3-L Model: A Model for Checking the Integrity Constraints of Mobile Databases

Hamidah Ibrahim
Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia, Malaysia
hamidah@fsktm.upm.edu.my

Zarina Dzolkhifli
Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia, Malaysia
zarinadzolkhifli@yahoo.com.sg

Lilly Suriani Affendey
Universiti Putra Malaysia, Malaysia
suriani@fsktm.upm.edu.my

Praveen Madiraju
Department of Mathematics, Statistics and Computer Science,
Marquette University, USA
praveen@mscs.mu.edu

Received 16 June 2009; Revised 25 August 2009; Accepted 21 September 2009

In this paper we propose a model for checking integrity constraints of mobile databases called Three-Level (3-L) model, wherein the process of constraint checking to maintain the consistent state of mobile databases is realized at three different levels. Sufficient and complete tests proposed in the previous works together with the idea of caching relevant data items for checking the integrity constraints are adopted. This has improved the checking mechanism by preventing delays during the process of checking constraints and performing the update. Also, the 3-L model reduces the amount of data accessed given that much of the tasks are performed at the mobile host, and hence speeds up the checking process.

Categories and Subject Descriptors: Abstract Interpretation [**Database Management**]: Systems – *Relational Databases*, *Transaction Processing*

General Terms: Algorithms, Management, Measurement, Performance
Additional Key Words and Phrases: Mobile Databases, Integrity Constraints, Constraint Checking

1. INTRODUCTION

Recently, there has been an increasing interest in mobile computing due to the rapid advances in wireless communication and portable computing technologies. Massive research efforts from academia and industry have been put forth to support a new class of mobile applications such as just-in-time stock trading, mobile health services, mobile commerce, and mobile games as well as migrating the normal conventional applications to mobile applications. Users of these applications can access information at any place at any time via mobile computers and devices such as mobile phone, palmtops, laptops, and PDA [Ken et al. 2006].

While technology has been rapidly advancing, various constraints inherited from limitations of wireless communication and mobile devices remain primary challenges in the design and implementation of mobile systems and applications. These constraints include: limited client capability, limited bandwidth, weak connectivity, and user mobility. Mobile devices generally have poor resources and thus it is usually impossible for them to store all data items in the network. In addition, disconnections occur frequently, which may be intentional or unintentional. These constraints make the wireless and mobile computing environments uniquely different from a conventional wired server/client environment [Ken et al. 2006].

A general architecture of a mobile environment is shown in Figure 1 [Chan and Roddick 2003 and Ken et al. 2006]. The architecture consists of base stations (BS) and mobile hosts (MH). The base station is a stationary component in the model and is responsible for a small geographic area called a cell. Each cell is connected to the other ones through fixed networks. The mobile host is the mobile component of the model and may move from one cell to another. These mobile hosts communicate with the base stations through wireless networks.

Since a mobile host is not capable of storing all data items in the network, thus it must share some data item with a database in the fixed network. Any update operation or transaction that occurs at the mobile host must guarantee *database*

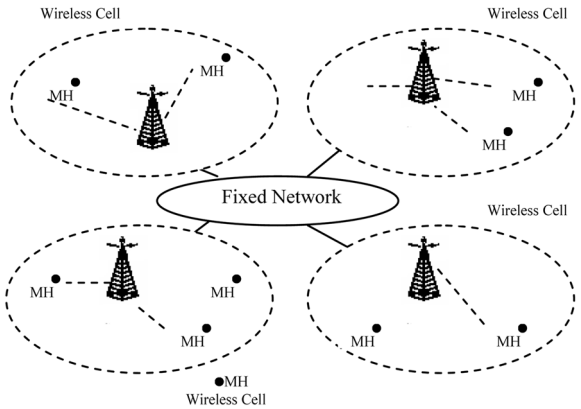

Figure 1. The Architecture of a Mobile Database Environment.

consistency. A database state is said to be consistent if the database satisfies a set of statements, called *integrity constraints*, which specify those configurations of the data that are considered semantically correct. The process of ensuring that the integrity constraints are satisfied by the database after it has been updated is termed *constraint checking*, which generally involves the execution of *integrity tests*. In a mobile environment, checking the integrity constraints to ensure the correctness of the database spans at least the mobile host and one other database (node), and thus the update is no longer local but rather distributed [Mazumdar and Chrysanthis 2004]. As mentioned in [Mazumdar and Chrysanthis 2004], the major problems in the mobile environment that are unbounded and unpredictable delays can affect not only the update but other updates running at both the mobile and the base stations, which is clearly not acceptable for most applications. With the same intuition as [Mazumdar and Chrysanthis 2004], we address the challenge of extending the data consistency maintenance to cover disconnected and mobile operations.

In this paper, a model called the Three-Level (3-L) is proposed where checking the consistency of mobile databases is performed at three different levels. This model is suitable for both intentional (planned) and unintentional (unplanned) disconnection. This model differs from the approach proposed in [Mazumdar and Chrysanthis 2004] since it is intended to cater for the important and frequently used integrity constraints, i.e. those that are used in database applications. Mazumdar's approach [Mazumdar and Chrysanthis 2004] is restricted to set-based constraints (equality and inequality constraints). In our work, in order not to delay the process of checking constraints during disconnection, a similar concept as proposed in distributed databases [Ibrahim et al. 2001] is employed, namely localizing integrity checking by adopting sufficient and complete tests. Since sufficient test can only verify if a constraint is satisfied, we propose that the data items required for the checking to be cached at the mobile host during the relocation period. Our model not only treats the issue of disconnection but also reduces the amount of data accessed during the process of checking the consistency of the mobile databases. Hence, we achieve speed up in the constraint checking process.

This paper is organized as follows. In Section 2, the previous works related to this research are presented. In Section 3, the basic definitions, notations and examples, which are used in the rest of the paper, are set out. Section 4 describes the 3-L model while Section 5 discusses the performance of the 3-L model. Conclusions and further research are presented in the final section, 6.

2. RELATED WORK

Much of the research concerning integrity constraint checking has been conducted in the area of relational database systems. A comprehensive survey on the issues of constraint checking and maintaining in centralized, distributed and parallel databases is provided in [Ibrahim 2006]. A naïve approach is to perform the update and then check whether the integrity constraints are satisfied in the new database state. This method, termed *brute force checking*, is very expensive, impractical and can lead to prohibitive processing costs because the evaluation of integrity constraints requires large amounts of data, which are not involved in the database update transition.

Hence, improvements to this approach have been reported in many research papers. Many approaches have been proposed for constructing efficient integrity tests, for a given integrity constraint and its relevant update operation, but these approaches are mostly designed for a centralized environment [Martinenghi 2005]. As centralized environment has only a single site, the approaches concentrate on improving the checking mechanism by minimizing the amount of data to be accessed during the checking process. Hence, these methods are not suitable for mobile environment as the checking process often spans multiple nodes and involves the transfer of data across the network.

Although there are a few studies that have been conducted to improve the checking mechanism by reducing the amount of data transferred across the network in distributed databases such as [Alwan et al. 2007; Gupta 1994; Ibrahim et al. 2001; Madiraju and Sunderraman 2004], but these approaches are not suitable for mobile databases. These approaches reformulate the global constraints into local constraints (local tests) with an implicit assumption that all sites are available, which is not true in mobile environment, since a mobile unit may be disconnected for long periods. Even though failure is considered in the distributed environment, but none of the approach caters failure at the node where the update is being executed, i.e. disconnection at the target site. Nevertheless, the localization concept proposed in distributed databases is used in our approach.

Other approaches such as [Hanandeh 2006; McCarroll 1995] focus on the problems of checking integrity constraints in parallel databases. These approaches are not suitable for mobile databases as the intention of their approach is to speed up the checking process by performing the checking concurrently at several nodes. To the best of our knowledge, PRO-MOTION [Mazumdar and Chrysanthis 2004] is the only work that addresses the issues of checking integrity constraints in mobile databases. The difference between our work and the work in [Mazumdar and Chrysanthis 2004] has been highlighted in the previous section.

3. PRELIMINARIES

Our approach has been developed in the context of relational databases. A database is described by a database schema, D, which consists of a finite set of relation schemas, $<R_1, R_2, ..., R_m>$. A relation schema is denoted by $R(A_1, A_2, ..., A_n)$ where R is the name of the relation (predicate) with n-arity and A_i's are the attributes of R. Database integrity constraints are expressed in prenex conjunctive normal form with the range restricted property. A conjunct (literal) is an atomic formula of the form $R(u_1, u_2, ..., u_k)$ where R is a k-ary relation name and each u_i is either a variable or a constant. A positive atomic formula (positive literal) is denoted by $R(u_1, u_2, ..., u_k)$ whilst a negative atomic formula (negative literal) is prefixed by $\neg$. An (in)equality is a formula of the form $u_1\ \theta\ u_2$ (prefixed with $\neg$ for inequality) where both u_1 and u_2 can be constants or variables and $\theta \in \{<, \leq, >, \geq, \neq, =\}$.

In the database literature, many types and variations of integrity tests have been described [Ibrahim et al. 2001; McCarroll 1995]. The classifications of integrity tests are based on some of their characteristics as explained below:

(a) Based on when the integrity test is evaluated: (i) *post-tests* - allow an update

operation to be executed on a database state, which changes it to a new state, and when an inconsistent result is detected undo this update. The method that applies these integrity tests is called the detection method. (ii) *pre-tests* - allow an update to be executed only if it changes the database state to a consistent state. The method that applies these integrity tests is called the prevention method.
(b) Based on region: (i) *local tests* – verify the consistency of a database within the local region, i.e. by accessing the information at the local site. The method that adopts these integrity tests is called the *local method*. (ii) *global tests* - verify the consistency of a database outside the local region, i.e. by accessing the information at the remote site(s). The method that adopts these integrity tests is called the *global method*.
(c) Based on its properties as defined by McCarroll [McCarroll 1995]: (i) *sufficient tests* - when the test is satisfied, this implies that the associated constraint is satisfied and thus the update operation is safe with respect to the constraint. (ii) *necessary tests* - when the test is not satisfied, this implies that the associated constraint is violated and thus the update operation is unsafe with respect to the constraint. (iii) *complete tests* - has both the sufficiency and the necessity properties. In a mobile environment due to the limited capacity of a mobile host, a test defined as complete in [McCarroll 1995] has the sufficiency property and *not* necessarily the necessity property. For example, consider the test $(\exists x \exists y \exists z)(dept(b, x, y, z))$ which is a complete test for the integrity constraint I_1: $(\forall t \forall u \forall v \forall w \exists x \exists y \exists z)(emp(t, u, v, w) \rightarrow dept(u, x, y, z))$ and update operation insert($emp(a, b, c, d)$), i.e. when inserting a new employee record into the *emp* relation, the referential integrity constraint I_1 is not violated if the *dno* = *b* is found in the *dept* relation (sufficiency property) and violated otherwise (necessity property). Since a mobile host is not capable of storing all data items in a network, thus if the information required, *dno* = *b*, is not found in the mobile host, this does not imply that the initial constraint I_1 is violated (*not* holding the necessity property). Therefore, the execution of integrity tests in mobile databases is different from the traditional databases.

Throughout this paper, the following symbols and their intended meaning, which are related to integrity constraints, are used:·

- $I^{\upsilon} = \{I_1, I_2, ..., I_M\}$, the set of integrity constraints of an application in the whole mobile system.
- $I^{Bi} = \{I^{Bi}_1, I^{Bi}_2, ..., I^{Bi}_N\}$, the set of integrity constraints at the base station, *i*.
- $I^{Mh} = \{I^{Mh}_1, I^{Mh}_2, ..., I^{Mh}_O\}$, the set of integrity constraints at the mobile host, *h*.
- $T_i = \{T_{i1}, T_{i2}, ..., T_{iw}\}$, the set of integrity tests for a given constraint I_i of I^{υ}.

From the above, $(\cup^{P}_{i=1} I^{Bi}) \cup (\cup^{Q}_{h=1} I^{Mh}) = I^{\upsilon}$, where P and Q are the number of base stations and mobile hosts, respectively in the mobile system.

Similarly, the following are the symbols and their intended meaning that are related to the data items in the mobile system. Here, data item refers to relation or fragments of relations that appear in the specification of an update operation.

- $R^{\upsilon} = \{R_1, R_2, ..., R_S\}$, the set of relations or fragments of relations in the mobile system.
- $R^{Bi} = \{R^{Bi}_1, R^{Bi}_2, ..., R^{Bi}_T\}$, the set of relations or fragments of relations at the base station, *i*.
- $R^{Mh} = \{R^{Mh}_1, R^{Mh}_2, ..., R^{Mh}_U\}$, the set of relations or fragments of relations at the

mobile host, h.

From the above, $(\cup^{P}_{i=1} R^{Bi}) \cup (\cup^{Q}_{h=1} R^{Mh}) = R^{\upsilon}$, where P and Q are the number of base stations and mobile hosts, respectively in the mobile system. Also, we assume that for each data item, $R^{Mh}_{\upsilon} \in R^{Mh}$, the same data item appears in one of the base station, i.e. $R^{Mh}_{\upsilon} \in (\cup^{P}_{i=1} R^{Bi})$ [EPFL et al. 2004]. Thus, $(\cup^{Q}_{h=1} R^{Mh}) \subset (\cup^{P}_{i=1} R^{Bi}) = R^{\upsilon}$.

Update operation in a mobile environment can occur at two different levels:

• $U^{Bi}(R)$, an update operation over the relation R, submitted by a user at the base station, i. This type of update operation is not considered in this paper, as this is similar to the update operation in distributed databases. Note that R can also be a fragment of relation.

• $U^{Mh}(R)$, an update operation over the relation R, submitted by a user through his/her mobile host, h, where R is located at the mobile host. Note also that R can be a fragment of relation.

The symbol, $C(R)$, is used to denote the list of relations, R that occurs in the specification of a construct, C, where C can be an update operation, an integrity constraint or an integrity test. Throughout this paper the *company* database is used,

Schema: emp(eno, dno, ejob, esal); dept(dno, dname, mgrno, mgrsal); proj(eno, dno, pno)
Integrity Constraints:
'The *dno* of every tuple in the *emp* relation exists in the *dept* relation'
I_1: $(\forall t \forall u \forall v \forall w \exists x \exists y \exists z)(emp(t, u, v, w) \rightarrow dept(u, x, y, z))$
'The *eno* of every tuple in the *proj* relation exists in the *emp* relation'
I_2: $(\forall u \forall v \forall w \exists x \exists y \exists z)(proj(u, v, w) \rightarrow emp(u, x, y, z))$
'Every employee must earn ≤ to the manager in the same department'
I_3: $(\forall t \forall u \forall v \forall w \forall x \forall y \forall z)(emp(t, u, v, w) \wedge dept(u, x, y, z) \rightarrow (w \leq z))$
'Any department that is working on a project P_1 is also working on project P_2'
I_4: $(\forall x \forall y \exists z)(proj(x, y, P_1) \rightarrow proj(z, y, P_2))$

Figure 2. The *Company* Static Integrity Constraints.

Table I. The Integrity Tests Derived Based on the Integrity Constraints Listed in Figure 2.

I^{υ}	Update Template	Integrity Test
I_1	insert($emp(a, b, c, d)$)	1. $(\exists x \exists y \exists z)(dept(b, x, y, z))^1$
		2. $(\exists t \exists \upsilon \exists w)(emp(t, b, \upsilon, w))^2$
	delete($dept(a, b, c, d)$)	3. $(\forall t \forall \upsilon \forall w)(\neg emp(t, a, \upsilon, w))^1$
I_2	insert($proj(a, b, c)$)	4. $(\exists x \exists y \exists z)(emp(a, x, y, z))^1$
		5. $(\exists \upsilon \exists w)(proj(a, \upsilon, w))^2$
	delete($emp(a, b, c)$)	6. $(\forall \upsilon \forall w)(\neg proj(a, \upsilon, w))^1$
I_3	insert($emp(a, b, c, d)$)	7. $(\forall x \forall y \forall z)(\neg dept(b, x, y, z) \vee (d \leq z))^1$
		8. $(\exists t \exists \upsilon \exists w)(emp(t, b, \upsilon, w) \wedge (w \geq d))^2$
I_4	insert($proj(a, b, P1)$)	9. $(\exists z)(proj(z, b, P_2))^1$
		10. $(\exists z)(proj(z, b, P_1))^2$
	delete($proj(a, b, P2)$)	11. $(\forall x)(\neg proj(x, b, P_1))^1$
		12. $(\exists z)(proj(z, b, P_2) \wedge (z \neq a))^2$

Note: a, b, c and d are generic constants; [1]: complete test; and [2]: sufficient test

as given in Figure 2. Here, we assumed that there are three relations, namely: employee (*emp*), department (*dept*), and project (*proj*). Due to space limitation, only referential and general semantic integrity constraints are used in the examples. Table I presents some of the integrity tests generated based on the set of integrity constraints given in Figure 2. Here, we categorized the type of tests based on the definitions given in [McCarroll 1995]. The derivation of the integrity tests is omitted here since this is not the focus of this paper. Interested readers may refer to [Ibrahim et al. 2001].

4. THE 3-L MODEL

As mentioned earlier, this research proposes a model, called the 3-L model to ensure that the consistency of mobile databases is maintained. As the name implies, the model consists of three distinct levels, as depicted in Figure 3.

When a user submits an update operation $U^{Mh}(R)$, through a mobile host h, the list of constraints, I^{Mh}, at the mobile host is checked. Violation of any of the constraints will abort the operation. Otherwise, if the checking process does not require information from the other sites, then I^{Mh} is said to be satisfied and the update operation is safe to be performed. The second level is invoked if the information stored at the mobile host is not sufficient to validate whether the constraint I^{Mh} is violated or not. At the first level, the process of checking the constraints spans only the mobile host, i.e. local to the mobile host. The type of test suitable for this level is the sufficient test with the existential quantifier since the mobile host has limited capacity and thus the information (relations) stored at the mobile host is limited. Referring to the Table I, $(\exists t \exists v \exists w)(emp(t, b, v, w))$ is an example of a sufficient test, which checks the existence of at least an employee who is currently working in the department b when an insert operation insert($emp(a, b, c, d)$) is executed. If such information is available at the mobile host, then we conclude that the initial constraint, I_1, is satisfied. If there is no such information, then further checking needs to be performed. The properties of the sufficient test can be *upgraded* to be similar to

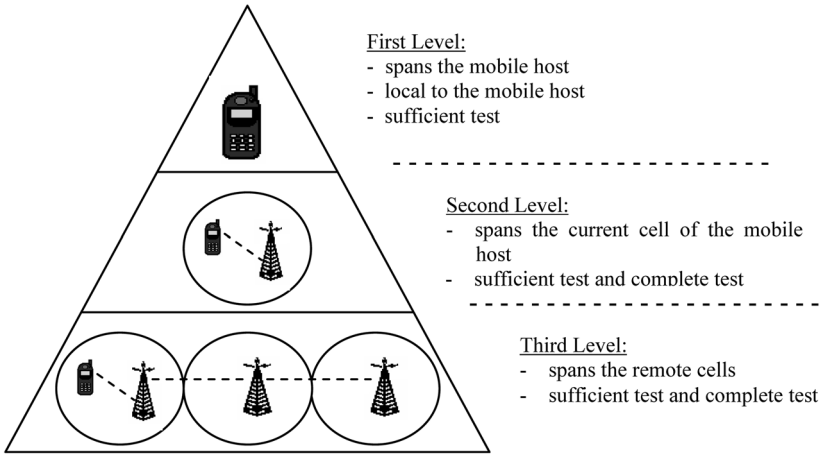

Figure 3. The 3-L Model.

the properties of the complete test if all possibilities of values for the required data item are cached to the mobile hosts. For example, referring to the sufficient test ($\exists t \exists v \exists w$)(*emp*($t$, b, v, w)), one notices that comparison is performed against the values of the *dno*. Assuming that the company has four departments, a vertical fragment of the *dept* relation consisting of the distinct *dno* is cached to the mobile host, then performing the test against these data items can verify whether the test is satisfied or not, and eventually verify if the initial constraint is satisfied or violated. Caching can be performed during the relocation period.

The second level commences if the mobile host failed to validate the truth of the I^{Mh}. The base station in the current position of the mobile host is responsible for checking the constraints. The base station checks the validity of the constraints against the data stored at its location. At this level, the process of checking the constraints spans the current cell of the mobile host, i.e. local to a cell of the current location of the mobile host. The types of test suitable for this level are the sufficient test and the complete test with the existential quantifier. If the information stored at the base station is not sufficient then the next level is invoked. However, if violation is detected then the base station notifies the mobile host to abort the update operation. The update operation is safe to be performed if no violation is detected.

The next level, third level, spans the remote base station(s), checks the validity of the constraints against the data stored at the remote site(s). Depending on the protocol of the mobile environment, either the flooding technique or the broadcasting technique is used to perform the constraint checking at this level. Here, the types of test that can be used are sufficient as well as complete test.

Below are the procedures that are employed in the Three-Level (3-L) model and their algorithms (in pseudo code) are presented in Figure 4.

• Procedure *LocateIntegrityConstraints&Tests*(I^v, I^{Mh}, I^{Bi}, T^{Mh}, T^{Bi}) – this procedure is invoked to determine the location of each of the integrity constraints in the I^v. An integrity constraint with its associated integrity tests is located either at the mobile host or base station or both depending on the data items referred in the specification of the constraint and the data items stored at the mobile host and base stations. For example, $I^{Mh} = \{I_1, I_2, I_3\}$ and $T^{Mh} = \{I_1{:}\{1, 2\}, I_2{:}\{6\}, I_3{:}\{7, 8\}\}$ are located at a mobile host, h, which stored part/fragment of the *emp* relation.

• Procedure *CacheDataItem*(T^{Mh}, *DataItem*) – this procedure analyses the list of integrity tests located at the mobile host, T^{Mh}, and caches the required data items to the mobile host. This is performed by the base station of the current cell of the mobile host during the relocation period. There are several techniques that can be used to minimize the amount of data items to be cached. One of these techniques proposed by [Dzolkhifli et al. 2008a] analyses the relationships between the integrity tests to be evaluated for a given update operation. If it is known that a data item, D, is frequently needed for evaluating several integrity tests, then the data item D should be cached at the mobile host to increase the number of local processing and number of hit ratio. For example, an update operation, insert(*emp*(a, b, c, d)), will trigger the following integrity tests, ($\exists t \exists v \exists w$)(*emp*($t$, b, v, w)) (test 2) and ($\exists t \exists v \exists w$)(*emp*($t$, b, v, w) $\wedge$ ($w \geq d$)) (test 8). Analyzing the relationship between these tests, one noticed that both tests required the data item, $dno = b$. Assuming that the data item is not

Procedure *LocateIntegrityConstraints&Tests*(I^o, I^{Mh}, I^{Bi}, T^{Mh}, T^{Bi})
Input: the initial list of integrity constraints, I^o and its associated integrity tests
Output: the list of integrity constraints I^{Mh} and its associated integrity tests T^{Mh} located at mobile host h and the list of integrity constraints I^{Bi} and its associated integrity tests T^{Bi} located at the base station i
1. Begin
2. $I^{Mh} = \{\}$, $I^{Bi} = \{\}$, $T^{Mh} = \{\}$, $T^{Bi} = \{\}$
3. For each I_j in I^o do
4. Begin
5. For $h = 1$ to Q do If $I_j(R) \cap R^{Mh} \neq \{\}$ then $I^{Mh} = I^{Mh} \cup I_j$, $T^{Mh} = T^{Mh} \cup T_j$
6. For $i = 1$ to P do If $I_j(R) \cap R^{Bi} \neq \{\}$ then $I^{Bi} = I^{Bi} \cup I_j$, $T^{Bi} = T^{Bi} \cup T_j$
7. End
8. End

Procedure *CacheDataItem*(T^{Mh}, *DataItem*)
Input: the list of integrity tests located at h, T^{Mh}
Output: the list of data items to be cached to h
1. Begin
2. *DataItem* = {}
3. For each T^{Mh}_j in T^{Mh} do
4. Begin
5. Identify the required data item, D
6. *DataItem* = *DataItem* $\cup$ D
7. End
8. End

Procedure *SelectIntegrityConstraints*($U^{Mh}(R)$, I^{Mh}, *Selected-*I^{Mh})
Input: the update operation, $U^{Mh}(R)$ and the list of integrity constraints located at h, I^{Mh}
Output: the list of integrity constraints that might violate the $U^{Mh}(R)$, *Selected-*I^{Mh} with its associated integrity tests
1. Begin
2. *Selected-*I^{Mh} = {}
3. For each I^{Mh}_j in I^{Mh} do
4. Begin
5. If the relation and the operation of the update operation $U^{Mh}(R)$ = the relation and operation of the update template (see Table 1 as examples) of the I^{Mh}_j then
6. Begin
7. *Selected-*I^{Mh} = *Selected-*$I^{Mh} \cup I^{Mh}_j$
8. For each test T_{ji} of I^{Mh}_j do parameterize the test T_{ji} with details of the update $U^{Mh}(R)$
9. End
10. End
11. End

Procedure *FirstLevel*($U^{Mh}(R)$, *Selected-*I^{Mh}, *Action*)
Input: the update operation, $U^{Mh}(R)$ and the selected integrity constraints that might violate the $U^{Mh}(R)$, *Selected-*I^{Mh} with its associated integrity tests
Output: a call to the second level or action to perform the update $U^{Mh}(R)$
1. Begin
2. *SecondLevelI* = {}
3. For each I^{Mh}_j in *Selected-*I^{Mh} do
4. Begin
5. Invoke the sufficient test, *ST-*I^{Mh}_j of I^{Mh}_j if any

Figure 4. The Procedures of the 3-L Model.

6. If there is no $ST\text{-}I^{Mh}_j$ or $ST\text{-}I^{Mh}_j$ is false then $SecondLevelI = SecondLevelI \cup I^{Mh}_j$
7. End
8. If $SecondLevelI \neq \{\}$ then
   Action: Call *SecondLevel*(*MHId*, *BSId*, $U^{Mh}(R)$, *SecondLevelI, Action*)
   Else Action: Perform the$U^{Mh}(R)$
9. End

Procedure *SecondLevel*(*MHId, BSId,* $U^{Mh}(R)$, *SecondLevelI*, *Action*)
Input: the mobile host *ID*, *MHId*, the base station *ID*, *BSId*, the update operation, $U^{Mh}(R)$ and the integrity constraints that need checking, *SecondLevelI* with its associated integrity tests
Output: a call to the third level or a notification to perform the update $U^{Mh}(R)$
1. Begin
2. $ThirdLevelI = \{\}$
3. For each I^{Mh}_j in *SecondLevelI* do
4. Begin
5. Invoke the sufficient test, $ST\text{-}I^{Mh}_j$ of I^{Mh}_j if any
6. If there is no $ST\text{-}I^{Mh}_j$ or $ST\text{-}I^{Mh}_j$ is false then
7. Begin
8. Invoke the complete test, $CT\text{-}I^{Mh}_j$ of I^{Mh}_j
9. If ($CT\text{-}I^{Mh}_j(R) \cap R^{BC} = \{\}$, where *BC* is the current base station) or ($CT\text{-}I^{Mh}_j$ with existential quantifier is false) then $ThirdLevelI = ThirdLevelI \cup I^{Mh}_j$
   Else If the $CT\text{-}I^{Mh}_j$ with universal quantifier is false then
10. Begin
11. Action: Notify the mobile host that violation occurs, *Notify(MHId, BSId, Abort)*
12. Exit()
13. End
14. Else If $CT\text{-}I^{Mh}_j$ with universal quantifier is true then
15. Begin
16. $ThirdLevelI = ThirdLevelI \cup I^{Mh}_j$
17. Action: Vote: perform the $U^{Mh}(R)$, *Notify(MHId, BSId, Vote: Perform)*
18. End
19. End
20. If $ThirdLevelI \neq \{\}$ then Action: Call *ThirdLevel*(*MHId, BSId,* $U^{Mh}(R)$, *ThirdLevelI, Action*) Else Action: Notify the mobile host to perform the $U^{Mh}(R)$, *Notify(MHId, BSId, Perform)*
21. End
22. End

Procedure *ThirdLevel(MHId, BSId,* $U^{Mh}(R)$*, ThirdLevelI, Action)*
Input: the mobile host *ID*, *MHId*, the base station *ID*, *BSId*, the update operation, $U^{Mh}(R)$ and the integrity constraints that need checking, *ThirdLevelI* with its associated integrity tests
Output: a notification to abort or perform the update $U^{Mh}(R)$
1. Begin
2. For each I^{Mh}_j in *ThirdLevelI* do
3. Begin
4. Invoke the sufficient test, $ST\text{-}I^{Mh}_j$ of I^{Mh}_j if any
5. If there is no $ST\text{-}I^{Mh}_j$ or $ST\text{-}I^{Mh}_j$ is false then
6. Begin
7. Invoke the complete test, $CT\text{-}I^{Mh}_j$ of I^{Mh}_j
8. If $CT\text{-}I^{Mh}_j$ with universal quantifier is false then

Figure 4. Continued.

9. Begin
10. Action: Notify the mobile host that violation occurs, *Notify(MHId, BSId, Abort)*
11. Exit()
12. End
13. End
14. End
15. Action: Vote: perform the $U^{Mh}(R)$, *Notify(MHId, BSId, Vote: Perform)*
16. End

Procedure *Notify(MHId, BSId, Action)*
Input: the mobile host *ID*, *MHId*, the base station *ID*, *BSId*, the action, *Action*
Output: execute the action, *Action*
1. Begin
2. Read *Action*
3. If *Action* = *Abort* then abort the update operation, $U^{Mh}(R)$, and other *Action* from other base station (if any) will be ignored, Exit()
4. If *Action* = *Perform* then perform the update operation, $U^{Mh}(R)$ and other *Action* from other base station (if any) will be ignored, Exit()
5. If *Action* = *Vote: Perform* then analyze the votes received so far. The mobile host waits until it receives an action from one of the base station whose action falls under steps 3 or 4 or majority of the base stations vote for *Perform*.
6. End

Figure 4. Continued.

located at the requested mobile host and is one of the frequently accessed data item, then the caching of this data item at the mobile host will prevent delays during the process of checking constraints and performing the update as the mobile host can still perform the process locally even during disconnection. As it is not the intention of this paper to discuss the caching strategies, the procedure presented in this paper is rather general with the assumption that any caching strategies can be applied. Interested readers may refer to [Dzolkhifli et al. 2008b].

• Procedure *SelectIntegrityConstraints*($U^{Mh}(R)$, I^{Mh}, *Selected-*I^{Mh}) – this procedure is invoked to identify and select only those constraints from the list I^{Mh}, that might be violated given the update operation, $U^{Mh}(R)$. This approach known as the incremental checking has been adopted by many researchers. The procedure is executed at the mobile host. For example, a mobile host, *h*, which stored part of the *emp* and *dept* relations, will have I^{Mh} = {I_1, I_2, I_3} and T^{Mh} = {I_1:{1, 2, 3}, I_2:{6}, I_3:{7, 8}} located at its repository. Given an update operation $U^{Mh}(R)$, insert(*emp*(*a*, *b*, *c*, *d*)), then *Selected-*I^{Mh} = {I_1, I_3} as I_2 is proven to be true for the given update operation. This is based on the well-known update-theorem proposed by [Nicolas 1982]. Further optimization can be performed towards the set of integrity tests that have been selected. For instance, analyzing further the tests 2 and 8 shows that test 2 is subsumed by test 8 and thus if test 8 is true this implies that test 2 is also true. This can further reduce the number of integrity tests to be evaluated. Several optimization techniques can be applied at this stage. Interested readers may refer to [Ibrahim et al. 2001].

• Procedure *FirstLevel*($U^{Mh}(R)$, *Selected-*I^{Mh}, *Action*) – this procedure is invoked at the mobile host to check the validity of each of the constraint in the *Selected-*I^{Mh}, given the update operation $U^{Mh}(R)$. Based on the result, an appropriate action is

performed. Consider the example given in the previous procedure (without applying the optimization technique) then the mobile host h evaluates each of the following sufficient tests, 2 and 8, against its local data including the cached data. If both tests can be verified locally, then the update operation which performed else each test that the mobile host h failed to verify based on its data, is collected to be evaluated at the next level.

• Procedure *SecondLevel*(*MHId*, *BSId,* $U^{Mh}(R)$, *SecondLevelI*, *Action*) – this procedure is invoked at the current base station of the mobile host to check the validity of each of the constraint in the *SecondLevelI*, given the update operation $U^{Mh}(R)$. Based on the result, an appropriate action is performed. Continuing from the above example, assume that test 8 is failed in the *First Level* then this test is evaluated at the base station. If the test is true, then the mobile host is notified to perform the update operation. Else an alternative test (complete) is evaluated. For this example, test 7 is evaluated. Note that test 7 is a complete test with universal quantifiers. For such test, checking its validity should be performed at the entire system, i.e. at all remote sites that have a copy of the relation as specified in the test. Assuming that test 7 is true at the current base station (if it is false then the mobile host is notified to abort the update operation), then tests 7 and 8 are send to the *Third Level* to be evaluated at the remote base station(s).

• Procedure *ThirdLevel*(*MHId, BSId,* $U^{Mh}(R)$, *ThirdLevelI*, *Action*) – this procedure is invoked at the remote base station(s) to check the validity of each of the constraint in the *ThirdLevelI*, given the update operation $U^{Mh}(R)$. Based on the result, an appropriate action is performed. The steps performed are similar to the steps presented in the Procedure *SecondLevel* except these steps are performed at the base stations which are remote to the mobile host.

• Procedure *Notify*(*MHId*, *BSId*, *Action*) – this procedure is invoked at the mobile host to perform the action as indicated by *Action*. The action is either (i) perform the update operation when it is known that all the integrity tests in the *Selected-*I^{Mh} are true or (ii) abort the update operation when it is known that at least one of the integrity tests in the *Selected-*I^{Mh} is false.

The following theorem proves that given an update operation submitted to a mobile host it is sufficient to check only those integrity constraints that have been located at the mobile host.

Theorem 1: Given an update operation, $U^{Mh}(R)$, submitted at a mobile host, it is sufficient to check I^{Mh}, i.e. if I^{Mh} is satisfied then this implies that I^{υ} is satisfied.

Proof : For each $I_j \in I^{\upsilon}$, if $I_j(R) \cap R^{Mh} \neq \{\}$, then I_j is located at h, i.e. $I_j \in I^{Mh}$. Given an $I_k \in I^{\upsilon}$ and $I_k \notin I^{Mh}$, then $I_k(R) \cap R^{Mh} = \{\}$. Let $I^{\neg Mh}$ denotes this set of constraints, thus $I^{\upsilon} = I^{Mh} \cup I^{\neg Mh}$. Since the set of constraints $I^{\neg Mh}$ does not contain the relation R in its specification therefore the $I^{\neg Mh}$ is not violated with respect to $U^{Mh}(R)$. While the set of constraints I^{Mh} contains the relation R in its specification thus this set of constraints needs to be checked. Thus checking the I^{Mh} is sufficient.

Note that the above theorem does not state that the process of checking I^{Mh} is performed only at h, it might involve the whole mobile system depending on the scope covered by the I^{Mh}, more specifically by the test selected which is associated to the I^{Mh}.

Now let us consider a simple example to clarify the above model. Referring to the

Table II. Example Flows of the 3-L Model for $M1$.

Level	Variable	I	Test	Action
1 $M1$	$Selected\text{-}I^{Mh} = \{I_1, I_3\}$	I_1	2	Assume that the $ST\text{-}I^{Mh}$, 2, is true, thus the I_1 is satisfied and further checking at the second level is not required.
	$SecondLevelI = \{I_3\}$	I_3	8	Assume that the $ST\text{-}I^{Mh}$, 8, is false. Thus, I_3 needs to be checked at the second level. (Otherwise, the I_3 is satisfied and checking at the second level is not required.)
2 $B1$	$SecondLevelI = \{I_3\}$	I_3	8	Assume that the $ST\text{-}I^{Mh}$, 8, is false, then $CT\text{-}I^{Mh}$, 7, is checked. (Otherwise, the I_3 is satisfied and $M1$ can perform the update operation.)
	$ThirdLevelI = \{I_3\}$	I_3	7	Assume that $CT\text{-}I^{Mh}_j(emp) \cap R^{B1} = \{\}$, thus the I_3 needs to be checked at the third level. (Otherwise, if the $CT\text{-}I^{Mh}_j$, 7, is not satisfied then violation is detected, else I_3 needs to be checked at the third level.)
3 $B2$	$ThirdLevelI = \{I_3\}$	I_3	8	Assume that the $ST\text{-}I^{Mh}$, 8, is false, then $CT\text{-}I^{Mh}$, 7, is checked. (Otherwise, the $CT\text{-}I^{Mh}$, 7, is omitted and $M1$ can perform the update operation.)
		I_3	7	If the $CT\text{-}I^{Mh}_j$, 7, is satisfied $M1$ can perform the update operation otherwise violation is detected.

example given in Figure 2, assume that an update operation, insert(*emp*(*a*, *b*, *c*, *d*)), is submitted by the mobile host $M1$. Also, assume that there are only two base stations, $B1$, which is in the same cell as $M1$ and $B2$, which is the remote base station. Due to the limited capacity of the mobile host, only part of the *emp* relation is located at $M1$, while other relations are scattered between the base stations. Table II presents the possible flows in the Three-Level (3-L) model. Note that $I^o = \{I_1, I_2, I_3, I_4\}$, $I^{Mh} = \{I_1, I_2, I_3\}$, and $Selected\text{-}I^{Mh} = \{I_1, I_3\}$.

The above process can be simplified by applying the optimization technique [Ibrahim et al. 2001], where the $Selected\text{-}I^{Mh}$ is reduced to $\{I_3\}$. For another example, consider a transaction $T1$ = {insert(*emp*(*a*, *b*, *c*, *d*)), insert(*proj*(*a*, *b*, *e*))} is submitted by the mobile host $M2$. Also, assume that there are only two base stations, $B1$, which is in the same cell as $M2$ and $B2$, which is the remote base station. Due to the limited capacity of the mobile host, only part of the *emp* and *proj* relations are located at $M2$, while other relations are scattered between the base stations. Note that $I^o = I^{Mh} = Selected\text{-}I^{Mh} = \{I_1, I_2, I_3, I_4\}$. For the first part of the transaction, we assume the same flow as shown in Table II, while for the second part of the transaction, integrity constraints I_2 and I_4 need to be evaluated. Analyzing the integrity tests of I_2 (tests 4 and 5), it is clear that the tests are true if there exists an employee *a* in the *emp* or the *proj* relations. If the first part of the transaction $T1$ is successful, then these tests are ignored as the tests are verified as true based on the new inserted tuple. Hence, this leaves us with the integrity constraint I_4 and the possible flow is as shown in Table III. Meanwhile, if the first part of the transaction $T1$ is failed, then the whole

Table III. Example Flows of the 3-L Model for $M2$.

Level	Variable	I	Test	Action
1 $M2$	$Selected\text{-}I^{Mh} = \{I_4\}$ $SecondLevelI = \{I_4\}$	I_4	10*	Assume that the $ST\text{-}I^{Mh}$, 10 is false. Thus, I_4 needs to be checked at the second level. (Otherwise, the I_4 is satisfied and checking at the second level is not required.)
2 $B1$	$SecondLevelI = \{I_4\}$ $ThirdLevelI = \{I_4\}$	I_4	10	Assume that the $ST\text{-}I^{Mh}$, 10, is false, then the I_4 needs to be checked at the third level. (Otherwise, the I_4 is satisfied and $M2$ can perform the update operation.)
3 $B2$	$ThirdLevelI = \{I_4\}$	I_4	10	If the $ST\text{-}I^{Mh}$, 10, is satisfied $M2$ can perform the update operation otherwise violation is detected.

*Note that as both the tests 9 and 10 are specified over the same relation with existential quantifiers, thus test 10 should be evaluated rather than test 9. Note also as discussed earlier the test 10 is defined as complete test in [McCarroll 1995] but for mobile databases, this test has only the sufficiency property.

transaction is aborted due to the atomicity property of a transaction.

Based on the given example, we observe the 3-L model has the following benefits:

• The process of checking the constraints is performed at three different levels that span different sizes of areas. This reduces the amount of data accessed in particularly if the checking process involves only the first-level without having to go through the second and third levels.

• The model which supports both types of tests, namely: complete and sufficient, further reduces the complexity of checking the constraints. Adopting the sufficient test in this model is due to the characteristics of the test, which are (i) able to infer the information stored at the remote site(s) and (ii) give the opportunity to utilize as much as possible the information stored at the local site. These characteristics are suitable for a mobile environment in particular when the mobile hosts are disconnected from the entire system.

• Utilizing the appropriate caching strategy that caches frequently accessed data items for the purpose of checking integrity constraints can also enhance the performance of the checking mechanism. In addition, filtering the set of integrity constraints to be evaluated and optimizing the set can further improve and simplify the process of checking integrity constraints.

5. DISCUSSION OF PERFORMANCE

The key problem in integrity checking is how to efficiently evaluate the proposed checking mechanism. In this section, we estimate the performance of the 3-L model with respect to the following parameters [Ibrahim et al. 2001]. We use the symbol $C(R_1, R_2, ..., R_n)$ to denote the set of relations or fragment relations specified in the constraint or simplified form (test) C; and Mh, LBi and RBi to represent mobile host, local base station and remote base station, respectively.

• $\mathcal{A}^L$ provides an estimate of the amount of data accessed, which is related to the number and the size of the relations or fragment relations specified in a given

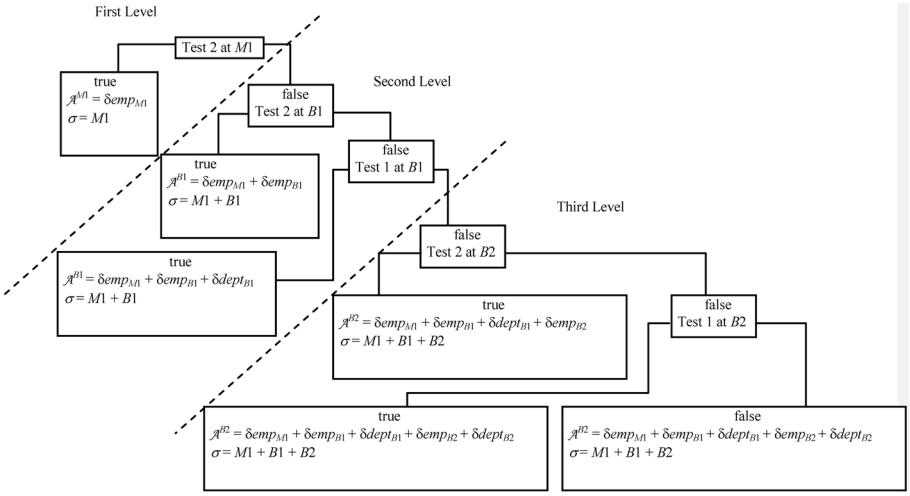

Figure 5. The Evaluation of I_1.

constraint or test, where L denotes Mh, LBi or RBi. This measurement indirectly indicates the size of the checking space. It is based on the following formula: $\mathcal{A}_{C(R1, R2, ..., Rn)} = \delta R_1 + \delta R_2 + ... + \delta R_n$ where the R_i's are the relations or fragment relations specified in C and δR_i is the size of R_i. To simplify δST^L and δCT^L, represent the amount of data accessed during the evaluation of the sufficient test and complete test at L. $[\mathcal{A}_{min}, \mathcal{A}_{max}]$ is a *range* where $\mathcal{A}_{min}$ ($\mathcal{A}_{max}$) is the minimum (maximum) amount of data that might be accessed.

• σ gives a rough measurement of the size of area that might be involved in validating the constraint.

Figure 5 presents the possible flows during the evaluation of I_1. Similar flows are observed for the other integrity constraints. Some conclusions can be made as follows:

• The first level which spans only the mobile host, i.e. $\sigma = Mh$, accessed small amount of data, $\mathcal{A}^{Mh} = \delta ST^{Mh}$, where $\delta ST^{Mh} < \delta R$. This is due to the characteristics of the sufficient test, which only accesses the data from the target relation (relation which appears in the specification of the update operation) and $\delta ST^{Mh} < \delta R$ since only part of the relation R is stored at the mobile host due to its limited capacity. Therefore, at this level it is important to have a high rate of success. This can be achieved by caching the relevant data items that are required by each of the test to the mobile host during relocation period.

• The second level spans the local base station of the mobile host. Since this level is embarked once the first level failed to validate the constraints, therefore, $\sigma = Mh + LBi$ (the operator + denotes that the size of area covered by the checking process include both the mobile host and the local base station). The amount of data accessed, $\mathcal{A}^{LBi} = [\mathcal{A}^{Mh} + \delta ST^{LBi}, \mathcal{A}^{Mh} + \delta ST^{LBi} + \delta CT^{LBi}]$, i.e. $\mathcal{A}^{Mh} + \delta ST^{LBi}$ if the sufficient test can verify if the initial constraint is not being violated. The worst case if when complete test needs to be evaluated and thus the amount of data accessed up to this level is $\mathcal{A}^{Mh} + \delta ST^{LBi} + \delta CT^{LBi}$. Therefore, at this level, it is important to have a high rate of success of the sufficient test or the complete test.

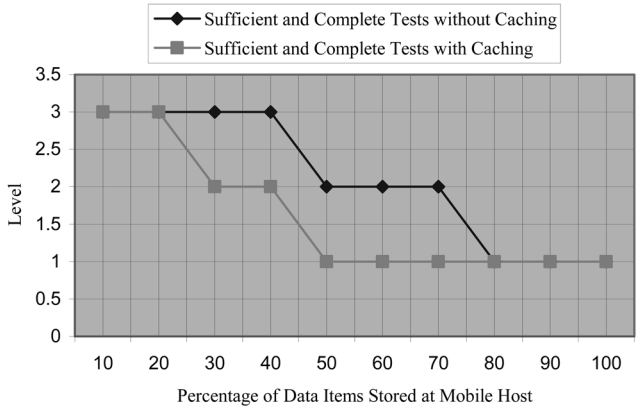

Figure 6. The Behavior of the 3-L Model.

• The third level spans the remote base station, which can involve more than one remote base station. Since this level is embarked once the second level failed to validate the constraints, therefore, $\sigma = Mh + LBi + RBi$. The amount of data accessed, $\mathcal{A}^{RBi} = [\mathcal{A}^{LBi} + \delta ST^{RBi}, \mathcal{A}^{LBi} + \delta ST^{RBi} + \delta CT^{RBi}]$, i.e. $\mathcal{A}^{LBi} + \delta ST^{RBi}$ if the sufficient test can verify if the initial constraint is not being violated. The worst case if when complete test needs to be evaluated and thus the amount of data accessed up to this level is $\mathcal{A}^{LBi} + \delta ST^{RBi} + \delta CT^{RBi}$. This level is similar to the brute force strategy, which spans the entire mobile system. Nevertheless, it is seldom the case that the entire mobile system is enforced to validate the constraints. As one can notice the scenario represented at this level is the same scenario as appeared in the distributed databases and the parameters presented above can be significantly reduced by applying the same optimization strategies as used in distributed databases [Ibrahim et al. 2001].

We have performed a simple experiment to analyze the behavior of the model when various percentages of amounts of data items are stored at the mobile host. There are 10 sets of data items altogether with minimum of 10% of the whole database and maximum of 100% that represents a complete database. A set of update operations is randomly generated and the same set is executed for each set of data items. This is to ensure that the same set of integrity tests are being evaluated. Figure 6 shows the results of this experiment. We have captured the behavior of the model when (i) sufficient and complete tests are utilized without any caching strategies and (ii) sufficient and complete tests are utilized with caching strategies with cache capacity 50% of the size of the database. From the figure as expected, if the whole database is stored at the mobile host (which is not possible in real application), checking the integrity constraints can be performed locally at the mobile host, i.e. at the first level (labeled as 1 in the graph). Also, whenever we increase the number of data items stored at the mobile host, the level of checking decreases. Furthermore as expected, utilizing the caching strategy can increase the number of local processing. Notice that with only 50% of the whole database stored at the mobile host we can achieve local processing. Finally, it is important to note that the figures might change when different set of update operations are executed but the pattern and the behavior of the

model are still the same.

6. CONCLUSION

This paper has presented the 3-L model, which is designed for checking database integrity in a mobile environment. The model has three levels and the process of checking the constraints embarks on at the first level. In this level, the information that is stored at the mobile host is accessed in order to check for the constraint violations. The second level which checks the validity of the constraints is performed at the base station of the mobile host by accessing the information stored at the station. The third level is invoked only if the second level fails to guarantee the constraint violations. The third level accesses the information stored at the remote base station(s). This model which adopts the simplified forms of integrity constraints, namely: sufficient and complete tests, together with the idea of caching the relevant data items during the relocation period for the purpose of checking the integrity constraints has reduced the amount of data accessed given that much of the tasks are performed at the mobile host. Eventually the checking mechanism of mobile databases is improved as delay during the process of checking the integrity constraints and performing the update is reduced. For future work, we plan to measure the performance of the 3-L model with respect to the time taken in checking the integrity of the mobile databases.

REFERENCES

ALWAN, A. A., H. IBRAHIM, AND N. I. UDZIR. 2007. Local integrity checking using local information in a distributed database. *Proceedings of the 1st Aalborg University IEEE Student Paper Contest 2007 (AISPC'07)*.

CHAN, D. AND J. F. RODDICK. 2003. Context-sensitive mobile database summarization. *Proceedings of the Twenty-Sixth Australian Computer Science Conference (ACSC 2003)*.

DZOLKHIFLI, Z., H. IBRAHIM, AND L. S. AFFENDEY. 2008a. Analyzing integrity tests for data caching in mobile databases. *Proceedings of the International Conference on Distributed Computing and Internet Technologies (ICDCIT 2008)*, Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, New Delhi (India), 157–165.

DZOLKHIFLI, Z., H. IBRAHIM, AND L. S. AFFENDEY. 2008b. Data caching strategies for checking integrity constraints of mobile database. *Proceedings of the 10th @WAS International Conference on Information Integration and Web-based Applications & Services (iiWAS2008)*, Association for Computing Machinery (ACM), 186–192.

EPFL, GRENOBLE, U., INRIA-NANCY, INT-EVRY, MONTPELLIER, U., PARIS, U. AND VERSAILLES, U. 2004. Mobile databases: a selection of open issues and research directions. *SIGMOD Record*, *33*, 2, 78–83.

GUPTA, A. 1994. Partial information based integrity constraint checking. *PhD Thesis*, Stanford University, USA.

HANANDEH, F. A. H. 2006. Integrity constraints maintenance for parallel databases. *PhD Thesis*, UPM, Malaysia.

IBRAHIM, H. 2006. Checking integrity constraints – how it differs in centralized, distributed and parallel databases. *Proceedings of the Second International Workshop on Logical Aspects and Applications of Integrity Constraints (LAAIC'06)*, 563–568.

IBRAHIM, H., W. A. GRAY, AND N. J. FIDDIAN. 2001. Optimizing fragment constraints – a performance evaluation. *International Journal of Intelligent Systems – Verification and Validation Issues in Databases, Knowledge-Based Systems, and Ontologies*, *16*, 3, 285–306.

KEN, C. K. L., L. WANG-CHIEN, AND M. SANJAY. 2006. Pervasive data access in wireless and mobile computing environments. *Journal of Wireless Communications and Mobile Computing*.

MADIRAJU, P. AND R. SUNDERRAMAN. 2004. A mobile agent approach for global database constraint checking. *Proceedings of the ACM Symposium on Applied Computing (SAC'04)*, 679–683.

MARTINENGHI, D. 2005. Advanced techniques for efficient data integrity checking. *PhD Thesis*, Roskilde University.

MAZUMDAR, S. AND P. K. CHRYSANTHIS. 2004. Localization of integrity constraints in mobile databases and specification in PRO-MOTION. *Proceedings of the Mobile Networks and Applications*, 481–490.

MCCARROLL, N. F. 1995. Semantic integrity enforcement in parallel database machines. *PhD Thesis*, University of Sheffield, UK.

NICOLAS, J. M. 1982. Logic for improving integrity checking in relational data bases. *Acta Informatica*, *18*, 3, 227–253.

Hamidah Ibrahim is currently an associate professor at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. She obtained her PhD in computer science from the University of Wales Cardiff, UK in 1998. Her current research interests include databases (distributed, parallel, mobile, bio-medical, XML) focusing on issues related to integrity constraints checking, cache strategies, integration, access control, transaction processing, and query processing and optimization; data management in grid and knowledge-based systems.

Zarina Dzolkhifli received the degree in Computer Science from Universiti Putra Malaysia. Currently, she is a master student of computer science in database majoring at Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. Her research interests include integrity constraints, constraints checking, data caching, and mobile database.

Lilly Suriani Affendey received her Bachelor of Computer Science Degree from University of Agriculture, Malaysia in 1991 and MSc in Computing from the University of Bradford, UK in 1994. In 2007, she received her PhD from Universiti Putra Malaysia. Her research interest is in Video Databases, Integration of Heterogeneous Databases, and Data Mining Application. She is currently a senior lecturer in Universiti Putra Malaysia.

Praveen Madiraju is an assistant professor of the Department of Mathematics, Statistics and Computer Science, Marquette University, Milwaukee WI 53201-1881 USA. His research interests include Databases: Global Constraint Checking and Querying in Multidatabases, XML Databases, Constraints in XML Databases; Distributed and Mobile Computing: Middlewares for Wireless Personal Communication Devices (iPAQ's), Distributed and Mobile Computing Systems, Mobile Agents; and Data Mining: Web Usage Mining, Web Logs.