MULTIVERSION CONCURRENCY CONTROL SCHEME
FOR A DISTRIBUTED DATABASE SYSTEM

- A TRIAL TO BREAK CONCURRENT UPDATE OF REDUNDANT COPIES -

Shojiro Muro*, Tadashi Mizutani**, and Toshiharu Hasegawa*

* Department of Applied Mathematics and Physics
Faculty of Engineering, Kyoto University, Kyoto 606, Japan
**Fujisawa Development Laboratory–IBM Japan
1, Kirihara-cho, Fujisawa, Kanagawa 252, Japan

Abstract – A new multiversion concurrency control scheme for a distributed database system is proposed in this paper. Each data object in our database model has two copies allocated in different sites in the system. Unlike the usual distributed database systems with redundant copies, these two copies are not concurrently updated, but only one is updated for a write request. For each data object, the copy with a newly updated value is called the new version and the other is called the old version. Since two versions are accessible for each read request to an object and concurrent update of two copies of each data object is not necessary, our scheme allows increased concurrency. Our concurrency control scheme employs both timestamp mechanism and locking mechanism with two different modes which is based on the (r,a,c)-locking protocol proposed by Bayer et al., and it grants a version for every read request without causing inconsistency. Transactions with write requests which would cause inconsistency are aborted. It has been proven that our concurrency control scheme works correctly; namely, it preserves consistency without deadlock or livelock among operations of the transactions in the system.

1. Introduction

In conventional database systems (abbreviated DBS), many users access shared data concurrently. Unless some kind of discipline is imposed on
users' transactions, data in the system may be modified in an unintoshed way. The reader is referred to [12] for problems that may arise. Concurrency control is the activity of synchronizing read and write operations issued from concurrently executed transactions on a shared database in order that it may realize a high level of concurrency without destroying database consistency caused by undesirable interactions among transactions. Particularly, the purpose of concurrency control is to produce an execution of operations with the same effect as (i.e., equivalent [11] to) a serial (noninterleaved) one. Such executions are called serializable (see, e.g., [8], [11], [23]). Many concurrency control schemes, most of which are based on serializability theory, have been proposed in this decade for both centralized DBS and distributed database system (abbreviated DDBS) models (see, e.g., [1], [10], [11], [18], [23], [26], [32], also see the references of the survey paper [6]).

It appears that the idea of using multiple versions of data objects was first tried in a Honeywell file system [13], and the first theoretical analysis of a multiversion DBS appeared in [30]. If we keep multiple versions of each data object, there is more likelihood of being able to grant read requests that arrive "too late", since older versions are saved for future read requests.

A number of important concepts and results concerning multiversion DBS's are presented in [30], and many multiversion concurrency control schemes have been recently investigated to achieve serializability by supporting multiple versions of the data objects (see, e.g., [1], [2], [5], [7], [9], [14], [16], [21], [24], [26], [29], [31]).

Although many concurrency control schemes have been proposed for DDBS's, they are usually complex and hard to understand (this sentence is quoted from the survey paper [6]). As what makes the problem hard, we list up the following two factors:

(F1) Users may concurrently access data object stored in many different sites in a system.

(F2) A controller at one site cannot instantaneously get the information of conflict relationship with other sites, which is mainly due to the communication delay between sites.

One of the most critical consistency problems of DDBS's is concurrent updating (writing) of multiple (redundant) copies of a data object, which is caused by (F2).

Multiversion concurrency control schemes have already been proposed in the environment of distributed systems (e.g., [2], [7], [9], [26], [29], [31]). However, in the models of these papers, versions of each