TRANSACTION MANAGEMENT IN DISTRIBUTED GEOGRAPHICAL DATABASES.

Britt-Inger Nilsson
Dept.of Environmental Planning and Design
Luleå University of Technology
S-971 87 Luleå, Sweden.
bini@sb.luth.se

ABSTRACT

The purpose of this paper is to study transaction management in a standard distributed data processing environment and to analyze methods for transaction management between independent spatial databases.

In distributed systems concurrency control is even more complex than in centralized systems. A transaction may access data stored at more than one site. Consistency of the databases involved must be guaranteed.

Standard database management systems of today cannot manage spatial data in an optimal manner. For that reason special spatial databases have been developed. Users of geographical information systems (GIS) are demanding the same sort of flexibility as in nongeographical database management systems.

The purpose of this paper is to compare different methods for concurrency control especially considering spatial data. The object is to elucidate restrictions and advantages of a distribution of geodatabanken, which is the national geographical database of the Swedish Land Survey.

In a GIS the user is working with long transactions. The disadvantages of different methods for concurrency control will be even more obvious. Which method to choose for concurrency control should be based on predictions not only about the current use of data but also the future use.

This study is primarly made with respect to geodatabanken. Geodatabanken is currently using the so called optimistic method for concurrency control. Also, a heterogeneous database environment is currently being considered within the organisation.

Preliminary studies of geodatabanken conclude that the optimistic methods for concurrency control seem to be enough at present. In consideration of probable change of the access patterns and the transaction types, a change of transaction management is recommended.

INTRODUCTION

The majority of present-day database systems are relational, and they also tend to be multi-user. These systems are developed to manage administrative data and therefore presume short transactions. Relational systems have well developed methods to guarantee data integrity.

Management of spatial information has requirements which traditional database managers cannot meet today. One problem is how to manage long transactions. Most GIS-vendors offer special solutions developed for spatial information.

Concurrently with the increase of information and the use of computerized systems the demand for service, access, flexbility, security, and local control will also increase. Distributed databases have been developed for nongeographical systems. Distributed systems make it possible to store data at different sites. A user at any site can access data stored at any site, without knowing where any given piece of data actually is stored. This in turn increases the requirements for security controls to maintain consistency between sites.

Today the techniques on how to handle distributed geographical databases are quite undeveloped. However, the problem has been stated before, see for example (Webster, 1988), (Sun, 1989), (Frank, 1992), (Pollock & McLaughlin, 1991), (Edmondson, 1989) and (Devine, Rafkind & McManus, 1991). The interest in distributed geographical databases will increase as the use of GIS technology spreads.

The Swedish Land Survey is currently in the process of establishing a continuous database of Sweden. Data will be stored in Geodatabanken which is the national geographical database developed by the Swedish Land Survey. These nation-wide databases will be very large. It is desirable that parts of these databases be stored at local sites. If these local databases consist of a subset of the central nation-wide database, all databases must have identical information in overlapping areas.

The purpose of this paper is to describe the methods for transaction management and to analyze methods on how to distribute geographical databases, especially geodatabanken.

TRANSACTION MANAGEMENT

The normal assumption in a traditional DBMS is that a transaction is short. In a banking system, for example, a transaction can be to transfer an amount of money from one account into another. Since this process is short, a few seconds at most, other users will not become aware of it.

In a GIS, a transaction can be very long, possibly hours or months. Here a transaction might be to digitize a map sheet or to redesign parts of an old utility network. Other users should not be prevented from working during this time.

Causes of failure

To create a reliable system which is able to recover automatically, all possible failures need to be identified. These are classified under four headings (Bell & Grimson, 1992).

- Transaction-local failure.
  - Transaction-induced abort. The programmer has written code to trap a particular error.
  - Unforseen transaction failure arising from bugs in the program.
  - System-induced abort. For example when the transaction manager explicitly aborts a transaction to break a deadlock or because it conflicts with another transaction.

- Site failures.
  - Occurs as a result of failure of the local CPU or a power supply failure resulting in a system crash.

- Media failures.
  - A failure which results in some portion of the stable database being corrupted. Disk head crash.

- Network failures.
  - Communication failure, for example, resulting in the network being partitioned into two or more sub-networks.

Concurrency control

While most DBMSs are multi-user systems there are tools needed to ensure that concurrent transactions do not interfere with each other. The purpose of consistency control is to guarantee consistency of the database.

A lot of problems can occur if concurrent transactions can interfere with one another, for example:

- Lost updates. A transaction that seems to be successfully completed can be overridden by another user.

- Inconsistent retrieval. A transaction can obtain inaccurate results if it is allowed to read partial results of incomplete transactions which are simultaneously updating the database.

There are three basic techniques in traditional DBMSs for concurrency control. Locking methods and timestamp methods are conservative methods. They check if a conflict occurs every time a read or write operation is executed. Optimistic methods are based on the premise that conflicts are rare and allow transactions to proceed. When a transaction wishes to commit, the system checks for conflicts. Version management is a special form of optimistic method developed to manage long transactions in spatial databases.

Locking methods

Locking methods are the most widely used approach to handle concurrency control. A transaction puts a read or write lock on a data item before a read or write operation is executed. Since read operations cannot conflict, it is permissable for more than one transaction to hold read locks on the same data item at the same time. A write lock gives a transaction exclusive access to the data item, no other transaction can read or update that data item as long as the write lock is hold. The write operations will not be visible to other transactions until the actual transaction releases its write locks.

Some systems allow upgrading and downgrading of locks. This means that if a transaction holds a read lock on a data item, this lock can be upgraded to a write lock on condition that the transaction is the only one holding a read lock on that data item. Upgrading of locks allows a transaction to examine the data first and then decide whether or or not it wishes to update it. Where upgrading is not supported, a transaction must hold write locks on all data items which it might want to update some time during the execution of the transaction.

The most common locking protocol is known as two-phase locking (2PL). The method got its name becuse it operates in two distinct phases. It consist of a growing phase during which the transaction acquires locks and a shrinking phase during which it releases those locks.

The rules for transactions which obey 2PL are:

- Transactions must be well-formed. A transaction must acquire a lock on a data item before operating, and all locks held by a transaction must be released when the transaction is finished.

- Rules for locking must be consistent. No conflicts like write-write locks or read-write locks are allowed to exist.
- Once the transaction has released a lock, no new locks are acquired.
- All write locks are released together when a transaction commits.

Deadlocks can occur when a transaction waits for a lock to be released by another transaction which in turn is waiting for a lock to be released by the first transaction. In a distributed system these deadlocks are even more complex to detect since a number of different sites can be involved in transactions.

Timestamp methods

No locks are involved in timestamp methods, and therefore no deadlocks can occur. Transactions involved in conflicts are rolled back and restarted. The goal of timestamping methods is to order transactions in such a way that older transactions (transactions with smaller timestamps) get priority in case a conflict occurs.

If a transaction tries to read or write a data item, the read or write will only be allowed if the last update on that data item was made by an older transaction. Otherwise the requesting transaction is restarted given a new timestamp.

Timestamp methods produse schedules from the timestamps of transactions which are committed successfully. Timestamps are used to order transactions with respect to each other. Each transaction is assigned a unique timestamp when it is started. No two transactions can have the same timestamp. In a centralized system they can be generated from the system clock or alternatively from a simple counter. When a transaction is started it is assigned the next value from the counter, which is then increased. To avoid very large timestamp values, the counter can periodically be reset to zero.

In a distributed system there is no central system clock or centalized counter. Each node has it own clock, and there is no guarantee that these clocks are syncronized with each other. Two (or more) transactions can start at the same time at different sites in the network. A simple approach is to define global time as the concatenation of the local site clock time with the site identifier.

A transaction must guarantee atomicity. It can easily be done by locking methods. With timestamp methods we do not have any locks, thus it is possible for other transactions to see partial updates. This can be prevented by not writing to the database until the actual transaction is committed. The updates are written to buffers, which in turn are written to the database when the transaction commits. This approach has the advantage that when a transaction is aborted and restarted no changes need to be made in the database.

Optimistic methods

Optimistic methods are based on the premise that conflicts are rare and that the best approach is to allow transactions to continue. Timestamps are only assigned to transactions. To ensure atomicity of transactions, all updates are made to local copies of the data. When a transaction commits, the system checks for conflicts. During detection of conflicts the system checks if there are transactions which overlap each other in time and, if so, if they affect the same data item. In the event of conflict, the local copy is discarded and the transaction is restarted. If no conflicts have been detected, the local copy is propagated to the database.

In environments where read-only transactions dominate, optimistic methods are attractive because they allow the majoriy of transactions to proceed without hindrance.

Version management

The different methods for concurrency control as mentioned above have been developed for short transactions. When using GIS the concurrency problem are even more complex because of long transactions. Version management is a method managing many users involved in long transactions. This method is essentially an optimistic approach on the assumption that, in the majority of situations, there probably will not be any conflicts between versions.

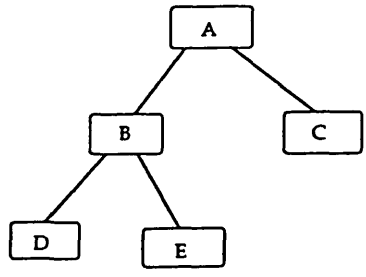
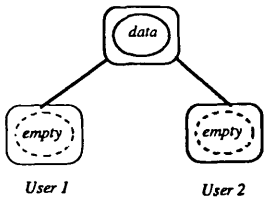


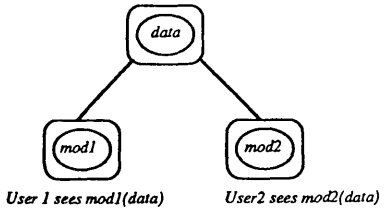
Figure 1. Example of a Version Tree.

Versions are thought of as having a hierarchical arrangement so that later versions are derived from earlier ones. At the top is the master database. At the leaves of the tree, each user has an alternative in which to work without conflicting with others. Each version has a unique route back to the root of the tree. This route can be called an access path since it determines precisely which part of the database each user can see and also what can be modified.

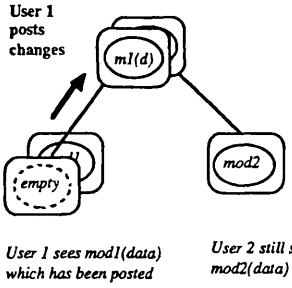
Changes are made visible (posted) one level at a time. Provided that the version at the higher level is the one from which the changed "leaf" version was derived, the changes will be made. But if the version at the higher level has since been updated, the version about to post must first be refreshed with superior updates. The other users will not notice that their versions suddenly change. They will have to make a refresh operation in order to see changes made at the higher level.



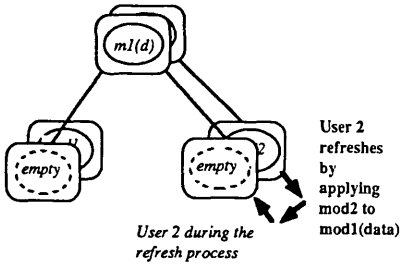
a) Two users see original data.



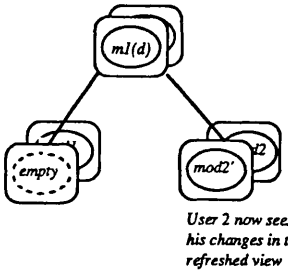
b) Each user makes changes independently of the other.



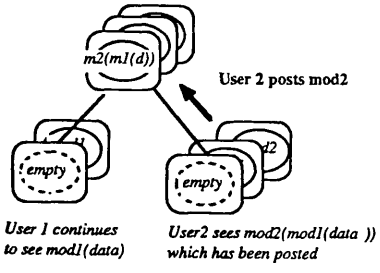
c) User 1 posts changes, making them potentially visible. User 2 doesn't see them yet.



d) User 2 begins the refresh process, in order to see user 1's changes.



e) User 2 sees the combination of changes.



f) User 2 posts changes. User 1 doesn't see them yet.

Figure 2. Illustrating the modify/post/refresh cycle for a pair of users. (Easterfield et al.)

Comparison

The different methods for concurrency control deal with conflicts differently which gives each method its own advantages and disadvantages.

Table 1. Comparison between different methods for concurrency control.

	Locks	Timestamps	Optimistic methods	Version management
How are conflicts dealt with?	The transaction waits.	The transaction is rolled back and restarted.	The transaction is rolled back and restarted.	Every single object is validated manually.
When are conflicts detected?	When a lock is applied for.	When reading/ writing a data item.	When a transaction wishes to commit.	When posted to a higher level.
Advantages	Waiting is not as expensive as restarting.	No deadlocks. No waiting.	No deadlocks. No waiting. Attractive where read-only transactions.	No deadlocks. No waiting. Possible to keep versions.
Disadvantages	Deadlocks. Enormous message overhead . Waiting is unacceptable when long-transactions	Rollback and restart is expensive. In distributed systems analyses will be very complex.	Rollback and restart is expensive. In distributed systems analyses will be very complex.	Rollback and restart is expensive. Inconsistency can occur as a result of manual validation.

GEODATABANKEN

Geodatabanken is a national spatial database developed by Swedish Land Survey. Its purpose is to store spatial data independent of map types or scales. Geodatabanken can briefly be described as:

- Feature oriented, which means that feature type is stored in conjunction with coordinates.
- Seamless, no map sheet partitioning.
- Multi-user system.
- Check-out/check-in routines with a possibility to append data.
- History.
- Optimistic methods for concurrency control.
- Network data structure.
- Inhouse developed security system.

Geodatabanken uses a time-dimension which gives information about the history of an object. It contains information about when the object was created, when it was edited or deleted, and who did the operations. The time is necessary for appending data, and time is also needed for concurrency controls.

Local databases

The Swedish Land Survey is currently in the process of establishing a continuous database of Sweden. Data for the northern region is being collected at the regional office in Luleå. Since the nation-wide database is to be stored in geodatabanken, the office in Luleå is regularly checking data in and out of geodatabanken.

A nation-wide database will be very large. The office in Luleå requires a local database containing a subset of the nation-wide database, more exactly, all data from the local region. A problem that will occur is that all databases must have identical information in overlapping areas.

In addition to the requirements of concurrency controls and consistency controls additional demands have to be fulfilled such as:

- an object has to have an unique identity,
- all objects have to be able to store time.

It has been discussed that the regional office should use a commercial GIS database management system, for instance ArcStorm from ESRI or GeoData Manager from Intergraph, instead of the inhouse developed system Geodatabanken. This would lead to a distributed heterogeneous environment. A requirement would be to maintain consistency. It would be necessary to preserve all information in both systems. It would also be necessary for their concurrency control methods, at least to some extent, to be able to communicate and co-operate. All information needed in a system to maintain consistency has to be transferable and differences in dataformat has to be manageable.

DISCUSSION

The course of checking-out - checking-in represents a transaction in Geodatabanken. Several users can simultaneously work in the same area. Geodatabanken uses optimistic methods for concurrency control when trying to check-in data. If any conflict occurs, the user has to add data from geodatabanken through the earlier check-out operation and then validate the data manually.

Today, work in Geodatabanken is concentrated in building new databases. The nation-wide data collection is divided into regions and thereby conflicts are rare.

Due to the optimistic methods, data is always accessible in geodatabanken. This is an advantage because almost all of the transactions are long, and in these circumstances it is unreasonable to expect other users to be locked out. The disadvantage with this method is the enormous overhead involved in restarting a transaction in case of a conflict. In the worst situation, important decisions can be made on the basis of inconsistent data.

Preliminary studies of work done in theSwedish Land Survey shows that the main part of the work uses local data. If there is occasionally interest to use other than local data, it is almost always dealing with read operations only. These facts lead to the conclusion that the optimistic methods for concurrency control seem to be enough at present.

The nation-wide databases will be very large. It is desirable that parts of these databases will be stored at local sites. By spreading these databases to the regions, better access to regional data is possible,and it might also give better quality data.

Some of the advantages of distributing local copies of Geodatabanken are:

- Local autonomy.
- Response time will be reduced.
- Greater reliability. All work in the region can continue even if a failure at the central database occurs, and vice versa, in case of failure of a local database, all data are available from the central database.
- Communication costs will be reduced.

Producing a true distributed spatial database-management system is difficult. Besides all the complicated problems which occur by distributing traditional databases, the developer, among other things, also has to deal with long transactions. Even in non-spatial database environments there is a lack of experience considering distributed systems.

To be able to make conclusions on whether the optimistic methods are enough for concurrency control in the future, the future access pattern has to be predictable. Due to the limitations of this method, the transaction management might have to be changed if the access pattern changes. Questions to pay attention to:

What will the future use of spatial data look like?
Who is responsible for updating these data?
Who will be interested in using these data?

REFERENCES

AutoKa-PC-projektet, 1993.Datorprogram, Beskrivning, Flyttfil för Geodatabanken och AutoKa-PC. Dnr 162-93-609. Lantmäteriverket, Gävle.

Baker T. and Broadhead J., 1992. Integrated Access to Land-related Data from Heterogeneous Sources. URISA Proceedings. Papers from the Annual Conference, Washington, DC, USA. Vol 3, pp 110-121

Batty P., 1992. Exploiting Relational Database Technology in a GIS. Computer & Geosciences, Vol. 18, pp 453-462.

Bell D. and Grimson J., 1992. Distributed Database Systems. Addison-Wesley Publishing Company.

Brown R. and Mack J., 1989. From the Closed Shop to the Organizational Mainstream: Organizational Impacts of a Distributed GIS. GIS/LIS '89 Proceedings. Annual Conference. Vol 1 pp 152-160.

Ceri S. and Pelagatti G., 1984. Distributed Databases - Principles and Systems. McGraw-Hill Inc. USA.

Date C.J., 1986. An Introduction to Database Systems. Vol 1, Fourth Edition. Addison-Wesley Publishing Company.

Degerstedt K., 1993. Synkronisering av geodatabanker. Utkast nr 2. Funk spec. Lantmäteriverket, Gävle.

Degerstedt K. and Ottoson P., 1993. En introduktion till Geodatabanken. Lantmäteriverket, Gävle.

Degerstedt K. and Ottoson P., 1993. AutoKa-Geodatabanken, Handbok. Lantmäteriverket, Gävle.

Devine H.A., Rafkind C.D. and McManus J., 1991. A New Approach to GIS Implementation: Colonial National Historic Park. Technical Papers - 1991 ACSM-ASPRS Annual Convention, Baltimore, USA. Vol 4, pp 51-60.

Easterfield M., Newell R.G. and Theriault D.G.. Version Management in GIS - Applications and Techniques. Technical Paper 8. Smallworld Systems Ltd., Cambridge, England.

Edmondson P.H., 1989. Transition from a Closed Shop GIS to a True Distributed GIS. GIS/LIS '89 Proceedings. Annual Conference. Vol 1, pp 161-170

Frank A.U., 1992. Acquiring a Digital Base Map: A Theorethical Investigation into a Form of Sharing Data. URISA Journal, Vol 4, pp 10-23.

Lidén J., 1992. Versionshantering av geografiska databaser. Smallworld Svenska AB. Stockholm.

Newell R.G. and Easterfield M., 1990. Version Management - the problem of the long transaction. Technical Paper 4. Smallworld Systems Ltd., Cambridge, England.

Pollock R.J. and McLaughlin J.D., 1991. Data-Base Management System Technology and Geographic Information Systems. Journal of Surveying Engineering, Vol 117, No 1, pp 9-26.

Salen F., 1992. Kartbiblioteksystem - för hantering av geografiska databaser. Högskolan i Luleå.

Sun Chin-Hong, 1989. Development of the National Geographic Information in Taiwan. GIS/LIS '89 Proceedings. Annual Conference, Vol 1, pp 270-275.

Webster C., 1988. Disaggregated GIS architecture - Lessons from recent developments in multi-site database management systems. International Journal of Geographical Information Systems. Vol 2, no 1, pp 67-79.

Özsu T.M and Valduriez P., 1991. Principles of Distributed Database Systems. Prentice-Hall Inc, Englewood Cliffs, New Jersey.