A Programming Solution for Moving Mobile Transaction

Osman Mohammed Hegazy, Ali Hamed El Bastawissy, and Romani Farid Ibrahim

Abstract—In this paper, our concern is the management of mobile transactions in the shared area among many servers, when the mobile user moves from one cell to another in online partially-replicated distributed mobile database environment. We defined the concept of transaction and classified the different types of transactions. Based on this analysis, we propose an algorithm that handles the disconnection due to moving among sites.

Keywords—Concurrency, mobile database, transaction processing, two phase locking.

I. INTRODUCTION

THE database systems evolved from manual database systems to computerized based centralized database systems, then to distributed database systems based on wired networking technology. Now it evolves to mobile database systems based on the wireless networking technology. We are in the age of mobile devices and wireless networks, accessing data anywhere-anytime-anyway will be real but this must be consistent. The mobile database, or embedded database on a mobile device, is starting to become an important player in all practical fields, for example, business, traveling, police, military, medical, etc. The data will be entered approximately in its real time, no delay between the events time and the entering time to the database.

The mobile computing environment consists of stationary and mobile components (Fig. 1). Mobile Hosts (MH) - also termed Mobile units (MU)- are computing devices that can connect to a stationary (fixed) network via wireless links. These stationary hosts are called mobile support stations (MSS) or base stations which perform the transaction and data management functions with the help of transaction managers (TMs) and data managers (DMs), respectively. Each MSS is responsible for all the mobile hosts within a given small geographical area, known as a cell. At any given instant, a MH communicates only with the MSS responsible for its cell [1]. MSSs are the gateways between mobile units and fixed hosts [2]. A MH may have some server capability to perform concurrency control and logging etc.

Prof. Osman Mohammed Hegazy is with the Faculty of Computers and Information, Cairo University, Egypt (e-mail: ohegazy@idsc.net.eg)**.**

Dr. Ali Hamed El Bastawissy is with the Faculty of Computers and Information, Cairo University, Egypt (e-mail: alibasta@hotmail.com).

Romani Farid Ibrahim is PhD student in the Faculty of Computers and Information , Cairo University, Egypt (e-mail: rfarid1@yahoo.com).

As an application, we are considering, mobile hosts are laptop computers belonging to members of big salespersons team.

Transaction is defined as a means by which application programmer can package together a sequence of database operations so that the database can provide a number of Guarantees, known as the ACID (Atomicity, Consistency, Isolation, and Durability) properties of a transaction [3].
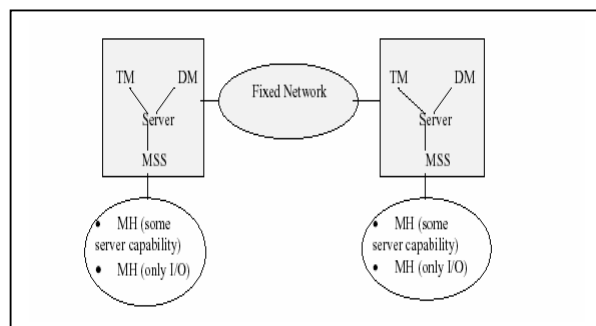

Fig. 1 Mobile Architecture [1]

Mobile transaction is a transaction where at least one mobile host takes part in its execution [5]. The mobile user, by nature, is moving from one place to another so the mobile transaction should follow the user anywhere, which is not supported in distributed database transactions.

The problems that we consider are:

Most of mobile database researchers assume that the mobile database system is a group of contiguous service areas as straight line, and the mobile user moves among them using a car (ex. Fig. 2).
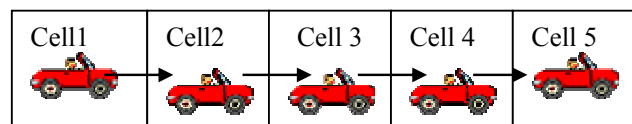

Fig. 2 Movement of Mobile Unit through Cells

If the cells are not contiguous as a straight line, but they are contiguous as in Fig. 3 [15],

If the mobile user moves from the cell covered by the Host H1 (a) to the shared area among the three hosts (b):

- How will the transaction that started at H1 continue its work?

- How will the refreshment process refresh the whole system replicas?
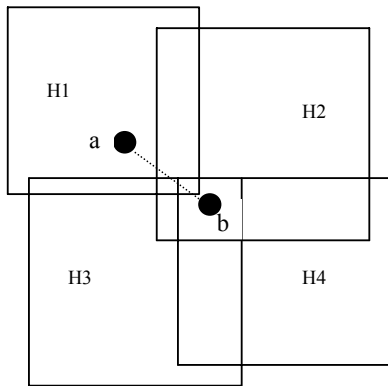- What is the next host for the mobile unit at point b?

Fig. 3 Shared area among hosts (H = Host)

In this paper, we will try to answer these questions. We assume involuntary and very short disconnection because of moving from cell to cell.

This paper is structured as follows: Section 2 describes the related work. Section 3 shows our view for the program and transaction, analysis and classification of transactions according to the relation among their subtransactions and our shadow technique for transactions handling. Section 4 describes the implementation environment for a simple system that applies the shadow technique. Section 5 includes the conclusions and future works.

II. RELATED WORK

Most of the papers that handle the transaction processing in mobile database assume long disconnection and working offline as [6] [7] [8] [9] [10] [11] [12] [13] [14] [16].

They relax or redefine the strict ACID properties of traditional transaction model in order to fit to mobile computing environment [17].

The Kangaroo model [16], supports the moving of the transaction from cell to cell by moving the control and the history of the transaction among MSSs. The PRO-MOTION model [6], uses the compact structure to support disconnection and working offline. The Moflex model [17], based on set of dependencies, acceptable goals and rules. It supports location dependent subtransactions. The MDSTPM (Multidatabase System Transaction Processing Manager) model [18], uses the message and queuing facility to support MH requests, data exchange, and disconnection mode. The Clustering model [19] divides the items of the database into clusters. Clusters are the units of consistency, when an MH is disconnected it becomes a cluster by itself. It uses two copies of objects to maintain consistency.

Most of the papers do not consider replication in the update process and assume the long duration transaction consists of totally independent subtransactions and doesn't consider the case when the subtransactions are dependent.

III. ANALYSIS AND SOLUTIONS

We can analyze the moving of the mobile user to 4 areas (Fig. 4) [15]:
1. Moving from one server area to many servers-shared area which contains the current server of the mobile user as from a→b
2. Moving to shared area that does not contain the current server of the mobile user as b →c
3. Moving to another one-server area as d→e
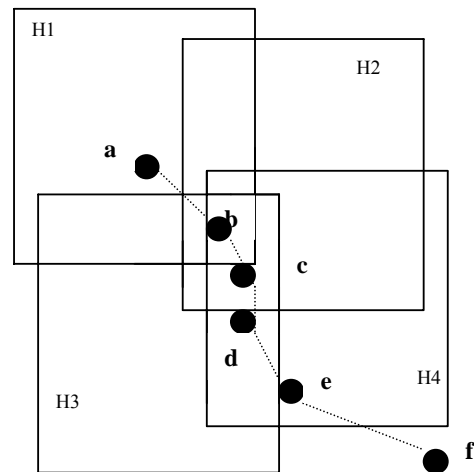4. Moving to area that has no servers at all as e→ f

Fig. 4 Moving of the mobile user

A. Description for Program and Transaction

Our view of a user program that access the database as a structure consists of three parts (sets).

1. read set : includes the select statements in the program and retrieves the dataset from the server.

2. calculation set : is the user inputs and modifications on the data set according to the business logic (rules).

3. write set : includes all the (insert , update , delete) statements in the program. It is written inside begin transaction and end transaction (commit or abort) statements. Its effects is directly on the database and can commit or rollback.

The read set and the write set are executed by the computer and they are very fast and accurate. The calculation set is very slow because of user interaction; it takes effort and time from the user.

Transaction concept is the key to the database consistency and integrity. Our view for the transaction, that it is the three parts together not only the write set. Because the write set depends on the read set and the calculation set, the write set alone has no meaning. We see the whole program is the transaction and the write set is a subtransaction that completes and finalizes the work and the logic of the program.

Consistency is the responsibility of the DBMS through the execution of the transaction, which means that, the database

will not change till the transaction commit. Consistency is the responsibility of the programmers through the design time, to ensure that the logic of the transaction leaves the database consistent.

We define transaction as: Transaction is a program in execution in which write set satisfies the ACID properties.

Our approach keeps the calculation set available while disconnection time in main memory and after closing the program on secondary storage and supporting the user by giving him the ability to continue running his program after short disconnection or long disconnection.

We classified compound transactions (that consists of two or more subtransactions) according to the relationship among their subtransactions to three classes: independent, fully dependent and partially dependent subtransactions.

- Independent Subtransactions

Every subtransaction is complete by itself and its success or failure doesn't depend on any other subtransaction, so they are semantically independent. The subtransaction that completes its work commits on the active database as a separate transaction and unlocks all its locked data items. If the subtransaction failed, the next subtransaction in the compound transaction should start its execution, because there is no dependency between them.

- Partially dependent subtransactions

The compound transaction consists of some dependent subtransactions and some independent subtransactions. We will study the update techniques for this type of transactions in our future study in another paper.

- Fully dependent subtransactions

Every subtransaction depends on the success of execution and commitment of the previous subtransaction. Failure of one subtransaction will stop and fail the processing of the long duration transaction because they are semantically dependent. The order of execution is important and can not be changed, so there is only one way to execute this compound transaction and only one success state for the compound transaction [15].

B. Transaction Processing with and without Disconnection

In Table I, T1 and T2 start from the same initial state S0 and pass through the intermediate states without losing any state, the sequence is the same until the state Sm which is the last state before disconnection and the initial state after disconnection. Both go through the states from Sm+1 to Sn which is the final state, so both are equivalent. Our goal is to build a technique for mobile database transactions to prove this analysis [15].

TABLE I	
TRANSACTION PROCESSING WITH AND WITHOUT DISCONNECTION	
Transaction T1 Without Disconnection	Transaction T2 With Disconnection
Initial state : S0 , S1 , S2 , . . Intermediate Sm, State	Initial state : S0 , S1 , S2 , . . Intermediate Sm, (before disconnection)
Sm+1, Sm+2 , . .	Disconnection Point . .
Sn = final	Sn = final

C. Managing Compound Transactions Updates (Shadow Technique)

We assume that the system is partially replicated distributed database system, because it is the most frequent practical and acceptable. We also assume that the mobile unit has a software package that can contact with the server and send and receive data from it.

We classified the computers that are involved in the update transaction into two groups:

- The basic group: consists of primary site and mobile unit. They are enough to complete the transaction.
- The complementary group: consists of all the remaining sites (replicas) that are involved in the update operation and we assume using lazy replication protocol for refreshment.

D. Scenario for Managing Compound Transactions Updates (shadow technique)

1 When the mobile user switches on the mobile unit and connects to the server. The program on the mobile unit sends request to the server to retrieve the current available dataset on it (we call this read set phase).

2 If a disconnection happened through the read set phase, the mobile unit (the program) tries to reconnect with the server for a predefined period,

3 If the mobile unit succeeds, then it reissues the request statement only (select statement only) for retrieving the read set , if it fails to reconnect with the server, the user has the options to close the program or tries to reconnect

with the server again. The select statement is not a transaction, it is only statement, because it does not effect the database and we can not rollback the result of select statement.

4   When the user receives the current dataset from the server, the program copies the current dataset (we call it original dataset) as shadow dataset (we call it shadow copy).

5   The user starts to do his updates (insertion, deletion, modification) on the shadow copy.

6   If the disconnection happened while the user doing his updates on the shadow copy (because of moving from the current cell to new cell or network failure), the program would not detect the network disconnection, but the operating system of the mobile unit would detect it and gives indication to the user.

7   When the user finishes his updates and the network state is active and the connection is available, then

8 Independent Case

which means that the relationship among the records in the dataset is independent, so the relationship among the subtransactions of the program is also independent. The control of the writeset transaction would be on the server. The server issues the begin-transaction and commit or rollback statements (We implemented it as stored procedures on the server).

8.1   The mobile unit sends the datasets (record by record) to the server, the update process starts as the following:

a)   The current record in the dataset is Modified

- The mobile unit sends the original record and the shadow record to the server.
- The primary site server starts the validation phase of the write set transaction for the shadow record by comparing its original values with the current value of the same record in the active database,
  - a. If both are equal, the primary site server modifies the active database from the shadow record in exclusive mode (Serializable mode).
  - b. If both are not equal, the primary server does not modify the active database, aborts the write set transaction for the current shadow record and sends failure message to the mobile unit.

b)   The current record in the dataset is Added

- The mobile unit sends the shadow record only to the primary server.
- The primary site server starts the validation phase of the write set

transaction for the shadow record by checking that the record is not exist in the active database :
  - a. If the result of the comparison is true, the primary server inserts the shadow record into the active database, in exclusive mode (Serializable mode).
  - b. If the result of the comparison is false, the primary server aborts the write set transaction for the current shadow record and sends failure message to the mobile unit.

c)   The current record in the dataset is Deleted

- The mobile unit sends the value of the primary key field of the shadow record to the primary server.
- The primary server starts the validation phase of the write set transaction for the shadow record by checking that the record is exist in the active database:
  - a. If the result of the comparison is true, the primary server deletes the record whose key field equals the shadow key field from the active database in exclusive mode (Serializable mode).
  - b. If the result of the comparison is false, the server aborts the writeset transaction for the shadow record.

8.2   In the three cases, if the writeset transaction committed or rollbacked according to the integrity constrain of the database, the server sends committed or rollbacked message to the mobile unit, and the mobile unit would remove the shadow record from the shadow copy.

8.3   If the disconnection happened after updating some records and before completing the updates of all records in the shadow dataset (we assume that the user has big number (10,100,1000 ,…) of records for update) then :

8.3.1   The mobile unit tries to reconnect with the server (directly or through any local server on the network), for a predefined period (short disconnection),

8.3.2   If the mobile unit succeeds, the program reissues the write set statement only for the current shadow record which the disconnection happened through its update (because all variables and datasets still available in the main memory).

8.3.3   The primary server starts the validation phase of the write set transaction for the shadow record (which the disconnection

happened through its update) and committing or rolls backing its transaction, and then the program would continue as the disconnection not happened and updates the remaining records in the shadow dataset.

8.3.4 If the reconnection to the primary site failed (long disconnection), the program saves the datasets (the original dataset and the remaining elements of the shadow dataset including the shadow record that the disconnection happened through its update) as external files (we implemented it as XML files) on the mobile unit secondary storage.

8.3.5 When reconnection (after long disconnection) with the primary site is available (directly or through any local server on the network), the program searches the secondary storage of the mobile unit for external files (XML files). If it not find any file, the program starts normally with blank form, If it found XML files related to the database operations, the program loads the XML files automatically and presents it to the user.

8.3.6 The user can (modify or delete or insert) any record in the loaded dataset, after he finishes, he clicks update, the program sends the original and the shadow datasets to the primary server, which start the validation and update for the remaining element in the shadow dataset as the disconnection not happened.

8.3.7 When the writeset transaction complete, the program deletes the external files and displays the current active dataset from the server after update.

9 Fully Dependent Case

which means that the relationship among the records in the dataset is dependent, so the relationship among the subtransactions of the program is also dependent. The control of the writeset transaction would be on the mobile unit (the client). The mobile unit issues the begin-transaction and commit or rollback statements (We implemented it as stored procedures on the server without transaction).

9.1 The mobile unit starts the write set transaction (issues begin-trans statement in exclusive mode (Serializable mode),then it sends the dataset record by record to the server and the update process starts as the following:

a)   The current record in the dataset is Modified

- The mobile unit sends the original record and the shadow record to the server
- The primary site server starts the validation phase of the write set transaction for the current shadow record by comparing its

original values with the current value of the same record in the active database,

a- If both are equal, the primary server modifies the active database from the shadow record and sends success message to the mobile unit when the modification completes.

b- If both are not equal, the primary server does not modify the active database, and sends failure message to the mobile unit,

c- The mobile unit aborts the write set transaction for all shadow records and clears the memory to start new transaction.

b)   The current record in the dataset is Added

- The mobile unit sends the shadow record only to the server
- The server starts the validation for the shadow record by checking that the record not exist in the database:

a- if the result of the comparison is true, the primary server inserts the shadow record into the active database and send success message to the mobile unit when the insertion completes.

b- If the result of the comparison is false, the primary server sends failure message to the mobile unit.

c- The mobile unit aborts the write set transaction for all shadow records and clears the memory to start new transaction.

c)   The current record in the dataset is Deleted

- The mobile unit sends the value of the primary key field of the shadow record to the primary server.
- The server starts the validation for the shadow record by checking that the record exist in the database :

a- If the result of the comparison is true, the primary server deletes the record whose key field equals the shadow key field from the active database and sends success message to the mobile unit when the deletion completes.

b- If the result of the comparison is false, the primary server sends failure message to the mobile unit.

c- The mobile unit aborts the write set transaction for all shadow records and clears the memory to start new transaction.

9.2 In the three cases, if the mobile unit receives success message from the primary server, it continues sending another record from the shadow copy until all records update. If it receives one failure message for any record of the shadow copy, it issues rollback command to the primary server to undo any effected records on the active database and the mobile unit clears the memory to start new transaction.

9.3 If the disconnection happened after updating some records and before completing the updates of all records in the shadow dataset (we assume that the user has big number (10,100,1000 ,…) of records for update) then :

9.3.1 The primary server rollbacks any effected records on the active database.

9.3.2 The mobile unit tries to reconnect with the server (directly or through any local server on the network), for a predefined period,
    If it succeeds (after short disconnection) then:
    The mobile unit program reissues the write set transaction as a new transaction for the shadow dataset. All the previous updates are rollbacked on the primary server because the transaction not exists.
    Because the writeset transaction reissued as a new transaction, the program would continue as the disconnection not happen and update all the records of the shadow dataset because all variables and datasets still available in the main memory.

9.3.3 If the reconnection with the primary site failed (long disconnection), the program saves the datasets (the original dataset and the shadow dataset) as external files (we implemented it as XML files) on the mobile unit secondary storage.

9.3.4 When reconnection (after long disconnection) with the primary site is available (directly or through any local server on the network), the program searches the secondary storage of the mobile unit for external files (XML files). If it not finds any file, the program starts normally with blank form, if it found XML files related to the database operations, the program loads the XML files automatically and presents it to the user.

9.3.5 The user can modify or delete or insert as he wants in the loaded dataset. When he finishes, he clicks update button. The program sends the loaded dataset to the primary server to validate and update the active database from it as a new transaction (as the disconnection not happened).

9.3.6 When the writeset transaction completes, the program deletes the external files and displays the current active dataset from the server after update.

E. The Advantage of the Shadow Approach

1- The effort and time of user not lost because the calculation set not lost.

2- No transfer of logs or transaction history among sites.

3- New precise definition for the transaction and also more flexible.

4- No need to load the mobile unit with DBMS and replica, because the program saves the datasets as XML files and can retrieve it at reconnection time or at any time the user wants.

5- The program differentiates between the short disconnection and the long disconnection.

6- We exactly know where and when in the program the disconnection would happen, so we design the program with capability to handle it.

7- We manage the unpredictable disconnection.

8- All ACID properties are reserved.

9- No storage lost on the primary server or on the mobile unit, because after the transaction committed or rollbacked, the program deletes the XML files.

10- In the independent case only, the program reissues the writeset transaction for the record that the disconnection happened through its update, and the transaction (program) continues its work. The user can move among servers and the write set subtransactions can commit on the primary server from different local servers connections and all the subtransactions would be valid. So in this case, we can say that reading from any where and writing from any where is true and valid.

11- In the dependent case, the writeset transaction commit only from the last connection to the primary site through the last local server, This is logical, because write set transaction is not divisible by nature, so, the program can read from any where, but must writes (commits) from one location (direct connection or through local server) to the primary site.

12- The locking of records at the primary server is very short, only at the writing time.

13- The load on the primary server would be more lite, because there is no waiting time, the program contacts it to validate and update.

14- More control over the network disconnection, especially in wireless networks which its property is frequently disconnection.

15- The user can work offline, and the program would handle it as long disconnection.

16- The disconnection can happen many times through the program life time in the main memory, and the program can continue its work or generate XML files by the uncommitted records.

17- This technique decreases the deadlock rate, because the primary site locks the data items when its transaction is online, otherwise it unlocks the data items. Therefore, it increases the performance of the system but the cyclic restart problem may occur.

IV. Implementation Environment

We implemented our programs using Visual Basic .Net and SQL Server 2000 because it is easy to install them and they support many new features as writing and reading XML files.

We assume that the replication handling is solved as a distributed database problem using the lazy replication technique among fixed hosts.

V. Conclusion and Future Works

The mobile computing and moving objects area is very interesting, it includes many other subjects as networking concepts, operating system concepts, database concepts ,programming concepts, so it is a rich area for research.

In this paper, we concentrate on the disconnection that occurs when the mobile user moves from one service area to another passing through the shared area among the local servers. We defined the concept of transaction and classified the different types of transactions. Based on this analysis, we implemented our shadow technique that handles the disconnection because of moving among sites.

There are many points for future study as the following:

- If the whole network is wireless, how will be the transaction management?
- If the relationship among subtransactions is partially dependent, How will be the update processing?
- What is the optimal solution for selecting the next server in the shared area among many servers to decrease the number of disconnections?
- We implemented our simple system as homogeneous distributed database with partially replication, but if the database was heterogeneous distributed how will be the implementation?

References

[1] S. K. Madria," *Transaction Models for Mobile Computing*", Proc. of the 6th IEEE Conf. on Networks, SICON'98 ,World Scientific Publishing, 1998.

[2] Phyoung Jung Kim ,Young Ju Noh, "*Mobile Agent System Architecture for supporting Mobile Market Application Service in Mobile Computing Environment*", Proceedings of the 2003 International Conference on Geometric Modeling and Graphics (GMAG'03)© 2003 IEEE.

[3] Patrick O'neil and Elizabeth O'nell , "*Database Principles Programming Performance*" , Academic press 2001.

[4] Peter Rob and Carlos Coronel, "*Database systems-Design, Implementation, and Management*", Boyd & Fraser publishing 1995.

[5] Patricia Serrano-Alvarado_, Claudia L. Roncancio, Michel Adiba, "*Analyzing Mobile Transactions Support for DBMS*" ,Proceedings of the 12th International Workshop on Database and Expert Systems Applications (DEXA'01) © 2001 IEEE.

[6] Gary D. Walborn and Panos K. Chrysanthis, "*PRO-MOTION: Management of mobile transactions*", Proceedings of the 1997 ACM symposium on Applied computing April 1997.

[7] T. Imielinski and B. R. Badrinath, "*Mobile wireless computing: challenges in data management*", Communications of the ACM 1994.

[8] Ravi A. Dirckze and Le Gruenwald , "*A pre-serialization transaction management technique for mobile multidatabases*", Mobile Networks and Applications December 2000 Volume 5 Issue 4.

[9] Subhasish Mazumdar , Mateusz Pietrzvk,Paanos K Chrysanthis, "*Caching Constrained Mobile Data*", Proceedings of the Tenth International Conference on Conference on Information and Knowledge Management October 2001.

[10] Olaf Zukunft, "*Rule based adaptation in mobile database systems*", Proceedings of the 1997 ACM symposium on Applied computing April 1997.

[11] Omran Bukhres and Stuart Morton, "*Mobile Computing in Military Ambulatory Care*", Proceedings of the 10th IEEE Symposium on Computer-Based Medical Systems, Aug 1997.

[12] Cris Pedregal-Martin and Krithi Ramamritham, "*Support for Recovery in Mobile Systems*", IEEE Transactions on Computers, October 2002.

[13] Joanne Holliday , Divyakant Agrawal , Amr El Abbadi, "*Disconnection modes for mobile databases*", Wireless Networks July 2002 ,Volume 8 Issue 4.

[14] Gary D. Walborn and Panos K. Chrysanthis, "*Transaction processing in PRO-MOTION*", Proceedings of the 1999 ACM symposium on Applied computing February 1999.

[15] Osman Hegazy, Ali El Bastawissy and Romani Ibrahim, "*Technique for Handling Transactions that Move among Hosts in Mobile Databases*", Proceedings of the International Conference on Computational Intelligence, December 2004, Istanbul, Turkey (ICCI 2004).

[16] Margaret H. Dunham , Abdelsalam Helal b and Santosh Balakrishnan , "*A mobile transaction model that captures both the data and movement behavior*", Mobile Networks and Applications 2 (1997), Baltzer Science Publishers BV.

[17] Yin-Huei Loh, Takahiro Hara, Masahiko Tsukamoto, Shojiro Nishio, "Moflex Transaction Model for Mobile Heterogeneous Multidatabase Systems", Proceedings of the symposium on Applied computing March 2000 ACM.

[18] Yeo, L. H. and A. Zaslavsky. "*Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment.*" Proceedings of the 14th International Conference on Distributed Computing Systems. 1994.

[19] E. Pitoura and B. Bhargava. *Building Information Systems for Mobile Environments*. In N. R. Adam, B. Bhargava, and Y. Yesha, editors, Proc. of the 3rd Int. Conf. on Information and Knowledge Management (CIKM'94), ACM Press, 1994.