UDC:004.63
Original scientific paper

TRANSACTION DESIGN FOR DATABASES WITH HIGH PERFORMANCE AND AVAILABILITY[1]

Lars Frank
Department of Informatics, Copenhagen Business School, Frederiksberg, DENMARK
frank@CBS.dk

Abstract: *When many concurrent transactions like ERP and E-commerce orders want to update the same stock records, long duration locking may reduce the availability of the locked data. Therefore, transactions are often designed without analyzing the consequences of loosing the traditional ACID (Atomicity, Consistency, Isolation and Durability) properties. In this paper, we will analyze how low isolation levels, optimistic concurrency control, short duration locks, and countermeasures against isolation anomalies can be used to design transactions for databases with high performance and availability. Long duration locks are defined as locks that are held until a transaction has been committed, i.e. the data of a record is locked from the first read to the last update of any data used by the transaction. This will decrease the availability of locked data for concurrent transactions, and, therefore, optimistic concurrency control and low isolation levels are often used. However, in systems with relatively many updates like ERP-systems and E-commerce systems, low isolation levels cannot solve the availability problem as all update locks must be exclusive. In such situations, we will recommend the use of short duration locks. Short duration locks are local locks that are released as soon as possible, i.e. data will for example not be locked across a dialog with the user. Normally, databases where only short duration locks are used do not have the traditional ACID properties as at least the isolation property is missing when locks are not hold across a dialog with the user. The problems caused by the missing ACID properties may be managed by using approximated ACID properties, i.e. from an application point of view the system should function as if all the traditional ACID properties had been implemented.*

Keywords: *ACID properties, database availability, short duration locks, multi-databases, client/server technology, ERP systems and E-commerce.*

1. INTRODUCTION

In this paper, we will analyze how concurrency methods like optimistic concurrency control and low isolation levels can be integrated with short duration locks and countermeasures against the absence of the traditional ACID properties in order to design transactions for databases with high performance and availability. Table 1 gives an

[1] A preliminary version of this paper appeared in *Proceedings of the 6th International Conference on Enterprise Information Systems*, ICEIS2004, Porto, Portugal, April 4 -17, 2004.

overview of the properties of the methods used to increase concurrency. Updating database transactions consists often of several subtransactions. For example, when a user is going to update some data in a remote database, the user normally starts a query that reads the data, which is going to be updated. Next, the user makes corrections to the data in user's PC or workstation. Finally, the corrections are sent to the remote database location where the data is updated. If short duration locks are used in the transaction described above, all the subtransactions will have local ACID properties, but the global transaction consisting of all the subtransactions will not have the traditional ACID properties. In this paper, we will use extended transaction models where only approximated ACID properties are implemented.

Table 1: Evaluation overview of methods to increase concurrency.

Evaluation criteria	Optimistic concurrency control	Low isolation levels	Short duration locks	Counter-measures
Deadlock	Eliminates deadlock, but restarts will occur in case of conflicts	Can eliminate read-write conflicts	Can eliminate both read-write and write-write conflicts	Special countermeasures against deadlock may be implemented
Hotspots	Hotspots will cause many restarts	Can eliminate read-write conflicts in hotspots	Can diminish locking time in hotspots	Special counter-measures may be designed against hotspot problems
The atomicity property	No problems	No problems	Extended transaction models should be used	
The consistency property	No problems	May cause inconsistency, and therefore countermeasures may be used	Will cause inconsistency, and therefore countermeasures should be used	Asymptotic consistency should be used
The Isolation property	No problems	May cause isolation anomalies, and therefore countermeasures may be used	Will cause isolation anomalies, and therefore countermeasures should be used	Countermeasures should be used to manage isolation anomalies
The durability property	No problems	No problems	No problems if atomicity is implemented	No problems if atomicity is implemented
Distribution options	Distributed concurrency control will decrease performance and availability	May be implemented in a distributed DBMS without further problems	Retriable subtransactions may be necessary to implement atomicity	Many different distributed countermeasures are available
Development costs	A DBMS facility	A DBMS facility	Extra costs for compensation implementation	Extra costs for countermeasure implementation

That is, the global atomicity property is implemented by using compensatable, pivot and retriable subtransactions in that order. The global consistency and isolatio n properties are managed by using countermeasures as described by Frank and Zahle (1998). The global durability property is implemented by using compensation and/or the durability property of the local DBMS. By using this transaction model it is possible to increase the availability of both central and distributed databases because only short duration locks are used. The major disadvantage of using short duration locks is the problems of managing the consistency of data. However, these problems can be reduced/solved by using countermeasures against the isolation anomalies that occur when the isolation property is missing. The paper is organized as follows: Section 2 will describe an extended version of the countermeasure transaction model, which includes central databases, and we will describe the most important countermeasures against the isolation anomalies used in central databases. In section 3, we will illustrate how to design database transactions optimized for high availability by using short duration locks. Concluding remarks are presented in section 5. Related Research: The transaction model described in section 2 is an extended version of *the countermeasure transaction model* described by Frank and Zahle (1998), Frank (1999), and Frank and Kofod (2002). This model owes many of its properties to e.g. Garcia - Molina and Salem (1987); Mehrotra et al. (1992); Weikum and Schek (1992) and Zhang (1994).

2. THE TRANSACTION MODEL

A *multidatabase* is a union of local autonomous databases. *Global transactions* (Grey and Reuter, 1993) access data located in more than one local database. In recent years, many transaction models have been designed to integrate local databases without using a distributed DBMS. The countermeasure transaction model (Frank and Zahle, 1998) has, among other things, selected and integrated properties from these transaction models to reduce the problems caused by the missing ACID properties in a distributed database that is not managed by a distributed DBMS. In the countermeasure transaction model, a global transaction involves a *root transaction* (client transaction) and several single site *subtransactions* (server transactions). Subtransactions may be nested transactions, i.e. a subtransaction may be a *parent transaction* for other subtransactions. All communication with the user is managed from the root transaction, and all data is accessed through subtransactions. A subtransaction is either an execution of a *stored procedure* that automatically returns control to the parent transaction or an execution of a *stored program* that does not return control to the parent transaction. All remote subtransactions are accessed through one of the following types of middleware:

Remote Procedure Call (RPC)

From a programmer's point of view, a RPC functions as a remote procedure call or submission of a SQL query. From a performance and an atomicity point of view, RPCs have the following important properties:

- If a parent transaction executes several RPCs, the corresponding stored procedures are executed one at a time.
- A stored procedure or SQL submission has only local ACID properties.
- The stored procedure or SQL submission automatically returns control to the parent transaction.

Update Propagation (UP)

In this context, UP is used in the sense of propagating any data (not just updates) in such a way that the data is transferred and stored/executed with atomicity and durability properties. UPs have the following properties, which are important from a performance and an atomicity point of view:

- If a parent transaction initiates several UPs, the corresponding, stored programs may be executed in parallel.
- A stored program initiated from a UP has atomicity together with the parent transaction, i.e. either both or none are executed.
- The stored program does not automatically return control to the parent transaction. The following subsections will give a broad outline of how approximated ACID properties are implemented in the countermeasure transaction model.

2.1 THE ATOMICITY PROPERTY

An updating transaction has the *atomicity property* and is called *atomic* if either all or none of its updates are executed. In the countermeasure transaction model, the global transaction is partitioned into the following types of subtransactions executed in different locations: The *pivot* subtransaction that manages the atomicity of the global transaction. The global transaction is committed when the pivot subtransaction is committed locally. If the pivot subtransaction aborts, all the updates of the other subtransactions must be compensated. The *compensatable* subtransactions that all may be compensated. Compensatable subtransactions must always be executed before the pivot subtransaction is executed to make it possible to compensate them if the pivot subtransaction cannot be committed. A compensatable subtransaction may be compensated by executing a *compensating* subtransaction. The *retriable* subtransactions that are designed in such a way that the execution is guaranteed to commit locally (sooner or later) if the pivot subtransaction has been committed. A UP tool is used to resubmit the request for execution automatically until the subtransaction has been committed locally, i.e. the UP tool is used to force execution of the retriable subtransaction. The global atomicity property is implemented by executing the compensatable, pivot and retriable subtransactions of a global transaction in that order. For example, if the global transaction fails before the pivot has been committed, it is possible to remove the updates of the global transaction by compensation. If the global transaction fails after the pivot has been committed, the remaining retriable subtra nsactions will be (re)executed automatically until all the updates of the global transaction have been committed.

2.2. THE CONSISTENCY PROPERTY

The consistency property is not useful in distributed databases with approximated ACID properties because such a database is almost always inconsistent.

2.3 THE ISOLATION PROPERTY

The isolation property is normally implemented by using *long duration locks*, which are locks that are held until the (global) transaction has been committed (Frank and Zahle, 1998). To ensure high availability in locked data, short duration locks should be used in all subtransactions, just as locks should be released before interaction with a user. This is not a problem in the countermeasure transaction model as the traditional isolation property is lost anyway. When transactions are executed without isolation, the so-called *isolation anomalies* may occur. If there is no isolation and the atomicity property is implemented, the following isolation anomalies may occur (Berenson et al., 1995 and Breibart, 1992). *The lost update anomaly* is by definition a situation where a first transaction reads a record for update without using locks. Subsequently, the record is updated by another transaction. Later, the update is overwritten by the first transaction. *The dirty read anomaly* is by definition a situation where a first transaction updates a record without committing the update. Subsequently, a second transaction reads the record. Later, the first update is aborted (or committed), i.e. the second transaction may have read a non-existing version of the record. *The non-repeatable read anomaly* or *fuzzy read* is by definition a situation where a first transaction reads a record without using locks. Later, the record is updated and committed by a second transaction before the first transaction has been committed. In other words, it is not possible to rely on the data that have been read. *The phantom anomaly* which is not dealt with in this paper. The countermeasure transaction model (Frank and Zahle, 1998) describes countermeasures that reduce the problems of the anomalies. *The pessimistic view countermeasure* used in section 3 reduces or eliminates the dirty read anomaly and/or the non-repeatable read anomaly by giving the users a pessimistic view of the situation. In other words, the user cannot misuse the information. The purpose is to eliminate the risk involved in using data where long duration locks should have been used. A pessimistic view countermeasure may be implemented by using:

- Compensatable subtransactions (or the pivot transaction) for updates that limit the users' options.
- Retriable subtransactions (or the pivot transaction) for updates that increase the users' options.

For example, when updating stocks, compensatable subtransactions should be used to reduce the stocks and retriable subtransactions should be used to increase the stocks.

2.4. THE DURABILITY PROPERTY

Updates of transactions are said to be *durable* if they are stored in stable storage and secured by a log recovery system. The global durability property will automatically be implemented, as it is ensured by the log-system of the local DBMS systems (Breibart et al., 1992).

3. TRANSACTION DESIGN IN E-COMMERCE SYSTEMS

In this section, we will illustrate how to use our transaction model in business -to-business E-commerce. We will assume that the seller has a customer file with the names, addresses, account balances and credit limits for all his customers. Therefore, the banks of the customers are not involved in the following description of the order transaction. At first, the buyer reads the offers made by the seller. If the buyer wants to make an order, the root transaction in the location of the buyer calls a compensatable subtransaction at the location of the seller. This subtransaction creates an order record with relationship to the customer record at the same location. Now, the buyer can make order-lines. For each new order-line made by the buyer, the root transaction starts a compensatable subtransaction, and this subtransaction creates an order-line at the location of the seller and updates the stock of the product ordered in the order-line by using *the pessimistic view countermeasure*. Pease notice that making order lines cannot cause deadlock when only short duration locks are used. If an order-line cannot be fulfilled, the field "quantity-delivered" in the order-line is updated. When the order form has been completed, the pivot subtransaction updates the account balance of the customer. If the credit limit of the customer is not violated, the pivot subtransaction will also confirm the deal for the customer. Alternatively, the buyer will be the asked to reduce the amount of the balance in order to avoid violating the credit limit.By executing a subtransaction that reduces the quantity ordered in an order-line, the amount in the order-line can be reduced and the stock of the product increased. Finally, the buyer can retry to execute the pivot subtransaction. The reread countermeasure should be used as the customer record has been read earlier.

4. CONCLUSIONS

In this paper, we have analyzed how low isolation levels, optimistic concurrency control, short duration locks, and countermeasures against isolation anomalies can be used to design transactions for databases with high performance and availability. The methods are independent of each other and, therefore, a mixture of the methods may give the best performance and availability. Using low isolation levels will increase the availability of locked data. If an application cannot accept the anomalies that are caused by using a low isolation level, it may be possible to minimize the time that data is locked by substituting long duration exclusive locks with short duration exclusive locks and countermeasures against the anomalies that may occur when a major database transaction is split into minor database transactions performing the same operations. The extra costs of this solution include the implementation of approximated ACID properties where e.g. DBMS aborts are substituted by compensations. Even if all applications can accept the anomalies caused by using a low isolation level, problems may occur as there is no isolation level that allows exclusive locks not to exclude conflicting updates. Therefore, in hotspots where many concurrent transactions update the same records we recommend to use short duration exclusive locks and countermeasures against the anomalies that may occur when long duration exclusive locks are substituted by short duration locks. Short duration locks and countermeasures against the isolation anomalies should be mandatory for *long-lived transactions* (Grey and Reuter, 1993) as long duration locks per definition cannot be recommended for such transactions. We have illustrated how to use countermeasures against isolation anomalies in E-commerce examples.

REFERENCES

[1] Berenson, H., Bernstein, P., Gray, J., Melton, J., O'Neil, E. and O'Neil, P., 1995, "A Critique of ANSI SQL Isolation Levels", *Proc ACM SIGMOD Conf.*, pp. 1-10.

[2] Breibart, Y., Garcia-Molina, H. and Silberschatz, A., 1992, "Overview of Multidatabase Transaction Management", *VLDB Journal*, 2, pp 181-239.

[3] Frank, L., 1999, "Evaluation of the Basic Remote Backup and Replication Methods for High Availability Databases", *Software -Practice & Experience*, Vol. 29, issue 15, pp 1339-1353.

[4] L. Frank and Uffe Kofod, 2002, 'Atomicity Implementation in E-Commerce Systems', *Proc of the Second International Conference on Electronic Commerce, ICEB 2002,* Taipei.

[5] Frank, L. and Zahle, T, 1998, "Semantic ACID Properties in Multidatabases Using Remote Procedure Calls and Update Propagations", *Software - Practice & Experience*, Vol.28, pp. 77-98.

[6] Gallersdörfer, R. and Nicola, M., 1995, "Improving Performance in Replicated Databases through Relaxed Coherency", *Proc 21st VLDB Conf,* 1995, pp 445-455.

[7] Garcia-Molina, H. and Salem, K., 1987, "Sagas", *ACM SIGMOD Conf*, pp 249-259.

[8] Garcia-Molina, H. and Polyzois, C., 1990, "Issues in disaster recovery", *IEEE Compcon.*, IEEE, New York, pp 573-577.

[9] Gray, J. and Reuter, A., 1993, "Transaction Processing", *Morgan Kaufman*, 1993.

[10] Kung, H and Robinson, J, 1981, 'On Optimistic Methods for Concurrency Control', *ACM TODS 6*, No. 2.

[11] Humborstad, R., Sabaratnam, M., Hvasshovd, S. and Torbjornsen, O., 1997, "1-Safe algorithms for symmetric site configurations", *Proc 23th VLDB Conf, 1997*, pp 316-325.

[12] Mehrotra, S., Rastogi, R., Korth, H., and Silberschatz, A., 1992, "A transaction model for multi-database systems", *Proc International Conference on Distributed Computing Systems*, pp 56-63.

[13] O'Neil, P., "The Escrow Transaction Mode", *ACM TODS* 11, No. 4, 1986.

[14] Polyzois, C. and Garcia-Molina, H., 1994, "Evaluation of Remote Backup Algorithms for Transaction-Processing Systems", *ACM TODS*, 19(3), pp 423449.

[15] Weikum, G. and Schek, H., 1992, "Concepts and Applications of Multilevel Transactions and Open Nested Transactions", *A. Elmagarmid (ed.): Database Transaction Models for Advanced Applications*, Morgan Kaufmann, pp 515-553.

[16] Zhang, A., Nodine, M., Bhargava, B. and Bukhres, O., 1994, "Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems", *Proc ACM SIGMOD Conf*, pp 67-78.

Received: 10 June 2004
Accepted: 08 December 2004