CHAPTER 7

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

This concluding chapter recounts the research contributions with a brief discussion on the merits of the proposed semantic multi-granular locking models for OODBMS and OODS and deadlock handling techniques for OODS. It also reveals a few open problems in the focused area of research.

7.1 Conclusions

Nowadays business domains and engineering applications are emerging as distributed applications. They require support of complex data modeling and long duration transactions. Application of object oriented paradigm provides support of complex data modeling. These domains are also continuously evolving in nature to provide better service to their clients. This requires provision of high concurrency and reliable consistency. Existing conventional concurrency control techniques are not equipped to meet these requirements of the recently emerging applications. Semantic concurrency control techniques exploiting the features of object oriented paradigm have shown better performance than the conventional concurrency control techniques.

In this research, semantic multi-granular lock models and deadlock handling techniques are proposed for distributed object environments. The developed models and algorithms were evaluated by conducting simulation experiments using the extended version of 007 benchmark. Semantic multi-granular lock models have been proposed for distributed object environments supporting stable as well as continuously evolving domains. A deadlock prevention algorithm has been proposed based on resource ordering and access ordering. The resource ordering and access ordering principles are based on the semantics of object oriented paradigm. Resource ordering policy breaks the circular wait. Access ordering policy eliminates the starvation in wealth and starvation in poverty. Existing probe based deadlock detection algorithm by Chandy1983 is very popular for its easy implementation. It uses probe messages to detect deadlock. It has the limitations of not being fault...
tolerant. It requires a separate resolution phase. These limitations are overcome by the proposed fault informant probe based algorithm and weight based victim selection algorithm. The fault informant probe based algorithm detects the status of the system as whether it is in deadlock or live lock state or having site failures. The weight based victim selection algorithm selects a victim based in the weights assigned to the desirable parameters of the system like throughput, fairness, resource utilization etc. This algorithm is used to dynamically select a victim by communicating the victim selected so far in the probe. Thus it eliminates the need for separate resolution phase. The features of the developed algorithms are narrated in the subsequent paragraphs.

(i) A Consistency Ensured Semantic Multi-Granular Lock Model (CESGML) is proposed for OODBMS implementing stable domains. In stable domains, the runtime transactions are more in number and design time transactions are rare. The model provides fine granularity of design time operations and ensures consistency of runtime transactions. It covers all the design time operations mentioned in the literature. It exploits the semantics of object oriented paradigm to provide rich set of lock modes by identifying the mutually exclusive operations. The need for access vectors is minimized by defining fine granular lock modes. Apart from the access vectors for attributes and methods, another access vector called Class Dependency Vector (CDV) is proposed for classes. This provides finer granularity for class definition and class relationship access. This improves parallelism for node level and link level design time operations.

Extended 007 benchmark for OODBMS is used to test its performance. The CESGML is compared with the latest existing techniques based on Semantic MGLM based on compatibility of relationships namely ORION and Semantic MGLM based on commutativity of operations namely Jun2000 scheme. CESMGL scheme is better than Orion by 68% and Jun2000’s scheme by 32.1%. ANOVA is also performed by comparing the three techniques by varying the number of transactions, design time to runtime ratio and varying the types of design time operations. In ANOVA and Duncan range test, it is found that CESGML provides very less response time when compared to the other two schemes.
(ii) Two semantic multi-granular lock models namely semantic MGLM using lock rippling and MGLM using access control lists are proposed for OODBMS implementing continuously evolving domains. In continuously evolving domains, more number of design time transactions arrives along with runtime transactions. The existing optimal models for OODBMS require apriori knowledge of the structure of classes. They use access vectors along with commutativity matrix to provide high concurrency. They incur the search and maintenance overhead of access vectors. This overhead increases linearly with the number of design time transactions. The proposed models eliminate the need for access vectors and hence reduce the response time when compared to the existing models.

Semantic MGLM using lock rippling extends the ORION locking scheme. In ORION scheme, the locking is always from the parent to the leaf for both runtime as well as design time transactions and intension locks are used to provide multi-granular lock support. In object oriented paradigm, there is an upward dependency from children to parents for runtime transactions. There is a downward dependency from parents to children in the case of design time transactions. This principle is not utilized in ORION. Further, intension locks do not convey what semantic operation is done on the fine granules and only S and X lock modes are provided for all read and write operations. In lock rippling, this lacuna is remedied by rippling the lock mode to convey the semantic operation taking place. A commutativity matrix is specified to define the conflicting operations. Then more than one lock can be placed on the same class as long as they are not conflicting. This helps to provide more concurrency than ORION. It does not require any access vectors for its operation. So the access vector overhead is nil in lock rippling mechanism.

Semantic MGLM using access control lists provides the same level of concurrency as in CESGML scheme without the limitations of access vectors, by splitting the lock table into three lists namely Available, Shared and Exclusive lists. This eliminates the need for maintaining vector tables along with lock table. The maintenance overhead is minimized as access vectors are not needed. In Available list, the attributes and methods of each class that are currently available are included. In Shared list, the attributes and methods of each class that are currently in shared (read) lock mode are included. In Exclusive list, the attributes and methods of each
class that are currently in exclusive (write) lock mode are included. The search time is
minimized as the lock table is split into three lists. Any transaction requires searching
only one of these lists instead of all of them. This reduces the search overhead to
roughly about one third. In order to save search time further, list search policies are
given. Exclusive lock mode is allowed for a requested resource only if the resource is
currently present in *Available* list. Shared lock mode is allowed, only if the resource is
not in *Exclusive* list.

Extended 007 benchmark is used to compare their performance with the
existing techniques. Semantic MGLM using lock rippling is better than CESGML by
14%. Semantic MGLM using access control list is better than CESGML by 21%. The
response time for lock rippling is more because of the coarse granularity of lock
modes for runtime transactions. However they do not need any apriori knowledge of
object structure. Thus, the proposed schemes perform better for continuously
evolving domains.

(iii) A semantic MGLM based on compatibility of relationships is proposed
for OODS. Lock modes and granularity of locks are proposed for attributes and
classes. Lock modes and granularity of locks are proposed for methods based on their
types and properties for each of the relationship namely inheritance, aggregation and
association. The granularity of runtime transactions is extended to attribute level. The
granularity of design time transactions are still coarse due to the limitations of
programming languages using which the domain is implemented.

(iv) A deadlock prevention algorithm is proposed for OODS based on
resource ordering. The resource ordering technique is based on the object semantics.
A formal model of the resource ordering technique is proposed using predicate
calculus. An expedient access ordering policy is also proposed to eliminate the
starvation in poverty and starvation in wealth. Informal and formal proofs are given.
Any DPA is expected to handle 3 conditions namely (1) Deadlock (2) Starvation on
Poverty (3) Starvation on Wealth. Deadlock is prevented in our algorithm by access
ordering. Transactions can access the resources only in specific order, i.e. resources
from lower IDs to higher IDs. Since FIFO ordering is followed, younger transactions
wait on older transactions. Hence, circular wait is broken. This ensures breaking of
cycles and hence deadlock is prevented. Transactions are satisfied on FIFO basis. This eliminates starvation in poverty. Starvation in poverty generally occurs when larger requests are kept pending permanently. This is because smaller transactions are favored over bigger transactions to increase throughput. However, in our algorithm, the transactions are served in FIFO basis. Hence, starvation in poverty is eliminated. Though, our algorithm favors FIFO ordering, when two transactions arrive at the same time, it favors the transaction requesting least number of resources. Hence, an expedient strategy is followed to speed up the computation, improve the resource utilization and alleviate starvation of wealth. Thus our algorithm has shown that deadlock prevention algorithm is possible for distributed object oriented systems.

(v) A probe based distributed deadlock detection algorithm using colored probes is proposed to provide fault tolerance. In this, Initiator always knows the status of probe whether deadlock or live lock or site failure. In existing algorithm, every non-faulty site tests other sites periodically for site failures. In the proposed algorithm the site failure is decided by acknowledgement messages. This improves the throughput of non-faulty sites. Fault identification is better than the existing fault diagnosis model, which needs 2t+1 processors to identify t failures. In the messaging mechanism, 2t-1 processors are enough to identify t failures. This algorithm is capable of detecting at most two failures, whereas in the existing algorithm only one site failure can be detected. The worst case message complexity is 4n where n is the number of transactions. This occurs when there is no site failure and deadlock occurs.

(vi) A weight based victim selection algorithm is proposed to dynamically select a victim based on the system parameters chosen for deadlock resolution. The victim selection algorithms are based on transaction attributes and resource attributes. The common transaction attributes or characteristics are age, history, code size, priority etc. The desirable attributes that could improve the system are higher throughput, better resource utilization and lesser deadlock latency. The desirable attributes in individual transaction execution is lower response time, fairness, no starvation and minimum roll back cost. It can be noticed that while each victim selection algorithm is optimal in one aspect, it is suboptimal in other aspects. Hence the proposed algorithm is based on assigning weights which can be configured based on user requirement. For example in real time systems, response time is more important than
other attributes In the proposed algorithm, each transaction is expected to possess an attribute list to maintain its rank in various aspects. Based on the effects of the various transaction attributes on the performance of the system and transactions, desirable system parameters are chosen and the transaction attributes influencing them are identified. Weights are assigned based on the choice of system parameters desired. The victim is thus computed in every site. When the probe circulates along the wait-for-edges in GWFG, the victim is updated by comparison. When the probe comes back to initiator, the victim ID is available for abortion.

7.2 Future Research Directions

In this research attempt is made to provide semantic concurrency control and deadlock handling for distributed object environments. The algorithms developed in this research can be further extended in the following aspects.

i. The semantic multi-granular lock models provide fine granularity using access vectors. They require prior knowledge of the structure of classes to provide high concurrency. There is a trade off between granularity and requirement of prior knowledge of class structure. Hence the models may be explored for providing fine granularity without the overhead of apriori knowledge of class structure.

ii. MGLM using lock rippling does not use any access vectors to provide concurrency control. However it provides coarse granularity as compared to MGLM using access control lists. Hence new lock modes may be explored to provide fine granularity of design time operations.

iii. Semantic MGLM proposed for OODS offers coarse granularity for design time transactions due to the limitations of programming languages. Hence possibilities may be explored to propose fine granule of design time operations.

iv. Semantic MGLM proposed for OODS is based on compatibility of relationships. A semantic MGLM based on commutativity of operations can be explored to compare their performance and identify the model that performs best.
v. Deadlock prevention is a proactive approach. Deadlock detection and resolution is reactive approach and hence favored. A deadlock detection algorithm can be explored by exploiting the semantics of objects. It may give better performance as it has done in semantic concurrency control and deadlock prevention.