SPECIAL ISSUE

7th SA COMPUTER RESEARCH SYMPOSIUM
The South African Computer Journal
An official publication of the South African Computer Society and the South African Institute of Computer Scientists

Editor
Professor Derrick G Kourie
Department of Computer Science
University of Pretoria
Hatfield 0083
Email: dkourie@dos-lan.cs.up.ac.za

Subeditor: Information Systems
Prof John Schochot
University of the Witwatersrand
Private Bag 3
WITS 2050
Email: 035ebrs@witsvma.wits.ac.za

Productio... Editor
Prof G de V Smit
Department of Computer Science
University of Cape Town
Rondebosch 7700
Email: gds@cs.uct.ac.za

Editorial Board
Professor Gerhard Barth
Director: German AI Research Institute
Professor Pieter Kritzinger
University of Cape Town

Professor Judy Bishop
University of Pretoria
Professor F H Lochovsky
University of Toronto

Professor Donald Cowan
University of Waterloo
Professor Stephen R Schach
Vanderbilt University

Professor Jürg Gutknecht
ETH, Zürich
Professor S H von Solms
Rand Afrikaanse Universiteit

Subscriptions
Annual Single copy
Southern Africa: R45,00 R15,00
Elsewhere $45,00 $15,00
to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685

Computer Science Department, University of Pretoria, Pretoria 0002
Phone: (Int)+(012)+430-2504 Fax: (Int)+(012)+43-6454 email: dkourie@dos-lan.cs.up.ac.za
SPECIAL ISSUE - 7th SA COMPUTER SYMPOSIUM

PREFACE

When the first SA Computer Symposium was held at the CSIR in the early eighties, it was unique. There was no other forum at the time for the presentation of research in computer science. In the intervening decade, conferences, symposia and workshops have sprung up in response to demand, and now there are several successful ventures, some into their third or fourth iteration. Each of these addresses a specific topic - for example, hypermedia, expert systems, parallel processing or formal aspects of computing - and attracts a specialised audience, well versed in the subject and eager to learn more. For the main part, the proceedings are informal, and certainly not archival.

SACRS, though, is still unique, in that it deliberately covers a broad spectrum of research in computing, and in addition, seeks to provide a lasting record of the proceedings. To achieve the second aim, we negotiated with the SA Institute of Computer Scientists for the proceedings to form a special issue of the SA Computer Journal, and the copy you have in front of you is the result. The collaboration between the symposium committee and the journal’s editorial board placed high standards on the refereeing and final presentation of the papers, to the symposium’s benefit, while we were still able to maintain a fresh, audience-oriented approach to the selection of papers.

This is SACR's first such special issue, and the largest issue (at 145 pages) to date. We hope that it is only the beginning of future such collaborations.

In all 29 papers were received, all were refereed twice, and 19 were chosen for presentation by the programme committee. All the papers were thoroughly revised by the authors on the basis of the referee’s comments, and the committee’s suggestions aimed at making the material more accessible to a broadly-based audience. Papers had to be new, and not to have been presented elsewhere, a requirement that is still unusual within the SA conference round.

A third goal of SACRS has been to invite keynote speakers, usually from overseas. This year, we are fortunate to present Dr Vinton Cerf, the father of the Internet and a world-renown expert on computer networks. Although his paper is not available for this special issue, it will appear later in SACJ. Through the good offices of Professor Chris Brink of UCT, we also have three other speakers from Germany, Canada and the US adding interest to the event, and two of their papers appear in this issue.

The programme committee originally devised a theme for the symposium - "Computing in the New South Africa". We received several queries as to the meaning of this theme, but unfortunately few papers that addressed it directly. One prospective author went as far as to enquire whether computer research would survive in the new South Africa. Another felt that his work was definitely not in the theme, as it was genuine, old world, basic, theoretical science! Nevertheless, there are two papers that consider one of South Africa’s key issues, that of language. Others look at the success we have achieved in applying technology to mining, and the future of low-cost operating systems. In all, the mix of papers represents a balance between the theoretical and the practical, the past and the future, all firmly based in the computing of the present.

Organising the symposium has involved the hard work of several people, and I would like to thank in particular:

- Derrick Kourie, my co-organiser, and the editor of SACJ for his invaluable advice and hard work throughout the planning and implementation stages;
- Riel Smit, the production editor, for attaining such a high standard in such a short time for so many papers;
- Gerrit Prinsloo and the staff at the CSSA for their efficient and quite delightfully unfussy organisation;
- Persetel for their very generous sponsorship of R25000, and Tim Schumann for taking a genuine interest in our events;
- the Foundation for Research Development for sponsoring Vint Cerf’s visit;
- and finally the Department of Computer Science of the University of Pretoria for providing the ideal working conditions for undertaking ventures of this kind, and especially Roelf van den Heever for his unfailing encouragement and support.

Judy M Bishop
Organising Chairman, SACRS 1992
Guest Editor, SACJ Special Issue
Referees

The journal draws on a wide range of referees. The following were involved in the refereeing of the papers selected for this special issue. Their role in certifying the papers and their contribution to enhancing the quality of papers is sincerely appreciated.

John Barrow
Ronnie Becker
Danie Behr
Sonia Berman
Liesbeth Botha
Theo Bothma
Chris Brink
Peter Clayton
Ian Cloete
Antony Cooper
Elise Ehlers
Quintin Gee
Andy Gravell
Wendy Hall
David Harel
Scott Hazellhurst
Derrick Kourie
Willem Labuschagne
Doug Laing
Dave Levy
Graham McLeod
Hans Messerschmidt
Deon Oosthuizen
Ben Oosthuysen
Neil Pendock
D Petkov
Martin Rennhackkamp
Cees Roon
Jan Roos
John Shochot
Morris Sloman
Riel Smit
Pat Terry
Walter van den Heever
Lynn van der Vegte
Herman Venter
Herna Viktor

UNISA
University of Cape Town
University of Pretoria
University of Cape Town
University of Pretoria
University of Pretoria
University of Cape Town
Rhodes University
Stellenbosch University

CSIR
Rand Afrikaanse University
DALSIG
University of Southampton
University of Southampton
The Weizmann Institute of Science
University of the Witwatersrand
University of Pretoria
UNISA
ISM
University of Natal
University of Cape Town
University of the Orange Free State
University of Pretoria
University of Pretoria
University of the Witwatersrand
University of Natal
Stellenbosch University
University of Pretoria
University of Pretoria
University of the Witwatersrand
Imperial College, London
University of Cape Town
Rhodes University
University of Pretoria
University of Pretoria
University of Fort Hare
Stellenbosch University
Network Partitions in Distributed Databases

H L Viktor and M H Rennhackkamp

Department of Computer Science, University of Stellenbosch, Stellenbosch, 7600, South Africa
e-mail: hlv@cs.sun.ac.za, mr@sunvax.sun.ac.za

Abstract

The technology of distributed databases is based on two other technologies which have developed a solid foundation: computer networks and centralized databases. A distributed database designer should therefore address database-related as well as network-related issues.

Provision has to be made for the detection of and recovery from failure of the database as well as the network components. One type of failure resulting from distribution is network partitions. During a network failure the sites of the distributed database are subdivided into two or more disjoint groups. These groups continue processing newly submitted transactions without any knowledge of the operations at the sites in the other partition groups. Since data may be replicated, the various copies of a data item may be updated inconsistently at the different partition groups.

When handling continued transaction processing during network partitions, a trade-off is made between the level of availability and the consistency provided by the system. Pessimistic partition processing strategies ensure database consistency by reducing availability. The token lists pessimistic partition processing strategy, which utilizes a reliable broadcasting facility, proved to be the best for the system under consideration.

Keywords: Consistency, distributed database, network partitions, recovery.
Computing Review Categories: H.2.2, H.2.4, H.2.7

Received March 1992, Accepted April 1992

1 Introduction

Distribution is an important issue in database development and research, since it fits naturally into the decentralized structures of many organizations. Powerful small computers provide many of the capabilities which were previously provided by large mainframes, but at much lower costs. The technology of distributed databases is based on two other technologies which have developed a solid foundation: computer networks and centralized databases. Thus a distributed database designer should address database-related as well as network-related issues.

No distributed database can be made totally failure free. Provision has to be made for the detection of and recovery from failure of the database as well as the network components. One type of failure resulting from distribution is network partitions. Network partitions are caused by network link failures. The sites of the distributed database are subdivided into two or more completely disconnected groups, since there are no communication path available between them. The various groups continue processing without any knowledge of the operations at the sites in the other partition groups.

When handling network partitions, a trade-off is made between the level of availability and the consistency provided by the system. In a highly available system where the correctness and currency of the data is not critical, transaction processing may be allowed to continue in all partition groups. Replicated copies of a data item may be updated inconsistently at the different partition groups. At the other extreme, transaction processing may be suspended until the resolution of the network partition. Usually transaction processing is allowed to continue at a particular subset of the database. This subset of the database is determined by using a suitable policy such as a voting scheme.

In the first section, distributed transactions are introduced. A distributed transaction is the unit of work when discussing recovery related concepts. It is a program unit which accesses and possibly updates data items located at various sites in the distributed database. This is followed by an overview of widely used partition processing strategies. The token lists strategy, a solution to network partitions which exploits a reliable broadcasting facility, is introduced. To date few studies have been done to compare partition processing strategies [3]. The results of a comparative modelling of the various strategies are presented, and it is shown that the token list strategy incurs the lowest overheads for the prototype under consideration. The paper is concluded with an evaluation of these results.

2 Distributed Transactions

A transaction is a sequence of actions considered to be an atomic logical unit of work. A sequence of operations constituting a transaction is delimited by BEGIN TRANSACTION and COMMIT or ABORT keywords. If the transaction completes successfully, it is committed. Otherwise, it is aborted and no effects of the transaction remain.

Transactions in a distributed environment transactions perform read and write operations on data located at various sites of a network. A distributed transaction is executed by
processes at each of these sites. These processes are able to communicate and cooperate with each other. The local transaction processing is known as subtransactions.

The actions performed by a distributed transaction should be indivisible. Either all of its actions should be properly reflected in the database, or none. This indivisibility requirement is achieved by a transaction satisfying four properties, referred to as the ACID-principle. It should:

1. be a single atomic action;
2. preserve the consistency of the database;
3. be isolated from other transactions; and
4. its results should be durable.

A distributed transaction that satisfies the ACID-principle is said to be successful. A distributed database is consistent if and only if it contains the results of successful distributed transactions [8]. A partition processing strategy is correct if the results thereof contain only successful transactions, i.e. if it preserves the consistency of the distributed database.

3 Partition Processing Strategies

Newly submitted transactions should be executed without violating the consistency of the database even in the presence of network partitions. Firstly, each partition should maintain consistency within the part of the database stored at the sites comprising the partition group. This can be achieved by adapting one of the standard replica control algorithms for unpartitioned systems. Secondly, the actions of a partition group should not conflict with the actions of other partition groups. Several strategies to ensure that the second requirement is met, have been developed.

Partition processing strategies can be classified in one of two ways.

Syntactic versus Semantic Strategies

The first trade-off concerns the type of information used to determine correctness. The two extremes are syntactic and semantic. Syntactic strategies use one-copy serializability as the sole criterion of consistency. One-copy serializability states that the concurrent execution of transactions on replicated data must be equivalent to a serial execution on non-replicated data. The serializability of transactions can be checked by examining the readsets and writesets of executed transactions [1].

Semantic strategies use either the semantics of the transactions or the semantics of the database in defining consistency. In general purpose systems, with the aim of supporting a wide spectrum of transactions and data models, no assumptions about the semantics of the database or transactions should be made. Semantic approaches will therefore not be given any further consideration.

Availability versus Consistency

For a high level of availability, the system’s normal processing should be disrupted as little as possible. Availability can be achieved by simply allowing all sites to process transactions as usual. Transactions may, however, produce inconsistent results since the databases in each group may diverge. In some applications such results may be acceptable. When partition groups are reconnected, incomplete transaction have to be committed or aborted and conflicts have to be corrected. Transactions missed by a partition group are executed and some transactions which cause conflicts are undone.

The consistency of a database can be guaranteed by simply suspending operations in all but one of the partition groups and forwarding updates at recovery. This solution severely compromises availability, and is not acceptable for applications where partitions either occur frequently or occur when access to the data is imperative.

Since it is impossible to satisfy both the availability and consistency goals simultaneously, one or both must be relaxed to some extent. Optimistic partition processing strategies achieve a high level of availability by relaxing consistency. The database in a partition group is consistent, but global consistency cannot be guaranteed. To resolve these inconsistencies, the transactions are undone or, alternatively, compensating or corrective transactions are executed. This study requires that transactions satisfy the ACID-principle. Since the durability of a transaction cannot be guaranteed when an optimistic approach is utilized, optimistic approaches will not be investigated further.

4 Pessimistic Syntactic Strategies

Pessimistic solutions prevent inconsistencies by limiting availability. Partition groups have to make worst-case assumptions about the behavior of the other partition groups. They operate under the pessimistic assumption that if an inconsistency can occur, it will. These strategies differ in the policy they use to restrict transaction processing and satisfy the ACID-principle. The following five strategies are of importance [4].

Primary Site Copy

One copy of a data item is called the primary copy. All read operations on a data item must be performed at the site at which the primary copy is kept. Thus only the partition group containing the primary copy can access the data item. A new primary copy is elected when the primary site for a data item fails. At recovery, outdated copies are updated to reflect the new values [2, 7].

All reads should be executed on the primary copy of each data item. This has detrimental effects on transaction throughput, since the primary copy data should be consulted prior to the reading of a data item. Also, the reading on a possibly remote data item increases the network traffic and could cause bottlenecks.

Token Site Copy

This approach is similar to the primary site copy approach. However, the primary copy can change for reasons other than site failure. Each data item has a token associated with it, permitting the bearer to access the item. In the event of a network partition, only the group containing the token
will be able to access the item [10]. Outdated copies are updated when the partition is resolved.

A new token is not elected when the token site fails. Rather, accessibility is lost. This limiting the availability of data is a serious drawback of this approach. If site failures within a network group are frequent, the number of transactions aborts due to unobtainable copies may be unacceptably high.

Voting
Each copy of a replicated item is assigned a number of votes. The weight of the vote is varied to reflect the accessibility level of an item. Protocols which use a weighted majority are called quorum-based protocols. Here, every transaction must collect a read quorum of \( r \) votes to read an item, and a write quorum of \( w \) votes to write an item. The total number of votes is denoted by \( v \). Quorums must satisfy two constraints:

1. \( r + w > v \).
   This constraint guarantees that an item cannot be read in one partition group and written in another.
2. \( w > v/2 \).
   This guarantees that writes cannot occur in two different partition groups on the same data item.

This approach reduces to a majority voting scheme [11] when each copy has exactly one vote and \( r \) and \( w \) are a simple majority [2].

A weakness of the quorum scheme is that reading an item is fairly expensive. A read quorum of copies must be read in this scheme, whereas a single copy suffices for the other strategies [4].

Missing Writes
Transactions are run in two modes, normal and failure. During normal processing, a transaction \( T \) reads one copy of an accessed data item and updates all the items in the writeset. If some copy cannot be updated, the transaction \( T \) enters failure mode. In failure mode, quorums must be obtained for each data item in the readset and writeset of a transaction. Missing updates which cannot be applied by the transaction are passed on to all transactions using results of \( T \). This missing update information is kept at the sites participating in \( T \).

In order to implement this strategy several files must be kept at each site.

1. A file for posting missing updates, with indications of which transactions need to be informed about the missing updates.
2. A file containing the values of missing updates.
3. A file indicating the transactions’ restarts, aborts or commits of which the site is aware.
4. A record of the missing updates that have been applied to the site.

Although these files can grow rapidly if the system is active during failures, they need only be maintained when failures are present in the system. Quorums are only required when a transaction is aware of missing updates. Performance overhead is thus only incurred when in failure mode. This method is very flexible since it requires no detection of failure other than the inability to perform updates. In addition, no special global action is needed to propagate updates when the failure is repaired [5].

Accessible Copies Algorithm
The principle idea of the Accessible Copies algorithm is the implementation of an abstract communication layer. The behavior of the new layer approximates that of an ideal network. The abstract communication layer creates and manipulates virtual partitions, analogous to the actual partition groups that occur in the real network. A virtual partition has three important attributes [6].

1. Creation time, which is the logical clock time of its creation.
2. The set of potential members, being the set of sites that are allowed to join the partition.
3. The set of actual members.

Virtual partitions are created explicitly according to a well-defined protocol. The steps of the creation protocol are as follows: A group of sites depart from their current virtual partition. The group of sites collectively determine the creation time and the potential members of the new partition. The set of potential members can include only those sites participating in the creation protocol. The sites in the group asynchronously become actual members of the new partition.

A data item can be read and written within a partition only if a majority of its copies reside on potential member sites of the partition. A read operation on an accessible data item is implemented by reading the nearest copy of the item residing on a potential member of the partition. Write operations on an accessible data item are implemented by writing all copies residing on potential members of the partition.

5 Token Lists
Token lists indicate the sites participating in a broadcast operation. A reliable broadcasting facility is used when all transaction related operations are executed. Message broadcasting is performed in the following steps: A site broadcasts its message. The token site receives the message and acknowledges it. When the sender receives the acknowledgement, it considers the message sent. The other sites receive the message as well as the acknowledgement. A message is committed when it is certain that more than a given number of sites have received the message.

This pessimistic strategy is utilized in a prototype DDBMS [9]. When a network partition is detected, the partition group which includes the token site continues operating. (It is assumed that site failures and network partitions can be distinguished.) The system is reconfigured to include the sites in the token group. The network must reform to exclude the other group. The site which detects the network partition initiates a reformation phase. Operational sites are invited to join a new token list. This new list is broadcasted to all sites in the group. All these
Table 1. Comparative results of modelling.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Disk NP</th>
<th>Disk TP</th>
<th>I/O NP</th>
<th>I/O TP</th>
<th>Message NP</th>
<th>Message TP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessible Copies</td>
<td>72,220</td>
<td>0</td>
<td>0</td>
<td>182</td>
<td>0</td>
<td>82</td>
</tr>
<tr>
<td>Majority Voting</td>
<td>48,000</td>
<td>0</td>
<td>126</td>
<td>63</td>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td>Missing Writes</td>
<td>0</td>
<td>140,470</td>
<td>0</td>
<td>60,910</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Primary Site Copy</td>
<td>20,480</td>
<td>0</td>
<td>63</td>
<td>126</td>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td>Token Lists</td>
<td>0</td>
<td>0</td>
<td>21</td>
<td>21</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Token Site Copy</td>
<td>20,543</td>
<td>0</td>
<td>126</td>
<td>189</td>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td>Voting</td>
<td>72,000</td>
<td>0</td>
<td>126</td>
<td>63</td>
<td>63</td>
<td>63</td>
</tr>
</tbody>
</table>

sites are required to confirm their participation in the new token list. After all sites have confirmed, a new token is generated and the protocol returns to normal phase. Transaction processing continues within the new partition group. Data items are accessed by using the normal replication and concurrency control algorithms. The standard site failure recovery techniques are utilized. If the token site fails during the network partition, the site which detects the failure will reinitiate the reconfiguration phase.

When the network partition is resolved, the network is reformed to include all operational sites. The results of transaction executions in the operational group are forwarded to the other group. The recovery techniques used for site failure recovery are used when updating the outdated copies.

6 Overhead Model

In a previous study an overhead model, which compared the overheads associated with various site failure related recovery techniques, was derived [13]. This model was simplified and subsequently extended to facilitate network partition processing strategy modelling.

Three types of overheads are of importance, namely disk storage, I/O and message overhead.

- The disk storage overhead is measured by considering the number of additional bytes incurred by the partition processing strategies.
- The I/O overhead consists of the number of I/O operations when reading and writing data related to partition processing strategies. The I/O operations occur synchronously with normal transaction processing.
- The number of additional messages imposed by the partition processing strategies is taken as a metric when discussing message overhead, since it directly influences the transmission delay and thus the overall transaction execution.

The overheads imposed by the strategies are determined by recovery processing. Recovery processing overheads are considering the overheads during normal, termination and implicitly included in the normal site failure recovery processing overhead, discussed elsewhere [12]. The recovering sites execute independent recovery, using the standard site failure recovery techniques. The overheads as incurred during normal processing and termination processing are of importance. During normal processing all database and network components are operating correctly. The termination processing overhead is defined as the overhead incurred by operational sites during a network partition.

7 Evaluation

The various strategies are uniformly modelled against the same distributed database specifications [13]. The specifications include the number of sites, the size of the local databases, and the number of various types of transactions executed at each site. The overheads are calculated by considering all sites in the distributed database. The resulting values reflect the overheads at the most expensive site. The modelling is pessimistic, since the maximum values incurred are used in the comparison. Table 1 shows the comparative results of an example modelling, incurred during normal processing (NP) and termination processing (TP).

The missing writes strategy incurs high disk and I/O overheads during termination processing. No overheads are, however, incurred during normal processing. In addition, no message overheads are incurred. This method is best suited for a system without a reliable broadcasting facility, with few network partition occurrences and where the message throughput is of importance.

The accessible copies strategy yields a high disk storage overhead during normal processing, due to the additional voting information maintained. The I/O and message overheads during termination processing are moderately high. The I/O and message overheads during normal processing are, however, low. This strategy is suitable in a system with a low network partition rate, where disk storage overhead is irrelevant.

The token lists approach yields the best overall overheads. This strategy is therefore the most suitable in a system which implements a reliable broadcasting facility.

8 Conclusion

Partition processing strategies should ensure that newly submitted transactions do not violate the consistency of the distributed database. Most of these partition processing strategies are based on the simple observation that a sufficient condition for consistency is that no two partition groups will execute conflicting data operations. Pessimistic
strategies ensure global consistency by limiting the availability of the distributed database.

An overview of widely used pessimistic partition processing strategies was presented. The token list strategy utilizes a reliable broadcasting facility and execution continues only within a single partition group. The global database state is consistent and the transaction execution satisfies the ACID-principle. It has been shown that this strategy incurs the least overheads, for the specific prototype, of the all strategies under consideration. In a system with reliable broadcasting facilities, few network partition occurrences and with real-world effects that cannot be undone, this seems to be the best option.

This paper concerns the occurrence of a single network partition. Simultaneous and multiple network partitions and site failures have not been discussed. An extension to this work should include the investigation of the effects of such failures on the partition processing strategies.

References

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research papers. However, non-refereed review and exploratory articles of interest to the journal’s readers will be considered for publication under sections marked as Communications or Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines:

- Use wide margins and 1 1/2 or double spacing.
- The first page should include:
  - title (as brief as possible);
  - author’s initials and surname;
  - author’s affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be numbered and titled. Figures should be submitted as original line drawings/printsouts, and not photocopies.
- References should be listed at the end of the text in alphabetic order of the (first) author’s surname, and should be cited in the text in square brackets [1, 2, 3]. References should take the form shown at the end of these notes.

Manuscripts accepted for publication should comply with the above guidelines (except for the spacing requirements), and may be provided in one of the following formats (listed in order of preference):

1. As (a) \LaTeX{} file(s), either on a diskette, or via e-mail/ftp – a \LaTeX{} style file is available from the production editor;
2. In camera-ready format – a detailed page specification is available from the production editor;
3. As an ASCII file accompanied by a hard-copy showing formatting intentions:
   - Tables and figures should be on separate sheets of paper, clearly numbered on the back and ready for cutting and pasting. Figure titles should appear in the text where the figures are to be placed.
   - Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.

Further instructions on how to reduce page charges can be obtained from the production editor.
4. In a typed form, suitable for scanning.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect scanning, typesetting, reproduction and other costs. Currently, the minimum rate is R20-00 per final page for \LaTeX{} or camera-ready contributions and the maximum is R100-00 per page for contributions in typed format.

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers in categories 3 and 4 above will be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author’s responsibility to ensure that such submissions are error-free. However, the editor may recommend minor typesetting changes to be made before publication.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to less than about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

References

## Contents

Forward .................................................................................................................. i
Referees .................................................................................................................. ii

Machine Translation from African Languages to English
DG Kourie, WJ vd Heever and GD Oosthuizen ...................................................... 1
Automatically Linking Words and Concepts in an Afrikaans Dictionary
PZ Theron and I Cloete ............................................................................................ 9
A Lattice-Theoretic Model for Relational Database Security
A Melton and S Shenoi .......................................................................................... 15
Network Partitions in Distributed Databases
HL Viktor and MH Rennhackkamp ......................................................................... 22
A Model for Object-Oriented Databases
MM Brand and PT Wood ...................................................................................... 27
Quantifier Elimination in Second Order Predicate Logic
D Gabbay and HJ Ohlbach .................................................................................... 35
Animating Neural Network Training
E van der Poel and I Cloete .................................................................................. 44
HiLOG - a Higher Order Logic Programming Language
RA Paterson-Jones and PT Wood .......................................................................... 53
ESML - A Validation Language for Concurrent Systems
PJA de Villiers and WC Visser .............................................................................. 59
Semantic Constructs for a Persistent Programming Language
SB Sparg and S Berman ....................................................................................... 65
The Multiserver Station with Dynamic Concurrency Constraints
CF Kriel and AE Krzesinski .................................................................................. 75
Mechanizing Execution Sequence Semantics in HOL
G Tredoux ............................................................................................................ 81
Statenets - an Alternative Modelling Mechanism for Performance Analysis
L Lewis .................................................................................................................... 87
N Pendock ............................................................................................................ 95
Galileo: Experimenting with Graphical User Interfaces
R Apteker and JM Bishop ...................................................................................... 99
Placing Processes in a Transputer-based Linda Programming Environment
PG Clayton, FK de-Heer-Menlah, EP Wentworth ................................................ 109
Accessing Subroutine Libraries on a Network
PH Greenwood and PH Nash .............................................................................. 117
A Multi-Tasking Operating System Above MS-DOS
R Foss, GM Rehmet and RC Watkins .................................................................. 122
Using Information Systems Methodology to Design an Instructional System
BC O'Donovan ....................................................................................................... 126
Managing Methods Creatively
G McLeod ............................................................................................................ 131
A General Building Block for Distributed System Management
P Putter and JD Roos ............................................................................................ 141