Schema Integration for Mobile Databases

J. Tan
Software Technology Department, De La Salle University, Manila
2401 Taft Ave., Malate, Manila 1004, Philippines
Email: ccsjlt@dlsu.edu.ph

Abstract Mobile workstations are becoming popular in many areas of the industry to provide information virtually at people's fingertips. Thus we find notebooks and personal digital assistants (PDA) being equipped with networking or mobile communication capabilities to become extensions of the corporate computing facilities. But the limitations of mobile workstations make it difficult to treat mobile connectivity in the same way as how a typical network host is connected. These limitations force us to augment networked systems and applications. This paper surveys a number of schema integration methodologies and explores how they can or cannot be applied to mobile database environments. We present existing methods and some related works that have been designed for multidatabase or distributed database systems. Then we classify them based on a number of areas, and we discuss how they may be applied in a mobile environment. Although mobile database systems allow for mobile workstations acting as servers for shared databases, we limit our study, however, to scenarios where mobile workstations are clients only.

1 Introduction

The primary goal of a database management system (DBMS) is to provide correct information to users who require it in an efficient manner. With the advances in networking technology, database systems can span the entire earth, connecting several independent systems. Connectivity at the database level, however, is not as simple as at the hardware level. While the technology for connecting to remote sites is available, there is the critical requirement of effective and meaningful sharing of information. In many cases, information must be shared among several systems that are not identical in platform or data model, and several projects attempt and succeed in varying degrees of success to make this connectivity a reality. Physical integration of DBMSs, while possible, is too expensive [13, 20], despite the benefits of uniformity in all database aspects. It is therefore necessary that the systems who wish to share information with others be unmodified and retain their *local autonomy* [13, 20]. There are many approaches to large-scale information sharing. A tabular taxonomy of information sharing systems is given in [13]. *Distributed database systems* are the most tightly coupled, followed by *multidatabase systems*, with *interoperable systems* being the most loosely coupled.

Distributed database systems (DDBS) are typically designed top-down, and the entire system of local and global functions is implemented as a single unit. The global system has control over local data and processing to implement replication and fragmentation features [7]. A *global schema* [1, 4, 18, 6], an integrated schema constructed from the schemas of the component databases, must exist. There is little or no local autonomy, although synchronization for serializability and optimization for global requirements becomes possible. The component systems are usually of the same data model.

Multidatabase systems (MDS) are designed bottom-up, as it integrates preexisting DBMSs that retain local autonomy and do not undergo costly modifications to join the multidatabase system. *Global schema multidatabase systems* are designed by constructing a global schema, treating the multidatabase as a very large database. However, it is often difficult to construct a global schema due to the number of schemas to integrate and the differences among them that are difficult to resolve [7, 8, 19]. *Federated databases* allow each DBMS to autonomously choose what information and services they are willing to share through *export schemas*. Local schemas are not integrated into a global schema, but remote schemas are integrated to form *import schemas*, representing remote information that the local applications will make use of. *Multidatabase language systems* use query language tools to integrate information from different databases, and each DBMS need not maintain a partial global schema. *Homogeneous multidatabase language systems* are variants that support only databases of the same data model. *Mobile databases* are multidatabase systems in which databases may reside in mobile computers and/or in stationary or non-mobile computers [17].

Interoperable systems, the most loosely coupled systems, have no support for full database functions, such as query processing. Global function is limited to simple message passing, and local interface is through an application above the local DBMS user interface.

Multidatabase systems have several advantages over the other systems. A key feature is the retention of local autonomy, which distributed database systems do not offer, and full database functionality, which interoperable databases lack. The major problems in designing multidatabase systems stem from two factors: the system must manipulate data from preexisting database sources that must not suffer modification overhead and they are located in heterogeneous sites of diverse hardware and software distributed across a possibly large computer network [4]. Current research issues on multidatabase interoperability can be grouped into four basic areas: *schema integration and semantic heterogeneity*, *transaction management*, *query optimization*, and *object-oriented multidatabases*. In this paper, we focus on schema integration, specifically for mobile database systems.

2 Schema Integration

Conventional database theory teaches us how database schemas, regardless of data model, are used to represent the logical structure of databases. Any entity in the database, and its relationship with other entities, should be represented in the schema. Combining entities means that the data manipulation facilities are able to reconcile their relationships based on the database schema. In this manner, they can be integrated properly. Attempting to do the same thing with several, autonomous databases can be difficult because each system's data manipulation facilities, be they language-based, view-based or in whatever form, are familiar only with the local database schema. The only way to go about this difficulty, therefore, is to be able to resolve the differences between the schemas of the involved databases. This is the basic goal of schema integration. *Schema integration* is the activity of integrating the schemas of existing or proposed databases [1, 25].

The integration of preexisting DBMSs faces several *domain mismatch* problems [4, 5, 24]. These problems include *naming, data type, data structure* or *data scaling conflicts*. There may also be cases of *missing data*, in which an object is represented in multiple sites but not with the same number of attributes. *Schematic discrepancies* are also possible, in which data in one database is equivalent to metadata in another database [4, 16, 23]. The first four conflict problems can be dealt with using data renaming, logical data restructuring, and scale or unit conversions. The missing data problem can be resolved by considering the objects semantically related, if there are common and identifying attributes. Another means to resolve missing data is with the use of *virtual attributes* and *domain mappings* [8]. Schematic discrepancies may arise as *data value attribute conflicts*, in which the data value of one database corresponds to an attribute of another, *attribute entity conflict*, in which an attribute in one database corresponds to a relation in another, or *data value entity conflict*, in which a data value in one database corresponds to a relation in another database [24]. The literature surveyed includes two solutions to schematic discrepancy problems. The first is by using extended operators based on aggregation and generalization [23], and the second is by using higher order expressions and views [16]. Both are general solutions, although they differ in that the latter is applicable to more loosely-coupled systems [23]. The authors of the former also argues that their solutions are simpler, although they have yet to consider database update features.

Whereas most structural differences in terms of schematic conflicts have been studied extensively, there is little work done on semantic issues [24]. Semantic similarities between objects must be studied to better understand how to integrate dissimilar objects properly. Obviously, schematic differences present difficulties. But it is even more difficult to resolve semantic issues, as the semantics behind database design are seldom represented in database design. A general policy of documenting semantics of databases and other provisions may assist in minimizing the damage that *domain evolution* causes [27]. The representation of semantics must be sufficient to describe domain values as well as the semantics of the domain. Metadata generation should be possible. Data/metadata synchronization is necessary so that specified values may be mapped to the matching semantic information. Metadata comparison should be possible, so that metadata operations can be defined and implemented. Relevance evaluation of semantic differences should be possible, so that not all semantic differences will call for costly resolution.

The problems mentioned can be classified under two basic areas: a structural or syntactic problems, which present difficulties in resolving structural differences between different representations of the same real-world entity, or semantic problems, which are more complex as the semantics and real-world relationships between entities are frequently not represented at all.

There are two means of classifying schema integration methods: according to *what* composite is produced and according to *how* the composite is defined. In the *global schema approach*, a global schema is constructed as the integration of the component databases, as it yields a global schema multidatabase system [6, 13, 14]. The *federated schemata approach* [26], which yields federated systems, requires portions of various local schema integrated to form several federated schemata based on the needs of applications. Lastly, the *multidatabase language approach* produces multidatabase language systems, and requires no composite schema. Whenever the queries cross schema boundaries, translation and transformation functions are employed. In the second scheme, based on how the composite is produced [25], the approach is either the *assertional approach*, which uses assertions to define the relationship between objects, or the *superview approach*, which integrates objects using ad-hoc view operators. The former is richer in terms of semantic expressiveness, but can be more complex.

2.1 General Outline of Activities in Schema Integration

A general schema integration methodology will have two phases: *analysis* and *integration* [2]. This can be expanded into a mix of four phases given below, although existing methodologies do not always go through each of them nor do they always follow a standard order nor standard terminology. Note that these steps may be applied even if the composite is temporary as happens in multidatabase language systems.

1. *Pre-integration* [25], in which local schemas are translated into a common data model and made to adhere to a common set of modeling guidelines. This will make it easier to analyze and integrate the local schemas. It can also be referred to as *schema translation* [2] or *syntactic transformation* [15]. It is defined differently in [1], referring instead to the analysis of schemas and definition of the integration policy.

2. *Schema analysis* [25], in which the local schemas are analyzed to discover relationships between data objects. [1] divides this phase between a different definition of the pre-integration phase and a *schema comparison* phase. In [2] schema analysis is distributed in different phases. Some portions such as method selection and conflict discovery are considered part of the pre-integration phase, while others are performed earlier.

3. *Integration* [25, 2], in which actual integration is carried out. This can be divided into *schema conforming* to resolve conflicts [1], and *merging*, the process of combining objects which may involve several intermediate steps producing several intermediate schemas, culminating in the final integrated schema.

4. *Restructuring* [25, 1, 2] which involves optimization of the intermediate results of integration, if necessary. [1, 2] define restructuring as a component of integration, and . [25] mentions that restructuring is intermingled with the integration phase iteratively as the progress of the integration phase progresses.

An earlier study of schema integration methods classifies them according to several categories [1]. One is in the integration strategy used. There are two general strategies: *binary* and *n-ary*. They are further divided according to *balance*: binary *ladder* or binary *balanced*, or *n*-ary *one-shot* or *n*-ary *iterative*. This study compared them based on a set of general steps that schema integration will undertake to achieve its goals. It also categorized them according to the class of conflicts they are capable of resolving, although these were limited to only two conflict types: naming conflicts and structural conflicts.

The existing schema integration methods surveyed follow variants of the given phases above. They vary in many details such as the basic approach used (global schema, federated, multidatabase language), choice of canonical data model (one of the known data models, a variant, or perhaps none), degree of automation, terminologies and other policy details and mechanism details. We withhold any judgments on whether any of these steps strictly require human interaction or not. At present, fully automated methods are very difficult to design due to the largely analytical nature of schema integration resulting from the need to study differences and relationships among real world entities modeled in the database.

2.2 Current Research

Amoco Distributed Database System (ADDS). ADDS (Amoco Distributed Database System) is a tightly-coupled federated system that supports multiple federated schemata [26] but is also classified as a global schema multidatabase system in [14]. This confusion may be the result of the flexibility in ADDS. Several schemata are integrated using composite database or CDB definitions to produce a composite database schema, which uses the extended relational model. Either a singular composite schema can be defined to integrate all the schemata or several may be defined, stored in the fully replicated ADDS data dictionary. In the case of a single composite schema, several views may also be defined from it according to the needs of ADDS applications. CDB definitions, which are akin to view definitions except on a multidatabase level, are usually defined for each application. A uniform data definition language is provided for uniform and transparent distributed database access. ADDS can integrate relational, network and hierarchical databases. An Application Program Interface or API is one of the standard user interface facilities available. Another is an interactive interface using a query language. The API is a set of callable procedures that users or programs may invoke to submit queries. Frequently used queries are also stored in the query catalog, from which they may be selected, modified and invoked by users.

The Bellcore Schema Design and Integration Toolkit (BERDI). [25] presents BERDI, an X11-based software toolkit developed at Bellcore to assist database designers in schema integration. It uses *assertions* to relate objects with one another in both the attribute and entity levels. The authors argue that most existing analysis tools and techniques may not be applicable for large, industrial schema integration. Two possible approaches are identified: the *assertional approach* and the *superview approach*. The former uses *assertions* to relate objects, while the latter uses view functions to use views for integrating existing objects. BERDI uses a methodology consisting of four phases:

1. *Pre-integration*, which involves deriving an intermediate schema for each local schema, conforming to a uniform data model and set of modeling guidelines.

2. *Schema Analysis*, the discovery and documentation of relationships among objects, or correspondences among entities.

3. *Integration*, using the results of analysis to create new objects by merging the source objects. This results in an *integration schema*.

4. *Restructuring, analysis and documentation*, which is simultaneous with the integration process, which involves monitoring the progress and smoothing over conflicts and other details of the integration.

Since there is no claim otherwise, we postulate the possibility of using BERDI for constructing either several federated schemata or perhaps a single global schema, generally applicable to any multidatabase or distributed system.

METU Interoperable Database System (MIND). The METU Interoperable Database System or MIND [9] supports heterogeneous, federated DBMSs, based on the OMG distributed object management model. A common data model and a single global query language OQL, based on SQL, are some of the properties of MIND. A CORBA compliant ORB implementation, DEC ObjectBroker, is the system that runs on the heterogeneous sites, on top of which MIND operates. CORBA stands for Common Object Request Broker Architecture, which is an architecture that provides independent access to objects on heterogeneous systems. MIND is composed of a collection of local database agents (LDA) and global database agents (GDA). Each LDA maintains export schemas represented using the common data model, and translates queries from the OQL to the local query language and back. Each GDA contains a global integrated schema, a global query processor and a global transaction manager that ensures serializability without compromising the autonomy of local databases.

MIND implements schema integration with a four-level architecture composed of the *local schema*, the *export schema*, the *derived* or *federated schema* and the *external schema*. The derived schema is a combination of the independent export schemas, and contains information on data distribution. The external schema specifies a subset of the information in the federated schema that is not stored in the local databases. Schema integration is a two-step process involving *investigation*, where commonalities and differences should first be manually investigated and identified, and *integration* itself, in which the DBA will solve export schema conflicts and generate integrated schema as a view over export schemas using an object definition language (ODL).

Multics Relational Data Store Multidatabase (MRDSM). MRDSM, which stands for the Multics Relational Data Store Multidatabase [19], uses the relational data model as the common data model. It provides data manipulation and data definition through MDSL (Multidatabase Data Sublanguage) and the MRDS (Multics Relational Data Store) definition language, respectively. MRDS is a popular relational database system, which can be integrated easily into the MRDSM system. DDL facilities allow the definition of export schemas and user access rights, whereas the DML facilities permit access and integration of data from different databases. Integration is therefore in two levels: at the DDL level available to the DBA defining export schemas, and at the DML level available for all users. Dependencies may also be defined on the multidatabase level: *manipulation dependencies*, *privacy dependencies* and *equivalence dependencies*. Several concepts including *multiple identifiers* and *semantic variables* are introduced as well. Multiple identifiers refer to names shared by different attributes, relations or databases in different sites. Semantic variables allow users to identify several relations with semantically related information with one name in user queries. MRDSM also allows the definition of *dynamic attributes*, which are functions over actual attributes that users may access.

The Remote-Exchange System. [10] discusses how to resolve *semantic heterogeneity* in the Remote-Exchange system. Heterogeneity may exist in several levels including the *meta-data language level*, where different data models may be used; the *meta-data specification level*, where semantic differences may exist even if the same data model is used; the *object comparability level*, where there can be dissimilarity at the object or entity level; the *data form or format level*, where semantically identical objects may be represented with different data types; and the *tool level*, where the tools used to access data may be different. The Remote-Exchange approach focuses on the meta-data specification, object comparability, and data form/format levels of heterogeneity. By providing the means of incorporating remote objects into a local database, the local database system will have the tools necessary to access these objects as if they were local objects. First, *discovery* of the remote object must occur, followed by *integration* of the remote object into the local database. Both steps will require the ability to recognize *object equivalence*. Object equivalence can be automated, to a degree, using either of *structural equivalence* or *behavioral equivalence*, which is based on the operations possible for each object. A *correspondence function* must be designed to establish the equivalence of objects.

An object-based common data model, the *Kernel Object Data Model* or KODM, is used so that heterogeneous objects may be shared. It provides *object uniformity, encapsulation, extensibility* and *remote database transparency*. A *sharing advisor* assists in discovering, establishing and refining *sharing patterns* for remote objects. The approach taken may use *structural and behavioral equivalence*, *constraint analysis*, or *heuristics*. A *semantic dictionary* is a component database that holds information about exported meta-data for sharable objects. The sharing advisor will come out with a *sharing plan* so that the remote object can be integrated as gracefully as possible into the local schema. A copy of the remote object may be ported locally, or a *handle* can exist for the remote object. The latter provides what appears to be direct access to the object which actually resides in its original, remote location. The methods that access the remote object may be local or the remote method that is originally encapsulated within that remote object maybe invoked. The fact that shared objects are encapsulated complies with the need for local autonomy of each component database, as well as limiting what can be done with each shared object..

Superview. The *superview* approach produces a *virtual database* which is generated as a pair $< schema, mapping >$ to provide mapping between the virtual (composite) database and its components [20]. There are two general tools defined for the superview approach: the *virtual database generator* or the *data definition language processor*, and the *virtual query processor*. Instead of the relational model for the integrated schema, a functional model, which makes much use of aggregation and generalization relationships, is used instead. Each database is treated as a set of classes, and either of two relationships are defined for these classes: *Att* or *Gen*. The former relates one class as being an attribute of the other, whereas the latter relates one class as being a generalization of the other. Each relation is also defined as a class, but not vice versa. Each class has a key, and, depending on the choice of the key, the semantic relationship between classes can be derived. Keys can therefore determine whether some relations may be joined or not. This technique is also the means by which semantic inconsistencies are avoided. Two logical schemas to be integrated are input, and queries, to produce the virtual database or the superview [1].

First, one of the two component schemas is restructured in a way as to make it appear similar to the other schema. This is done using extended language operators that are defined for superview construction. Next, the two schemas are linked at the primitive level, combining equivalent classes, if they model the same real world concept. Finally, operators based on generalization are used to connect attributes. A major deficiency of the superview approach is that it allows combinations only for classes which are entirely compatible.

2.3 Related Works

Domain Mapping and Virtual Attributes. [8] introduced a formal system that uses a relational data model and allows the derivation of an attribute value for mismatched domains. The mechanisms used are two: *domain mapping* and *virtual attributes*. Actual attributes may be logically integrated to form a virtual attribute, although other information in the database may also be used to define one. Domain mappings define how the domain of one attribute, actual or virtual, may map onto the domain of another. These features are provided by extending the relational algebra, introducing new constructs and concepts such as *definite* or *fully determined tuples* and *indefinite tuples*, *total* and *partial relations*, and *true* and *maybe* results. These constructs assist in making the integration of objects smoother.

Database Integration of Monomodel and Monolingual Systems. [12] discusses general solutions to integrate *monomodel* and *monolingual* DBMSs, pertaining to DBMSs supporting only one data model and data language. Three software approaches can be identified: *data-model transformation*, using a one-shot *database conversion* or a real-time *data transformer*; *data language translation*, in which queries crossing site boundaries are translated instead of the data; and *cross-model accessing*, which combines the two approaches so that a user can issue queries involving a remote database using his/her native data model and query language. There are four mapping schemes possible: (1) *Single-model-and-language to single-model-and-language* is a one-to-one mapping that lacks a high degree of heterogeneity; (2) *Single-model-and-language to multiple-model-and-*

language, a one-to-many mapping, is not flexible because of a strong belief in an all-powerful, best-choice data model and language, and users must become familiar with them; (3) *Multiple-model-and-language to single-model-and-language* mapping which involves a many-to-one mapping, which seems to be the most feasible; and (4) *Multiple-model-and-language to multiple-model-and-language* mapping involves a many-to-many mapping, which can be very costly in terms of overhead. Translation is two phase: source to *intermediate* model and language, then intermediate to target model and language. Hardware approaches may also be taken, but such solutions are not applicable to multidatabase systems. There can be no compromise to modify the existing DBMSs by modifying the hardware configurations as this goes against local autonomy.

The best approach seems to be the multiple-ML's-to-single-ML mapping, and this is the popular approach that is being taken. A kernel data model and language is used so that access to remote data is transparent and user interface need not change as the native data model and language are retained.

Integration Using Higher Order Expressions and Views. The occurrence of schematic discrepancies in relational databases requires new features to reconcile them [16]. *Higher order expressions* where variable can range over data and metadata, so that a unified view over original databases can be defined. *Higher order view definitions* that allow the number of relations or attributes to vary according to the state of the database.

A two-way mapping provides a unified view of the group of databases with similar information, preserves the "style" or format of each database and in a form that is familiar to the users. However, there should be an unambiguous definition of what defines "similar" databases and what "style" consists of. Accordingly, the view definition capability should be expressive and rich so that the transformations between databases can be specified clearly and transparently. Queries must be able to pertain to multiple databases as if they were one database, or to act upon them each in the same manner. The database must also remain familiar to the user, as if the original "style" still holds. These are referred to as *database transparency* and *integration transparency*.

A universe of databases is recursively defined as a nested relational model based on three object categories *set, tuple* and *atom*. Objects are value-based and have no notion of object identity. Tuples in a relation can have varying arity, so the sets (relations) contain heterogeneous objects. A more powerful data manipulation language, IDL or Interoperable Database Language, is proposed that provides higher order capabilities. Each database is modeled as a complex object wherein the metadata and data are explicitly represented. There is a need for the ability to specify higher order query expressions for higher order quantification (e.g. operations over relations/attribute names). Furthermore, higher order views must also be possible, so that views can be specified over a variable number of relations each with unique names.

Semantic Proximity – Measuring Semantic Similarity. *Semantic proximity* [24] can be used as a metric to characterize semantic similarities between objects to provide a classification of such similarities. Defining a model for semantic classification, semantic proximity between two objects O_1 and O_2 is given as

$$semPro(O_1, O_2) = < Context, Abstraction, (D_1, D_2), (S_1, S_2) >$$

where D_i is the domain of O_i and S_i is the state of O_i.

Context refers to the context in which the semantic similarity holds. Each object has a context, and two objects may be semantically closer to one another than to another object of the same context. *Abstraction* refers to the mechanism for mapping the domains of objects to each other or to that of a common third object. The *domain* of each object may denote the type of the object, and is not to be confused with its context. The *states* of the objects, also considered their *extensions*, is not the same as the actual, real world semantic state of the entity.

If two objects represent the same real world entity or concept, then they are said to be *semantically equivalent*. It should be possible for the abstraction to be defined as a total 1-1 value mapping. Otherwise, they may simply be said to be *semantically related*, when the abstraction is a partial many-one value mapping, or a generalization, or aggregation abstraction. If two objects are semantically related in the same context using *some* abstraction, meaning that this proximity is context-dependent, then they are *semantically relevant*. The weakest classification of semantic proximity is known as *semantic resemblance*, when two objects cannot be related by any abstraction in any context. They should, however, have the same role in their respective contexts. An object's *role* refers to the relationship between the object and the semantic context to which it belongs. If two objects cannot be related in any context nor any abstraction, nor do they have the same role in their respective contexts, then it can be concluded that they are *semantically incompatible*.

There is a certain amount of uncertainty, however, that comes with integrating data if semantic proximity is not near. This uncertainty can be modeled using fuzzy logic, "maybe" attributes, and other approaches.

3 A Tabular Summary of the Surveyed Methodologies

A tabular summary of the surveyed works is given below. Most of the complete methodologies are implemented in working prototypes. A method is considered object-oriented if it uses object-oriented concepts such as objects, object uniformity, object equivalence, or object classes. A schema integration method is considered *fully automated* if it can achieve its goals given the proper parameters without human interaction. It is only *partially automated* if it requires interaction with a human. A method is considered *manual* if it is employed without any support from a software component except for specifications (e.g. GUI). The symbols **B?** and *n***?** signify that there was no claim about the balance of the integration.

LEGEND:

Integration approach by composite output: *what*

GLOB : global schema approach   **FED** : schemata approach   **MDBLS** : multidatabase language approach
N : no claims

Integration approach by composite mechanism: *how* (given in parentheses)

SUP : superview approach   **ASSERT** : assertional approach

Design Characteristics:

A : Agent-based method   **C** : A common data model is specified   **L** : MDB query language exists
O : Object-oriented approach
F : Fully automated integration   **P** : Partially automated integration   **M** : Manual integration
Bb : Binary balanced   **Bl** : Binary ladder   **B?** : Binary but unknown balance
n1 : n-ary one-shot   **ni** : n-ary iterative   **n?** : n-ary but unknown iteration times

Table 1: Complete Schema Integration Methodologies.

Name of Methodology	Integration Approach	Characteristics	Integration Strategy
ADDS [5]	FED/GLOB(SUP)	CLMB?	ADDS uses CDB definitions to construct a composite schema (or several). CDB definitions are provided for specific views needed by applications, defined using the DDL. Logical data items are defined to resolve conflicts.
BERDI [25]	N(ASSERT)	Pni	1. Preintegration by converting local schemas to a form complying with the uniform data model and modeling guidelines. 2. Schema analysis to discover correspondences among objects. 3. Integration to produce an integration schema. 4. Restructuring, analysis and documentation.
MIND [9]	GLOB(SUP)	LOCMA	1. Manual investigation. 2. Manual integration using ODL to make global schema.
MRDSM [19]	MDBLS(SUP)	LCM	Integration is manual since it is embedded in DML queries or in DDL commands. MRDSM is a complete prototype but is limited to relational databases.
Remote-Exchange [10]	FED(ASSERT)	COPni	1. All local schemas must be converted to KODM. 2. *discovery* of what information should be shared 3. *integrate* remote objects • Consult the *sharing advisor* to get a *sharing pattern*. • Resolve conflicts using the *semantic dictionary*. • Create class in the local schema for the class to import or consider it a superclass of a preexisting equivalent class, if any.
Superview [20]	GLOB(SUP)	CPOB?	1. Merge two local schemas combining primitive classes. 2. Restructure superview when applicable, nesting if needed.

4 Schema Integration Applied to Mobile Database Systems

The usefulness mobile database systems is emphasized by the emergence of personal digital assistants (PDAs) and other mobile computers [3, 21]. Unfortunately, such environments are characterized by frequent loss and change of connections as the user moves from one place to another as well as restrictions in bandwidth, power and resources. In attempting to apply database theories and applications to a mobile environment, we explore new issues and attempt to resolve them.

There are special conditions to systems associated with mobile hosts [21]:

1. The search time should be short.
2. The bandwidth should be optimized.
3. Power consumption should be minimized.
4. Fast response time (i.e. first set of output values should be produced quickly) is necessary.
5. The precision of data must be controlled to minimize computation and communication expenses.

Schema integration for mobile databases can also be complicated by the necessity of *disconnected operations* [11, 28]. Disconnected operation is anticipated due to two factors: unintentional connection failure due to battery power failures or wireless connection failures can occur, and disconnection may be required to conserve battery life.

Transaction management issues are dealt with using protocols which can guarantee serializability and error recovery despite disconnected operations within transaction lifetimes. Database theories applied to mobile databases will have to be adapt to the above circumstances. A number of approaches are being explored at making information sharing in mobile environments effective i.e. [22, 28, 11]. A system based on agents has been proposed in [22] to provide access to the heterogeneous databases in the system. *Application agents* are submitted to the host that maintains the needed database to interact with the host's *database agents* to access the database. *Database agents* coordinate the access for the use of the application agents submitted. They also maintain the consistency of each database and handles error recovery in case of failure. Such a system differs from multidatabase systems in many aspects: agent coordination is decentralized; agent decomposition is adaptive; and unlike multidatabase systems, interface is through primitive methods instead of through a series of read and write operations. Each agent executes in the context of its local data, and each must be robust and recoverable. This pertains not only to the data but also to the computation state of the object itself. The nature of mobile database systems makes the agent model much suited to handle the interruptions, delays, disconnected operations and recovery measures whenever necessary. In such an environment, coordination of transactions becomes largely controlled by applications rather than users [22].

We examine each of the surveyed schema integration methodologies with respect to their applicability in a mobile environment in the following discussions. We may the universe of concern as a set of databases which may or may not be relational databases, and mobile workstations may request access. Mobile hosts are used only as clients, from which transactions originate. Thus our environment consists of mobile clients and non-mobile database servers. When a query from a mobile host is submitted for processing, mappings must be resolved, the query must be decomposed [2] into subqueries addressing respective remote databases, the subqueries are submitted to the DBMSs managing these databases, and intermediate results are gathered and combined to extract the query output. We define global transactions as transactions which manipulate data from databases residing in a site other than the source of the transaction. Otherwise the transaction is only a local transaction, and does not require any schema integration. The following subsections will focus on how each of the surveyed methods can or cannot be applied to our mobile multidatabase environment.

Amoco Distributed Database System (ADDS). ADDS provides a uniform query language and multiple composite schemata or multiple views based on a single global schema. Access to information in remote databases is through API calls which invoke procedures for application programs that provide transparent access. Another way is through an interactive interface for submitting queries through the ADDS query language or SQL. API routines as well as the ADDS data dictionary are fully replicated for query processing.

In operation, since the ADDS data dictionary is fully replicated, there is no need to perform real-time schema integration. However a mobile client will be limited to the CDB or view definitions available in the data dictionary. In the case of APIs being used, the queries are predetermined in the application. In the case of interactive queries, the user is expected to formulate the query on a logically perceived single unit – whichever CDB the query will refer to. But all this hinges on whether the mobile host has a replica of the data dictionary and the APIs or the ADDS interface components. Otherwise, some time will be expended to download them to the mobile host.

A mobile client can invoke integration to produce new CDB definitions, but there is no component for automatic conflict resolution in ADDS. This task must be performed by a valid integrator, which pertains to a DBA who is familiar with the databases to be integrated. It is therefore not feasible for schema integration to be carried out with a mobile workstation.

The Bellcore Schema Design and Integration Toolkit (BERDI). BERDI is a graphical toolkit for schema integration that involves a great deal of user (integrator or DBA) interface while schema integration is being performed. A system that has been integrated using BERDI will be supplied with a global schema that is produced through assertions on the relationships between objects. Global schema information are kept in the BERDI dictionary. Integrity constraints are also recorded to assure consistency of the integrated schema.

BERDI has little bearing on how a multidatabase will use the integrated schema generated. What may be unique with BERDI is the fact that the integrated schema is not a view but is defined by assertions that are stored in the BERDI dictionary. These assertions will be required to trace the mappings of objects in the integrated schema into actual objects (e.g. new objects created to merge equivalent objects). In operation, if the global schema were replicated in the mobile host, then the overhead for query processing will depend on how quickly the mappings can be resolved from the global schema specification. Without mathematical analysis or simulation, we cannot say for certain if this will cost more time and energy than mappings defined using view operators.

If BERDI must be used to integrate schemas using a mobile client, then the mobile client will not be used efficiently, as BERDI takes a lot of time and interaction with an integrator who must do a lot of analysis especially in cases of conflicts.

METU Interoperable Database System (MIND). MIND is based on the OMG distributed object management model, where two components, the LDA and the GDA, reside in each MIND client. The LDA has access to export schemas and can translate queries received in OQL into the local language and back. The GDA has access to the global integrated schema (or several federated schemata), a global query processor that also provides query optimization, and a global transaction manager ensuring serializability.

Given the responsibilities of the GDA, the mobile client will expend a considerable amount of time and energy in query decomposition, optimization and transaction management. The mobile client may be more efficiently used if GDA responsibilities for query processing and transaction management are delegated to some land-based server. The global transaction manager is designed to be resilient to site failures, which we can consider frequent (whether intentional or not) in a mobile host session.

If the mobile client must perform integration of remote information into the local federated schema, the case is the same with BERDI, in which a lot of time will be spent with the integrator analyzing manually and perhaps doing integration by trial and error. As with other methods, the semantics of the schema to be integrated should be clear to the integrator.

Multics Relational Data Store Multidatabase (MRDSM). MRDSM is an interoperable system of a federation of databases, providing either DDL-defined composite schemas using export schemas or direct multiple database manipulation through the DML.

In operation, the mobile client would consult the export schemas, or use the DML to gather data. While the DML provides interoperability [19], this may be inconvenient for users since they must embed the integration in their queries. Its advantage is that there is no requirement to integrate them and store a replica of an export schema. However, if an export schema were stored locally, then mapping resolution would be simpler and the user need not carry the burden of how to integrate the multiple databases. However, MRDSM, although a complete prototype, does not deal with physical data distribution and does not deal with query optimization issues [4], which is crucial for mobile systems.

Schema integration may occur using a mobile workstation whenever DML or DDL commands are used to combine information from different databases. It is direct and not too complex, although the component databases must be relational or at least present a relational view. However, the integrator must have knowledge about the schema of the multidatabase it wishes to integrate. This is a disadvantage as the component schemas must be downloaded to the mobile host unless copies are already available locally. And since integration is basically manual, this would take a lot of time.

The Remote-Exchange System. Integration in the Remote-Exchange system involves consulting the sharing advisor to derive a sharing pattern, resolving conflicts and performing integration of remote objects into local schema. Assuming that such integrated schemas of local and remote objects are in place, the mobile client will access the remote object as if it was a local object. The mapping is consulted for each access and is used to find remote information. Similar to other approaches above, the crux of overhead or expenses is in the reception of remote data and join processing.

If the remote object necessary is not yet integrated locally, then that is the time it becomes costly to use the sharing advisor, update the semantic dictionary, and perform integration into the local schema. Integration will depend on the sharing patterns available or applicable. Sharing by reference, for instance, requires keeping a copy of local handles for the remote objects, or RPC can be used to access remote methods. In any case, it is clear that the sharing behavior will affect the applicability of this technique to mobile databases. Note that the remote object itself may be copied, or perhaps only a local handle representing it is necessary. The best method would be to keep only a local handle, rather than copying the remote object completely, which can waste a lot of time and energy. However, should the access to that remote data become extensive, then it may waste the same amount, especially if the same data is accessed again and again. Aside from this, conflict resolution must be performed locally.

An interesting property of the method is that it employs complex analysis in semi-automatically establishing object equivalence, and addresses a higher level of semantic issues than others. If this analysis can become accurate, then schema integration using this method can be very efficient. The remaining barrier would seem to be the fact that all component schemas must be initially translated to KODM format. This activity may not be performed with a mobile host, and the specifics of converting to KODM was not covered by the reviewed literature.

Superview. The superview approach performs integration in a binary manner to produce a superview, which can again be used to integrate other schemas. As a ready to use view, the superview is no different from other global schema systems described above, in terms of actual usage. Meaning that since the virtual database is already in existence, then integration is not necessary unless a new superview is needed.

But in terms of integration, it is interesting to note that the superview approach will integrate only equivalent objects. Also, superview is a binary integration method, and it may prove costly due to the iterations. However, if the strong assumption that all objects to be integrated will not present conflicts, the superview approach is straightforward. As long as the integrator is familiar with the databases to integrate, generating the superview is not complex. Its designers intended that superview generation should be simple so that it can be easily assimilated in typical database systems without need for expert or sophisticated handling [20].

During operation, it is necessary that a replica of the needed superview exists. In the case that superviews would be generated locally, a replica of each component schema should also be stored locally.

5 Conclusion

We surveyed a number of schema integration methods for distributed and multidatabase systems, and we discussed them in the context of mobile database systems with mobile clients and non-mobile databases. In operation, systems with a global schema (ADDS, MIND, Superview) or federated schemata (Remote-Exchange) were most applicable since as long as the mobile client possesses a replica of the global schema and related information, there is no need for schema integration. They take less time and energy to process queries than multidatabase language systems. In performing actual schema integration with a mobile workstation, we can conclude that, regardless of integration method, it should be discouraged. This is not to say that the methodologies themselves are at fault. But the process of schema integration, no matter how high the degree of automation, is too time-consuming for a mobile workstation. However, it can be argued that mobile clients may have no choice in some cases. Multidatabase language systems require what can be considered real-time schema integration, since a global schema does not exist, and neither do federated schemata.

Automatic conflict resolution is usually not feasible [25, 1, 9] due to conflicts that will probably arise as the number of databases to integrate increase. This is very true for large, industrial databases. Close interaction with designers and users has always been necessary. In a highly automated environment such as in mobile computing environment, database access must be carried out without frequent user intervention. Otherwise, network transmission costs would become too expensive and the power supply of mobile clients will be wasted on user setup time for the integrators and experts making integration decisions. If good semantic representations can be developed, we believe that perhaps automatic integration will become more possible even with mobile clients. The overhead that hinders this is mostly on time spent by the user who analyzes the component schemas and performs the integration manually or interactively. The object-oriented paradigm as well as the agent model show much promise for mobile database applications. Object equivalence [10] can be a strong basis for conflict resolution based on semantics. The surveyed literature shows how much interest has been generated over agent-based systems and object-based systems.

We may also recommend an alternative: we may define an additional component, which we refer to as a *mobile transaction broker* or MTB, as a non-mobile host in the network which may coordinate global transactions on behalf of mobile clients. The results may be hoarded [11] as the processing goes on, while the mobile client runs in disconnected operation mode and the results may be gathered later. This portrait of the MTB can be general enough to apply to systems requiring real-time schema integration such as multidatabase language systems, or to composite schema systems such as global and federated schemata systems. Details can differentiate them as multidatabase language systems do not require one or any number of composite schemata, so the only information to forward to the mobile client would be the results of the query coordinated by the MTB. The major deterrent to this scenario is the fact that the MTB must exist in the first place. In a truly loosely coupled system of DBMSs, such a server may not exist.

We recommend further research along two areas: the possibility of using an object-oriented, agent-based mobile transaction broker in mobile database systems, and further work on semantic representation, analysis and operations. For this endeavor, formal methods of representation and manipulation of data or semantic information should also be explored.

Acknowledgement The author would like to thank the staff of the Department of Computer Technology of Monash University, especially Prof. Srinivasan and Dr. Zaslavsky, for their support and guidance. This research was made possible through their generosity and that of the Department of Science and Technology, Philippines.

References

[1] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. In A. Gupta, editor, *Integration of Information Systems: Bridging Heterogeneous Databases*, pages 72–112. IEEE Press, 1989.

[2] A. R. Bobak. *Distributed and Multi-Database Systems.* Multiscience Press, Bantam Books, 1993.

[3] A. Bouguettaya and A. Delis. On mobile database management systems. Paper Draft, School of Information Systems, Queensland University of Technology, Gardens Point Campus, 2 George St., GPO Box 2434, Brisbane 4001, Queensland, Australia, November 1994.

[4] Y. Breitbart. Multidatabase interoperability. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 1, pages 41–48. IEEE Computer Society Press, 1993.

[5] Y. Breitbart, P. L. Olson, and G. R. Thompson. Database integration in a distributed heterogeneous database system. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 5, pages 231–240. IEEE Computer Society Press, 1993.

[6] M. W. Bright, A. R. Hurson, and S. H. Pakzad. A taxonomy and current issues in multidatabase systems. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 1, pages 3–12. IEEE Computer Society Press, 1993.

[7] S. Ceri, B. Pernici, and G. Wiederhold. Distributed database design methodologies. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 4, pages 166–179. IEEE Computer Society Press, 1993.

[8] L. G. DeMichiel. Performing operations over mismatched domains. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 1, pages 103–112. IEEE Computer Society Press, 1993.

[9] A. Dogac, C. Dengi, E. Killic, G. Ozcan, S. Nural, C. Evrendilek, U. Halici, B. Arpinar, P. Koksal, N. Kesim, and S. Mancuhan. Metu interoperable database system. *SIGMOD Record*, 24(3):56–61, September 1995.

[10] D. Fang, J. Hammer, and D. McLeod. The identification and resolution of semantic heterogeneity in multidatabase systems. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 3, pages 52–59. IEEE Computer Society Press, 1993.

[11] R. Gruber, F. Kaashoek, B. Liskov, and L. Shrira. Disconnected operation in the thor object-oriented database system. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications.* IEEE, December 1994.

[12] D. K. Hsiao and M. N. Kamel. Heterogeneous databases: Proliferations, issues, and solutions. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 2, pages 60–77. IEEE Computer Society Press, 1993.

[13] A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors. *Multidatabase Systems: An Advanced Solution for Global Information Sharing*. IEEE Computer Society Press, 1993.

[14] A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors. *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 5, pages 217–230. IEEE Computer Society Press, 1993.

[15] W. Klas, P. Fankhauser, and P. Muth. Database integration using the open object-oriented database system vodak. In O. Bukhres and A. K. Elmagarmid, editors, *Object Oriented Multidatabase Systems: A Solution for Advanced Applications*. Prentice Hall, Englewood Cliffs, N.J., 1995.

[16] R. Krishnamurthy, W. Litwin, and W. Kent. Interoperability of heterogeneous databases with schematic discrepancies. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 3, pages 118–125. IEEE Computer Society Press, 1993.

[17] S. J. Lai, A. Zaslavsky, G. P. Martin, and L. H. Yeo. Cost efficient adaptive protocol with buffering for advanced mobile database applications. In T. W. Ling and Y. Masunaga, editors, *Proceedings of the Fourth International Conference on Database Systems for Advanced Applications (DASFAA '95)*, Singapore, April 1995. World Scientific Publishing Co. Pte Ltd.

[18] W. Litwin. From databases to multidatabases: Why and how. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 1, pages 13–40. IEEE Computer Society Press, 1993.

[19] W. Litwin and A. Abdellatif. Multidatabase interoperability. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 4, pages 180–188. IEEE Computer Society Press, 1993.

[20] A. Motro. Supveriews: Virtual integration of multiple databases. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 5, pages 292–305. IEEE Computer Society Press, 1993.

[21] E. Pitoura and B. Bhargava. Dealing with mobility: Issues and research challenges. Technical Report CSD-TR-93-070, Department of Computer Science, Purdue University, West Lafayette, IN 47907, November 1993.

[22] E. Pitoura and B. Bhargava. A framework for providing consistent and recoverable agent-based access to heterogeneous mobile databases. *SIGMOD Record*, 24(3):44–49, September 1995.

[23] F. Saltor, M. G. Castellanos, and M. Garcia-Solaco. Overcoming schematic discrepancies in interoperable databases. In D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, editors, *IFIP DS-5 Semantics of Interoperable Database Systems*, volume 1, pages 184–198, Lorne, Victoria, Australia, November 1992. IFIP and CITRI and Australian Computer Society and La Trobe University.

[24] A. Sheth and V. Kashyap. So far (schematically) yet so near (semantically). In D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, editors, *IFIP DS-5 Semantics of Interoperable Database Systems*, volume 1, pages 282–301, Lorne, Victoria, Australia, November 1992. IFIP and CITRI and Australian Computer Society and La Trobe University.

[25] A. P. Sheth and H. Marcus. Schema analysis and integration: Methodology, techniques and prototype toolkit. Technical Memorandum TM-STS-019981/1, Bell Communications Research, March 1992.

[26] G. Thomas, G. R. Thompson, C-W. Chung, E. Barkmeyer, F. Carter, M. Templeton, S. Fox, and B. Hartman. Heterogeneous distributed database systems for production use. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 5, pages 306–335. IEEE Computer Society Press, 1993.

[27] V. Ventrone and S. Heiler. Semantic heterogeneity as a result of domain evolution. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 3, pages 113–117. IEEE Computer Society Press, 1993.

[28] L. H. Yeo and A. Zaslavsky. Submission of transactions from mobile workstations in a cooperative multidatabase processing environment. In *Proceedings of the 14th International Conference on Distributed Computing Systems*, pages 372–379, Los Alamitos, California, June 1994. IEEE, IEEE Computer Science Press.