AN STUDY ON DATA CONSISTENCY IN SPATIAL DATABASE SYSTEM

Zhu Xinyan Li Deren Gong Jianya
National Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing
Wuhan Technical University of Surveying and Mapping
129 Luoyu Road, 430079, Wuhan, China
zxy@rcgis.wyusm.edu.cn

KEY WORDS: Data Sharing, Data Consistency, Lock, Undo/Redo Transaction, Notification-Reread Method.

ABSTRACT

In this paper, two cases of data consistency problems are discussed in Multi-user Geographical Information System. When a user updates some spatial data that have been accessed by other users in database, or execute an undo/redo operation, data consistency probably will be destroyed. To solve the problem caused by the first case, a "Notification-Reread Method" is introduced. As for the second case, there are many problems to be studied.

1 INTRODUCTION

Geographical Information System (GIS) is widely used in many fields. With the rapid development of computer network, GIS users care more about data sharing in networks. In traditional relational database, data consistency was controlled by consistency control mechanism When a data object is locked in a sharing mode, other transactions can only read it, but can't update it. This is appropriate in traditional relational databases that store attribute data and mainly deal with short transactions. In spatial databases, because of vast amount of data and complex topological relations, long transaction are met frequently. If the traditional consistency control method has been used yet, the system's concurrency will be badly influenced. So there come many new requirements about the consistency control in the field of GIS. There are many aspects of data consistency problems in spatial databases, such as the inconsistency between attribute and geometry data; the inconsistency of topological relations after geometry objects is modified. In this paper, other two cases of data consistency are discussed in multi-user geographical information system.

In GIS, there are many forms of data, such as geometry data, attribute, image data, and DEM data. In this paper , we only discuss spatial geometry data.

2 UPDATE CONSISTENCY

Considering the case in Figure 1, a group of users are manipulating spatial data in the same range. Suppose that there is a spatial object D. At one moment, one user want to update D, but at this time, other users have read it.

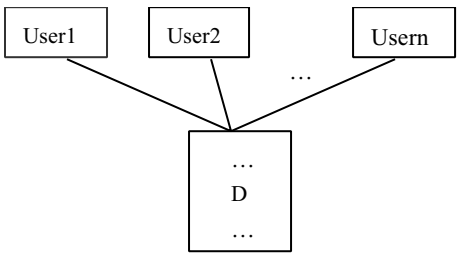

Figure 1. A block of Spatial data accessed by multi-users

2.1 Traditional Method in Relational Databases

In traditional relational databases, DBMS deal with this problem by concurrency control protocol. Transaction was introduced to implement this control protocol. In order to prevent any transaction from reading or updating data that is being updated by another transaction we require a locking mechanism, which guarantees a transaction exclusive access to an item of data while a lock is in force. The are two kind of locks: read-lock (sharing lock) and write-lock (exclusive

lock). A read-lock gives read-only access to a data item and prevents any other transaction from updating the item. Any number of transactions may hold a read-lock. A write-lock gives read/write access to a data item and, while in force prevents any other transaction from reading or writing to the data item. It gives a transaction exclusive access to a data item.

Using the traditional locking mechanism means if any one of the users read the object D, other users cannot update it. In traditional relational databases which mainly deal with attribute, and as the data items a transaction concerned usually are not very much, long transactions are not frequently happened, so the locking mechanism is properly. In spatial database, because of vast volumes of geometry data, usually, users manipulate a large scope of data items and hold for a long time. So long transactions are met frequently, the traditional locking mechanism is not satisfied in spatial database management. For example, a user read 16 maps with 1:10000 scale for display or spatial analysis, one way is holding a read-lock for these maps and keep them in client side memory until the operations end, in this case, other users cannot update these 16 maps, so the concurrency of the system is reduced. Another way is to lock a spatial object when it is used, and the objects the maps contain don't be kept in the client side memory, in this case, we need to read object from the spatial database frequently and the system would become low efficiency.

2.2 Notification-Reread Method

The locking system in the traditional database management system is conventional. The basic premise is that the inconsistency and imperfection of data is not allowed without considering how to solve the problem once it happens. To a certain extent, the traditional database management system is very passive. In fact, if DBMS is able to feed back the database update (adding, deletion, modification) of the committed transactions to the application programs, the problems may be solved to a large extent.

Suppose we do some modifications to the locking mechanism as follows: data with a sharing lock may allows a transaction (at most one transaction) to add an exclusive lock. This kind of modification will certainly cause the data inconsistency. As Figure 1 shows, user one read data D for query, and hold them until client process ends, while user two read data D for modification. Once the modification transaction is successfully committed, data in the database will be inconsistent with the data on the client side of user one, because user one does not know the modification. In order to avoid the situation, after the modification is successfully committed, the modified information must be feed back to user one in order to keep the data consistency.

The key point of the improvement is to inform client side of the database modification promptly. Send client sides related to the modified data a message so that client sides will reread the modified data after receiving the message. One method is to reread all data, which will cause low efficiency of the system, however. The other method is to reread the modified part of the data. It is demanded that the notification message should include information of the modified data. In the object-oriented system, information of the modified data can be represented by object identification.

2.3 Realization of Notification-Reread Method

The notification-reread method must be supported by both DBMS and the client side. When implementing the method, DBMS and the client programs should be improved accordingly.

2.3.1 Improvement of DBMS

Under the general circumstance of multi-user environment, DBMS on the server side should maintain the computer number of client side, user number and process number etc. The key to realize the notification-reread method is to inform the client process locking the data with a sharing mode about the modification information in time. An important task of DBMS is to implement "notification". Therefore, when locking the data with read mode, besides recording the type and status of the lock, it is required to know the client process number which manipulate the data. Thus, a more complicated locking mechanism is needed. There are two methods to realize it. One is to record the relevant data object locked with the sharing lock by means of client process number. The client process may be found according to the data object on the contrary. The other is to record the relevant client process number by means of the data object. Then the client process number may be found directly through the data object. The notification-reread method has increased the budget of DBMS maintenance.

The processing of modification transaction of data D is as follows:
(1)Judge whether D has an exclusive lock. If yes, wait until D is unlocked;
(2)Judge whether D has a sharing lock. If yes, mark it as a sharing lock;
(3)Write-locks on data D;

(4)Implement modification on the database;
(5)Commit the transaction. After it is committed successfully and before the transaction is finished, judge whether data D has a sharing lock. If yes, make a "notification" , inform each client sides about the task according to the computer number, user number, process number, data-use-range and current updated data object ID;
(6)End the transaction.

2.3.2 Improvement of Client Sides

The major task of the client sides of notification-reread method is to respond to the updated information sent by the DBMS on the server promptly. Therefore, the client program needs a process to monitor the information sent from DBMS. When data is to be found modified, the monitoring process records the ID of the modified data object in time and informs the application program. The application program will reread the data according to the object ID after receiving the notification.

The process of modification transaction of data D in client side is as follows:
(1) Read data in the database. Carry out client task processing and monitor the update notification sent by DBMS;
(2) If the notification of data update is received, reread the relevant modified data on database according to the notification and update the data on the client side;
(3) If data D is to be updated, add an exclusive lock on it;
(4) Update the data;
(5) Commit the transaction.

3 INCONSISTENCY AFTER UNDO/REDO

In the GIS data management, graphics data take a big part. In the process of data modification, UNDO/REDO is an important operation provided by the system. UNDO demands to cancel the previous modification, while REDO is to cancel the last UNDO.

3.1 Inconsistency Caused by UNDO/REDO

In the circumstance that data D in Figure 1 is a folded line, the situation is shown in Figure 2. If a user wants to move the coordinate of point 3, the point will turn to 3' after being moved. Suppose that there are two transactions as T_1 and T_2. Inspect on the sequence in Figure 3(a), in which T_{ij} represents the *j*th time of the *i*th transaction.

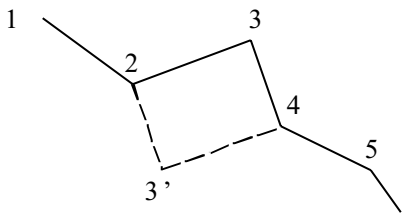

Figure 2. Move vertex 3 to 3 ’

T_{11}: T_1 reads D.
T_{12}: T_1 write-locks on data D.
T_{13}: T_1 update data D.
T_{14}: T_1 commits. Unlock.
T_{21}: T_2 read-locks on data D.
T_{15}: User 1 UNDO
T_{31}: T_3 write-locks on data D.
T_{32}: T_3 recovers D.
T_{33}: T_3 commits. Unlock.

(a) An inconsistency caused by UNDO

T_{11}: T_1 reads D.
T_{12}: T_1 write-locks on data D..
T_{13}: T_1 update data D.
T_{14}: T_1 commits. Unlock.
T_{21}: T_2 read-locks on data D.
T_{15}: User 1 UNDO
T_{31}: T_3 write-locks on data D.
T_{32}: T_3 recovers D.
T_{33}: T_3 commits, T_3 inform the relevant users.
T_{34}: T_3 unlocks.
T_{22}: T_2 rereads data D.

(b) Transaction sequence after improvement

Figure 3 . An inconsistency caused by UNDO operation and its improvement

In the sequence shown in Figure 3(a), at the time of T_{15}, since user 1 did the operation of UNDO, the data read by transaction T_2 is inconsistent with the data in the database. In fact, the operation of UNDO requires a modification transaction as T_3, whose process is similar to that of T_{12} to T_{14}. If the result of UNDO has been inform to the user application program before the UNDO operation is finished, the user application program will reread the relevant data. Thus, the inconsistency has been solved. The transaction sequence is shown in Figure 3(b). The inconsistency caused by REDO is as similar to UNDO. If UNDO/REDO operation is used frequently, frequent notification and reread are to be brought about.

In the process from T_{31} to T_{34} in Figure 3(b), the transaction T3 is in fact implementing the UNDO operation of user 1. The UNDO/REDO operation of a transaction has caused the recovery of the database update, the transaction may be called an UNDO transaction or a REDO transaction, or an UNDO/REDO transaction. The roll-back of the UNDO/REDO transaction is different from the common transactions. The roll-back of the common transactions is generally before the transaction is committed, due to some reason, the database should go back to the status before the transaction starts. However, to the UNDO/REDO transaction, because users have implemented the UNDO/REDO operation, the database is forced to go back to the status before the transaction has been committed. An UNDO/REDO transaction must correspond with a common transaction. It is the recovery transaction of this common transaction.

3.2 A Chain Reaction UNDO/REDO Transaction

The modification above is just a simple case. If take the object division and merge as well as the topological relations of spatial objects into consideration, the situation will be more complicated. In Figure 4(a), there are five arcs and two surfaces as S_1 and S_2 composed by the five arcs. Arc 1, 2 and 5 compose S_1, while arc 5, 4 and 3 compose S_2. Figure 4(b) is the situation that arc 5 has been divided into arc 51 and arc 52. At the time, S_1 is composed of arc 1, 2, 52 and 51, S_2 is composed of arc 4, 3, 51 and 52. Figure 4(c) represents the situation that arc 52 has been divided into arc 521 and 522. At the moment, S_1 is composed of arc 1, 2, 522, 521 and 51, while S_2 is composed of arc 521, 522, 4, 3 and 51. In the modification process of this case, the division of a line and reconstruction of topological relations must be carried out in one transaction so as to ensure the data consistency and accuracy.

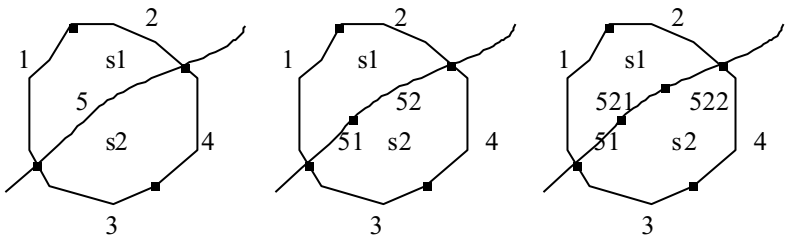

Figure 4. Object splitting with topology

Suppose that data D is the data block involving all data (including topological relation) of the case. Now take the transaction sequence of Figure 5 into account.

At the T_{27} moment of this sequence, user 2 wants to implement the UNDO operation. As user 1 has modified data D before user 2 carries out the UNDO operation, user 2 cannot implement the UNDO operation (add arc 51 and arc 52, delete arc 5, reconstruct S_1 and S_2) directly on the modified data D. Otherwise, inconsistency and inaccuracy will be caused, for arc 51 and 52 do not exist and the status of S1 and S2 has changed at the moment. In order that user 2 is able to carry out the UNDO operation accurately (suppose the corresponding transaction as T'_2), user 1 must implement the UNDO operation firstly (suppose the corresponding transaction as T'_1). Thus, a chain reaction UNDO/REDO transaction is brought about.

Suppose there is an transaction sequence as T_1, T_2, …, T_i, …, T_n, in which some transactions are the UNDO/REDO transactions of other transactions. As for the specified UNDO/REDO transaction T_i ($1<i<=n$), if another UNDO/REDO transaction T_j ($j<i$) exists, T_j must be executed before T_I, Then Tj is called the preorder UNDO/REDO transaction. In the above sequence, if an UNDO/REDO transaction has a preorder UNDO/REDO transaction, a chain reaction UNDO/REDO is engendered. Otherwise, this sequence does not have a chain reaction UNDO/REDO transaction. In the chain reaction UNDO/REDO transaction, when a user is carrying out an UNDO/REDO operation, the most important thing is to judge whether this UNDO/REDO transaction has a preorder UNDO/REDO transaction. If DBMS is to deal with the chain reaction UNDO/REDO transaction automatically, a more complicated control system is required, which has proposed a new research topic for the spatial data management system.

T_{11}: T_1 read-locks on D.
T_{21}: T_2 read-locks on D.
T_{22}: T_2 write-locks on D.
T_{23}: T_2 modifies data D (add arc 51 and 52, reconstruct S_1 and S_2, see Figure 4(b))
T_{24}: T_2 commits and sends the modification notification.
T_{25}: T_2 ends.
T_{12}: User 1 rereads data D.
T_{13}: T1 write-locks on D.
T_{14}: T1 modifies data D (add arc 521 and 522, delete arc 52, reconstruct S_1 and S_2, see Figure 4(c))
T_{15}: T1 commits and sends the modification notification.
T_{16}: T1 ends.
T_{26}: User 2 rereads data D.
T_{27}: User 2 carries out UNDO.

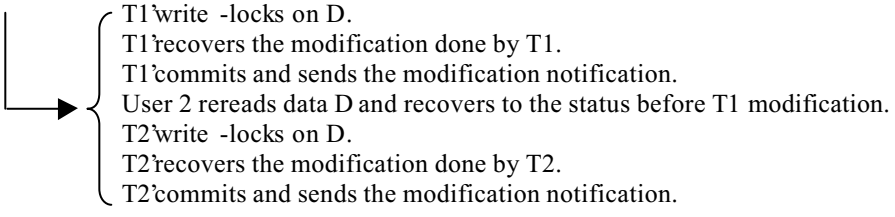

Figure 5. A chain reaction of UNDO/REDO transaction

4 CONCLUSIONS

In order to satisfy the GIS data sharing demanded by multiple users, spatial database management system has to solve the problem of data consistency in a multi-user environment. To the update causing the data inconsistency , "Notification-Reread Method" requires DBMS send a "notification", the client program will "reread" according to the "notification". With regard to the data inconsistency caused by the UNDO/REDO operation, if there is no chain reaction of UNDO/REDO transaction, the "Notification-Reread Method" may settle the problem. Otherwise, further research works should be done on the system.

The "Notification-Reread Method" needs the support from both spatial database management system and customer programs. The spatial database management system needs to know information as user's computer number, user number, process number, data-use-range and the ID of the currently updated data object.

REFERNENCES

John G. Hughes, 1988, DATABASE TECHNOLOGY: A Software Engineering Approach, PRENTICE HALL, NEW YORK.

D. AGRAWAL AND A. EL ABBADI, 1994, "A Nonrestrictive Concurrency Control Protocol for Object-Oriented Databases", DISTRIBUTED AND PARALLEL DATABASE OBJECTED MANAGEMENT, edited by Elisa Bertino and M. Tamer Qzsu, KLUWER ACADEMIC PUBLISHERS, Boston.

Yucai FENG, 1993,The Fundamental of Database System, Wuhan Science and Engineering University of Central China Press, 2nd Version.

Burrogh P A , 1986, Principles of Geographical Information System, For Land Resources Assessment. Oxford: Clarendon Press.