Application of Radial Basis Function for Concurrency Control in CAD with KBMS

[1]P.Raviram
Research Scholar, Dept of Computer Science Engg.
Vinayaka Missions University
Salem, Tamilnadu – 636 308, INDIA
E:mail: ravirampedu@gmail.com
Tel: 00-91-9843968884

[2]Umarani S D
Research Scholar, Dept of Elec. & Comm. Engg
Government College of Engineering,
Salem – 636011, Tamilnadu, INDIA
Email id : umaraviram@gmail.com
Mobile : 00-91-9843688818.

[3]R.S.D.Wahidabanu
Professor and Head, Dept of Elec. & Comm. Engg
Government College of Engineering, Salem - 636 011, INDIA
Tel: 00-91-9443008886. E:mail: rsdwb@yahoo.com

Abstract—**Manufacturing database stores large amounts of interrelated data. Each designer accessing an entity tries to modify the design parameters meeting the requirements of different customers. Sister concerns of the same group of company will be modifying the data as per design requirements. When information is updated with new modification by different group of designers, what is the order in which modification of the data has to be allowed. When simultaneous access of the information is done, sufficient care has to be taken to maintain consistency of data. In this research work, we have proposed an artificial neural network (ANN) method for managing the locks assigned to objects and the corresponding transactions are stored in a data structure. The main purpose of using the ANN is that it will require less memory in storing the lock assignment information for objects. We have attempted to use Radial Basis Function (RBF) neural network for storing and managing lock information when multi users are working on computer Aided Design (CAD) database with screw as a main object. The memory requirements of the proposed method is minimal in processing locks during transaction.**

Keywords-transaction locks; Radial Basis Function; knowledge management; Concurrency Control

I. INTRODUCTION

Knowledge management in advanced database has been considered as an interesting research area in the recent past. Researchers concentrate on the integration of active and real-time database systems. New problems are evolved in concurrency control (CC) [1-4, 12-15] of real-time database systems. Conventional CC protocols are more concerned about the serializability but real-time database systems also focus on transaction deadlines.

Transaction is a series of actions, which accesses and changes contents of database. It is a basic unit of work on the database. Transaction transforms database from one consistent state to another. During this process, consistency may be violated [5-6]. The process of managing simultaneous operations on the database without having them interfere with one another is called concurrency. It prevents interference when two or more users are accessing database simultaneously. Even though two transactions may be correct in themselves, interleaving of operations may produce an incorrect result. Important problems caused by concurrency are lost update, inconsistent analysis and uncommitted dependency.

Incorrect updates are prevented by using locks allotted during transactions. A transaction must claim a shared (read) or exclusive (write) lock on a data item before read or write. Lock prevents another transaction from modifying item or even reading it, in the case of a write lock. Rules of locking are, if transaction has shared lock on item, can read but not update item, and if transaction has exclusive lock on item, can both read and update item, Reads cannot conflict, so more than one transaction can hold shared locks simultaneously on same item, Exclusive lock gives transaction exclusive access to that item.

II. PROBLEM DEFINITION

One of the shortcomings of traditional general purpose database management system (DBMS) is the inability to provide consistency in the database when long transactions are involved. The transaction with undefined time limit will not be able to identify if there is any violation of database consistency during the time of commitment, then it indicates a wastage of huge amount of time and resources. The activities of many users working on shared objects is not serializable. Existing two phase locking and optimistic transactions will result in deadlock in case of long transaction (LT). Two phase locking forces to lock resources for long time even after they have finished using them. Other transactions that need to access the same resources are blocked. The problem in optimistic mechanism with timestamping is it causes repeated rollback of transactions when the rate of conflicts increases significantly. We are using a RBF to manage the locks allotted to objects and locks are claimed appropriately to be allotted for other objects during subsequent transactions.

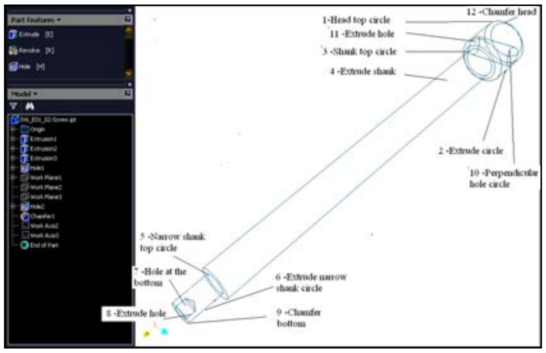

Figure 1. Screw

Inbuilt library functions for the screw (Figure 1) are available in standard CAD [7-11]. We assume initially the drawing of the screw is available in the central database. Subsequently due to customer requirements at different locations, the designer accesses the screw in the central database and modifies different features of the screw. During the process of modifications of different features, consistency of the data has to be maintained. In such case the following sequences of locking objects have to be done whenever a particular user access a specific feature of the screw. Each feature is treated as an object. The features are identified with numbers and corresponding feature names. In this explanation, O1 refers to object / feature marked as 1.

In general, the following sequences are formed when creating screw. Eventhough library files are available for screw drawing, customized drawing screw file is discussed. The major parameters involved in creating the screw are diameter (outer diameter / inner diameter), depth (hole) / length (extension or extrusion or projection) and chamfering (slanting). The various constraints that have to be imposed during modifications of features by many users on this screw are

During chamfering both length and diameter have to be write locked at the first transaction T1.

During diameter modification, chamfering has to be locked

During length or depth modification, diameter has to be locked.

This screw has two major entities.

1) *Features 1-2-10-11-12 (set 1)*
2) *3-4 (set 2)*
3) *3-5-6-7-8-9(set 3).*

Set 1, set 2 and set 3 can be made into individual drawing part files (part file 1 part file 2 and part file 3) and combined into one assembly file (containing the part files 1, 2 and 3 are intact). When the users are accessing individual part files , then transactions in part file 1 need not worry about the type of transactions in part files 2, 3 and vice versa. However, when the part files 1, 2 and 3 are combined into a single assembly file, then inconsistency in the shape and dimension of the set 1, set 2 and set 3, during matching should not occur. Hence, provisions can be made in controlling the dimensions and shapes with upper and lower limits confirming to standards. At any part of time when a subsequent user is trying to access locked features, he can modify the features on his system and store as an additional modified copy of the features with timestamping and version names (allotted by the user / allotted by the system).

III. RADIAL BASIS FUNCTION (RBF)

An artificial neural network is an abstract simulation of a real nervous system that contains a collection of neuron units, communicating with each other via axon connections. Such a model bears a strong resemblance to axons and dendrites in a nervous system. Due to this self-organizing and adaptive nature, the model offers potentially a new parallel processing paradigm. This model could be more robust and user-friendly than the traditional approaches. ANN can be viewed as computing elements, simulating the structure and function of the biological neural network. These networks are expected to solve the problems, in a manner which is different from conventional mapping. Neural networks are used to mimic the operational details of the human brain in a computer. Neural networks are made of artificial 'neurons', which are actually simplified versions of the natural neurons that occur in the human brain. It is hoped, that it would be possible to replicate some of the desirable features of the human brain by constructing networks that consist of a large number of neurons. A neural architecture comprises massively parallel adaptive elements with interconnection networks, which are structured hierarchically.

RBFs can be represented by a network structure, as any approximation based neural networks. Figure 1 shows the network representation of a Multi-Input Multi-Output RBF. The input layer distributes each element of the input vector to all the hidden nodes. Each node in the hidden layer contains one of the RBF's centres and applies the basis function ϕ to the euclidean distance between the input vector and its centre. Each node in the hidden layer produces a scalar value, based upon the centre it contains.

The outputs of the hidden layer nodes are passed to the output layer via weighted connections. Each connection between the hidden and output layers is weighted with the relevant coefficient. The node in the output layer sums its inputs to produce the network output. If an output of many dimensions is required, then several output nodes are needed, one for each output dimension. Several sets of coefficients will also be required, one set for the connections to each output node. The number of centres will however be unaffected. The training and testing procedure of RBF is given in Table1.

TABLE I. RBF ALGORITHM

(a) Training RBF

```
Step 1: Apply Radial Basis Function.
        No. of Input = 2
        No. of Patterns = 1
        No. of Centre = 3
      Calculate RBF as
        RBF = exp (-X)
      Calculate Matrix as
        G = RBF
        A = GT * G
      Calculate
        B = A-1
      Calculate
        E = B * GT
Step 2: Calculate the Final Weight.
        F = E * D
Step 3: Store the Final Weights in a File.
```

(b) Testing RBF

```
Step 1:. Read the Input
Step 2: Read the final weights
Step 2 Calculate.
      Numerals = F * E
Step 3: Check the output with the templates for the type of lock
```

IV. PROPOSED INTELLIGENT LOCKING STRATEGY

Let us assume that there are two users editing the screw. User1 edits O1 and hence O2 and O12 will be locked sequentially (Table 2). Immediately user2 wants to edit O2 and hence O1, O12, O10, O11 will be locked in sequence. However, already O12 is locked by the user1 and hence user2 cannot lock O12 once again during editing of O2. The user2 has to wait until user1 has released O1. However, user2 or any other user can try to access O10. Even now during subtransaction of O2 and O11, other user cannot complete locking as O2 is already locked by user1. Any other user can access O3 or O4 and not both. Similarly, any one group G6 / G7 / G8 can be accessed and edited and not more than one among G6 / G7 / G8.

TABLE II. SHAPE AND DIMENSION CONSISTENCY MANAGEMENT

Group	First feature	Remaining feature to be locked
G1	1	2 , 12
G2	5	6,7,8 ,9
G3	2	1, 10, 11, 12
G4	10	2,11
G5	3	4
G6	4	3
G7	6	5,8
G8	8	5,9,7

V. IMPLEMENTATION

The variables used for training the ANN about locks assigned to different objects are transaction id, object id, lock mode (Table 3).

Transaction id represents the client or any other intermediate transactions

Object id represents the entire feature or an entity in the file

Mode represents type of lock assigned to an object.

exclusive (X) mode. Data item can be both read as well as written.

shared (S) mode. Data item can only be read..

intention-shared (IS): indicates explicit locking at a lower level of the tree but only with shared locks.

intention-exclusive (IX): indicates explicit locking at a lower level with exclusive or shared locks

shared and intention-exclusive (SIX): the subtree rooted by that node is locked explicitly in shared mode and explicit locking is being done at a lower level with exclusive-mode locks.

An intention locks allow a higher level node to be locked in S or X mode without having to check all descendent nodes.

In Table3, column 1 represents the lock type. column 2 represents the value to be used in the input layer of the ANN in module 1. Column 3 gives binary representation of Lock type to be used in the output layer of module 1. The values are used as target outputs in the module during lock release on a data item.

This work uses four modules of algorithms which work using RBF given in Table 1. The modules given in Table 4 gives their usage for learning and finding the lock states. OML(Object, Mode, Lock) and OL (Object, Lock)

TABLE III. BINARY REPRESENTATION OF LOCK TYPE

Lock type	**(input layer representation numerical value).**	**Binary representation in target layer of the RBF**
Object Not locked	0	000
S	1	001
X	2	010
IS	3	011
IX	4	100

TABLE IV. MODULES USED FOR LEARNING THE LOCK STATUS OF AN OBJECT

Module	Name	Training / Testing	RBF Topology
1	OML	Training (Figure 2)	2{Transaction number and object id} x {no. of nodes in hidden layer} x 3{Lock value}
2	OML	Testing (Figure 3)	2{ Transaction number and object id } x (no. of nodes in hidden layer) x 3(Lock value)
3	OL	Training (Figure 4)	1{object id} x 2 {no. of nodes in hidden layer} x 3{lock value}
4	OL	Testing (Figure 5)	1{object id} x 2 {no. of nodes in hidden layer} x 3{lock value}

In the fourth column of this table, 3 values are given in the order, no. of nodes in the input layer, no. of nodes in the hidden layer which can be anything and no. of nodes in the output layer which is 3 (fixed)

OML training (Figure 2)

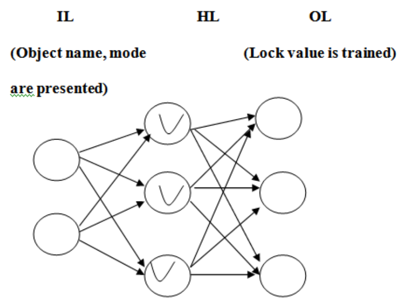

Figure 2. OML training

OML testing (Figure 3)

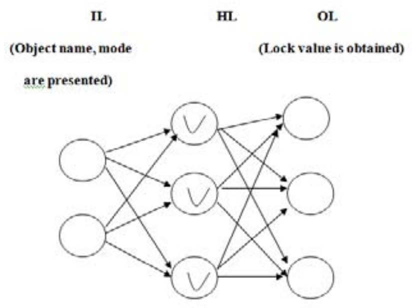

Figure 3. OML testing

OL training (Figure 4)

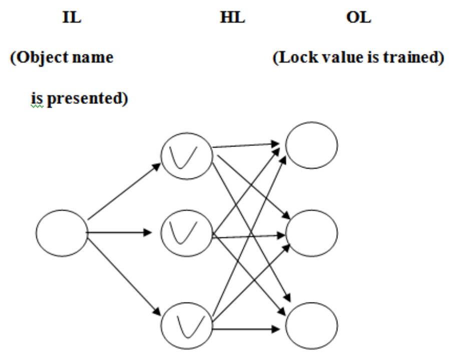

Figure 4. OL training

OL testing (Figure 5)

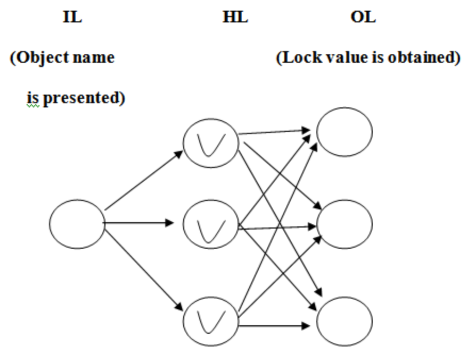

Figure 5. OL testing

VI. RESULTS AND DISCUSSIONS

Initially, user 1 and user 2 have opened the same screw file from the common database. The following steps shows sequence of execution.

Step 1:For the first time, a pattern is generated as soon as a user accesses the object. The pattern generated is given in Table 5. Find distance between input pattern and the centres. The centres are predefined based on the object information automatically. The centre is nothing but one of the patterns generated. T1 edits O1 with write mode. Table 5 shows pattern formed for the OML training.

TABLE V. FIRST TIME PATTERN USED FOR TRAINING OML RBF

Object number	**Input pattern**	**Target output pattern**
O1	[1 1]	[0 1 0]

Similarly based on the patterns generated, the following pattern is developed for the OL training

(Table 6). The patterns given in Table 5 and Table 6 are the same but for the number of inputs.

TABLE VI. FIRST PATTERN USED FOR TRAINING OL RBF

Object number	Input pattern	Target output pattern
O1	[1]	[0 1 0]

OML and OL are trained separately.

Step 2: The transaction manager locks objects mentioned in the third column of Table 2. Now, repeat step 1 with the patterns given in Table 7.

TABLE VII. ADDITIONAL PATTERNS USED FOR TRAINING OML RBF

Object number	Input pattern	Target output pattern
O1	[1 1]	[0 1 0]
O2	[2 1]	[0 1 0]
O12	[12 1]	[0 1 0]

Similarly train OL module with the patterns given in Table 8. The patterns in Table 7 and 8 are same but for the number of inputs.

TABLE VIII. ADDITIONAL PATTERNS USED FOR TRAINING OL RBF

Object number	Input pattern	Target output pattern
O1	[1]	[0 1 0]
O2	[2]	[0 1 0]
O12	[12]	[0 1 0]

Step 3: A new transaction T2 accesse O5. A pattern is formed to verify if lock has been assigned to O5 and its associated objects O12. Only when the locks are not assigned to O5 and O12 then T2 is allowed.

The following input patterns are presented to the OL testing module to find if the output [0 0 0] is obtained in the output layer. During OL testing, the final weights obtained during OL training will be used. Otherwise, it means that lock has been assigned to either O5 or O12 or both. In such case, transaction is denied for T2. Else the following patterns in Table 9 is presented in step 1

TABLE IX. ADDITIONAL PATTERNS USED FOR TRAINING OML RBF

Object number	Input pattern	Target output pattern
O1	[1 1]	[0 1 0]
O2	[2 1]	[0 1 0]
O12	[12 1]	[0 1 0]
O5	[5 2]	[0 1 0]

Step 4: To know the type of lock value assigned to an object and for a transaction, OML testing is used. OML testing uses the final weights created by OML training

The proposed RBF for lock state learning and lock state finding have been implemented using Matlab 7. Module 1 and Module 3 are trained using distance measure.

VII. CONCLUSION

An approach has been attempted to implement RBF in concurrency control to maintain consistency in the CAD database. A screw has been considered that contains 12 objects. The 12 objects have categorized into 8 groups. The transaction behavior and concurrency control by the two users on the 12 objects have been controlled using RBF network. We have found less memory required for storing lock information about the objects. The computational complexity is very minimal.

REFERENCES

[1] Pei-Jyun Leu and Bharat Bhargava, “Clarification of Two Phase Locking in Concurrent Transaction Processing,” IEEE Transactions on Software Engineering, Vol. 13, No 1, January (1988).

[2] A. A. Akintola, G. A. Aderounmu, A.U. Osakwe and M.O. Adigun, “Performance Modeling of an Enhanced Optimistic Locking Architecture for Concurrency Control in a Distributed Database System,” In: Journal of Research and Practice in Information Technology, 37 (4): 365-380, (2005).

[3] K. Vidyasankar, “A Non-Two-Phase Locking Protocol for Global Concurrency Control in Distributed Heterogeneous Database Systems,” IEEE Transactions on Knowledge and Data Engineering, Vol. 3, No. 2, June (1991).

[4] Abraham Silberschatz, “A Case for Non-Two-Phase Locking Protocols that Ensure Atomicity, IEEE Transactions On Software Engineering,” Vol. SE-9, No. 4, July (1983).

[5] C. Mohan, Donald Fussell, Zvi M. Kedem and Abraham Silberschatz, “Lock Conversion in Non-Two-Phase Locking Protocols,” IEEE Transactions On Software Engineering, Vol. SE-11, No. 1, January (1985).

[6] Bharat Bhargava, “Concurrency Control in Database Systems,” IEEE Transactions On Knowledge And Data Engineering, Vol. 11, NO. 1, Proceedings of DETC’00: January / February (1999).

[7] Raymond C. W. Sung, Jonathan R. Corney, and Doug E. R. Clark, “Octree Based Recognition Of Assembly Features,” Proceedings of DETC’00:, Baltimore, Maryland September 10-13, (2000).

[8] M. L. Brodie, B. Blanstein, U. Dayal, F. Manola and A. Rosenthal, “CAD/CAM Database Management,” IEEE Database Engineering, 7 (2): 12-20, (1984).

[9] Alexandtos Biliris and Huibin Zhao, “Design Versions in a Distributed CAD Environment,” IEEE, pp:354-359, (1989).

[10] M. A. Ketabchi and V. Berzins, “Modeling and Managing CAD Databases,” IEEE Computer, pp: 93-102, (1987).

[11] Hamideh Afsarmanesh and David Knapp, “An Extensible Object-Oriented Approach to Databases for VLSI / CAD,” Proceedings of VLDB, Stockholm, pp 13-24,(1985).

[12] Min Li, J.Y.H. Fuh, Y.F. Zhang and Shuming Gao, “Adaptive Granular Concurrency Control for Replicated Collaborative Feature Modeling,” 978 -1-4244-1651-6/08/ © IEEE(2008).

[13] Xue WANG, tie-min MA, YAN yan, qiu-yu ZHANG, yong LIU, “The research of Conflict-detection algorithm of concurrency control Based on Rough Set,” International Conference on Computer Science and Information Technology, 978-0-7695-3308-7/08 © 2008 IEEE (2008).

[14] Zongtao Zhao1, Jun Wei, Li Lin, and Xiaoning Ding1, “A Concurrency Control Mechanism for Composite Service Supporting User-Defined Relaxed Atomicity,” Annual IEEE International Computer Software and Applications Conference, 0730-3157/08 © IEEE(2008)

[15] Martin Kot, “Modeling Real-Time Database Concurrency Control Protocol Two-Phase-Locking in Uppaal,” Proceedings of the International Multiconference on Computer Science and Information Technology pp. 673–678, 978-83-60810-14-9/08 IEEE (2008).