RESEARCH ON DATA CONSISTENCY IN SPATIAL DATABASE SYSTEM

ZHU Xinyan
WEN Yi
LI Deren
GONG Jianya

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 executes a UNDO/REDO operation, data consistency will probably 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 not 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 for 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 has been 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 Fig. 1, a group of users are manipulating spatial data in the same range. Suppose that there is a data object $D$, at one moment, one user wants to update $D$, but at this time, other users have read it.

Fig. 1 A block of spatial data accessed by multi-users
2.1 Traditional method in relational databases

In traditional relational databases, DBMS deals with this problem by concurrency control protocol\(^1\). Transaction was introduced to implement this control protocol\(^2\). 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. There are two kinds of locks: read-lock and write-lock. A read-lock (sharing 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 (exclusive lock). A write-lock gives read/write access to a data item, while in force and prevents any other transaction from reading or writing for 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 can not update it. In traditional relational databases which mainly deal with attribute, and as the data items that a transaction concerned usually are not so many, long transactions are not frequently happened, thus the locking mechanism works well. In spatial database, because of vast volume of geometry data, usually, users manipulate such a large scope of data items and hold for such a long time that long transactions are met frequently, the traditional locking mechanism is no longer satisfying in spatial database management. For example, a user read 16 maps at 1:10 000 scale for display or spatial analysis, one way is holding a read-lock for these maps and keep them in client 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 other objects are not kept in the client memory. In this case, we need to read object from the spatial database frequently and the system would be of 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 some application programs of relative transactions, the problems may be solved to a large extent.

Suppose we do some modifications to the locking system as follows; data with a sharing lock may allow a transaction (at most one transaction) to add an exclusive lock. This kind of modification will certainly cause the data inconsistency. As Fig. 1 shows, user 1 reads data \(D\) for query, and holds them until client process ends, while user 2 reads data \(D\) for modification. Once the modification transaction is successfully committed, data in the database will be inconsistent with the data on the customer side of user 1, because user 1 does not know the modification transaction. In order to avoid this situation, after the modification has been successfully committed, the modified information must be feed back to user 1 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, however, will cause low efficiency of the system. 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 operation, 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 end should maintain the computer number, user number and process number of client task. The key to realize the notification-reread method is to inform the client process
of the sharing lock about the modification information in time. An important task of DBMS is to implement "notification". Therefore, when locking the data with a sharing lock, besides recording the type and status of the lock, it is required to record the client process number of the data operation. Thus, a more complicated locking system is necessary. There are two methods to realize it. One is to record the relevant data object of the sharing lock by means of client process number. On the contrary, the client process may be found according to the data object. 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) Check whether D has an exclusive lock. If yes, wait until D is unlocked;
2) Check whether D has a sharing lock. If yes, identify it as a sharing lock;
3) Write the exclusive lock of data D;
4) Implement modification operation on the database;
5) Commit the transaction. After it is committed successfully and before the transaction is finished, check whether data D has a sharing lock. If yes, 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 number of the relevant users;
6) End the transaction.

2.3.2 Improvement of client sides

The major task of the client side of notification-reread method is to respond to the updated information sent by the DBMS on the server end promptly. Therefore, the customer program needs a process to monitor the information sent from DBMS. When data is to be found modified, record the ID of the modified data object in time and inform 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 data on the server end according to the notification to 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 takes a big part. In the process of data editing, UNDO/REDO is an important operation provided by the system. UNDO demands to cancel the previous editing, while REDO is to cancel the last UNDO.

3.1 Inconsistency caused by UNDO/REDO

In the circumstance that data D is a folded line in Fig. 1, the situation is shown in Fig. 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 Tl and T2. Inspect on the sequence in Fig. 3(a), in which $T_{ij}$ represents the jth time of the ith transaction.

Fig. 2 Move vertex 3 to 3
the relevant data. Thus, the inconsistency is cancelled. The transaction sequence is shown in Fig. 3(b). The inconsistency caused by REDO is similar to UNDO. If UNDO/REDO operation is used frequently, frequent notification and rereading are to be brought about.

\[ \begin{align*}
T_{11} & : T_1 \text{ reads } D. \\
T_{12} & : T_1 \text{ writes-locks on data } D. \\
T_{13} & : T_1 \text{ updates data } D. \\
T_{14} & : T_1 \text{ commits. Unlock. } \\
T_{21} & : T_2 \text{ reads-locks on data } D. \\
T_{15} & : \text{ User 1 UNDO } \\
T_{31} & : T_3 \text{ writes-locks on data } D. \\
T_{32} & : T_3 \text{ recovers } D. \\
T_{33} & : T_3 \text{ commits. Unlock. } \\
T_{14} & : T_1 \text{ commits. Unlock. } \\
T_{21} & : T_2 \text{ reads-locks on data } D. \\
T_{31} & : T_3 \text{ writes-locks on data } D. \\
T_{32} & : T_3 \text{ recovers } D. \\
T_{33} & : T_3 \text{ commits. T_3 informs the relevant users. } \\
T_{15} & : \text{ User 1 UNDO } \\
T_{31} & : T_3 \text{ unlocks. } \\
T_{22} & : T_2 \text{ rereads data } D.
\end{align*} \]

(a) An inconsistency caused by UNDO

(b) Transaction sequence after improvement

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

In the process from \( T_3 \) to \( T_{34} \) in Fig. 3(b), the transaction \( T_1 \) is in fact implementing the UNDO operation of user 1. The UNDO/REDO operation of users 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 taken place before the transaction is committed, for 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 topologic relations of spatial objects into consideration, the situation will be more complicated. In Fig. 4(a), there are five arcs and two surfaces as \( S_1 \) and \( S_2 \) composed of the five arcs. Arcs 1, 2 and 5 compose \( S_1 \), while arcs 5, 4 and 3 compose \( S_2 \). Fig. 4(b) is the situation that arcs 5 has been divided into arc 51 and arc 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52. At the time, \( S_1 \) is composed of arcs 1, 2, 51 and 52.

Fig. 4 Object splitting with topology

Suppose that data \( D \) is the data block involving all data (including topologic relation data) of the case. Now take the transaction sequence of Fig. 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 arcs 51 and 52, reconstruct $S_1$ and $S_2$) directly on the modified data $D$. Otherwise, inconsistency and inaccuracy would be caused, as arcs 51 and 52 do not exist and the status of $S_1$ and $S_2$ 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 a transaction sequence $T_1, T_2, \ldots, T_i, \ldots, T_n$, in which some transactions are the UNDO/REDO transactions of other transactions.

As for the specified UNDO/REDO transaction $T_i$ ($1 < i \leq n$), if another UNDO/REDO transaction $T_j (j < i)$ exists, $T_j$ must be implemented before $T_i$. $T_j$ 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 of UNDO/REDO is engendered. Otherwise, this sequence does not have a chain reaction of UNDO/REDO transaction. In the chain reaction of 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.

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. For the update of data consistency, "Notification-Reread Method" requires DBMS to 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 work should be done on the system.

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

References
1 Hughes J G. Database Technology: A Software Engineering Approach. New York: Prentice Hall, 1988
2 Agrawal D, Abbadi A E. A Nonrestrictive Concurrency Control Protocol for Object-Oriented Databases. Dis-
tributed and Parallel Database Objected Management. Bertino E, Qasu M T, eds. Boston: Kluwer Academic Publishers, 1994
3 Feng Y C. The Fundamental of Database System. 2nd ed. Wuhan: Press of Huazhong University of Science and Technology, 1993 (in Chinese)
4 Burrough P A. Principles of Geographical Information System, for Land Resources Assessment. Oxford: Claren
don Press, 1986