A Transaction Model for Multidatabase Systems

Timuçin Devirmiş and Özgür Ulusoy

Department of Computer Engineering and Information Science
Bilkent University
Bilkent, Ankara 06533, TURKEY

1 Introduction

In this paper, we present a new transaction model for multidatabase systems (MDBSs). This model captures the formalism and semantics of various extended transaction models and adopts them to an MDBS environment. The extended models constituting our transaction model are the nested transactions [4], the flexible transaction model that provides various dependency relations among transactions [7], and the model that involves a relaxed version of transaction atomicity, namely the semantic atomicity, to increase the level of concurrency [2], [5]. While including the semantics of all those transaction models, the global serializability in our execution model was ensured through the use of the ticketing method [3].

2 A Multidatabase Transaction Model

In a multidatabase environment, some subtransactions can be committed independent of its global transaction. If a subtransaction's effects on the database can be semantically undone by executing a compensating transaction, the subtransaction can be allowed to commit earlier. A subtransaction that reserves a seat in an airline reservation system is compensatable by a transaction that cancels the reservation. Another kind of commit independent transactions is the retriable transactions which eventually commit if they are retried a number of times. A retriable transaction can be committed later than the global transaction. Crediting a bank account is an example of retriable transactions. We will consider three transaction types (TT) in our model:

- Compensatable (C),
- Retriable (R), or
- Ordinary (O) (neither compensatable nor retriable).

In the following, we provide a formal definition for subtransactions processed in our MDBS and the dependency relation types among subtransactions.

Definition 1. A subtransaction S is a 2-tuple S=(TT,CT) where

- TT is the transaction type of S;
- CT is the set of compensating transactions of S, if TT is compensatable.

Definition 2. Let S_i and S_j be two subtransactions. We define four types of dependency relation between S_i and S_j.

- Precedence relation ($<$), $S_i < S_j$ means that S_j cannot begin execution until S_i successfully finishes its execution.
- Alternative relation ($\diamond$), $S_i \diamond S_j$ means that S_j and S_i are alternative of each other and any of them can be executed. It is also possible to execute them together, but only one of them should be committed.
- Preference relation ($\triangleright$), $S_i \triangleright S_j$ means that among two alternative subtransactions S_i and S_j, S_i is preferred to S_j. If they are executed together, S_j can be committed only if S_i fails. If they are not allowed to execute together, S_i should execute first, and if it fails, S_j can be executed.
- No-dependency relation ($\Box$), $S_i \Box S_j$ means that S_i and S_j can execute independently.

A global transaction in our model is syntactically a nested transaction with extended semantics. A global transaction consists of a set of child transactions each of which is either a subtransaction or again a global transaction. This transaction model can be represented as a tree where the internal nodes are global transactions and the leaf nodes are subtransactions. The height of a transaction tree can vary depending on the transaction complexity.

Definition 3. A global transaction G is a 3-tuple G=(ST,DT,TO) where

- ST is the set of global transactions and/or subtransactions that are the children of G;
- DT is the dependency type among the transactions in ST;
- TO is the total order on ST according to the dependency specified in DT.

3 An Execution Architecture for the Proposed Transaction Model

In our execution model, the local transactions are directly submitted to local database management systems (LDBSs), while global transactions use a common MDBS interface. A global transaction, submitted to the global transaction manager (GTM), is divided into a number of subtransactions, and each subtransaction is sent to the relevant site where the required data items reside. A set of application programs called *agents* is built on top of the LDBSs to act as an interface between GTM and each local site in controlling the execution of subtransactions.

The objectives of GTM are to avoid inconsistent retrieval of data, and to preserve global consistency and atomicity. The LDBS at each site ensures the local consistency and isolation properties by generating serializable schedules.

Global serializability can be provided by obtaining the information of relative serialization order of subtransactions at each local site and guaranteeing the same relative order at all those sites [6].

3.1 Ensuring Global Atomicity

We need to extend the traditional atomicity to capture the semantics of dependency relations among subtransactions. The execution of a global transaction G preserves the semantic atomicity, if the following conditions are satisfied:

- When a precedence or a no-dependency relation exists among its children, G can commit if all of its child transactions commit. If one of its child transactions is aborted, G is aborted and the other child transactions are either aborted or the effects of committed ones are undone.
- If an alternative or a preference relation exists, G can commit if one of its child transactions commits. When a child transaction commits, other child transactions that are executing are aborted.

The execution of a global transaction containing only ordinary children proceeds as follows.

- First, the global transaction is constructed with the initial execution state.
- GTM spawns the children of the global transaction according to the specified dependency type:
  - If either a no-dependency, or an alternative, or a preference dependency exists, all of the child transactions are created.
  - Otherwise (if a precedence relation is specified), the children are created on the basis of the given total order.
- If GTM reaches a leaf node in the nested transaction tree and creates a subtransaction, it submits the subtransaction to the corresponding site through the agents.
- When a subtransaction finishes its database operations, the agent of that site sends a ready-to-commit message to GTM.
- After receiving a ready-to-commit message for a subtransaction, GTM checks the dependency type associated with the parent of the subtransaction to find out what to do next.
  - If a precedence relation exists among its children, the next child transaction in the given order is created by GTM. If all of the child transactions enter the ready-to-commit state, the parent also enters the ready-to-commit state.
  - If an alternative relation exists, the parent enters the ready-to-commit state and GTM sends messages to the relevant agents to abort the other child transactions.
  - If a preference relation exists, the parent enters the ready-to-commit state if the completed subtransaction is the most preferred one. When the parent becomes ready to commit, GTM broadcasts the abort message for the other child transactions.

  - If a no-dependency relation exists, the execution state of the parent becomes ready-to-commit after all of its children enter the ready-to-commit state.
- If the root transaction enters the ready-to-commit state, GTM decides to commit or abort the transaction according to the concurrency control algorithm executed.
- After a commit or abort is issued for the root transaction, GTM broadcasts a message to child transactions down to the leaves of the transaction tree to commit or abort the subtransactions at local sites.

3.2 Ensuring Global Serializability

The global serializability is ensured in our execution model by employing a ticketing-based concurrency control for global transactions. The ticket values obtained by subtransactions are transferred to their parents up to the root transaction. GTM ensures the same relative serialization order at all sites of the global root transaction using the ticket values obtained. Two possible methods that can be used to control concurrent execution of global transactions are the optimistic ticketing method, and the conservative ticketing method [3]. Due to the space limitation the implementation details of these two methods in our execution model are not included in this paper. Interested readers are referred to [1], which also includes the details of the execution strategies for commit independent subtransactions. The performance implications of the proposed transaction model are also discussed in [1].

References

1. T. Devirmiş: Transaction Execution in Multidatabase Systems. M.S. Thesis in preparation, Department of Computer Engineering and Information Science, Bilkent University (1996)
2. A. Elmagarmid, Y.Leu, W. Litwin, M. Rusinkiewicz: A Multidatabase Transaction Model for Interbase. VLDB Conference (1990) 507–518
3. D. Georgakopoulos, M. Rusinkiewicz, A.P.Sheth: Using Tickets to Enforce the Serializability of Multidatabase Transaction. IEEE Transactions on Knowledge and Data Engineering **6** (1994) 166–180
4. J.E.Moss: Nested Transactions: An Approach to Reliable Distributed Computing. MIT Press (1985)
5. S. Mehrotra, R. Rastogi, H.F. Korth, A. Silberschatz: A Transaction Model for Multidatabase Systems. Technical Report TR-92-14, Department of Computer Science, University of Texas at Austin (1992)
6. M. Rusinkiewicz, P.Krychniak, A. Cichocki: Towards a Model for Multidatabase Transactions. Technical Report UH-CS-92-18, Department of Computer Science, University of Houston (1992)
7. A. Zhang, M. Nodine, B. Bhargava, O. Bukhres: Ensuring Relaxed Atomicity for Flexible Transaction in Multidatabase Systems. ACM SIGMOD Conference (1994) 67–78