The South African Institute of Computer Scientists and Information Technologists

The 1996 National Research and Development Conference

Interaction Conference Centre, University of Natal, Durban

26 & 27 September

Hosted by

The Department of Computer Science and Information Systems
University of Natal, Pietermaritzburg

Sponsored by IBM

PROCEEDINGS

Edited by Vevek Ram
The South African Institute of Computer Scientists
and
Information Technologists

Proceedings

of the

1996 National Research and Development Conference

Industry meets Academia

Interaction Conference Centre, University of Natal, Durban.
26 & 27 September

Edited by
Vevek Ram
FOREWORD

This book is a collection of papers presented at the National Research and Development Conference of the Institute of Computer Scientists and Information Technologists, held on 26 & 27 September, at the Interaction Conference Centre, University of Natal, Durban. The Conference was organised by the Department of Computer Science and Information Systems of The University of Natal, Pietermaritzburg.

The papers contained herein range from serious technical research to work-in-progress reports of current research to industry and commercial practice and experience. It has been a difficult task maintaining an adequate and representative spread of interests and a high standard of scholarship at the same time. Nevertheless, the conference boasts a wide range of high quality papers. The program committee decided not only to accept papers that are publishable in their present form, but also papers which reflect this potential in order to encourage young researchers and to involve practitioners from commerce and industry.

The organisers would like to thank IBM South Africa for their generous sponsorship and all the members of the organising and program committees, and the referees for making the conference a success. The organisers are indebted to the Computer Society of South Africa (Natal Chapter) for promoting the conference among its members and also to the staff and management of the Interaction Conference Centre for their contribution to the success of the conference.

On behalf of the Organising Committee
Vevek Ram
Editor and Program Chair
Pietermaritzburg, September 1996
Organising Committee

Conference General Chairs
Mr Rob Dempster and Prof Peter Warren (UNP)

Organising Chair
Dr Don Petkov (UNP)

Secretariat
Mrs Jenny Wilson

Program Chair
Prof Vevek Ram (UNP)

Program Committee
Prof Peter Wentworth, Rhodes
Dr Milan Hajek, UDW
Prof Derek Smith, UCT
Prof Anthony Krzesinski, Stellenbosch
Dr Don Petkov, UNP
Mr Rob Dempster, UNP
Prof Peter Warren, UNP
Table of Contents

Foreword i
Organising Committee ii
List of Contributors vi

Keynote Speaker

The Role of Formalism in Engineering Interactive Systems 1
M D Harrison and D J Duke

Plenary

Industry-Academic-Government Cooperation to boost Technological Innovation and People Development in South Africa 15
Tjaart J Van Der Walt

Checklist support for ISO 9001 audits of Software Quality Management Systems 17
A J Walker

The IS Workers, they are a-changin' 29
Derek Smith

Research

Examination Timetabling 35
E Parkinson and P R Warren

Generating Compilers from Formal Semantics 43
H Venter

Efficient State-exploration 63
J. Geldenhuys

A Validation Model of the VMTP Transport Level Protocol 75
H.N. Roux and P.J.A. de Villiers

Intelligent Systems

Automated Network Management using Artificial Intelligence 87
M Watzenboeck

A framework for executing multiple computational intelligent programs using a computational network 89
H L Viktor and I Cloete

A Script-Based prototype for Dynamic Deadlock Avoidance 95
C N Blewett and G J Erwin

Parallelism: an effective Genetic Programming implementation on low-powered Mathematica workstations 107
H. Suleman and M. Hajek

Feature Extraction Preprocessors in Neural Networks for Image Recognition 113
D Moodley and V Ram
Real-Time Systems

The real-time control system model - an Holistic Approach to System Design
T Considine

Neural networks for process parameter identification and assisted controller tuning for control loops
M McLeod and VB Bajic

Reference Model for the Process Control Domain of Application
N Dhevcharran, A L Steenkamp and V Ram

Database Systems

The Pearl Algorithm as a method to extract information out of a database
J W Kruger

Theory meets Practice: Using Smith’s Normalization in Complex Systems
A van der Merwe and W Labuschagne

A Comparison on Transaction Management Schemes in Multidatabase Systems
K Renaud and P Kotze

Education

Computer-based applications for engineering education
A C Hansen and P W L Lyne

Software Engineering Development Methodologies applied to Computer-Aided Instruction
R de Villiers and P Kotze

COBIE: A Cobol Integrated Environment
N Pillay

The Design and Usage of a new Southern African Information Systems Textbook
G J Erwin and C N Blewett

Teaching a first course in Compilers with a simple Compiler Construction Toolkit
G Ganchev

Teaching Turing Machines: Luxury or Necessity?
Y Velinov

Practice and Experience

Lessons learnt from using C++ and the Object Oriented Approach to Software Development
R Mazhindu-Shumba

Parallel hierarchical algorithm for identification of large-scale industrial systems
B Jankovic and VB Bajic
Information Technology and Organizational Issues

A cultural perspective on IT/End user relationships
A C Leonard 243

Information Security Management: The Second Generation
R Von Solms 257

Project Management in Practice
M le Roux 267

A Case-Study of Internet Publishing
A Morris 271

The Role of IT in Business Process Reengineering
C Blewett, J Cansfield and L Gibson 285

Abstracts

On Total Systems Intervention as a Systemic Framework for the Organisation of the Model Base of a Decision Support Systems Generator
D Petkov and O Petkova 299

Modular Neural Networks Subroutines for Knowledge Extraction
A Vahe and I Cloete 300

Low-Cost Medical Records System: A Model
O A Dami and T Seipone 301

A Methodology for Integrating Legacy Systems with the Client/Server Environment
M Redelinghuys and A L Steenkamp 302

Information Systems Outsourcing and Organisational Structure
M Hart and Kvavatzandis 303

The relational organisation model
B Laauwen 304

The Practical Application of a New Class of Non-Linear Smoothers for Digital Image Processing
E Cloete 305

A Technology Reference Model for Client/Server Software Development
R C Nienaber 306

The Feasibility Problem in the Simplex Algorithm
T G Scott, J M Hattingh and T Steyn 307

Author Index 309
List of Contributors

Vladimir B Bajic  
Centre for Engineering Research,  
Technikon Natal,  
P O Box 953  
Durban 4000

C N Blewett  
Department of Accounting  
University of Natal  
King George V Avenue  
Durban 4001

Justin Cansfield  
Department of Accounting  
University of Natal  
King George V Avenue  
Durban 4001

Tom Considine  
Apron Services (Pty) Ltd  
P O Johannesburg  
International Airport  
1600

Eric Cloete  
School of Electrical Engineering  
Cape Technikon  
Box 652  
Cape Town

I Cloete  
Computer Science Department  
University of Stellenbosch  
Stellenbosch  
7600

O A Daini  
Department of Computer Science  
University of Botswana  
Gaborone  
Botswana

Nirvani Devcharan  
Umgeni Water  
Box 9  
Pietermaritzburg  
3200

P J A de Villiers  
Department of Computer Science  
University of Stellenbosch  
Stellenbosch  
7700

Ruth de Villiers  
Department of Computer Science and  
Information Systems  
UNISA  
Box 392, Pretoria, 0001

G J Erwin  
Business Information Systems  
University of Durban-Westville  
Private Bag X54001  
Durban 4000

G Ganchev  
Computer Science Department  
University of Botswana  
PBag 0022  
Gaberone, Botswana

J Geldenhuys  
Department of Computer Science  
University of Stellenbosch  
Stellenbosch  
7700

Louise Gibson  
BIS, Dept Accounting & Finance  
University of Durban  
Pvt Bag X10  
Dalbridge 4014

Mike Hart  
Department of Information Systems  
University of Cape Town  
Rondebosch  
7700

M. Hajek  
Department of Computer Science  
University of Durban-Westville  
Pvt Bag X54001  
Durban 4000

A C Hansen  
Dept of Agricultural Engineering  
University of Natal  
Private Bag X01  
Scottsville 3209

J M Hattingh  
Department of Computer Science  
Potchefstroom University for CHE  
Potchefstroom 2520
Boris Jankovic  
Centre for Engineering Research  
Technikon Natal  
P O Box 953,  
Durban 4000

Paula Kotze  
Department of Computer Science and Information Systems  
UNISA  
Box 392  
Pretoria, 0001

J W Kruger  
Vista University  
Soweto Campus  
Box 359  
Westhoven 2124

A C Leonard  
Dept of Informatics  
University of Pretoria  
Pretoria 2000

Ben Laauwen  
Laauwen and Associates  
P O Box 13773  
Sinoville 0129

Mari Le Roux  
Information technology, development: project leader  
Telkom IT 1015  
Box 2753  
Pretoria 0001

P W L Lyne  
Dept of Agricultural Engineering  
University of Natal  
Private Bag X01  
Scottsville 3209

Rose Mazhindu-Shumba  
Computer Science Department  
University of Zimbabwe  
Box MP167  
Harare, Zimbabwe

Meredith McLeod  
Centre for Engineering Research,  
Technikon Natal,  
P O Box 953  
Durban 4000

D Moodley  
Computer Management Systems  
Box 451  
Umhlanga Rocks 4320

Andrew Morris  
P O Box 34200  
Rhodes Gift 7707

R C Nienaber  
Technikon Pretoria  
Dept of Information Technology  
Private Bag X680  
Pretoria 0001

E Parkinson  
Department of Computer Science  
University of Port Elizabeth  
Box 1600  
Port Elizabeth 6000

Don Petkov  
Department of Computer Science and Information Systems  
University of Natal  
PBag x01  
Scottsville 3209

Olga Petkov  
Technikon Natal  
Box 11078  
Dorpspruit 3206  
Pietermaritzburg

N Pillay  
Technikon Natal  
Box 11078  
Dorpspruit 3206  
Pietermaritzburg
A J Walker  
Software Engineering Applications  
Laboratory  
Electrical Engineering  
University of Witwatersrand  
Johannesburg

P Warren  
Computer Science Department  
University of Natal  
P/Bag X01  
Scottsville 3209

Max Watzenboeck  
University of Botswana  
Private Bag 0022  
Gaberone  
Botswana
A COMPARISON OF TRANSACTION MANAGEMENT SCHEMES IN MULTIDATABASE SYSTEMS

Karen Renaud and Paula Kotzé
Department of Computer Science and Information Systems,
University of South Africa, P.O. Box 392, Pretoria, 0001.
Email: {renaukv,kotzep}@riscl.unisa.ac.za

Abstract

In many applications today multiple pre-existing database systems are integrated into a single multiple database system called a multidatabase system. There are various characteristics of these systems which distinguish multidatabase systems from one another — autonomy, heterogeneity and distribution. One of the biggest problems with respect to transaction management in a multidatabase system is the question of how to maintain global concurrency control. Researchers have proposed various transaction management models for multidatabase systems. In this paper a core group has been formed by choosing the work of seven different research groups. Their respective approaches to transaction management will be appraised and the concurrency control methods they employ will be briefly discussed as well.

Introduction

A multidatabase system (MDBS) (also called a multidatabase (MDB)) is a system which is composed of autonomous or semi-autonomous, pre-existing database systems (DBSs), together with a multidatabase management system (MDMS) software layer, built on top of them, to control access to all data in all the component databases from the point of view of the user of the multidatabase system, while the component database systems still function independently.

Transactions submitted to the MDBS are called global transactions, while transactions submitted to the component DBSs are called local transactions. The MDMS facilitates access and manipulation of data at local sources (the component databases), possibly distributed among nodes of a computer network, by both users at the local database systems (LDBSs) as well as users of the MDBS. Transaction management in an MDB environment has many functions. It must undertake to ensure global consistency by ensuring that local and global transactions can execute concurrently without violating MDBS consistency, and maintain freedom from deadlocks. This must be done in the presence of local transactions and in the face of the inability of local database management systems (DBMSs) to coordinate execution of MDB transactions, under the assumption that no design changes are allowed in local DBMSs [Bright 1992]. This paper will examine the maintenance of MDB consistency during concurrent transaction execution.

One cannot, and indeed may not, assume that the LDBSs are aware of the existence of other participating LDBSs, and thus one cannot assume any communication between them. As a result, while local DBMSs can be depended upon to preserve consistency of the local database, it cannot be expected to ensure that the concurrent execution of local and global transactions preserve MDBS consistency.

What is meant by MDBS consistency? In most centralized DBMSs the serializability correctness criterion is used to ensure that the consistency of a database is maintained while concurrent transactions carry out operations on the database. Achieving serializability in an MDBS is not a trivial task. Some researchers have felt that global serializability is too stringent an expectation for MDBSs. Various researchers have proposed alternative approaches to MDB consistency [Barker 1990, Du 1989, Elmagarmid 1990, Rastogi 1993].

If global transactions were to be executed serially in the MDBS, one would not have a problem. The challenge in an MDBS is to achieve the maximum degree of concurrency while maintaining MDBS consistency. To do this, a software module, referred to as the global transaction manager (GTM), which is part of the MDBS, coordinates the execution of global transactions in an MDBS environment [Breitbart 1995]. A correctness criterion is chosen for the MDBS and the GTM will employ mechanisms to ensure that concurrent execution of transactions will not violate these criteria.
In order to give an idea of how various researchers have addressed the problem of MDBS transaction management, a core group of seven different transaction management schemes will be introduced. A brief synopsis of the intricacies of their mechanisms will be given, as well as details about the concurrency control mechanisms being employed. To set the stage, the characteristics of MDBSs will be discussed, identifying the autonomy characteristic as the one which can be used to differentiate schemes, and to evaluate the "goodness" of a proposed transaction management scheme. We will then take a look at each of the transaction management schemes which comprise our core group and then, after evaluation, put forward the optimal scheme which scores the best on the autonomy stakes.

Characteristics of Multidatabase Systems

There are three features which characterize MDBSs: distribution, heterogeneity and autonomy [Özsu 1990]. Distribution and heterogeneity are self-explanatory but autonomy, the most relevant characteristic, deserves a short discussion. Autonomy indicates the degree to which individual component databases in an MDBS can operate independently.

The preservation of local autonomy is both desirable and necessary in an MDBS since applications which have been developed prior to integration should continue to run afterwards too. Local DBMSs should also have the same measure of control over local databases after integration, while participating DBSs should be added to, or removed from, the MDBS with ease [Bradshaw 1993, Rastogi 1993]. Barker [Barker 1994] has set out a number of guidelines for the quantification of autonomy in MDBSs. He proposes a method for quantifying the autonomy violation of a particular transaction management scheme and arriving at a single value indicating the autonomy violation. A zero value would indicate no violation, while the ceiling value of 3 would indicate that autonomy had been violated to the maximum extent.

Using these measurements, the terms fully autonomous, semi-autonomous and non-autonomous can be defined.

1. An MDBS is said to be fully autonomous if the individual DBSs making up the system are stand-alone DBSs systems that are unaware of the existence of other component DBSs. The value for the autonomy violation would be 0.

2. An MDBS is said to be semi-autonomous if the component DBSs can operate independently but have decided to participate in an MDBS in order to make their local data shareable. They typically require certain changes to be made to their DBMSs or to the data in order to participate in the MDBS. The autonomy violation would probably be midway between 0 and 3.

3. An MDBS is said to be non-autonomous if a single image of the entire database is available to any user who wants to share the information which may reside in the MDBS. The individual DBSs will typically not operate independently. The overall autonomy violation would probably be close to 3.

Integrating Various Concurrency Control Methods

The various local DBMSs integrated by the MDMS may, and must be assumed to, use different concurrency protocols. The concurrency control mechanism in an MDBS has to be able to synchronize global transactions with purely local, autonomous transactions which are under the control of the local DBMS and ensure that the consistency of the database is maintained. Global transactions will typically address more than one of the component DBSs. The global transaction is therefore broken up into a set of subtransactions, each of which operates on a single DBS. To maintain consistency, either all these subtransactions must commit or all must abort.

The autonomy requirement means that once the global transaction submits a subtransaction to the local DBMS, it effectively relinquishes control over it. The local DBMS will assume
all responsibility and will decide whether to commit or roll-back the transaction. Hence, some local DBMS could commit one subtransaction and another could abort another subtransaction of the same global transaction, thereby destroying the atomicity of the global transaction and compromising the consistency of the MDBS. One of the biggest problems arises when a global transaction and a local transaction work on the same data item. The local DBMS will handle the conflict by causing the transactions to execute serially, but this could cause the resulting execution of the global transactions to be non-serial — as is shown in the following example.

Example

In this example the following notation will be used: \( GT_n \) signifies global transaction \( n \), \( r_n(x) \) denotes a read from data item \( x \) by local transaction \( n \), \( L_n \) denotes local transaction \( n \), \( LDB^n \) indicates LDBS number \( n \), \( w_n(x) \) denotes a write to data item \( x \) by transaction \( n \), and \( c_n \) denotes that transaction \( n \) has decided to commit. \( LS_n \) refers to the local schedule at DBS number \( n \) — showing the order in which transaction operations were executed at the local system.

Consider the following two non-conflicting global transactions:

\[
GT_1 : r_1(d); r_1(s); \quad GT_2 : r_2(e); r_1(t);
\]

In addition, consider the following local transactions; \( L_3 \) executing at \( LDB^1 \) and \( L_4 \) executing at \( LDB^2 \).

\[
L_3 : w_3(d); w_3(e); \quad L_4 : w_4(s); w_4(t);
\]

Now consider a history in which transaction \( GT_1 \) first executes at sites \( LDB^1 \) and \( LDB^2 \) followed by the execution of transaction \( GT_2 \) at both \( LDB^1 \) and \( LDB^2 \). It is possible for the local transactions \( L_3 \) and \( L_4 \) to execute in such a manner that \( GT_1 \) is serialized before \( GT_2 \) at \( LDB^1 \), while \( GT_2 \) is serialized before \( GT_1 \) at \( LDB^2 \). For example:

\[
LS^1 : r_1(d); c_1; w_3(d); w_3(e); c_3; r_2(e); c_2;
\]
\[
LS^2 : w_4(s); r_1(s); c_1; r_2(t); c_2; w_4(t); c_4;
\]

As far as the GTM is concerned, global transactions \( GT_1 \) and \( GT_2 \) are executed serially. At \( LDB^1 \), the resulting execution is serial: \( GT_1, L_3 \) and \( GT_2 \). At \( LDB^2 \), the resulting execution is also serial: \( GT_2, L_4 \) and \( GT_1 \).

However, the global execution is non-serializable because, to be serializable, \( GT_1 \) should always precede \( GT_2 \), or vice versa.

In order to integrate heterogeneous local concurrency control algorithms to satisfy the chosen correctness criterion, two problems have to be considered [Yun 1993] — how can a global transaction be processed when it violates either serializability or some alternative correctness criterion, and how can an indirect conflict introduced by a local transaction be managed.

The traditional approaches to integrating heterogeneous concurrency control algorithms can be classified into two groups — bottom-up and top-down approaches [Breitbart 1995]. The bottom-up (optimistic) approach collects local information from each local site at the global level and thereafter checks for global serializability or for an alternative criterion. The top-down (pessimistic) approach maintains a global serialization order at local sites, which has been determined already at the global level.

Mullen et al [Mullen 1992] has proved that it is impossible to synchronize local and global transactions while still preserving local autonomy. In other words, some autonomy has to be sacrificed in order to maintain MDBS consistency in the presence of concurrently executing transactions. One either has to restrict the types of global transactions allowed to execute, or impose restrictions of some sort on the structure of the local concurrency control mechanisms.

In line with this, most existing MDBSs support data retrieval only — they only allow retrieval of data by global transactions and do not have any concurrency control schemes to
all responsibility and will decide whether to commit or roll-back the transaction. Hence, some local DBMS could commit one subtransaction and another could abort another subtransaction of the same global transaction, thereby destroying the atomicity of the global transaction and compromising the consistency of the MDBS. One of the biggest problems arises when a global transaction and a local transaction work on the same data item. The local DBMS will handle the conflict by causing the transactions to execute serially, but this could cause the resulting execution of the global transactions to be non-serial — as is shown in the following example.

**Example**

In this example the following notation will be used: GTn signifies global transaction n, rn(x) denotes a read from data item x by local transaction n, Ln denotes local transaction n, LDBn indicates LDBS number n, wn(x) denotes a write to data item x by transaction n, and cn denotes that transaction n has decided to commit. LSn refers to the local schedule at DBS number n — showing the order in which transaction operations were executed at the local system.

Consider the following two non-conflicting global transactions:

GT1 : r1(d); r1(s);  GT2 : r2(e); r1(t);

In addition, consider the following local transactions; L3 executing at LDB1 and L4 executing at LDB2.

L3 : w3(d); w3(e);  L4 : w4(s); w4(t);

Now consider a history in which transaction GT1 first executes at sites LDB1 and LDB2 followed by the execution of transaction GT2 at both LDB1 and LDB2. It is possible for the local transactions L3 and L4 to execute in such a manner that GT1 is serialized before GT2 at LDB1, while GT2 is serialized before GT1 at LDB2. For example:

LS1 : r1(d); c1; w3(d); w3(e); c3; r2(e); c2;
LS2 : w4(s); r1(s); c1; r2(t); c2; w4(t); c4;

As far as the GTM is concerned, global transactions GT1 and GT2 are executed serially. At LDB1, the resulting execution is serial: GT1, L3 and GT2. At LDB2, the resulting execution is also serial: GT2, L4 and GT1.

However, the global execution is non-serializable because, to be serializable, GT1 should always precede GT2, or vice versa.

In order to integrate heterogeneous local concurrency control algorithms to satisfy the chosen correctness criterion, two problems have to be considered [Yun 1993] — how can a global transaction be processed when it violates either serializability or some alternative correctness criterion, and how can an indirect conflict introduced by a local transaction be managed.

The traditional approaches to integrating heterogeneous concurrency control algorithms can be classified into two groups — bottom-up and top-down approaches [Breitbart 1995]. The bottom-up (optimistic) approach collects local information from each local site at the global level and thereafter checks for global serializability or for an alternative criterion. The top-down (pessimistic) approach maintains a global serialization order at local sites, which has been determined already at the global level.

Mullen et al [Mullen 1992] has proved that it is impossible to synchronize local and global transactions while still preserving local autonomy. In other words, some autonomy has to be sacrificed in order to maintain MDBS consistency in the presence of concurrently executing transactions. One either has to restrict the types of global transactions allowed to execute, or impose restrictions of some sort on the structure of the local concurrency control mechanisms.

In line with this, most existing MDBSs support data retrieval only — they only allow retrieval of data by global transactions and do not have any concurrency control schemes to
co-ordinate the execution of global transactions. Even with this restriction, the problem of dirty, or unrepeatable reads, must be addressed.

Transaction Management Approaches

This section introduces the research done into transaction management in a selected core group of schemes. This is by no means a complete list but serves to give an indication of related work done in the area of transaction management in MDBSs.

Barker and Özsu’s basic MDB model

Barker and Özsu [Barker 1990, Barker 1991, Özsu 1991] propose a very basic MDB model which is illustrated in Figure 1. This model serves to give a good basic understanding of MDBS architecture. The MDBS consists of various LDBSs, each having its own DBMS, each of which manages a possibly different type of LDBS. The MDMS provides a layer of software that runs on top of these LDBSs and allows users to access the various DBSs. The MDMS layer consists of a GTM, a global scheduler and a global recovery manager. Each DBMS has its own transaction processing components: the local transaction manager, the local data manager, and a local scheduler. The MDMS is simply seen as another user from which transactions are received and to whom results are presented.

Barker [Barker 1990] proposes a new correctness criterion called m-serializability which is an extension of serializability theory. MDB serialization graphs are developed to make it easy to determine when an MDB history is m-serializable. Barker and Özsu require strict schedules to be generated. The autonomy quantification for Barker and Özsu’s model is shown in Figure 2.
Table 1

<table>
<thead>
<tr>
<th>Modification Dimension</th>
<th>System</th>
<th>0.75</th>
<th>Reliability protocol requires strictness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data</td>
<td>0</td>
<td>Data remains untouched</td>
</tr>
<tr>
<td></td>
<td>Design</td>
<td>X</td>
<td>Not discussed</td>
</tr>
</tbody>
</table>

\[ \bar{m} = \sqrt{0.75^2 + 0.5^2} = 0.75. \]

| Execution Dimension   | Local Transaction | 0    | Execute as usual                       |
|                       | Global Transaction| 0.5  | Handshake required at commit point      |

\[ \varepsilon = \sqrt{0.5^2 + 0.5^2} \times \sqrt{1.5} = 0.612. \]

| Information Exchange Dimension | Execution | 0.25 | Failures require MDMS to query DBMS |
|                                | Data      | X    | Not discussed                         |
|                                | Schema    | 0    | External Schema only                  |

\[ \iota = \sqrt{0.25^2 + 0.612^2} = 0.25. \]

The overall autonomy violation is
\[ \sqrt{0.75^2 + 0.612^2 + 0.25^2} = 1.0. \]

Figure 2: Autonomy Quantification for Barker and Özsu’s Scheme

Figure 3: Pu’s MDB transaction processing model

[Pu 1988, p146]

Pu’s hierarchy of superdatabases

Pu [Pu 1988] describes MDBs in terms of a hierarchy of superdatabases as illustrated in Figure 3. Each participating DBS (called an element database) can be pictured as the leaves on a tree and each internal node as a superdatabase that manages the element databases in a hierarchical structure. Each element database operates independently but global activities are managed at the node level of the tree. Transactions that cross multiple element databases are called supertransactions and are posed against a superdatabase. Superdatabases utilize serializability as a transaction correctness criteria. Serializability is ensured by having each element database provide the superdatabase with information about the ordering of its local transactions by means of what he calls an O-element. Pu claims this is not necessary when element databases provide strict schedules. He assumes that each local DBMS can be modified to return the serialization order of each global transaction executed at the local site to the GTM. The GTM then uses the serialization orders from all the local sites to validate the execution of a global transaction. The approach provides a high level of concurrency at the expense of local autonomy.

There are some problems with Pu’s approach. Formation of the O-element ordering requires that the local site keep the superdatabase informed of decisions made locally. This violates local autonomy. The superdatabase, and not the local databases, makes arbitrary decisions about whether transactions may commit, which also violates autonomy. The overall autonomy viola-
tion for Pu's scheme is 1.25.

**Breitbart et al's work**

Breitbart et al [Breitbart 1995] have done research into many aspects of MDBSs. Their approach is based on splitting up data into mutually exclusive groups of *locally* and *globally updateable* data. Breitbart and co-workers propose a transaction processing model where the software module includes a GTM at the MDB site, and a set of *servers*, one associated with each participating DBS. The MDB transaction processing model is illustrated in Figure 4. The GTM submits operations to the local DBMSs through the server, which acts as a liaison between the GTM and the local DBMS. Operations belonging to a single global subtransaction are submitted to the local DBMS by the server as a single local transaction.

A major drawback of this scheme is that it requires local transactions not to update certain data items [Mehrotra 1993, Rastogi 1993]. The overall autonomy violation for Breitbart et al's scheme is 1.0897.

**Elmagarmid et al's work**

Elmagarmid et al [Elmagarmid 1986a, Elmagarmid 1987, Elmagarmid 1986b] have done much work in the area of transaction management in MDBSs. They have defined a correctness criterion called *quasi-serializability* which is a weaker form of serializability. They also present a framework for designing concurrency control control protocols using a top-down approach [Elmagarmid 1987]. This means that the global serialization order of global transactions must be determined at the global level before their being submitted to the local sites. They present two mechanisms for ensuring global serialization at the local sites. The first controls the submission of global subtransactions by using a stub process, and the second controls the execution of global subtransactions by modifying local schedulers. The overall autonomy violation in the stub approach is 1.116 while in the modification of the local scheduler approach the autonomy violation is 1.73.

**Chen et al's distributed MDMS**

Chen et al [Chen 1993] extended Elmagarmid et al's work by proposing a *distributed* MDMS which is not vulnerable to failures and thus addressing the reliability and recovery aspects of the model. The regular architectures which have been described up to now have a central node
which, if it fails, incapacitates the whole system. The MDMS described by Chen et al consists of a GTM and a set of interfaces located at each site. The GTM controls execution of all MDMS transactions. For each MDMS transaction $T_i$, a GTM process $GTM_i$, which is responsible for the consistent and reliable execution of $T_i$, is issued. $GTM_i$ is therefore coincident with the life cycle of $T_i$. The interface accepts and schedules the execution order of subtransactions on the local system where it resides and creates a server procedure for each subtransaction in the system. The server is coincident with the lifecycle of the subtransaction.

The architecture of Chen et al's model is illustrated in Figure 5. Before a transaction is executed, it requests all corresponding MDMS interfaces to arrange the scheduling order of its subtransaction on the corresponding LDBSs so as to prevent any MDMS inconsistencies its execution may cause. When executing a global transaction, a GTM process only interacts with relevant MDMS interfaces, without the need to communicate with other GTM processes. This scheme makes no assumptions about the characteristics of the underlying LDBSs but rather exploits those characteristics in order to achieve global database consistency.

The disadvantages of this approach are that performance is lowered by additional network delays caused by the additional network traffic. This scheme also reduces concurrency compared to other algorithms. The network delays can be alleviated by high speed networks and the reduced concurrency is offset by the fact that no global transactions will be aborted due to deadlocks or nonserializable executions [Bukhres 1993]. The overall autonomy violation for Chen et al's scheme is 0.829.

Kang and Keefe's decentralized GTMs
Kang and Keefe [Kang 1993] propose an MDB model where the GTM is totally decentralized. Their model caters for multiple versions and is illustrated in Figure 6. This scheme differs from Chen et al.'s scheme [Chen 1993] because in their scheme the GTM is located at the site where the transaction is submitted and that GTM communicates with interfaces at all the other local database sites, whereas Kang and Keefe's scheme locates a GTM at each local site which does all the work — there are no servers. At each site there is a GTM accepting global transactions from users and receiving subtransactions from other GTMs via the network. The GTM maintains a global directory and therefore can determine the appropriate sites at which a global transaction will execute.

Kang and Keefe's scheme also partitions data items which once again cal autonomy. They propose a distributed strict timestamp ordering scheme (DSTO) which is globally serializable. In DSTO, each global transaction is assigned a unique global timestamp when it starts. Each subtransaction carries the parent's timestamp. The GTM at each site executes strict timestamp ordering (TO). Strict TO blocks transactions attempting to read or write an object until the transaction that previously wrote it has either committed or aborted. The GTM ensures that conflicting operations are executed at the local site in global timestamp order by aborting transactions whose operations arrive too late. All global subtransactions are required to take-a-ticket (ticket being an object not updateable by local transactions). They only assume that the local data manager at each site outputs serializable and cascadeless schedules. Kang and Keefe prove that the DSTO scheme produces globally serializable histories in the face of failures and also prove that the scheme is deadlock free. The overall autonomy violation for Kang and Keefe's scheme is 0.93.

Garcia-Molina and Salem's sagas

Garcia-Molina and Salem [Garcia-Molina 1987] have proposed a nested transaction model intended to deal with long lived transactions. Their model uses nested transactions called sagas, with only two levels of nesting. A saga is not executed as an atomic unit. This means that the results of a subtransaction's execution are visible as soon as it commits and not only after commitment of the entire saga. Sagas are written so that they are interleavable with any other transactions, which makes concurrency control at the saga level unnecessary. Because of this design factor, the introduction of local transactions does not cause any concurrency control problems. In this model two assumptions are made which make this approach unsuitable in MDBS. Firstly, the model is not applicable to all MDB environments since it may be too re-
strictive to require that sagas be interleavable with other transactions. Secondly, it may not always be possible to write the compensating transactions that this model requires. The overall autonomy violation is 0.93.

Evaluation and Appraisal

The transaction management and concurrency control scheme proposed by the researchers in each of the research efforts which comprise the core group is summarized in Table 1. Assessing this table reveals the optimal approaches for transaction processing and concurrency control in MDBs.

The optimal MDB transaction processing model

After due consideration of the research done in this field, we have decided on the distributed GTM model presented by Chen et al [Chen 1993] as the most promising transaction management scheme. The reasons for this are that the GTM is distributed, failure resistant, and allows considerable leeway in how global transactions are handled. The interface can either send transaction operations to the local DBMS an operation at a time, or send through predetermined service requests to the DBMSs at the sites. It is also important to note that no assumptions are made about the local sites involved in the MDBS. Most importantly, LDBS autonomy is maintained and new DBSs are very easily added to and removed from the MDBS. A strong recommendation is that this scheme has been successfully implemented in the InterBase system at Purdue University.

Global concurrency control

We have reviewed various concurrency control schemes in this paper. Most of them suffer from one or more problems. They expect specific conditions to be satisfied in the LDBSs, or they are not failure resilient, or they violate local database autonomy to a lesser or greater degree, or they allow local transactions to only update a portion of the available data items, or they generate unacceptably high overhead, or they expect users to specify correct interleavings of subtransaction operations, or they could result in global deadlock, or they relax the atomicity requirement of transactions.
We propose the use of the customized global concurrency control algorithm as outlined in [Chen 1993], since it was designed for the transaction management scheme designed by Chen et al and because it achieves global serializability with the minimum violation of local autonomy. The algorithm utilizes the semantics of global transactions and the concurrency control strategies of the underlying LDBSs to customize a global concurrency control algorithm. It guarantees that the relative serialization order (RSO) of global transactions on different sites is consistent with their pre-determined relative scheduling order, ensuring that sites have the same RSO at all sites. It has been proved by Breitbart and Silberschatz [Breitbart 1988] that when global transactions have the same RSO at all sites, global serializability is preserved in the presence of local transactions.

Summary

In this article the transaction management schemes in the core group of research into transaction management and global concurrency control in MDBSs have been appraised, and the relative strengths and weaknesses of each scheme have been outlined. The optimal transaction processing model which scores well in the autonomy violation stakes, and is also reliable and failure resistant, has been proposed.

References


