARUMOTO Masahiko TSUKAMOTO Shojiro NISHIO

Department of Information Systems Engineering
Graduate School of Engineering, Osaka University
2-1 Yamadaoka, Suita 565-0871, Japan
E-mail: {budiarto,harumoto,tuka,nishio}@ise.eng.osaka-u.ac.jp

Abstract The mobile computing paradigm introduces new technical issues in the area of database systems. In mobile environment, for an efficient tracking of user movements, the users' location data are stored in location databases. As the location of user is a frequently changing piece of data, queries to location databases may get inconsistent results. In this paper, we discuss the problem of processing aggregate queries on the location databases and propose the two step position locking protocol that can guarantee the queries to get the consistent results.

モーバイル環境における位置管理データベースのための施錠方式

Budiarto 春本 要 塚本 昌彦 西尾 章治郎

大阪大学大学院工学研究科情報システム工学専攻
〒 565-0871 吹田市山田丘 2-1
E-mail: {budiarto,harumoto,tuka,nishio}@ise.eng.osaka-u.ac.jp

あらまし モーバイルコンピューティングはデータベースシステム分野に新たな研究課題をもたらしている．ユーザの移動とともにシステムは新しい環境に適応しなければならない．特にモーバイルコンピューティング環境ではユーザの位置データは頻繁に更新される．よって，集約問合せなどで位置データが対象になった場合，その問合せに対する結果データの一貫性に問題が起こる可能性がある．本稿ではこれらの問題を避けるために２段階位置施錠方式を提案する．この施錠方式を用いることにより，位置に関する集約問合せは一貫した結果データを得ることができる．

1 Introduction

The mobile computing paradigm introduces new technical issues in the area of system software, including the database systems[1]. For example, many techniques for traditional distributed database management have been based on the assumption that the location of hosts and connections among them in the distributed system do not change[11]. However, in mobile computing environment (from now, we will refer to this term simply as *mobile environment*), these assumptions are no longer valid.

Mobility of hosts engenders a new kind of locality that migrates as hosts move[12]. As a consequence, the existing solutions for traditional distributed database management may not be readily applicable to the mobile environment. In particular, a measure to manage the location of mobile host must be taken, so that users do not need aware on their mobility while using applications on a mobile environment.

Location management is one of the most important mobile computing technologies. Integrating location management into mobile environment will provide location transparency to its users. Users can always be identified regardless of their locations in the environment.

Since it is possible to track the user location in mobile environment, we can think of using this facility to have a knowledge on user distribution at a particular location in the environment. In our real life, there are many applications that can make use of such a knowledge. The dynamic resource allocation possibly becomes the most important kind of the application. Examples include the adaptive allocation of communication channel capacity and the adaptive traffic light control.

Getting the information of user distribution at a particular location will involve querying and aggregating the location information. Such a process can be done in some ways, for example, by multicasting the paging message or by letting the location management do it for us. However, since users' location is a frequently changing piece of data, the information may probably contain errors.

Not all of applications will reject to work with this erroneous information. *Sampling*, for example, can be used for getting a sufficiently accurate information from some erroneous data. In this situation, the important thing is how we can limit the errors to the minimum level. In this paper, we discuss the technique for processing aggregate queries on location data and propose a technique called *two step locking protocol* that can be used to improve the accuracy of the query result.

This paper is organized as follows. In section 2, a generalized and simplified model of location management is presented. After that, the technical aspects of processing aggregate queries on location data is presented in section 3. The concurrency control for location databases which becomes the main contribution of this paper is discussed in section 4. After discussing the related work in section 5, finally, we provide some concluding remarks in section 6 .

2 The Model of Location Management in Mobile Environment

The mobile environment consists of mobile hosts and fixed hosts. Each mobile user (user that is using the mobile host) in the environment is assigned with a unique ID called *UID*. The UID of particular user is fixed and does not changed even if the user moves. Further, since in general a mobile host is used and possibly owned by a single user, for the simplicity, we will use the terms mobile user and mobile host interchangeably.

In mobile environment, the whole geographic area is partitioned into *cells*. Each cell is assigned with a unique ID called *CID*. Each cell is covered by a fixed host called *mobile support station* (MSS). Augmented with wireless communication capability, MSS provides mobile users in the cell a gateway to connect to the entire network.

Every cell is attached to a *location server* (LS). The LS supports *query location* and *move* operations. In general, an LS covers some cells which is located closely (from the view of location or network) each other. Each LS has a *location database* for storing the location data of users which exist inside its coverage. The entry that describes a users' location is a tuple composed by two attributes, i.e., CID and UID. In general, the location database is indexed to improve the data lookup performance. The most logical scheme is probably to make CIDs as its key values. Therefore, logically, the location database forms a tree as shown in Figure 1.

LSs in the mobile environment are connected each other by a dedicated network. In general, this network forms a hierarchy where messages related to the location management are exchanged among LSs. The depth of the hierarchy varies according to the size of cells

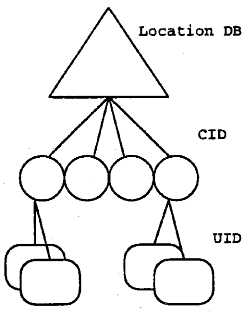

Figure 1: A hierarchical structure of data.

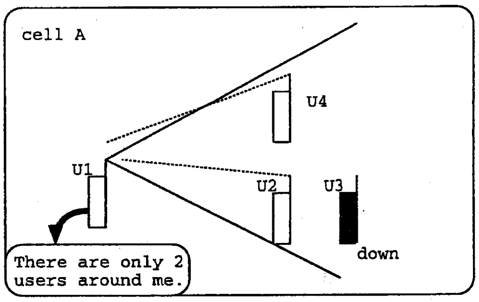

Figure 2: An unidentified user.

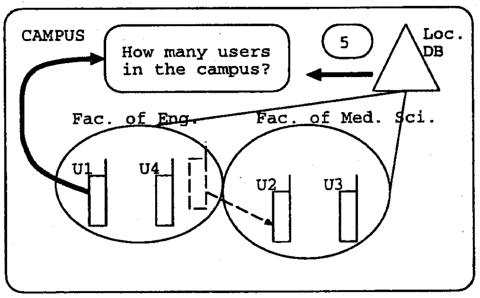

Figure 3: Identified more than once.

and network bandwidth. Generally, the smaller the size of cells is, the deeper the LS hierarchy.

Being connected, all location databases in the mobile environment forms a distributed database environment. For implementing the query location and move operations on an LS, the location database provides *read*, *insert*, and *delete* operations of data item. In particular, a query location operation is translated into a read operation on the corresponding key values in the location database. On the other hand, a move operation is translated into deletion of an entry, which is composed of the UID of user that moves and the CID of previous cell, and then, it is followed by insertion of a new entry, which is composed of the UID of the user and the CID of newly entered cell, into the location database.

3 Processing Aggregate Queries on Location Data

In mobile environment, processing the aggregate query involves collecting the information about all users reside in the area being queried. When the location becomes the subject of an aggregate query, the characteristics of mobile environment, including the fast changing nature of the location data, may introduce some errors into the query result, such as in the following examples.

Example 1 (Unidentified users)
Suppose that a mobile user A is about to query the number of mobile users other than itself, which are located in the cell where this user is located now (see Figure 2). For this purpose, A throws a paging message into the *common link* (like the command *rusers* of UNIX) and by simply relying on response packets issued by other mobile users, A is expected to know the number of all mobile users other than A itself. However, this can be true only if the message issued by A is received by all users in the cell and all users can give response to this message. Some users probably do not receive the message due to errors caused by the echoing effect (note that the message is propagated on the air) of many skyscrapers around. The problem could also occur if some mobile hosts are down due to the power expiration, so that they cannot give response to the message.

Example 2 (Identified more than once)

In Figure 3, suppose that a mobile user U1 is about to query the number of students in some parts of the Campus. A student U2 is in the Faculty of Engineering when the location management begins to process the query, thus its location becomes the subject of this query. In the middle of query processing, after U2 had its location data evaluated, he/she moves into the Faculty of Medical Science. Due to administrative reasons, it is located in different cell from the Faculty of Engineering. Unfortunately, when U2 moves into Faculty of Medical Science, the query processing has still been collecting the data, therefore the location data of U2 will be evaluated again though it had been done when U2 was in Faculty of Engineering.

Above examples show how prone is the aggregate query to get a wrong result when it deals with the location of mobile users. The situation will be worse in the future, where a high density of mobile users moving across very small cells.

Besides the errors caused by the physical nature of mobile environment, the problem could also occur due to uncontrolled accesses to the location database, such as in Example 2. In essence, the user that is querying the location data can be regarded as reading the data. On the other hand, the mobile user can be considered as updating its location data as it moves crossover the cell boundary. These two kinds of accesses conflict, but unfortunately, in Example 2, we allow these conflicting accesses to be executed simultaneously. This makes the consistency of location database broken from the view of the query.

4 Concurrency Control for Location Databases

Allowing simultaneous accesses to the database will result in performance gain of the entire database system. However, if the simultaneous accesses are allowed in uncontrolled fashion, they may break the consistency of database. Concurrency control is the important mechanism of a database system that allows simultaneous accesses to the database and protects the database consistency from possible anomalies.

To perform control on simultaneous accesses, the accesses are grouped into an abstract unit called the *transaction*. The level of grouping is not critical as long as it satisfies the *ACID* properties. In our model of location management, for example, the move operation should be considered as a transaction. The concurrency control then determines the scheduling of the transactions execution so that it satisfies the correctness criteria, such as the *serializability* property.

As the location data is stored in the location databases, every user in the mobile environment relies on the location databases for tracking its location and those of other users. Due to its important role in the location management, the performance of location database systems will determine the performance of the entire location management system. Thus, the performance of concurrency control is among the most critical issues of location database systems.

Considering the concurrency control performance of location database system, it is important to employ the concurrency control mechanism for the location database that is bearable in the update intensive accesses. Further, since the location databases will also be involved in the query processing, the non-blocking property is also among the important requirements.

4.1 The (One Step) Locking Protocol

The most common concurrency control mechanism employed by database systems is probably the (two-phase) locking. As compared to other concurrency control mechanisms, such as those are using conflict graphs associated with read/write database accesses[7], the locking protocol is easier to implement.

In a locking protocol, two locking modes, i.e., R and W, are used. A transaction has to lock the data item in R (W) mode before it performs read (update, i.e., delete or insert) operation on the data item. If the data item

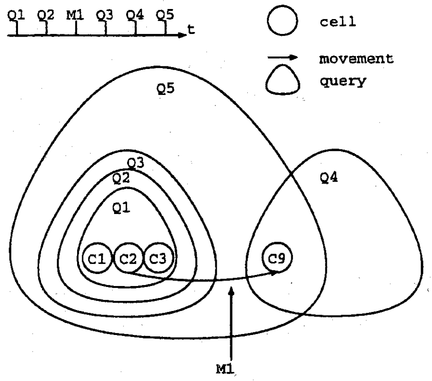

Figure 4: A blocked move.

is locked by more than one transaction, the locks conflict if at least one of them is W mode. Otherwise, they are compatible.

Before a new locking request on a data item is granted, the concurrency control system will test its compatibility with the locks being set on the data item. If it is compatible, then the request will be granted immediately. Otherwise, it must wait until the conflicting locks are released (i.e., the accesses represented by the conflicting locks are finished).

Since the locking protocols enforce *wait* when conflicts are detected, a transaction that conflicts with the transaction being executed will be blocked. Further, if the conflicting transactions wait each other releasing the locks set on the data to be accessed, a deadlock occurs. The fact that blocking and deadlock may occur, unfortunately, makes the existing locking inappropriate to be employed in location database systems. An example below shows a situation where the locking performs bad.

Example 3 (A blocked move)
Suppose that 5 aggregate queries (Q1 ... Q5) are submitted to the location server for processing (see Figure 4). After the beginning of processing Q2, a mobile user relocates from cell C2 to C9. since the entry of this user is currently being locked in R mode by Q1 and Q2, the entry cannot be deleted, thus, the move operation M1 is blocked waiting for Q1 and Q2 to complete (note that the move operation must be atomic). In the middle of processing Q1 and Q2, queries Q3, Q4, and Q5 are submitted consecutively and all of the queries will lock in R mode all the entries of users existing in their query areas. As a result, M1 will still be blocked until Q5 completed. On the other hand, Q4 will get 0 as the result even though one user that has moved from C2 exists in its query area.

4.2 The Two Step Position Locking Protocol

We propose the two step position locking protocol to overcome the problems of using one step locking protocols in location databases. Especially, it is dedicated to make the movement of mobile users reflected into the location databases as soon as possible while guaranteeing the aggregate queries getting the correct results.

Besides R and W, the two step position locking uses two additional locking modes, i.e., VR and VW (the conflict relations are not symmetrical, see Table 1 for the compatibilities among locking modes). In general, locking is performed in two steps. In the first step (the *volition step*), the data item to be read (deleted) is locked in VR (VW) mode. The purpose of this step is only for indicating that in the near future, *if possible*, the data item is about to be read (deleted) by the transaction that is setting the VR (VW) lock.

After a transaction completely setting volition locks on all data to be accessed, the transaction enter the second step (the *execution step*) (i.e., the volition step and the execution step are two-phase). Entering the execution step, the transaction will try to *upgrade* the volition lock modes into the execution lock modes. That is, the VR (VW) lock will be upgraded to R (W) lock. If the upgrade can be done successfully, then the transaction will execute read (delete) operation on the data item.

As the volition lock is only showing the volition to access the data item, it does not prevent other transactions, that have succeeded to set the R (W) lock prior to the setting of the volition lock, from reading (deleting) the data item.

Once a VW or W lock is set on a data item, no further locking request will be granted on the data item, since the data item will be obsoleted soon (will be explained later). The rejection of locking request, however, is not fatal. The transaction can just continue to run and perform accesses on other data items.

Since W lock conflicts with R lock, the VW lock is *non-upgradeable* while R locks have not been released and it must wait. As the VW lock becoming *upgradeable* (all R locks have been released), it will be upgraded to W lock immediately. If a transaction has succeeded to upgrade its VW lock on a data item to the W lock, then all volition locks being set on the data item become *definitely non-upgradeable.* Thus, the corresponding future accesses to the data item will be *impossible.* In this situation, the transactions, that have set the non-upgradeable volition locks, have to *quit* from accessing the data item, but may continue to run (to perform accesses on other data items).

So far, we have discussed only the read and delete operations inside the two step position locking protocol. So, how about the insert operation and its relation to the delete operation in the two step locking protocol?

In the one step locking protocols, insert operation is considered as an update. Unfortunately, this causes a blocking which may continue to a deadlock, such as in Example 3. In the two step position locking protocol, we allow an insert operation to be done without locking prior to the insertion. Further, the inserted data item will *inherit* all the VR locks being set on all data items below its current key value, except those are set by transactions that have set the same VR locks on its obsolete data item and already upgraded the locks to R modes.

In the one step locking protocols, the insert operation is executed only after the delete operation is executed successfully. In the two step position locking protocol, however, we do not need to do this. Instead, as the transaction *successfully* set the VW lock on the data item to be deleted, the insertion operation can be executed immediately. The successfully criterion here means that the transaction set the VW lock and return a *list* of transactions that have upgraded their VR locks on the data item into R locks. This list, then, will be used in calculating the VR locking inheritance mentioned above.

Table 1: Locking mode compatibilities in the two step position locking protocol.

	Request			
	VR	R	VW	W
VR	√	√	√	√
R	√	√	√	×
VW	–	–	–	–
W	–	–	–	–

√ : compatible, × : wait, – : quit

To see the effectiveness of two step position locking protocol, we will demonstrate it on the scenario in Example 3. Firstly, Q1 and Q2 can successfully lock all entries below key values C1, C2, and C3 in VR mode, upgrade them to R mode, read them, and complete. for M1, there are 3 groups of possible cases.

- If both of Q1 and Q2 have completed, M1 can simply lock the moving users' entry below the key value C2 in VW mode, insert a new entry for the user below the key C9, upgrade VW mode to W mode, and finally, delete the obsolete entry below the key value C2.
- If the VR locks of Q1 and Q2 on the moving users' entry below the key value C2 have not been upgraded to R yet, M1 can simply, lock it in VW mode, insert a new entry for the user below the key value C9, upgrade the VW locks to W mode, and finally, delete the obsolete entry below the key value C2. Q1 and Q2, then, will never read the obsolete entry.
- If either Q1, Q2, or both of them have upgraded VR locks to the R mode, M1

can just set the VW lock, insert a new entry of the moving user below the key value C9, and wait. Therefore, only who have upgraded their VR locks on the obsolete entry to R mode will read the entry. After all R locks are released, M1 can upgrade its VW lock to W mode, and finally, delete the obsolete entry.

Q4 and Q5 will see only the new entry below the key value C9, since the obsolete entry below the key value C2 has been either deleted or set with VW lock by M1. Thus Q4 will get 1 as the result.

From the demonstration of the two step position locking protocol, we can see that only in the group case 3 (users go away from the query area immediately after R locks are set on their entries) the query will get the obsolete result. We found no solution to this problem. However, we can reduce the possibility of such problems to occur by inserting a longer time interval between setting the VR locks to upgrading them to R modes.

5 Related Work

A highly mobile and distributed environment creates new scenarios for query processing. Especially, processing aggregate queries on the transient data, such as user location, becomes an interesting research problem on the very basic, semantic level[10]. The aggregate query itself has a wide range of applications, especially, to help in dynamic allocation of resources.

Research on the location management of mobile environment was mainly focused on minimizing the cost of location update. For that purpose, the hierarchy structure becomes the common approach taken in the location management[2, 3, 4, 6, 8, 13, 14]. Besides assuming the hierarchy structure of location servers, we also assume a 3-level hierarchical data structure, i.e., the location DB in the highest level, CIDs in the middle, and UIDs in the lowest level.

In this work, we let more than one version of data item (UID) to exist. Due to such a reason, our work can be regarded as a multiversion protocol[7]. Further, since it has two types of transactions, i.e., the read-only transaction (the query operation) and the update transaction (the move operation), it is categorized under the mixed multiversion schemes. In general, the mixed multiversion scheme allows the read-only transaction to read the old version. In our work, however, we give a higher priority for the new version to be read, by rejecting any locking request on the data item being locked in VW mode.

The volition locks (VR and VW) reminiscent the *intention* locks (IR and IW) in the DAG locking protocol[9]. However, they are different in the fact that the conflict relations between the volition locks and other locks are not symmetrical, while they are symmetrical for the intention locks. Further, the volition locks have similar functions with the *marking* which is used in *Altruistic locking* scheme[15]. However, the volition locks have richer semantics as they are involved in determining the scheduling of accesses while markings are not.

6 Concluding Remarks

In mobile environment, the location of users is a frequently changing piece of data. Although there are a wide range of applications for aggregate querying users' location, the result of processing these aggregate queries may contain much errors caused by the physical and non-physical limitations of mobile environment. For example, sending paging messages will not guarantee all users to respond, especially, for the users who are down.

The use of location database can improve the accuracy of processing aggregate queries in mobile environment. However, the existing one step locking protocols, which become the most general concurrency control employed in database systems, are not appropriate for supporting the move operation. Deadlocks may frequently occur.

In this paper, we have proposed the two step position locking protocol for location databases. Due to its non-blocking property, the movement of user can be reflected into the location database as soon as possible. Further, since the locking is performed in two steps, the aggregate query can get a result that reflects the most recent situation. As a result, the accuracy of processing aggregate queries can be improved dramatically.

In the future, we will address the formalization issue of our locking protocol.

Acknowledgment

This research was supported by Grant-in-Aid for Scientific Research from the Ministry of Education, Science, Sports, and Culture, Japan under Grant No. 08244103.

References

[1] Alonso, R. and Korth, H., "Database Issues in Nomadic Computing," in *Proc. of ACM-SIGMOD Conf. on Management of Data*, pp. 388–392, 1993.

[2] Audestad, J. A., "GSM General Overview of Network Functions," in *Proc. of Intl. Conf. on Digital Land Mobile Radio Communications*, 1987.

[3] Awerbuch, B. and Peleg, D., "Sparse Partitions," in *Proc. of IEEE Symp. on FOCS*, pp. 514–522, 1990.

[4] Awerbuch, B. and Peleg, D., "Online Tracking of Mobile Users," in *Journal of the ACM*, Vol. 42, No. 5, pp. 1021-1058, 1995.

[5] Badrinath, B. R. and Imielinski, T., "Location Management for Networks with Mobile Users," in *Mobile Computing*, Kluwer Academic Publishers, p-p. 129–152, 1996.

[6] Badrinath, B. R., Imielinski, T., and Virmani, A., "Location Strategies for Personal Communication Networks," in *Proc. of Workshop on Networking of Personal Communication Applications*, 1992.

[7] Bernstein, P. A., Hadzilacos, V., and Goodman, N., *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, USA, 1987.

[8] EIA/TIA, "Cellular Radio-telecommunications Intersystem Operations," *Technical Report IS-41*, Revision B, 1991.

[9] Gray, J., "Notes on Database Operating Systems," in *Operating Systems – An Advanced Course, Lecture Notes in Computer Science*, Vol. 60, Springer-Verlag, 1978.

[10] Imielinski, T. and Badrinath, B. R., "Querying in Highly Mobile Distributed Environments," in *Proc. of VLDB Conf.*, No. 18, pp. 41–52, 1992.

[11] Imielinski, T. and Badrinath, B. R., "Mobile Wireless Computing: Solutions and Challenges in Data Management," *Technical Report DCS-TR-296*, Dept. of Computer Science, Rutgers University, U.S.A., 1993.

[12] Jing, J., Bukhres, O., and Elmagarmid, A., "Distributed Lock Management for Mobile Transaction," in *Proc. of ICDCS*, pp. 118–125, 1995.

[13] Mohan, S. and Jain, R., "Two User Location Strategies for Personal Communication Services," in *IEEE Personal Communications*, 1st Quarter, pp. 42–50, 1994.

[14] Ng, L. J., Donaldson, R. W., and Malyan, A. D., "Distributed Architectures and Databases for Intelligent Personal Communication Network," in *Proc. of ICWC*, pp. 300–304, 1992.

[15] Salem, K., Garcia-Molina, H., and Shands, J., "Altruistic Locking," in *ACM TODS*, Vol. 19, No. 1, pp. 117–165, 1994.