Concurrency Control in Distributed Databases and Advanced Database Applications

Database Reading Group
Henrique Paques
February 8, 2000

Concurrency Control in Distributed Database
(*Reference*: Ozsu, Chapter 11)

- The chapter reviews the concept for concurrency control in centralized databases.
- The serializability theory extends in a straightforward manner to the non-replicated (or partitioned) distributed databases. The schedule of transaction execution at each site is called a *local schedule*. If the database is not replicated and each local schedule is serializable, their union (called *global schedule*) is also serializable.
- In a replicated distributed database, the extension of the serializability theory requires more care. It is possible that the local schedules are serializable, but the mutual consistency of the databases is still compromised.
- The two-phase lock approach is further divided into the following categories:
  - **Centralized locking**: One of the sites in the network is designated as the primary site where the lock tables for the entire database are stored and is charged with the responsibility of granting locks to transactions
  - **Primary copy locking**: One of the copies of each lock unit (e.g., a table) is designated as the primary copy, and it is this copy that has to be locked for the purpose of accessing that particular unit.
  - **Decentralized locking**: The lock management task is shared by all the sites of a network. The execution of a transaction involves the participation and coordination of schedulers at more than one site. Each local scheduler is responsible for the lock units to that local site
- Timestamp-ordering protocols are classified into: basic, multi-version, and conservative.

Concurrency Control in Advanced Database Applications
(*Reference*: Naser S. Barghouti and Gail E. Kaiser. "Concurrency control in advanced database applications," ACM Computer Survey. 23(3) (Sep. 1991), pp. 269 - 317)

- The following are considered advanced database applications:
  - CAD/CAM
  - Software Development Environment (SDE)
  - Network Management
  - Financial instruments trading
- In traditional applications, the nature of the data and operations performed on the data are amenable to concurrency control mechanisms that enforce classical transaction model. Advanced applications have different kind of consistency constraints, and, in general, the classical transaction model is not applicable (e.g., real-time processing requirements).
- The key idea of the paper is the following:
  - Traditional DBMSs implements general-purpose concurrency mechanisms. They do not take into consideration the semantics of the application, mainly because it would be impossible to have a general-purpose tool that could accomplish that.

  - However, if the we know the semantics of the application, some non-serializable execution could be possible.
  - Another problem with these advanced applications is that they normally have *long transactions*, and this kind of transactions can lead to locking problems if we used the classical locking approaches (e.g., 2-phase lock).
- The new requirements for concurrency control in advanced applications are the following:
  - *Supporting long transactions*: Long transactions need different support than traditional short transactions. In particular, blocking a transaction until another commits is rarely acceptable for long transactions.
  - *Supporting use control*: In order to support user tasks that are non-deterministic and interactive in nature, the concurrency control mechanism should provide the user with the ability to start a transaction, interactively execute operations within it, dynamically re-structure it, and commit or abort it at any time.
  - *Supporting synergistic cooperation*: Cooperation among programmers to develop project components has significant implications on concurrency control. In CAD/CAM systems, SDEs,and other design environments, several users might have to exchange knowledge (i.e., share it collectively) in order to be able to continue. This kind of sharing and and exchanging is called *synergistic interaction*.
- The paper presents several concurrency models that try to provide the requirements above. However, none of them is able to provide all threes requirements at once.
- An example is Revision Control System (RCS), which implements the *checkout/check-in* mechanism (also called reserve/replace and reserve/deposit). It is used for version control in a software development environment.