Location Dependent Data and its Management in Mobile Databases

Margaret H. Dunham
Department of Computer Science and Engineering
Southern Methodist University
Dallas, Texas 75275-0122
mhd@seas.smu.edu

Vijay Kumar
Computer Science Telecommunications
University of Missouri-Kansas City
Kansas City, Missouri 64110
kumar@primus.cstp.umkc.edu

Abstract

Location Dependent Data is data whose value depends on its location. The objective of this paper is to introduce this topic and spawn further related research.

1 Introduction

In traditional ways of managing data, the relationship between the data and the geographical location of the organization it represents, is usually ignored. In wireless computing this property of "location transparency" is in fact often replaced by a "location dependency" property. Furthermore, the mode of issuing queries (the geographical location where the queries originate, the way they are issued, etc.) on such data determines the outcome. *Location Dependent Data (LDD)* refers to data whose values depend on location. The following examples illustrate these points.

Example 1 LDD - Hotel Information: *Suppose a traveler wants to find out information (location, room rent, etc.) about hotels in the middle of his journey. He issues a query to obtain this information. The answer to this query depends upon the geographical location. At one place, for example Dallas, the answer might be a Holiday Inn. and at another, for example Kansas City, the response might be a Best Western. It would even be possible for a traveler driving from Dallas to Kansas City to ask the same question en route but request the response using Kansas City data rather than local data.*

Example 2 LDD - TV Station Tuning: *While driving through Texas, Sarah wants to be able to ensure that her children can watch their favorite TV shows. She has a TV in the van and whenever the reception becomes poor, she wants to find out the local TV stations. So she periodically enters a query of the form "Find the local TV stations". The response will vary from locale to locale, but will list the TV station number and local affiliate.*

We observe the following from the above examples: (a) the semantics of a set of data and their values are tightly coupled with a particular location, (b) the answer to a query depends on the geographical location where the query originates, and (c) a query may be valid only at a particular location.

While the issue of location dependent queries has been proposed as an important issue related to mobile computing [2], little work has been done which examines the data itself and the impact on database processing issues. The contributions of this paper, thus can be summarized as follows:

- Introduction of the location dependent property of data and its formalization.
- Identifying the unique processing requirements and approaches for managing them efficiently.
- Use of mobile computing in processing such data.

2 Location Dependent Data

We use our earlier mobile platform model [2] in this paper. Base Stations (BSs) and Fixed Hosts (FHs) are connected through wired links and Mobile Units (MUs) are connected to BSs via a wireless network. We assume that a *BS* has the resources (disk space, CPU) needed to assist in data processing. We define the following and use them throughout this paper.

Definition 1 *The* **Geographic Domain**, *G, is the entire area covered by the Mobile Computing Platform. Thus a mobile unit can freely move around in G.*

Definition 2 *A* **Location** *is a precise point within the Geographic Domain. It represents the smallest identifiable position in the domain. It can be represented in terms of a latitude/longitude pair. Each location is identified by a specific id, L. Also,* $G = \cup L, \forall L$.

In a centralized database system it is usually assumed that there is only one copy of each data object. In a distributed environment there may be multiple copies, however there is only one "correct" value. In mobile computing, much research has examined how data may be cached at the MU [1]. It is normally assumed that the data cached at the MU is a replica of that in the fixed network and is usually considered to be a "secondary copy" much like replicas in a distributed environment. This framework is not adequate to handle location dependent data. Even with secondary replicas, it is assumed that there is only one value. With location dependent data we must be able to support multiple different values.

We introduce the concept of "data region" to explain our view of data. A data region is a logical

boundary within which a data object has only one correct value. In Example 2, the TV station number for NBC affiliate is different in Kansas City than in Dallas. The data values (station number) differ in each data region (Dallas and Kansas City). It is possible that the Dallas region or Kansas City region may be covered by a number of cells, in that case the data region for Dallas will have these cells all included within it. A data region is different from a cell of mobile computing.

Definition 3 *A* **Data Region**, *R, is a logical area within G, thus* $R \subseteq G$*, within which one correct value exists for a data object.*

With a conventional distributed database, each object has only one data region - the entire geographic domain.

2.1 Spatial and Temporal Replicas

In a distributed database we may have multiple replicas. We use the notation O_i to indicate a data object (e.g., a relation, a set of tuples, etc.) i where i is a unique integer value. If replication exists, we add an additional subscript to indicate the replica number: $O_{i,t}$. Here t is an integer value representing the replica identity. In a distributed environment, suppose that there existed four replicas for object O_1: $O_{1,1}, O_{1,2}, O_{1,3}, O_{1,4}$. At any point in time there is only one correct value for O_1.

Traditional databases are snapshots of temporal data. That is, they represent data values at one instant of time. Replicas may temporarily have different actual values, but it is the responsibility of some implemented replica control policy to maintain mutual consistency. Replicas may change their values over time under the same set of consistency constraints. We thus refer to these traditional replicas as "Temporal Replicas".

Definition 4
Temporal Replication *refers to copies of data objects all of which have only one consistent data value at any point in time. One of these copies is called a* **Temporal Replica**.

In a mobile computing database, however, multiple copies of a data object may have different correct values. Example 2 above returns different data in Dallas and Kansas City. The station number for "NBC" is not the same in each area. We use the term "Spatial Replica" to refer to these copies of the object.

Definition 5 Spatial Replication *refers to copies of data objects which may have different correct data values at any point in time. Each value is correct within a given region. One of these copies is called a* **Spatial Replica**.

Unlike temporal replicas, no attempt is made to ensure that different spatial replicas of an object have the same value. For this reason, spatial replicas cannot be processed using the *read-one write-all* version of distributed two-phase locking concurrency. We don't want to force all spatial replicas to be updated.

A superscript is added to show the spatial replica. Thus $O^s_{i,t}$ represents a replica of object O_i in data region s with replica number t. Each spatial replica has a unique superscript value. Figure 1 shows four data objects: O_1, O_2, O_3, and O_4. O_1, O_2 and O_4 do not have spatial replicas (Actually we could view that there is only one spatial replica for each, but by convention we omit the superscript of 1 in this case.) O_1 has three temporal replicas: $O_{1,1}$, $O_{1,2}$, and $O_{1,3}$, O_2 also has three temporal replicas: $O_{2,1}$, $O_{2,2}$, and $O_{2,3}$, and O_4 does not have any replicas. In this case we drop the second subscript. O_3 has both temporal and spatial replicas. We show three data regions for this object O^1_3, O^2_3, and O^3_3. The first spatial replica has two temporal replicas: $O^1_{3,1}$ and $O^1_{3,2}$. The third spatial replica also has two temporal replicas: $O^3_{3,1}$ and $O^3_{3,2}$. The second spatial replica has no temporal replicas: O^2_3. In this case we again drop the second subscript as there are no other copies of the spatial replica.

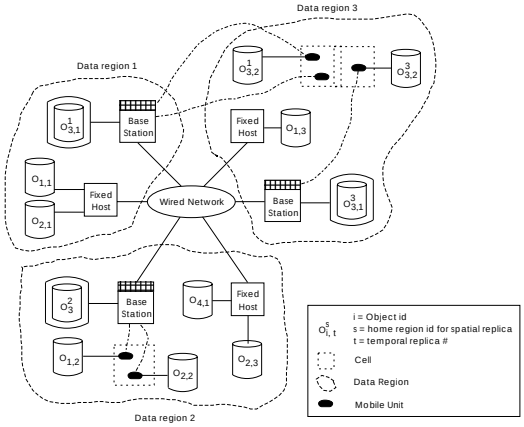

Figure 1: Different Replication Types

Notice that this notation indicates the data region associated with an object. Here $O^1_{3,2}$ is the second temporal replica of the spatial replica for object 3 in data region 1. Thus the schema of O_3 is replicated in regions 1, 2, and 3, but no attempt is made to keep these three spatial replicas of the same object consistent. Even if they contain the same data values, semantically they are different. In Figure 1 replica $O^1_{3,2}$ is stored at a mobile unit currently in data region 3. As the mobile unit moves around it may move into data region 2 or back to data region 1. However, it is always a copy of the object for data region 1 and should have the same value as the data from data region 1. Here data region 1 is the "home region" for the spatial replica $O^1_{3,1}$. As the mobile unit moves around, the actual data region where it resides changes, but the home region always remains the same. Every spatial replica has its home region and the spatial replica belongs to this region. A spatial replica of a home region may be "borrowed" by a foreign region for use after satisfying some criteria set by the home region. For example, the mobile unit moves into region 3, or there could even be a temporal replica of a spatial replica in a foreign region at a fixed host. Consider the following example:

Example 3 *Any two or more cities or countries may have a common geographical border. For managing*

activities at the border, two cities or countries may temporally replicate some spatial replica. For example, to control the traffic flow and smuggling at the USA/Mexico border, Mexico might replicate vehicle registration data from USA, and vice versa. Replica restrictions could indicate that these temporal replicas can not be updated in the foreign regions.

Temporal replicas of spatial replicas which exist in foreign data regions exist for the same reason as other temporal replicas: to improve performance. They may also exist because of the fact that boundaries between regions themselves may not always be clear cut. An MU close to the boundary between two data regions may actually prefer to receive data from a foreign region rather than its home region. For example, a traveler driving into a city currently on its outskirts might wish data for the city itself. Thus, even though we assume that the data regions precisely partition the geographic domain G, there are cases where data from foreign regions will coexist with the spatial replica for that region as well. For example, USA vehicle data (USA is the home region) overlaps in the Mexico region (foreign region of USA data). We assume that this temporal replica in another region will be subject to a set of update and/or access constraints as defined by the replica control policy in effect at the home region (USA in this case).

The spatial replica is tightly coupled with a specific region. To associate a spatial replica with the correct data region, we define a mapping function, which we refer to as "Data Location Mapping".

Definition 6 *Given a set of data objects $\mathcal{D}$ and a set of data regions $\mathcal{R}$, a* **Data Location Mapping** *DLM is a mapping $DLM : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{R})$ where $DLM(D)=\{R_1, R_2, ..., R_n\}$, $R_i \in \mathcal{R}$, $\bigcup_{i=1}^{n} R_i = G$, and $\forall i, j\ R_i \cap R_j = \emptyset$. (Here $P(R)$ is the power set of R.)*

Thus the set of data regions for a data object is a partitioning of the geographic domain. In the case where no spatial replication of a data object exists, then the number of data regions for that object is one. That is, for an object with no spatial replica $D(O) = G$.

Definition 7 *Given a set of data objects $\mathcal{D}$, a set of location ids $\mathcal{L}$, and a set of data regions $\mathcal{R}$, a* **Data Region Mapping** *DRM is a mapping $DRM : \mathcal{D} \times \mathcal{L} \rightarrow \mathcal{R}$ where $DRM(< D, L >) \in DLM(D)$.*

Notice that a data region is identified by the data object and the location. Thus as an MU moves, each location uniquely identifies for all data objects the data region to which each belongs. This gives us a technique to perform location dependent queries which return data values for the location from which they execute. Notice that this approach is consistent with that in a centralized or distributed environment where only one data region exists for all data objects. Notice that in Figure 1 one mobile unit is actually accessing a replica of O_3 from data region 1, $O^1_{3,2}$, even though it is physically located in the data regions 3 for this object. This could easily occur if the MU cached that replica while in data region 1 and is continuing to execute a query using this value. As we briefly discuss later, this points out issues associated with when and how a query is bound to a region. Even though we assume that based on a location there is a unique data region for each object, there are cases when temporal replicas of values from its home region are found within other foreign regions. The temporal replica, $O^1_{3,2}$, discussed above illustrates how this may occur with data cached at a MU. Our earlier example of overlapping USA vehicle data provides another scenario when this may occur.

2.2 Consistency in LDD

In a centralized or distributed environment there is only one correct value for each data object. The term *mutual consistency* is used to indicate that all the values converge to this same correct value [6]. A replicated database is said to be in a *mutually consistent state* if all copies have the exact same value [6]. In addition, a database is said to be in a *consistent state* if all integrity constraints identified for the database are followed [6].

Mobile computing complicates these concepts in much the same way as data replication. When performing location dependent queries, a consistent view is obtained if all data is seen with respect to one location. So, for example, if we get a list of hotels and restaurants it will be for one location. We won't get hotels for Dallas and restaurants for Kansas City. Consider issuing a query "List the names of all hotels". If this query is processed in a traditional distributed environment with temporal replicas, then it will list the names of all the hotels in the entire geographic domain. But if it is processed on spatial replicas, then it will list only the names of hotels situated in the corresponding data region. The consistency for spatial replication is localized to that region. Mutual consistency is guaranteed across temporal replicas of each spatial replica, but there is no such thing as mutual consistency across the spatial replicas themselves. We, therefore, define the concept of spatial consistency and refer to the traditional consistency as temporal consistency.

Definition 8 Temporal Consistency *indicates that all data object values must satisfy a given set of integrity constraints, independent of location. A database is in a* **Temporally Consistent State** *if all temporal replicas of an object have the same value.*

Definition 9 Spatial Consistency *indicates that all data object values (instances of a common schema) of a spatial replication are associated with one and only data region, and they satisfy consistency constraints as defined by the region. Thus there is 1:1 mapping between data value set and the region it serves.*

3 A Review of Previous Work

Our approach for dealing with LDD queries and transactions is quite different from that of the MOST [8] approach, which is the only other research project in this area of which we are aware. They introduce the concept of dynamic attributes whose value change continually as a function of time. In addition, a language, Functional Temporal Language (FTL), used in

the MOST model is proposed. Like our approach, they incorporate both spatial and temporal aspects into their location dependent processing. We don't require that specific location data be placed in the database (although this would be allowed with our approach). The database can be thought of as partitioned into locations. Each partition would have the correct data values for that location. For example, motel information in Dallas would not have motel information for Kansas City. Although the motel databases at each location could have the same schema, the instances stored would be different. (It is possible that there is an overlap between the data at multiple locations, however.) We would like to point out that our work does not conflict with that in [8]. Indeed location dependent data could be partitioned using the approach we introduce below and could as well (although it doesn't have to) follow the MOST approach.

4 Query Processing of LDD

In this section we explore some of the issues related to accessing location dependent data.

4.1 Possible Approaches

The examples introduced earlier illustrated the need to be able to provide location dependent answers to the same query based on location of the MU. We envision three approaches to correctly process these queries:

- Data Alone: Store data in such a way that when a query is executed it is routed to the correct data to be examined. Thus the query is not modified, but the data (somehow) is stored and accessed in such a way that different instances of the database will be used to give different results.

- Query Alone: Augment each query (transaction request) with location information. Thus when the data is accessed the correct answer will be generated. This may require extra location information to be added to the data, but the placement and access of data is handled as if the queries were location transparent. Only the query changes.

- Both Data and Query: This approach requires both changes to how the data is stored and how queries are executed. This is the approach that we propose.

Table 1 illustrates these options for Example 2. Both total data and data associated with a specific location are included. With the first option, when a query is issued in Dallas it would be directed to only the data associated with Dallas, Table 1b. Notice that with this approach, the size of the database to be examined would be reduced. We need the location for the global table to add to the affiliate attribute to make a key. Thus we see that use of the first option, is not a simple partitioning and that it reduces the size of the data to be examined.

With the second option, the table for the total data, Table 1a, would be examined. While this table could be replicated (and perhaps partitioned) at multiple sites for efficiency, the value of the data at all sites would be the same.

Sta	Loc	Affil
9	Dallas	ABC
3	Dallas	CBS
6	Dallas	FOX
10	Dallas	NBC
5	Austin	ABC
10	Austin	CBS
3	Austin	FOX
12	Austin	NBC

(a) Total Data

Sta	Affil
9	ABC
3	CBS
6	FOX
10	NBC

(b) Dallas Data

Sta	Affil
5	ABC
10	CBS
3	FOX
12	NBC

(c) Austin Data

Table 1: Tabular data of Example 2

We must be able to differentiate the type of queries. For example, consider the query "List the names of the hotels" issued by a customer from an MU. If the system processes this as a temporal replica query, then the answer will contain a list of all hotels in the geographic domain. Otherwise the list will contain names of the local hotels only. In conventional systems such queries may be regarded as a general query and for any specific query, the user would provide additional location information. Thus, to get the names of local hotels, the query would contain the name of the location. In mobile computing, however, the opposite is true. In our approach, therefore, every query is spatial by default and a temporal query is specifically identified.

The use of location dependent data has several desirable features. It would reduce the size of the tables to be examined, thus improving the efficiency of the queries. Scalability and maintainability would also be improved due to the fact that local data is stored and managed locally. This approach allows a global information service provider to implement multiple database sites similar to the local service providers. Users of global service providers could ask exactly the same queries from any site. The answers would differ because the LDD tables would be used to answer them.

Think of each user's query as identifying the spatial replica to be used. Instead of maintaining large databases, we envision that these fragments (or spatial replicas) will be located at different sites throughout the entire geographic domain. Each service provider will have these spatial replicas spread around to multiple sites with each site maintaining only the data appropriate for the MSCs (Mobile Services Switching Center) which will be connected to it. Thus when a request is sent to an MSC, it will be forwarded to the correct SCP (Service Control Point) for that MSC. The data at that SCP will be only for that MSC. This points out a problem if option 3 is not used. Namely, if the query is not changed at all, how can the correct spatial replica be identified? The current location of the MU could be used to determine the data region, however this may not be the value which the user wishes. Without option 3, a user in Dallas would not be able to query Kansas City data. Thus we must have both data and query modifications.

4.2 Binding

Since the MU is moving, the binding of the query to a location (and ultimately a data region and spatial replica for each object) has many possibilities. When and how to do this binding impacts the results of the

query. To process a query we must be able to determine the binding granularity, what binding location to use, whether to allow the binding to vary, and how to implement the binding. The binding granularity is ultimately a latitude longitude pair. This granularity might be too restrictive for some applications and a granularity higher (such as a cell associated with a BS might be more appropriate). The following options exist for choice of binding location: location of MU when query is first requested, location of MU when query is committed, location specifically identified in query, or projected location based on movement of MU. While there may be other choices these appear to be the most appropriate at this point. If indeed a location is identified in the query, then this will be the location. Future research must examine the other options to determine if there are other choices and which are most appropriate under what conditions.

4.3 Caching

Caching of data at MUs can improve performance and facilitate disconnected operation. Much research has been performed in the area of MU caching [1]. Caching issues are complicated by the use of LDD. Because of the fact that data which is cached can be viewed as a temporal replica of spatial data, as a MU moves into new data regions the cached data may become obsolete. This data is not stale because it is incorrect, but may not be desired because it is from a foreign region. Replacement policies need to be reexamined to include location information. For example, data from a foreign region should perhaps be replaced before data from the current home region even though the foreign data is more recently used. However, this is further complicated by the fact that ongoing or future queries could be bound to foreign regions. The MU mobility is such that the MU could very quickly move back into the home region for this data, making the replacement policy also subject to movement of the MU. All of these issues are beyond the scope of this paper, but certainly need to be studied.

5 Transaction Management

To facilitate the possibility that different portions of a transaction may be associated with different locations, we propose the use of execution fragments. A mobile transaction, T_i, is a database transaction requested from an MU. An execution fragment, e_ij is a part of a mobile transaction. Just as spatial replicas of a mobile transaction are associated with a data region, so too are execution fragments:

Definition 10 *Each execution fragment, e_ij, of a mobile transaction, T_i, is associated with a unique location. Given a set of execution fragments $\mathcal{E}$ we define a* **Fragment Location Mapping** *FLM is a mapping $FLM : \mathcal{E} \rightarrow \mathcal{L}$.*

The FLM identifies the location with respect to which each execution fragment is executed. This identifies the spatial replica to be used for each data object in that fragment. given the FLM which identifies the location for the fragment, the DRM mapping indicates the data region to be used for each data object. Thus we have a set of data regions associated with each fragment. In addition, it is used to ensure spatial consistency of fragments within a transaction.

5.1 Atomicity

The purpose of atomicity is to ensure the consistency of the data. However, in a mobile environment we have two types of consistency. Certainly atomicity at the execution fragment level is needed to ensure spatial consistency. However, transaction atomicity is not. We could have some fragments execute and others not.

Definition 11 *A mobile transaction, T_i, satisfies* **Spatial Atomicity** *iff each execution fragment, e_ij, of T_i is atomic. T_i is said to be* **Spatially Atomic** *iff each execution fragment, e_ij, is atomic.*

Theorem 1 *If all mobile transactions satisfy spatial consistency then spatial atomicity is required.*

Proof: (Proof by contrapositive) Suppose that a transaction does not satisfy spatial atomicity. Then there must be a fragment of this transaction which has only partially updated the database. As such, spatial consistency is then not necessarily maintained.

The converse of this theorem is not true. To be spatially consistent, a mobile transaction need not be atomic at the transaction level.

5.2 Isolation

We need to reevaluate transaction isolation when spatial consistency is present. As with consistency, isolation at the transaction level is too strict.
The important thing is to ensure that execution fragments satisfy isolation at the execution fragment level.

Definition 12 *A mobile transaction, T_i, satisfies* **Spatial Isolation** *iff each execution fragment, e_ij, of T_i is isolated from all execution fragments of T_i or any other transaction.*

Theorem 2 *If all mobile transactions satisfy spatial consistency then spatial isolation is required.*

Proof: Simple contrapositive proof is omitted.

5.3 Location Dependent Commit

To ensure spatial consistency, spatial isolation, and spatial atomicity, mobility forces that the commit also change. We introduce the concept of location dependent commit.

Definition 13 *An execution fragment, e_ij, satisfies a* **Location Dependent Commit** *iff the fragment operations terminate with a commit operation and a FLM exists. Thus all operations in e_ij operate on spatial replicas defined by a DLM on the location identified by the FLM. The commit is thus associated with a unique location, L. To indicate this we write $commit_L$.*

5.4 Transaction Definition

We now define a mobile transaction which uses the concept of location dependent data. Many different techniques for processing mobile transactions have been proposed [4], but this definition is independent of a particular technique for processing a query.

Definition 14 *An* **Execution Fragment** e_ij *is a partial order* $e_ij = \{\sigma_j, \leq_j\}$ *where*

- $\sigma_j = OS_j \cup \{N_j\}$ *where* $OS_j = \cup_k O_jk$, $O_jk \in \{read, write\}$, *and* $N_j \in \{abort_L, commit_L\}$. *Here these are a location dependent commit and abort.*
- *For any* O_jk *and* O_jl *where* $O_jk = R(x)$ *and* $O_jl = W(x)$ *for a data object* x, *then either* $O_jk \leq_j O_jl$ *or* $O_jl \leq_j O_jk$.
- $\forall O_jk \in OS_j, O_jk \leq_j N_j$

The only difference between an execution fragment and a transaction is that either a location dependent commit or abort is present instead of a traditional commit or abort. Every fragment is thus associated with a location. However, keep in mind that if the data object being updated has temporal replicas, then the fragment updates all replicas (based on some replica control policy). Thus it is not subjected to location constraints and appears as a regular transaction.

Definition 15 *A* **Mobile Transaction**, T_i, *is a triple* $< F_i, L_i, FLM_i >$ *where* $F_i = \{e_i1, ..., e_in\}$ *is a set of execution fragments,* $L_i = \{l_i1, ..., l_in\}$ *is a set of locations, and* $FLM_i = \{flm_i1, ..., flm_in\}$ *is a set of fragment location mappings where* $\forall j$, $flm_ij(e_ij) = l_ij$.

In traditional database systems, the transaction is assumed to be a unit of consistency. Even with spatial atomicity, this is still the case with a mobile transaction. A Mobile Transaction is a unit of consistency. That is, given a database state which is both temporally and spatially consistent, a mobile transaction T_i converts this state into another temporally and spatially consistent state.

6 Implementation of LDD

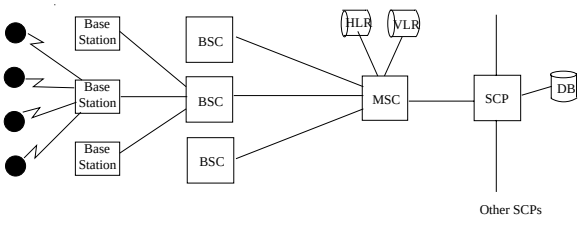

Figure 2: Implementation Architecture for LDD

Figure 2 shows our proposed approach to implement location dependent data. When a query is requested from an MU, it is routed to the MSC where the correct SCP to service this message is determined. Each MSC has a unique SCP for each service provider. The SCP has a database which contains the location dependent data. It has the temporal replicas for any data to be accessed from the cells in the MSC area as well as the spatial replica for those objects which have a home data region within the MSC area. All SCPs are connected in a network and appropriate distributed database concurrency control and recovery approaches are implemented across the SCPs. In the case that the amount of data is too large to be stored at the SCP, then corresponding database servers may exist elsewhere on the fixed network. In this case, the SCP routes the query to the correct location for processing. It is assumed that the MSC adds the location to the request. So that when the SCP receives this message, this default location binding has been determined. If the query contains another location, then this overrides the default location and the SCP forwards the query to that location for processing.

7 Conclusions and Future Research

In this paper we have introduced the concept of location dependent data and discussed some of the associated research issues. A further related discussion can be found elsewhere [5]. Specific topics requiring future research include location binding granularity and variability, query language changes needed to support LDD, transaction mangement issues including commit, and use of caching with LDD.

References

[1] Barbara, D., and Imielinski, T. Sleepers and Workaholics: Caching Strategies in Mobile Environments. *Proc. ACM SIGMOD Conf.*, Minneapolis, May, 1994.

[2] Dunham, M. H., and Helal, A., *Mobile Computing and Databases: Anything New?*, *SIGMOD Record*, Vol. 24, No. 4, pages 5–9, December 1995.

[3] Gray, J., and Reuter, A., *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann Publishers, 1993.

[4] Helal, S., Balakrishnan, S., Elmasri, R., and Dunham, M., Mobile Transaction Models, Purdue University, Department of Computer Sciences. Technical Report number 96-003, Jan 1996.

[5] Kumar V. and Dunham M. H., Defining Location Data Dependency, Transaction Mobility and Commitment, Technical Report 98-CSE-1, Southern Methodist University, February 1998.

[6] Ozsu, M. Tamer and Valduriez, Patrick, *Principles of Distributed Database Systems*, Prentice Hall, 1991.

[7] Samet, H. and Aref, W., Spatial Data Models and Query Processing, in *Modern Database Systems, The Object Model, Interoperability, and Beyond*, edited by Won Kim, Addison-Wesley Publishing, 1995.

[8] A. Prasad Sistla, O. Wolfson, S. Chamberlina, and S. Dao, "Modeling and Querying Moving Objects," *Proceedings of the IEEE International Conference on Data Engineering*, 1997, pp. 422-432.