A Study on the Management of Semantic Transaction for Efficient Data Retrieval

Shi-Ming Huang, Irene Kwan* and Chih-He Li**

Department of Information Management, National Chung Cheng University
160 San-Hsing, Min-Hsiung, Chia-Yi 621, Taiwan, R.O.C
* Department of Information Systems, Lingnan University
Tuen Mun, Hong Kong
**Department of Computer Science and Engineering, Tatung University
40 Chungshan N.Road, 3rd sec, Taipei 104, Taiwan, R.O.C
E-mail: smhuang@mis.ccu.edu.tw, *ijkwan@ms18.hinet.net and **chlee@mis.ttu.edu.tw

Abstract
Mobile computing technology is developing rapidly due to the advantages of information access through mobile devices and the need to retrieve information at remote locations. However, many obstacles within the discipline of wireless computing are yet to be resolved. One of the most significant of these issues is the speed of data retrieval, which directly affects the performance of mobile database applications. To remedy this problem, we propose here a revised methodology focusing on the management of mobile transactions. This paper investigates an extended semantic-based transaction management mechanism, and applies a model-based approach for developing a simulation model to evaluate the performance of our approach.

1. Introduction

Mobile computer systems have changed their way of access to shared data due to the emergence of wireless networking, which provides users with the ability to retrieve information that is stored at different sites/databases at any place and time through mobile devices.
Apart from the need maintain data consistency among different local database systems in a multi-database environment, the mobile computing environment is restricted in many ways. Mobile elements themselves are resource-poor (e.g. limited battery life and small storage capacity), and network connectivity is often achieved through low-bandwidth wireless links so it can be frequently lost for uncertain periods of time. In the new mobile database environment, the task of achieving the required performance has become more difficult than in the non-mobile database systems. To improve the execution, a semantic transaction model to describe the process of data transaction and analyze the semantic information of each transaction before submitting to the local databases is proposed herein.

This paper investigates a semantic-based transaction management mechanism and adopts a model-based approach for developing a case study. This mechanism enables us to reason about the transaction management and the various constraints of developing application systems in a mobile computing environment. As such, this paper has the following contributions:

1) Investigation of a semantic transaction model that supports transaction execution with higher efficiency in a wireless environment.
2) Examination of general data consistency control in the mobile computing environment.
3) Identification of major constraints in the wireless communication environment that influence the performance of transaction-based application.

Finally, the efficiency of the proposed approach is evaluated in terms of its efficiency by constructing a simulation model using SES/Workbench 3.1 simulation.

2. Related Work

Much research has been focused on transaction management in multi-database or mobile computing environments. For examples: Imielinski and Badrinath [1] have categorized research challenges in data management for mobile wireless computing in terms of mobility and disconnection. Kayan and Ulusoy [2] have discussed critical issues related to time-constrained transaction management and have provided a mobile database system model that supports real time applications. They constructed a transaction execution model with two alternative execution strategies: ESFH (Execution Site is Fixed Host) and ESMH (Execution Site is Mobile Host) for mobile transactions. Walborn and Chrysanthis [3] have examined various types of semantic information with respect to their applicability in the context of mobile transaction processing. Lim, et al. [4] have proposed concurrency control algorithm using global locking tables created with semantic information in a mobile heterogeneous system. Huang and Huang [5] have described a global semantic

transaction model in an active heterogeneous Multi-database, which relaxes the global serializability constraints as well as improves the performance of global transaction execution.
The major differences between a mobile database system and a multi-database system (MDBS) [4][6][7]:

1) Some of the distinctive issues for mobility are disconnection, storage capacities, processing abilities, and energy restrictions.
2) The connection of server and client site through wireless devices and environment.

Indeed both systems are intended to provide timely and reliable access to globally shared data. Many issues and solutions for a Mobile Computing System can be extracted from a MDBS, although there are added complexities introduced by the mobility and wireless connection of MDBS.

3. Semantic-Based Data Transaction Management for Mobile Database Systems

3.1 Mobile Database Systems

Figure 3.1 depicts a mobile computing environment based on the ESFH [3] architecture as considered in this approach. A transaction is submitted with a request message from a mobile host (MH) to the Mobile Support Station (MSS), the MSS will forward the transaction to Mobile Transaction Management Server (MTMS). The overall process flow can be described in the following steps:

Step1. A mobile host connects with fixed network through a wireless communication medium (such as Wireless Application Protocol (WAP) etc.) in the cell of MSS.

Step2. The MSS receives the transaction from the mobile host, then processes it with hand-off mechanisms and forwards the transaction to the MTMS.

Step3. The MTMS deal with the execution of transactions, according to the defined semantic transaction management mechanisms.

Step4. After the transaction is committed, the MTMS sends the results back to mobile host through MSS and the process is done.

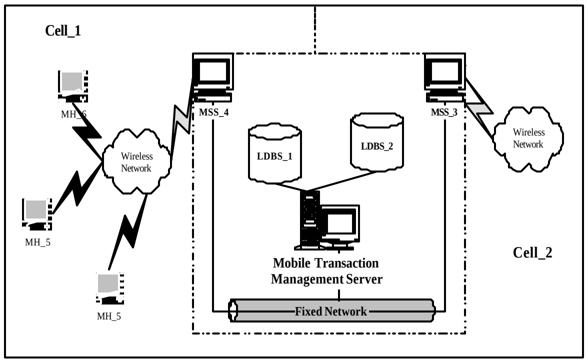


Figure 3.1: Mobile Database Environments.

Our solutions to remedy the major issues of mobile computing environment are highlighted as the following (Table 3.1):

Issues	Our solutions
Mobility–Handoff	A Location Table is in the MTMS to record the location of MH. It is updated by MSSs when the handoff of a mobile device occurs.
Disconnection	When MH are disconnected, in order to maintain the data consistency, we obtain a disconnect_log record that records the disconnect information of transaction.
Concurrency control	In our approach, the concurrent execution of transactions is controlled by ensuring the serializability. For this, we enclose a semantic-based transaction mechanism to relax the global serializability constraints and pessimistic mechanism to ensure that each GDST is executed in LDBS by following the global execution order.

Table 3.1: Solutions to remedy the major issues of mobile computing environment.

3.2 Semantic Data Transaction Management Model in Mobile Database Systems

A semantic data transaction management model requires mobility and disconnection control, and this proposed transaction model is an extension of our previous work [5]. It consists of two phases, design and execution. In the design phase, we define the applications and compatibility of the transaction model. After receiving the transactions submitted by the MHs, the MTMS will follow the defined applications and semantic mechanisms in the design phase and then constructs the transaction model in the execution phase. The following Figure 3.2 shows the structure of this model.

- Semantic Data Transaction Model -

Design phase

application(parameters)

compatibility(database, application, application, semantic)

Execution phase

GDTi(application(parameters), GDSTs, internal value dependency relationship, status, disconnect_status)

GDST ij(application(parameters), confirmation)

GDTi ¡Ÿ GDST i1 ¡ å GDST i2 ¡ å GDST i3 ..... ¡ å GDST in

if GDTi needs to access n different LDBs

disconnnect_status: the disconnect state of mobile transaction

GDTi: the ith Global Data Transaction

GDSTij: the jth global data subtransaction of the ith global data transaction

Figure 3.2: Semantic Data Transaction Model.

Table 3.2 describes the aspects of this model, the attributes and their descriptions as applied in our semantic transaction model:

Attributes	**Descriptions**
Application	In our semantic transaction model, the application is an instance of transaction that contains several database operations and input/output parameters.
Compatibility	To enable the transaction to be executed semantically, we define the compatibility between transactions that operate with LDBSs.
GDSTs	The global transaction is constructed by a set of global data sub-transactions.
Indicators	The indicator "$- >_{vd}$" represents internal value dependency relationship between GDSTs within a global transaction.
Status	The status attribute is used to state the process of transaction, including active (the transaction is executing), prepare-to-commit, abort, and commit states.
Disconnect_status	When the MSS receives a mobile transaction, the GTM will check this attribute and do the restore processes if needed. It contains two states: normal: a non-resubmit transaction from a mobile host, the initial state; disconnected: a resubmit transaction from a mobile host due to disconnection, dealing with restore operations if needed.
Confirmation	This attribute contains the transaction status, i.e. active, completely, commit, prepare-to-commit, prepare-to-commit-complete, prepare-abort or abort. If the GTM decides to commit the GDST and LDBMS have successfully committed, then the confirmation is "commit"

Table 3.2: Syntax of GDT and GDST.

3.3 Semantic-Based Transaction Mechanism

Our semantic-based transaction mechanism is based on Compatible Tables, which are defined by users to represent the interleaving between two GDSTs. A similar method has been was in [5] for a semantic-based transaction management in a stationary database, and we extend that method to a mobile database environment.

To enable the GDT to be executed semantically, users need to identify the compatibility of applications before the model definition, and these descriptions will construct the Compatible Table. Each LDBS has a Compatible Table to represent the semantic information of its application. The table contains four different options: Yes, No, Yes-SP, and Yes-DP [5].

3.4 Global Data Transaction Execution

To ensure the validity of GDT processing, the GTM needs to maintain the global serializability of the GDT execution. To maintain the global serializability, each GDST of the GDT needs to be executed by following the schedule of the global order on each LDBSs. Due to the local autonomy property in MDBS, the LDBSs may not follow the global order. Much research has been done to deal with this problem [9][10]. Unfortunately, these solutions have low concurrency and a low performance.

To improve these shortcomings, we use a hybrid approach which consists of a semantic-based transaction mechanism to relax the global serializability constraint and a pessimistic mechanism to ensure that each GDST is executed in LDBS by following the global execution order, represented as "**- >**". The result of this unique hybrid approach has proven feasible via our case study. This novel approach has significantly increased both the performance and degree of concurrency for the transaction management.

3.5 Global Serializability and Data Commit Protocol

* Global Serializability

Considering an MDBS that contains LDBS1 and LDBS2, which data item A in $LDBS_1$, and items B and C are in $LDBS_2$. There are applications to copy these data, GDT1 to copy the data from B to A, and GDT2 to copy the data from A to C. If we do not control the execution order in the LDBSs, then an indirect conflict problem will occur. For instance, a local transaction (i.e. L: $w_L(B)$, $w_L(C)$) is submitted at $LDBS_2$. If the local schedules of each LDBS are:

S_1: $w_{GDST11}(A)$, $r_{GDST12}(A)$

S_2: $w_L(B)$, $r_{GDST21}(B)$, $w_{GDST22}(C)$, $w_L(C)$ and $S=\{S_1,S_2\}$.

The execution order of S_1 is $GDST_{11}$ - > $GDST_{12}$, and S_2 is $GDST_{21}$ L - > $GDST_{22}$. This seems follow the global execution order, i.e. GDT_1 - > GDT_2. However, the LTM of LDBS2 may change the execution order to $GDST_{22}$ - > L - > $GDST_{21}$. This is due to $r_{GDST21}(B)$ operation conflicts with $w_L(B)$ and $w_L(C)$ operation conflicts with $w_{GDST22}(C)$. The LTM will change the execution schedule to $w_{GDST22}(C)$ - > $w_L(C)$ - > $w_L(B)$ - > $r_{GDST21}(B)$.

To deal with indirect conflict, a pessimistic method is used. If the GDSTs of different GDTs that do not provide the compatibility of GDSTs or their compatibility of GDSTs are not compatible, then those GDSTs executed on the LDBSs need to follow the global execution order. The pessimistic method is that the rear GDST could be dispatched to the LDBS if the front GDST has reached a commit point that is in the complete state to ensure GDSTs executing in LDBS will be the same as the global order:

*** Global Data Commit Protocol**

This global transaction is based on the 2PC protocol. Because the autonomy is different in LDBSs, the transaction process of a GTA has two types: (1) a 2PC agent for a LDBS that supports 2PC protocol, and (2) an Emulated 2PC agent for a LDBS that does not support 2PC protocol. To deal with the mobility of mobile host, we extend the 2PC protocol to involve mobility control mechanisms, described as follows:

Disconnection Process:

During the data transmission of a transaction, if the mobile host were suddenly disconnected, the GTM will record the information into the disconnect_status attribute of transaction model as 'disconnected' and the disconnect_log will also record the transaction information. The recovery and backup records will save the local database process status information that will be used for later recovery process.

Recovery Process:

When MSS receives a new mobile transaction, it will for search the transaction from the disconnect_log. If it exists in the disconnect_log, the GTM will retrieve the information from the recovery and backup records. The GTM will then recover the transaction status into the previous status and continue the transaction process.

4. Performance evaluation

A simulation study utilizing our proposed approach is presented here to verify and evaluate the performance of the proposed mechanism versus non-semantic ones. It is discussed in the following sections.

4.1 Construction of the semantic transaction model

Considering a global transaction for a travel information system, we assume that for the purpose of this trip the only applicable airline is China-airlines, the hotel is the Grand-Hotel and the car rental agency is the Nice car company. Through this system, a customer could order a trip package or make only an individual reservation for airline, hotel or car.
For instance, a tourist T_A1 wants to schedule a trip package, the process of his schedule may consist of the following 3 subtasks:

Subtask1: T_A1 connects a travel agent through a mobile device to schedule a trip package.

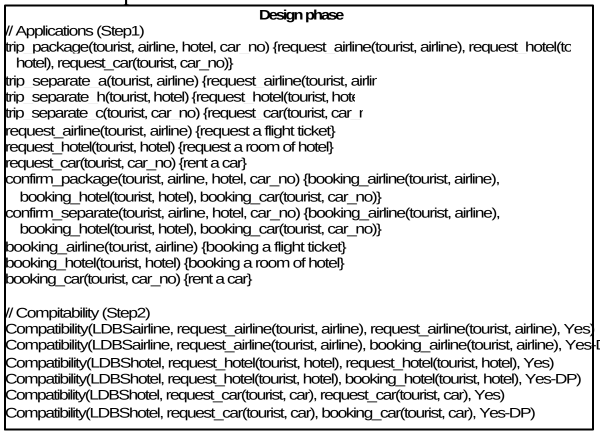
Subtask2: The travel agent negotiates with airlines for flight tickets, and hotels to reserve rooms, and car rental agencies for lease of cars.

Subtask3: The travel agent sends the available booking to T_A1 for confirmation of the reservations.

Through the following steps, we constructed the transaction model in the design phase shown in the following Figure 4.1:

Step1. Define applications.

Step2. Define the compatibility of applications. After constructing applications, we need to describe the compatibility of applications, i.e. semantic representation.

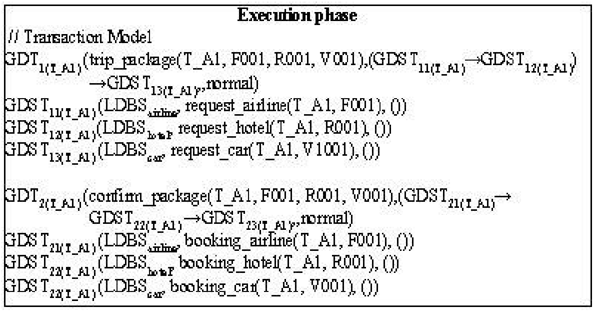


```
Design phase
// Applications (Step1)
trip_package(tourist, airline, hotel, car_no) {request_airline(tourist, airline), request_hotel(to
   hotel), request_car(tourist, car_no)}
trip_separate_a(tourist, airline) {request_airline(tourist, airlir
trip_separate_h(tourist, hotel) {request_hotel(tourist, hote
trip_separate_c(tourist, car_no) {request_car(tourist, car_r
request_airline(tourist, airline) {request a flight ticket}
request_hotel(tourist, hotel) {request a room of hotel}
request_car(tourist, car_no) {rent a car}
confirm_package(tourist, airline, hotel, car_no) {booking_airline(tourist, airline),
   booking_hotel(tourist, hotel), booking_car(tourist, car_no)}
confirm_separate(tourist, airline, hotel, car_no) {booking_airline(tourist, airline),
   booking_hotel(tourist, hotel), booking_car(tourist, car_no)}
booking_airline(tourist, airline) {booking a flight ticket}
booking_hotel(tourist, hotel) {booking a room of hotel}
booking_car(tourist, car_no) {rent a car}

// Compitability (Step2)
Compatibility(LDBSairline, request_airline(tourist, airline), request_airline(tourist, airline), Yes)
Compatibility(LDBSairline, request_airline(tourist, airline), booking_airline(tourist, airline), Yes-DP)
Compatibility(LDBShotel, request_hotel(tourist, hotel), request_hotel(tourist, hotel), Yes)
Compatibility(LDBShotel, request_hotel(tourist, hotel), booking_hotel(tourist, hotel), Yes-DP)
Compatibility(LDBShotel, request_car(tourist, car), request_car(tourist, car), Yes)
Compatibility(LDBShotel, request_car(tourist, car), booking_car(tourist, car), Yes-DP)
```

Figure 4.1: Design phase of Transaction Model Construction.

After constructions in the design phase, when a tourist submitted a tour package, the MTMS then constructed the transaction model (Figure 4.2) following the applications and semantic mechanisms defined in the design phase.



Execution phase

// Transaction Model

$GDT_{1(T\_A1)}$(trip_package(T_A1, F001, R001, V001),($GDST_{11(T\_A1)} \rightarrow GDST_{12(T\_A1)}$ $\rightarrow GDST_{13(T\_A1)}$),normal)

$GDST_{11(T\_A1)}$($LDBS_{airline}$, request_airline(T_A1, F001), ())

$GDST_{12(T\_A1)}$($LDBS_{hotel}$, request_hotel(T_A1, R001), ())

$GDST_{13(T\_A1)}$($LDBS_{car}$, request_car(T_A1, V1001), ())

$GDT_{2(T\_A1)}$(confirm_package(T_A1, F001, R001, V001),($GDST_{21(T\_A1)} \rightarrow$ $GDST_{22(T\_A1)} \rightarrow GDST_{23(T\_A1)}$),normal)

$GDST_{21(T\_A1)}$($LDBS_{airline}$, booking_airline(T_A1, F001), ())

$GDST_{22(T\_A1)}$($LDBS_{hotel}$, booking_hotel(T_A1, R001), ())

$GDST_{22(T\_A1)}$($LDBS_{car}$, booking_car(T_A1, V001), ())

Figure 4.2: Execution phase of Transaction Model.

4.2 Simulation modeling

According to the above process, we then constructed two simulation models: Semantic and Non-semantic. Each consisted of a single MTMS, three MSSs (MSS_A, MSS_B and MSS_C) that covered individual cells (Cell_A, Cell_B and Cell_C), ten MHs and three LDBSs($LDBS_{airline}$, $LDBS_{hotel}$ and $LDBS_{car}$). This is similar to the environment shown in Figure 3.1. The simulation was developed under the SES/Workbench 3.1 platform. The purpose of our simulation study is to compare the response time and utilization of a global transaction between semantic and non-semantic transaction management mechanism in mobile computing environment.

There are four main sub-modules in our simulation models, as described in Table 4.1 and the major system parameters applied in the simulation models are provided in Table 4.2

Sub-modules	Functionality
Wireless communication module	The communication interface between mobile hosts and fixed hosts
Transaction management module-Transaction monitor	Monitoring and process the status of transactions submitted from MHs
Transaction management module-Semantic/Non-semantic transaction management mechanism	Semantic/Non-semantic transaction management mechanisms
Database management module	Database resource management

Table 4.1 Main sub-modules in the model

The distinction between the semantic and non-semantic models is the transaction management mechanism that applied; comparison and analysis of the simulation results is presented in the following section.

Parameters	Values	Descriptions
Simulation run length	480min	From 9a.m. to 5p.m.
Number of MHs	10	10 users
Number of MSSs / Cells	3 / 3	MSS_A~C, Cell_A~C
Number of LDBSs	3	LDBS of airline, hotel and car
Power of MHs	5 units	Each MH contains 5 unit energy
Wireless communication time	Exponential distribution	
Mobile transactions request rate	Exponential distribution	
Local transactions request rate	Exponential distribution	
MTMS process time	Exponential distribution	
Disconnect probability	Probability distribution	

Table 4.2 System parameters

4.3 Simulation result and comparison

* Number of committed transactions

Figure 4.3 reveals that during the 480min-simulation time period, the number of committed semantic transactions was 391, which is higher than the 229 non-semantic thus actions. Here the authors utilize the semantic mechanism to encourage the concurrency control process of transactions, rather than the serializability obtains by non-semantic ones.

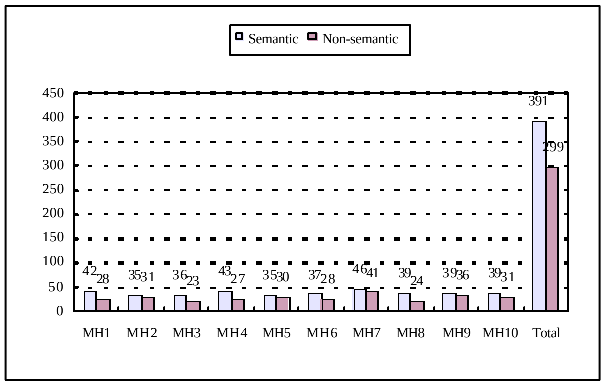


Figure 4.3 Number of committed transactions

* Process response time

From the simulation result shown in Figure 4.4, it is clear to realize that the response time of process is related to the committed rate of transaction. The higher commit rate, the shorter response time. As a result, the response time of the semantic mechanism is 0.97833min shorter than 1.2733min of non-semantic mechanism.

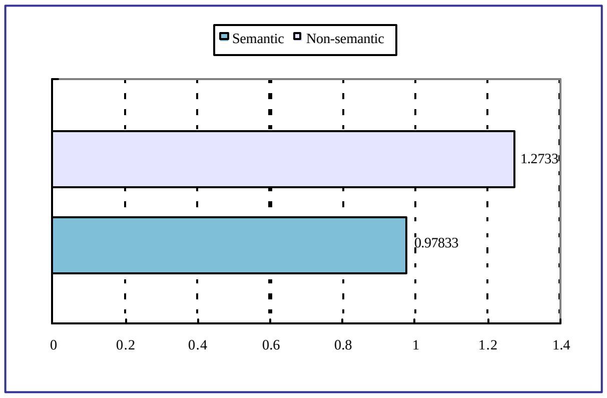


Figure 4.4 Process response time

*** Server utilization and execution rate of transactions**

From the simulation results, the utilization of MTMS service is seen to correspond to the committed rate of transactions, 1.0229 for the semantic and 0.78542 for the non-semantic. The utilization of the MTMS with a semantic mechanism is better than the non-semantic ones. The execution rate for the semantic mechanism is higher than for the non-semantic one, at 481 versus 367. Because the semantic mechanism adopted here is encourages the concurrency process of transactions, the MTMS can execute more transactions in parallel.

Finally, we conclude that the performance of the semantic transaction management mechanism used in our simulation model is more efficient than the non-semantic mechanism. With the semantic mechanism, we can relax the serializability constraint, and the execution order of the transactions can be changed to improve the execution performance.

5. Conclusions and Future Works

To summarize the work in this paper, we have illustrated in detail a semantic-based transaction management mechanism for mobile database segments. In addition, the simulation study has also allows us to reason about the transaction management and the constraints of developing mobile-based applications.

The next step in this research will be the implementation of our global transaction mechanism into a working prototype for mobile application system. Without investigating the feasibility of our approach on real operating application systems, the actual performance of our mechanism is difficult to identify and measure, especially in resource-constrained wireless environments. In conclusion, our research in this paper is theoretically sound and is ready to move to the next step for testing and/or building the proposed mobility control mechanisms for transaction management upgrade in an operating MDBS environment.

Acknowledgement

The National Science Council, Taiwan, under Grant No. NSC 90-2213-E194-006 has supported the work presented in this paper. We greatly appreciate their financial support and encouragement.

References

[1] Tomasz Imielinski and B.R.Badrinath, March 1993. "Data Management for Mobile Computing", SIGMOD RECORD, Vol.22, No1.

[2] Ersan Kayan and Ozgur Ulusoy, 1999. "Real-Time Transaction Management in Mobile Computing Systems", Proceeding of the 6th International Conference on Database Systems for Advances Applications, IEEE (Pub.), pp. 127 –134.

[3] Gray D. Walborn and Panos K. Chrysanthis, 1995. "Supporting semantics-based transaction processing in mobile database applications", Proceedings of the 14th Symposium on Reliable Distributed Systems, pp. 31 –40

[4] J.B. Lim, A.R.Hurson and Kavi, 1995. "Concurrent Data Access in Mobile Heterogeneous Systems", K.M, Systems Sciences. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference, pp. 10 -19

[5] Shi-Ming Huang and Chien-Ming Huang, 1998. "A semantic-based transaction model for active heterogeneous database systems", *Systems, Man, and Cybernetics*, 1998. 1998 IEEE International Conference on Volume: 3 , pp. 2854 -2859 vol.3

[6] R. Ortiz and P. Dadam, 1995. "Towards the Boundary of Concurrency", 2nd International Conference on Concurrent Engineering, Research and Applications.

[7] E. Pitoura and B. Bhargava, 1995. "Maintaining Consistency of Data in Mobile Distributed Environments", Proceedings of the 15th International Conference on Distributed Computing Systems, pp. 404-413.

[8] A.B. Omran, C. Jiansan, W. Du, and A.K. Elmagarmid, 1993. "InterBase: An Execution Environment For Heterogeneous Software Systems", *IEEE Computer*, IEEE (Pub.), pp. 57-68

[9] D. Georgakopoulous, M. Rusinkiewicz, and A. Sheth, 1991, "On Serializability of Multi-database Transactions through Forced Local Conflicts", Proceedings of the 7th International Conference Data Engineering, IEEE (Pub.), pp. 314-323.

[10] S. Mehrotra et al., 1991, "Non-serializability Executions in Heterogeneous Distributed Database Systems", Proceedings of the 1st International Conference Parallel and Distributed Information Systems, IEEE (Pub.), pp. 245-252.