15. Puustjarvi J. Using advanced transaction and workflow models in composing web services. In Adv. Comput. Sci. Technol., 2007.
16. Soparkar N., Levy E., Korth H.F., and Silberschatz A. Adaptive commitment for distributed real-time transactions. In Proc. Int. Conf. on Information and Knowledge Management, 1994, pp. 187–194.
17. Weikum G. Principles and realization strategies of multilevel transaction management. ACM Trans. Database Syst., 16(1): 132–180, 1991.

Semantics-based Concurrency Control

Krithi Ramamritham[1], Panos K. Chrysanthis[2]
[1]Indian Institute of Technology Bombay, Mumbai, India
[2]University of Pittsburgh, Pittsburgh, PA, USA

Definition

Specifications of data contain semantic information that can be exploited to increase concurrency. For example, two insert operations on a multiset object commute and hence, can be executed in parallel; further, regardless of whether one operation commits, the other can still commit. Applying the same rule, two push operations on a stack object do not commute and hence cannot be executed concurrently. Several schemes have been proposed for exploiting the semantics of operations have to provide more concurrency than obtained by the conventional classification of operations as *reads* or *writes*.

Key Points

In most semantics-based protocols, conflicts between operations is based on commutativity, an operation o_i which does not commute with other uncommitted operations will be made to wait until these conflicting operations abort or commit. Some protocols use operations' return value commutativity, wherein information about the results of executing an operation is used in determining commutativity, and some use the arguments of the operations in determining whether or not two operations commute. An example of the former, two increment operations on a counter object commute as long as they do not return the new or old value of the counter. An example of the latter, two insert operations on a set object commute as long as they do not insert the same item.

In the scheme reported in [1], non-commuting but *recoverable* operations are allowed to execute in parallel; but the order in which the transactions invoking the operations should commit is fixed to be the order in which they are invoked. If o_j is executed after o_i, and o_j is *recoverable relative to* o_i, then, if transactions T_i and T_j that invoked o_i and o_j respectively commit, T_i should commit before T_j. Thus, based on the recoverability relationship of an operation with other operations, a transaction invoking the operation sets up a dynamic commit dependency relation between itself and other transactions. If an invoked operation is not recoverable with respect to an uncommitted operation, then the invoking transaction is made to wait. For example, two pushes on a stack do not commute, but if the push operations are forced to commit in the order they were invoked, then the execution of the two push operations is serializable in commit order. Further, if either of the transactions aborts the other can still commit.

In [2] authors make an effort to discover, from first principles, the nature of concurrency semantics inherent in objects. Towards this end, they identify the dimensions along which object and operation semantics can be modeled. These dimensions are then used to classify and unify existing semantic-based concurrency control schemes. To formalize this classification, a graph representation for objects that can be derived from the abstract specification of an object is proposed. Based on this representation, which helps to identify the semantic information inherent in an object, a methodology is presented that shows how various semantic notions applicable to concurrency control can be effectively combined to improve concurrency. A new source of semantic information, namely, the ordering among component objects, is exploited to further enhance concurrency. Lastly, the authors present a scheme, based on this methodology, for *deriving* compatibility tables for operations on objects.

Cross-references

► ACID Properties
► Concurrency Control – Traditional Approaches

Recommended Reading

1. Badrinath B.R. and Ramamritham K. Semantics-based concurrency control: beyond commutativity. ACM Trans. Database Syst., 17(1):163–199, 1991.
2. Chrysanthis P.K., Raghuram S., and Ramamritham K. Extracting concurrency from objects: a methodology. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1991.