On Rigorous Transaction Scheduling

Yuri Breitbart, Dimitrios Georgakopoulos, Member, IEEE, Marek Rusinkiewicz, and Abraham Silberschatz

Abstract—The concept of serializability is commonly used as a basic criterion for determining the correctness of concurrent execution of multiple transactions in database systems. Ensuring serializability is complicated, because transactions' serialization order does not need to correspond to their execution order. This anomaly is particularly troublesome in an environment consisting of multiple autonomous database systems in which global transactions are executed under the control of a multi-database transaction scheduler, and local transactions are executed under the control of the local transaction scheduler. The above difficulties could be avoided if the serialization order of transactions (which is needed to determine the correctness of the schedules) was always the same as their execution and commitment order (which can be controlled by the scheduler). In this paper we define the class of transaction scheduling mechanisms in which the transaction serialization order can be determined by controlling their commitment order. This class of transaction management mechanisms is important, because it simplifies transaction management in a multidatabase system environment.

Index Terms—Serializability, strict scheduling, rigorous scheduling, multidatabase transactions.

I. INTRODUCTION

The concept of serializability is commonly used as a basic criterion for determining the correctness of concurrent execution of multiple transactions in a Database Management System (DBMS). Numerous concurrency control mechanisms for ensuring serializability have been proposed in the literature. Most of them determine the (correct) serialization order by controlling the execution of transactions by either delaying the operations that may violate serializability, or by aborting the transaction that issues such operations.

In addition to serializability, a DBMS must ensure transaction atomicity and recoverability. Although the problems of concurrency control, atomicity, and recovery are frequently discussed separately, they are actually closely related. To simplify transaction management, the DBMS produces schedules that are not only serializable, but also have the following additional properties [3]:

1) Cascadelessness: Each transaction reads only data items written by committed transactions

2) Strictness: No data item may be read or written until the transaction that previously wrote it either commits or aborts.

The above properties simplify transaction management by avoiding cascading aborts and allowing the use of before images for database recovery.

Ensuring serializability is complicated because transactions' serialization order does not need to correspond to their execution order. For example, it is possible that transaction $T_1$ is submitted executed and committed completely before another transaction $T_2$ is submitted to the DDMS, yet $T_2$ would precede $T_1$ in the equivalent serialization order (schedule $S_1$ from Section II, for instance, delivers a specific example of this phenomenon). In many cases this anomaly can be tolerated, however, in some situations it presents a serious problem.

This anomaly is particularly troublesome in an environment consisting of multiple autonomous database systems. Global transactions in such environments are executed under the control of a multidatabase transaction scheduler. Since the local DBMS's involved in the execution of global transactions are autonomous, the multidatabase transaction scheduler can determine the serialization order of global transactions only by controlling the order in which their operations are submitted for execution at the local DBMS's. However, even when the execution of a transaction is completed, its serialization order is not always determined. Until recently this anomaly was not completely understood, and, as a result, a surprisingly large number of published solutions were not successful.

These difficulties could be avoided if the serialization order of transactions (which is needed to determine the correctness of the schedules) were always the same as their execution and commitment order (which can be controlled by the scheduler). In this paper we define the class of transaction scheduling mechanisms in which the transaction serialization order can be determined by controlling their commitment order. This class of transaction management mechanisms is important because it simplifies transaction management and allows the hierarchical composition of transaction management mechanisms.

The remainder of the paper is organized as follows. In Section II we define the notion of analogous execution and serialization orders of transactions and formally introduce the concept of strongly recoverable and rigorous execution schedules. We then prove that rigorous schedulers always produce analogous execution and serialization orders. In Section IV we show that the systems using the rigorous scheduling can be naturally incorporated in hierarchical transaction management mechanisms. In particular, we prove that several multidatabase transaction management mechanisms proposed in the litera-