MASSACHUSETTS INSTITUTE of TECHNOLOGY
LIBRARIES

M.I.T. LIBRARIES - DEWEY

Dewey

HD28
.M414
no.
3658-
94

29

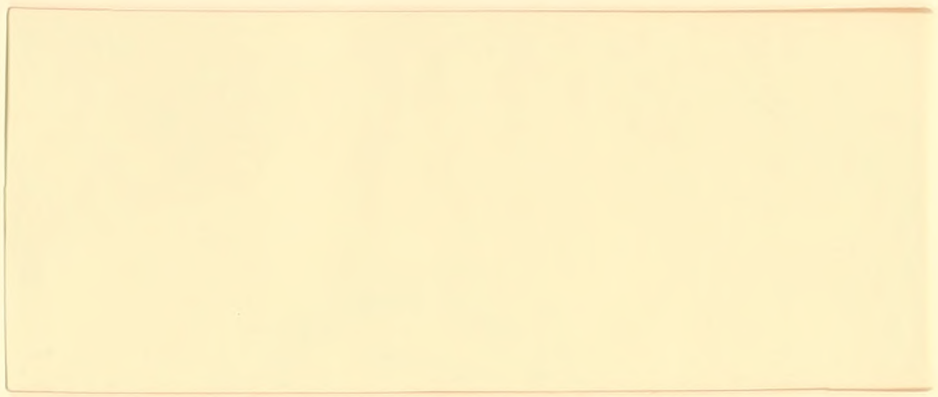
WORKING PAPER
ALFRED P. SLOAN SCHOOL OF MANAGEMENT

Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment

February 1994 WP# 3658-94
CISL WP# 94-01

Cheng Hian Goh
Stuart E. Madnick*
Michael Siegel**

MASSACHUSETTS
INSTITUTE OF TECHNOLOGY
50 MEMORIAL DRIVE
CAMBRIDGE, MASSACHUSETTS 02139



Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment

February 1994 WP# 3658-94
CISL WP# 94-01

Cheng Hian Goh
Stuart E. Madnick*
Michael Siegel**

* Stuart Madnick, E53-321
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139

** Michael Siegel, E53-323
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139

M.I.T. LIBRARIES

MAR 2 4 1994

RECEIVED

Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment*

Cheng Hian Goh Stuart E. Madnick Michael D. Siegel
Sloan School of Management
Massachusetts Institute of Technology

Abstract

Research in database interoperability has primarily focused on circumventing schematic and semantic incompatibility arising from autonomy of the underlying databases. We argue that, while existing integration strategies might provide satisfactory support for small or static systems, their inadequacies rapidly become evident in large-scale interoperable database systems operating in a dynamic environment. The frequent entry and exit of heterogeneous interoperating agents renders "frozen" interfaces (e.g., shared schemas) impractical and places an ever increasing burden on the system to accord more flexibility to heterogeneous users. *User heterogeneity* mandates that disparate users' conceptual models and preferences must be accommodated, and the emergence of large-scale networks suggests that the integration strategy must be *scalable* and capable of dealing with *evolving semantics*.

As an alternative to the integration approaches presented in the literature, we propose a strategy based on the notion of *context interchange*. In the context interchange framework, assumptions underlying the interpretations attributed to data are explicitly represented in the form of *data contexts* with respect to a *shared ontology*. Data exchange in this framework is accompanied by *context mediation* whereby data originating from multiple *source contexts* is automatically transformed to comply with the *receiver context*. The focus on data contexts giving rise to data heterogeneity (as opposed to focusing on data conflicts exclusively) has a number of advantages over classical integration approaches, providing interoperating agents with greater flexibility as well as a framework for graceful evolution and efficient implementation of large-scale interoperable database systems.

Keywords: context interchange, heterogeneous databases, semantic interoperability, shared ontology.

*This research is supported in part by NSF grant IRI-90-2189, ARPA grant F30602-93-C-0160, the Productivity From Information Technology (PROFIT) project, and the International Financial Services Research Center at MIT.

1 Introduction

The last decade or so have witnessed the emergence of new organizational forms (e.g., adhocracies and virtual corporations) which are critically dependent on the ability to share information across functional and organizational boundaries [21]. Networking technology however provides merely *physical connectivity*: meaningful data exchange (or *logical connectivity*) can only be realized when agents are able to attribute the same interpretation to data being exchanged. The quest for logical connectivity has been a major challenge for the database research community: there has been, in the short span of a few years, a proliferation of proposals for how logical connectivity among autonomous and heterogeneous databases can be accomplished. The body of research describing these endeavors have appeared under various headings in different places: for instance, "heterogeneous database systems" [28], "federated database systems" [33], and "multidatabase systems" [5]. In this paper, we use the generic phrase "interoperable database systems" to encompass all of these usage in referring to a collection of database systems (called *component database systems*) which are cooperating to achieve various degrees of integration while preserving the autonomy of each component system.

The autonomous design, implementation and administration of component databases brings about the problem of heterogeneity: *system heterogeneity* refers to non-uniformity in data models, data manipulation language, concurrency control mechanisms, hardware and software platforms, communication protocols and such; *data heterogeneity* refers to non-uniformity in which data is structured (*schematic discrepancies*) and the disparate interpretations associated with them (*semantic discrepancies*) [28]. Conflicts arising from system heterogeneity are often seen as surmountable through the provision of "wrappers" which provides for the translation of one protocol into another; on the other hand, the problem of data heterogeneity is much more intractable and has been a major concern among the database research community [5,28,33]. This in turn has led to a variety of research prototypes demonstrating how issues arising from schematic and semantic heterogeneity can be resolved [1,3,4,6,8,14,18,23].[1]

This paper stems from our concern for the inadequacies of existing integration strategies when applied to the construction of *large-scale* systems in a *dynamic* environment, an example of which is the Integrated Weapons System Data Base (IWSDB)[2] [41]. For the purpose of our discussion, we characterize a dynamic environment to be one in which *heterogeneous interoperating agents*, which include *data sources* (i.e., databases and data feeds) and *data receivers* (users and applications accessing data), may join or quit the system over time. This has a number of implications. First, because the requirements of data receivers are diverse (a fact which we refer to as *receiver* (or *user*) *heterogeneity*) and can change rapidly, it is impractical for data requirements to be captured and "frozen" (e.g., in shared schemas): i.e.,

[1] Surveys on a wide selection of prototype implementations can be found in [5,7,33].

[2] The IWSDB is a project aimed at providing integrated access to the databases supporting the design and engineering of the F-22 Advanced Tactical Fighter. The IWSDB spans databases containing information on technical specifications, design, manufacturing, and operational logistics. Work on the F-22 began in 1992, and the IWSDB is expected to operate to the year 2040. As of March 1993, more than 50 databases have already been identified, and this number is expected to grow greatly as more sub-contractors are being brought into the process.

data receivers must be given the flexibility of defining what their data needs are dynamically. Second, large-scale dynamic systems introduce considerable complexity and accentuate the need for better performance in managing system development, availability and query processing since mediocre solutions are likely to be felt with greater impact.

The goal of this paper is to highlight the above concerns and to illustrate how they can be overcome in the context interchange framework. The notion of context interchange in a *source-receiver system* was first proposed in [34,35]. In particular, it has been suggested that the context of data sources and receivers can be encoded using *meta-attributes*. In the same spirit as Wiederhold's mediator architecture [40], Siegel and Madnick [35] suggested that a *context mediator* can be used to compare the source (export) and receiver (import) contexts to see if there are any conflicts. Sciore, Siegel and Rosenthal [29] have proposed an extension to SQL, called *Context-SQL (C-SQL)* which allows the receivers' import context to be dynamically instantiated in an SQL-like query. Hence, the import context can be conveniently modified to suit the needs of the user in different circumstances. A theory of *semantic values* as a basis for data exchange has also been proposed in [30]. This work defines a formal basis for reasoning about conversion of data from one context to another based on context knowledge encoded using meta-attributes and a rule-based language. Our contribution in this paper is an integration of these results and the generalization to the scenario where there are multiple data sources and receivers.

The rest of this paper is organized as follows. Section 2 lays the foundation for subsequent discussion by examining both the classical concerns (i.e., over schematic and semantic heterogeneities) and the strategies for addressing them. Section 3 presents the issues pertinent to large-scale and dynamic interoperable systems which to-date have received little attention from this research community. Section 4 describes the context interchange framework, discusses the rationale underlying its design, and contrasts it with other competing integration strategies. Section 5 summarizes our contribution and describes work in the pipeline.

2 Interoperable Database Systems: Current Issues and Strategies

Research in interoperable database systems has traditionally focused on resolving conflicts arising from *schematic* and *semantic* heterogeneities. This section of the paper serves to elaborate on these concerns and gives a brief overview of the different strategies proposed for overcoming them.

2.1 Schematic and Semantic Heterogeneities

Site autonomy — the situation whereby each component DBMS retains complete control over data and processing — constitutes the primary reason for heterogeneous schema and semantics of data situated in different databases. In the rest of this subsection, we will enumerate some of the most commonly occurring conflicts belonging to the two genre.

Conflicts arising from *schematic heterogeneity* have been extensively documented in Kim

Database 1a

Date	StkCode	TradePrice
01/02/93	HPP	30.10
01/02/93	BBM	40.20
⋮	⋮	⋮

Database 1b

Date	HPP	BBM	···
01/02/93	30.10	40.20	···
⋮	⋮	⋮	⋮

Database 1c

relation HPP

Date	TradePrice
01/02/93	30.10
⋮	⋮

relation BBM

Date	TradePrice
01/02/93	40.20
⋮	⋮

Figure 1: Example of three stock databases exhibiting structural conflicts.

Database 2a

(Based on actual realtime feed provided on the Reuters Monitor Service)

Date	StkCode	Exchange	TradePrice	PE
13 Jan 94	IBM	NYS	$58\frac{1}{8}$	
13 Jan 94	TX	NYS	65*03	15.15
13 Jan 94	LVNI.O	NAS	5\05	
13 Jan 94	SAPG.F	FRA	1790.00	
13 Jan 94	RENA.PA	PAR	2380	
13 Jan 94	NA.TO	TOR	$10\frac{3}{4}$	11.39
13 Jan 94	BMO.M	MON	$27\frac{1}{2}$	11.53
⋮	⋮	⋮	⋮	⋮

Database 2b

(Hypothetical database for a US investor)

Date	StkCode	Price	Shares
12/20/93	IBM	58.50	200
12/23/93	SAP AG	1028.74	1000
01/10/94	Renault	402.03	500
⋮	⋮	⋮	⋮

Figure 2: Examples of conflicts arising from semantic heterogeneity

and Seo [12]. Two types of conflicts are frequently cited as belonging to this category. *Naming conflicts* such as *synonyms* (different attributes names referring to the same thing) and *homonymns* (the same attribute name having different meanings) arise from the uncoordinated assignment of names in a database schema. Examples of homonymns and synonyms are plentiful: an attribute name such as `name` is frequently used for referring to different things in different scenarios (e.g., name of a person and name of a company); on the other hand, an item such as "employee name" is frequently given a variety of attribute names such as `empName` or `employee-name` and so on. *Structural conflicts* came about because the same piece of information may be modeled as a relation name, an attribute name, or a value in a tuple. Figure 1 depicts how three stock databases might exhibit this type of conflict.[3]

Conflicts arising from *semantic heterogeneity* are much more complex and are less well understood despite attempts at classifying them (see, for instance, [32]). Figure 2 illustrates some of these conflicts and complications. The reader is encouraged to consider what knowledge

[3]This example is first discussed in [13].

is needed to determine how much money the investor (Database 2b) made as of January 13, 1994, using the information from the two databases shown. A few of the problems are discussed below.

Naming conflicts similar to those corresponding to schematic heterogeneity can occur at the semantic level: in this case, we can have homonymns and synonyms of *values* associated with attributes. For example, different databases in Figure 2 might associate a different stock code with the same stock (e.g., SAPG.F versus SAP AG). (Where the conflict concerns the key or identifier for a particular relation, this leads to the problem sometimes referred to as the *inter-database instance identification problem* [38].) *Measurement conflicts* arise from data being represented in different units or scales: for example, stock prices may be reported in different currencies (e.g., USD or marks) depending on the stock exchange on which a stock trade occurs. *Representation conflicts* arise from different ways in representing values, such as the disparate representations of fractional dollar values in Database 2a (e.g., 5\05 on the New York Stock Exchange actually mean $5\frac{5}{16}$ or 5.3125). *Computational conflicts* arise from alternative ways to compute data. For example, although "Price-to-Earnings Ratio" (PE) should just mean "Price" divided by "Earning", it is not always clear what interpretation of "Price" and "Earning" are used. Also, in the Reuters interpretation, PE is not defined when "Earnings" are negative. *Confounding conflicts* may result from having different shades of meanings assigned to a single concept. For example, the price of a stock may be the "latest closing price" or "latest trade price" depending on the exchange or the time of day. Finally, *granularity conflicts* occur when data are reported at different levels of abstraction or granularity. For example, the values corresponding to a location (say, of the stock exchange) may be reported to be a country or a city.

2.2 Classical Strategies for Database Integration

In the last couple of years, there has been a proliferation of architectural proposals and research prototypes aimed at achieving interoperability amongst autonomous and heterogeneous databases. One approach for distinguishing the various integration strategies is by observing the strength of data coupling at the schema level. This has been the basis for the taxonomy presented in [33], which we will describe briefly below.

- An interoperable database system is said to be *tightly-coupled* if conflicts inherent in the component databases are resolved, a priori, in one or more *federated schemas*. Where there is exactly one federated schema, this is sometimes referred to as a *global schema* multidatabase system; otherwise, it is called a *federated database system*. With this approach, users of the system interact exclusively with one or more federated schemas which mediate access to the underlying component databases. The notions underlying this strategy has its roots from the seminal work of Motro and Buneman [22] and Dayal and Hwang [9] who independently demonstrated how disparate database schemas can be merged to form a unified schema, for which queries can be reformulated to act on the constituent schemas. Examples of tightly-coupled systems include most of the early research prototypes, such as Multibase [14], and Mermaid [36].

- Advocates of *loosely-coupled* systems on the other hand suggested that a more general approach to this problem consists of providing users with a *multi-database manipulation language*, rich enough so that users may easily devise ways of circumventing conflicts inherent in multiple databases. For example, users of the system may define integration rules (e.g., a table which decides how to map letter grades to points) as part of the query. No attempt is made at reconciling the data conflicts in a shared schema a priori. The MRDSM system [18] (which is part of the French Teletel project at INRIA) is probably the best-known example of a loosely-coupled system.

In most instances, the tightly-coupled systems surveyed in [33] focus almost exclusively on the issue of schematic heterogeneity and there was little support (if any) for circumventing conflicts arising from semantic heterogeneity. Loosely-coupled systems, such as MRDSM, attempt to remedy this deficiency by providing language features for renaming, data conversions, and dealing with inconsistent data [17]. Unfortunately, the efficacy of this approach depends on users having knowledge of the semantic conflicts: support for conflict detection and resolution was either lacking or accomplished in some ad hoc manner. These observations were largely responsible for the emergence of a number of experimental prototypes aimed at achieving interoperability at the semantic level. We observed through our survey of the literature two genre of systems: the first adopting an object-oriented data model as a basis for integration, and the second, a knowledge representation language. These newcomers share a common feature: the ability to interoperate at the semantic level is derived from the adoption of a much richer language (compared to the earlier generation) which provides for semantic reconciliation. We will briefly describe each approach below.

Object-oriented multidatabase systems advocate the adoption of an object-oriented data model as the *canonical model* which serves as an *inter lingua* for reconciling semantic discrepancies amongst component database systems. Examples of these systems include Pegasus [1], Comandos Integration System (CIS) [4], Distributed Object Management (DOM) [6], FBASE [23] and others.[4] With this approach, semantic interoperability is accomplished via the definition of supertypes, the notion of upward inheritance, and the adoption of user-defined methods. Consider, for instance, the following granularity conflict: in the first database, a student's grade is represented as a letter grade; in the second, the corresponding information is represented as points in the range 0 to 100. In the Pegasus system [1], integration is achieved by introducing a supertype of the two types, allowing all attributes of the subtypes to be *upward inherited* by the supertype, and attaching a method to the supertype to provide the necessary conversion:

[4]See [7] for a comprehensive survey.

```
create supertype Student of Student1, Student2;
create function Score(Student x) ->
    real r as
    if Student1(x) then Map1(Grade(x))
    else if Student2(x) then Map2(Points(x))
    else error;
```

In the above example, Map1 and Map2 implement the conversions which must take place to transform data from different representations into a common one.

We contend that as powerful as object-oriented data models may be, this approach has a number of inherent weaknesses. First, procedural encapsulation of the underlying data semantics (as demonstrated in the above example) means that underlying data semantics will no longer be available for interrogation. For example, once the canonical representation for a student's score is chosen, the user has no way of finding what the original representations in the component databases are. Second, this integration strategy presupposes that there is one canonical interpretation which everyone subscribes to. For example, a particular user might prefer to receive the information as letter grades instead of raw scores. There are two ways of circumventing this: define a method which translates the canonical representation (score) to letter grade, or create a new supertype altogether by-passing the old one. Either implementation is clearly inefficient and entails more work than necessary. Third, the collection of conflicting representations from disparate databases into one supertype may pose a problem for the graceful evolution of the system. For example, the score function above would need to be modified each time a new database containing student information is added to the network.

The approach taken in database integration using a *Knowledge Representation (KR)* language constitutes a novel departure from the traditional approaches and in fact is largely pioneered by researchers from the Artificial Intelligence (AI) community. Two such attempts are the Carnot project [8] which employs the Cyc knowledge base as the underlying knowledge substrate, and the SIMS project [2,3] which uses Loom [20] as the underlying knowledge representation language. Integration in these *KR multidatabase systems* is achieved via the construction of a global semantic model unifying the disparate representations and interpretations in the underlying databases. In this regard, it is analogous to the global schema multidatabase systems, except that we now have a rich model capturing not just the schematic details but the underlying semantics as well. For instance, users of SIMS will interact exclusively with the semantic model and the system takes on the responsibility of translating queries on "concepts" (in the semantic model) to actual database queries against the underlying databases.

Data resource transparency (i.e., the ability to pose a query without knowing what databases need to be searched) has been an explicit design goal of KR multidatabase systems. While this might be useful in many circumstances, there are clear disadvantages in not providing users with the alternative of selectively defining the databases they want to access. In many instances (e.g., the financial industry), it is important for users to know where answers to their query are derived from since data originating from different sources may differ in their quality and this has an important impact in decision making [39]. In addition, some of the objections we have

for object-oriented multidatabase systems are also true for the KR multidatabase systems. For instance, semantic heterogeneity in these systems are typically overcome by having disparate databases map to a standard representation in the global semantic model. This means that, like the case of object-oriented multidatabase systems, data semantics remain obscure to the users who might have an interest in them, and changing the canonical representation to suit a different group of users may be inefficient.

3 Beyond Schematic and Semantic Heterogeneities

Attaining interoperability in a large-scale, dynamically evolving environment presents a host of different issues not addressed by existing integration strategies. In this section, we examine some of the issues which arise from *receiver heterogeneity*, the nature of *large-scale* systems, and their collective impact on the ability of the system to *evolve* gracefully.

3.1 Receiver Heterogeneity

Previous research in interoperable database systems is largely motivated by the constraint that data sources in such a system are autonomous (i.e., sovereign) and any strategy for achieving interoperability must be *non-intrusive* [28]: i.e., interoperability must be achieved in ways other than modifying the structure or semantics of existing databases to comply with some standard. Ironically, comparative little attention has been given to the symmetrical problem of *receiver heterogeneity* and their sovereignty. In actual fact, receivers (i.e., users and applications retrieving data from one or more source databases) differ widely in their conceptual interpretation of and preference for data and are equally unlikely to change their interpretations or preferences.

Heterogeneity in conceptual models. Different users and applications in an interoperable database system, being themselves situated in different real world contexts, have different "conceptual models" of the world.[5]. These different views of the world lead users to apply different assumptions in interpreting data presented to them by the system. For example, a study of a major insurance company revealed that the notion of "net written premium" (which is their primary measure of sales) can have a dozen or more definitions depending on the user department (e.g., underwriters, reinsurers, etc). These differences can be "operational" in addition to being "definitional": e.g., when converting from a foreign currency, two users may have a common definition for that currency but may differ in their choice of the conversion method (say, using the latest exchange rate, or using a policy-dictated exchange rate).

Heterogeneity in judgment and preferences. In addition to having different mental models of the world, users also frequently differ in their judgment and preferences. For example, users might differ in their choice of what databases to search to satisfy their query: this might be due to the fact that one database provides better (e.g., more up-to-date) data than another, but is more costly (in real dollars) to search. The choice of which database to query is not apparent

[5]This observation is neither novel nor trivial: the adoption of *external views* in the ANSI/SPARC DBMS architecture is evidence of its significance.

and depends on a user's needs and budget. Users might also differ in their judgment as to which databases are more credible compared to the others. Instead of searching all databases having overlapping information content, users might prefer to query one or more databases which are deemed to be most promising before attempting other less promising ones.

The preceding discussion suggests that receiver heterogeneity should not be taken lightly in the implementation of an interoperable database system. Integration strategies (e.g., the global schema database systems) do so at the peril of experiencing strong resistance because users deeply entrenched in their mental models will resist being coerced into changing their views of the world. Moreover, some of these receivers could be pre-existing applications and semantic incompatibility between what the system delivers and what the application expects will be disastrous.

There is at least one attempt in mitigating this problem. Sheth and Larson [33] have proposed that data receivers should be given the option of tailoring their own *external schemas* much like external views in the ANSI/SPARC architecture. This scheme however has a number of shortcomings in a diverse and dynamic environment. In many instances, the mechanism for view definition is limited to structural transformations and straight forward data conversions which might not be adequate for modeling complex and diverse data semantics. Moreover, the view mechanism presupposes that the assumptions underlying the receivers' interpretation of data can be "frozen" in the schema. Users however rarely commit to a single model of the world and often change their behavior or assumptions as new data are being obtained. For example, after receiving answers from a "cheap" but out-dated database, a user might decide that the additional cost of getting more up-to-date data is justifiable. Encapsulating this behavior in a predefined view is probably impractical. Once again, this problem will be accentuated in a dynamic environment where new and diverse users (with different views of the world) are frequently added to the system.

In brief, we take the position that a good integration strategy must accord data receivers (i.e., users and applications requesting data) with the same *sovereignty* as the component databases. In other words, users should be given the prerogative in defining how data should be retrieved as well as how they are to be interpreted.

3.2 Scale

A large-scale interoperable database environment (e.g., one with three hundred component databases as opposed to three) presents a number of problems which can have serious implications for the viability of an integration strategy. We suggest that the impact of scale can be felt in at least three areas: system development, query processing, and system evolution. The first two issues are described in this subsection and the last is postponed to the next.

System development. From the cognitive standpoint, human bounded rationality dictates that we simply cannot cope with the complexity associated with hundreds of disparate systems each having their own representation and interpretation of data. Integration strategies which rely on the brute-force resolution of conflicts simply will not work here. For example, the global

schema approach to database integration advocates that all data conflicts should be reconciled in a single shared schema. Designing a shared schema involving n different systems therefore entails reconciling an order of n^2 possibly conflicting representations. Clearly, the viability of this approach is questionable where the number of databases to be integrated becomes large.

Multidatabase language systems represent the other extreme since there are no attempts at resolving any of the data conflicts a priori. However, it appears that the problem did not simply go away but is instead being passed down to the users of the system. Instead of attempting to reconcile all conflicts a priori in a shared schema, the multidatabase language approach delegates completely the task of conflict detection and resolution to the user. The problem then becomes much more pronounced since users do not necessarily have access to underlying data semantics nor do they necessarily have the time and resources to identify and resolve conflicting data semantics.

Query-processing. A key concern for databases has always been that of efficient query processing. Large-scale interoperable database systems have the potential of amplifying poor query responses in a number of ways. In a small and controlled environment, it is often possible for "canned" queries to be carefully handcrafted and optimized. Such an approach again would be impractical in large (and especially, dynamic) environments. In addition, large-scale systems tend also to be more diverse and this is certain to lead to greater data heterogeneity. This implies that conversions from one data representation to another will have to be frequently performed. In those instances where integration is achieved by having each component database system mapped to a canonical representation which may then be converted to the representation expected by the receiver, this entails a great deal of redundant work which can significantly degrade system performance. In addition, the encapsulation of data semantics in these conversion functions means that they will remain inaccessible for tasks such as semantic query optimization.

3.3 System Evolution

Changes in an interoperable database system can come in two forms: changes in semantics (of a component database or receiver) and changes in the network organization (i.e., when a new component database is added or an old one removed from the system).

Structural changes. For integration strategies relying on shared schemas, frequent structural changes can have an adverse effect on the system since changes entail modifications to the shared schema. As we have pointed out earlier, the design of a shared schema is a difficult task especially when there is a large number of component systems. Modifications to the shared schema suffers from the same difficulties as in system development, and in some cases, more so because the system may have to remain online and accessible to geographically dispersed users throughout the day.

Domain evolution. Changes in data semantics have a more subtle effect on the system. Ventrone and Heiler [37] referred to this as *domain evolution* and have documented a number of examples why this may occur. Since we expect these changes to be infrequent (at least with respect to

a single database), the impact of domain evolution on small or stable interoperable database systems is likely to be insignificant. This however is no longer true for large-scale systems, since large numbers of infrequent changes in individual databases adds up to formidable recurring events at the system level. For example, assuming an average of one change in three years for any given database, an interoperable database system with three hundred databases will have to contend with 100 changes every year, which translates to two changes every week! Domain evolution has significant impact for integration strategies which rely on prior resolution of semantic conflicts (e.g., object-oriented multidatabase systems) since semantic changes entail modifying definitions of types (and corresponding conversion functions embedded in them). Consider for instance, when databases reporting monetary values in French francs are required to switch to European Currency Units (ECUs). This change will require modifying all type definitions which have an attribute whose domain (in some component databases) is a monetary value reported in French francs. Since there might be several hundreds of attributes having monetary value as its domain, it is not hard to imagine why a change like this would have dire consequences on the availability and integrity of the system.

4 The Context Interchange Framework

The key to the *context interchange* approach to achieving interoperability is the notion of *context*. We use the word "context" to refer to the (implicit) assumptions underlying the way in which an interoperating agent routinely represents or interprets data. Data contexts, like *event scripts* [31], are abstraction mechanisms which allow us to cope with the complexities of life. For example, when visiting a local grocery store in the US, we assume that the price on a shelve with boxes of Cheerios is reported in US dollars, that this price refers to unit price of a box (rather than say unit price per pound), that "3.40" refers to $3.40 and not 3.4 cents, and so forth. Given sufficient time, groups of individuals will tend to develop shared assumptions with respect to how one should interpret the data generated and owned by the group. These shared assumptions are desirable because it reduces the cost of communication among members of the group. In the same way, applications and individuals issuing queries to a data repository all have their own assumptions as to how data should be routinely represented or interpreted. All is well when these assumptions do not conflict with one another, as is usually the case if databases, applications, and individuals are situated in the same social context. When multiple databases, applications, and individuals transcending organizational or functional boundaries are brought together in an interoperable system, the disparate data contexts each brings to the system result in both schematic and semantic conflicts.

4.1 Context Interchange in Source-Receiver Systems

We will first exemplify the key notions underlying the context interchange strategy as depicted in Figure 3 with a simple scenario where there is only one data source (i.e., a database or some other data repository) and one data receiver (an application or user requesting for data) [34,35]. To allow data exchange between the data source and data receiver to be meaningful, data

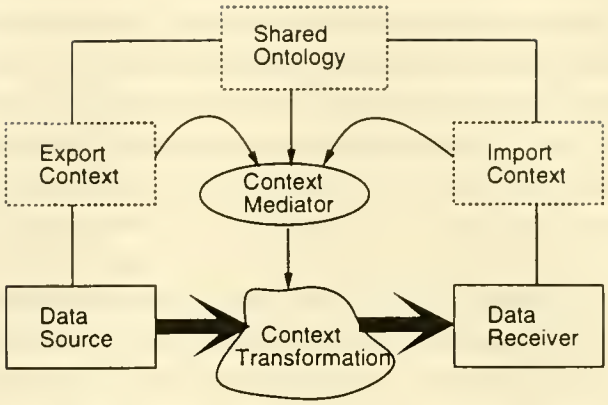


Figure 3: Context mediation in a simple source-receiver system.

contexts specific to both are captured in an *export context* and an *import context* respectively: i.e., the export context captures those assumptions integral to the "production" of data in the data source, and the import context captures those assumptions which the data receiver will employ in interpreting the data. The export and import contexts are defined with respect to a *shared ontology* [11] which constitutes a shared vocabulary for context definition. Intuitively, the shared ontology is needed because the only way disparity can be identified is when we have consensus for mapping real world semantics to syntactic tokens in a consistent manner. In this framework, data transmitted from the data source to the data receiver undergo *context transformation* supervised by a *context mediator*. The context mediator detects the presence of semantic conflicts (if any) between data supplied by the data source and data expected by the data receiver by comparing the export and import contexts, and calls upon conversion functions (if available) to reconcile these disparities.

To make the discussion more concrete, suppose the data source is a database containing information on stocks traded at the New York Stock Exchange and the data receiver is a stock broker in Tokyo. The NYSE database reports the "latest closing price" of each stock in US dollars; the stock broker however might be expecting to see the "latest trade price" in Yen. Both sets of assumptions can be explicitly captured in the export and import contexts, and the context mediator will be responsible for detecting any conflicts between data provided by the source and the interpretation expected by the receiver. When such a conflict does occur (as in this case), the context mediator will attempt to reconcile the conflict by automatically applying the necessary conversion functions (e.g., converting from US dollars to Yen). When relevant conversion functions do not exist (e.g., from "latest closing price" to "latest trade price"), data retrieved can still be forwarded to the data receiver but the system will signal the anomaly. If the receiver is indifferent to whether prices are "latest closing" or "latest trade", the disparity disappears and both types of prices will be received by the receiver without any distinction. Instead of passively defining the assumptions underlying their interpretation of data, receivers have also the option of defining customized conversion functions for reconciling

semantic conflicts. For example, a user might have reason to believe that exchange rates between two currencies should be something other than what is currently available to the system and might therefore choose to define his own conversion routines as part of his import context.

4.2 Context Interchange with Multiple Sources and Receivers

The strategy for achieving interoperability in a source-receiver system can be generalized to the scenario where there are multiple data sources and receivers. Figure 4 illustrates what the architecture of such a system might look like. This architecture differs from the source-receiver framework presented in Figure 3 in two significant ways. First, because we now have multiple data sources and receivers, it is conceivable that groups of interoperating agents may be situated in similar social contexts embodying large number of common assumptions. We exploit this fact by allowing commonly-held assumptions to be shared among distinct interoperating agents in a *supra-context*, while allowing the representation of idiosyncratic assumptions (i.e., those peculiar to a particular data source or receiver) in individual *micro-contexts*. The export or import context of each agent is therefore obtained by the summation of assumptions asserted in the supra- and micro-contexts. This nesting of one data context in another forms a *context hierarchy* which, in theory, may be of arbitrary depth. Second, we need to consider how a user might formulate a query spanning multiple data sources. As noted earlier, two distinct modes have been identified in the literature: data sources may be pre-integrated to form a *federated schema*, on which one or more *external views* may be defined and made available to users, or, users may query the component databases directly using their export schemas. We are convinced that neither approach is appropriate all the time and have committed to supporting both types of interactions in the proposed architecture. In each case, however, the context mediator continues to facilitate interoperability by performing the necessary context mediation. In the rest of this section, we shall present the rationale underlying the design of the proposed framework, and where appropriate, the more intricate details of the integration strategy.

Our concern for user heterogeneity, scalability and the capacity for graceful evolution (see Section 3) has led us to a number of considerations which prompted the incorporation of the following features in the context interchange architecture.

1. *Explicit representation of and access to underlying data semantics.*

   Instead of reconciling conflicting semantics directly in shared schemas, we advocate that semantics of data underlying disparate data sources be captured in export contexts independent of the method for reconciling semantic conflicts. This is achieved by associating every data element in the data source with a *semantic domain* which has a meaningful interpretation in the shared ontology. For example, instead of defining "Yen" by supplying a function for converting a stock price from Yen to USD, we suggest that underlying data semantics should be explicitly represented by first defining in the shared ontology, the concept "Yen" to be a kind of currency and that monetary amounts in one currency can be converted into another given an exchange rate, and associating attribute `stockPrice` with the semantic domain "a monetary amount <currency: Yen, unit: thousands>".

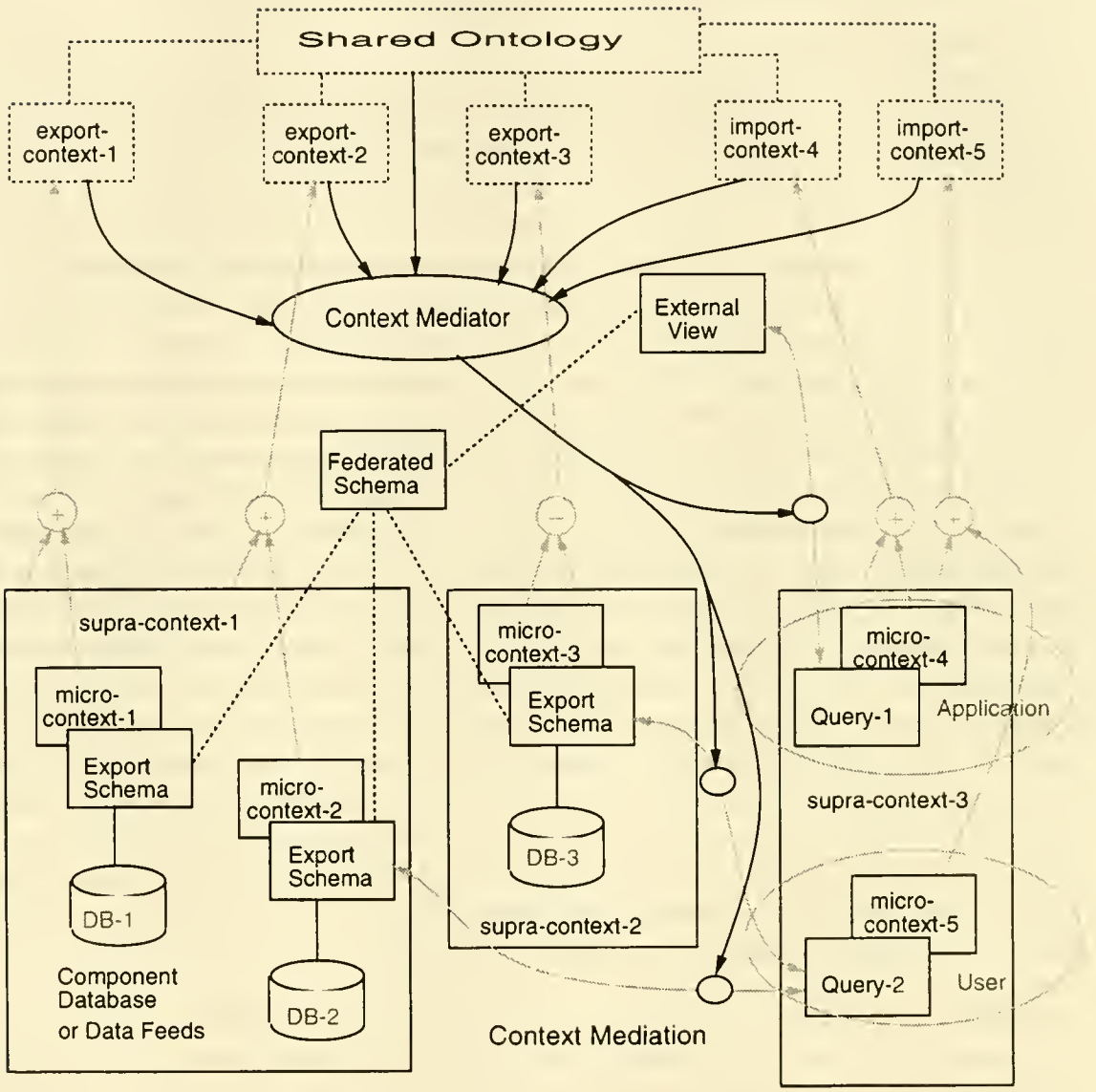


Figure 4: Context interchange with multiple data sources and data receivers.

Representation of data semantics in semantic domains independent of the database schema and the conversion functions used for achieving reconciliation has a number of advantages.

- By allowing data semantics to be defined without the use of shared schemas, we are able to support semantic interoperability even when no integrated schema exists.
- Because data semantics are represented explicitly (as opposed to being implicitly in conversion functions), intensional queries (e.g., "What currency is this data reported in?") can now be answered.
- More intelligent query processing behavior can now be expected of the system since we no longer commit ourselves to a single conflict resolution strategy. Consider for example, the query issued by a US broker (for whom all prices are in USD):

  select stkCode
  from Tokyo-stock-exchange.trade-rel
  where stkPrice > 10;

  If we have defined the semantics of "Yen" with respect to "USD" via only a unidirectional conversion function (from Yen to USD), this query would have lead to an unacceptable query plan (converting all stocks traded in Tokyo stock exchange to USD before making the selection required). A more sensible query plan consists of converting 10 USD to the equivalent in Yen and allowing the DBMS at Tokyo stock exchange to do the selection. This second query plan can be trivially produced by the context mediator. The point here is that converting from one data representation to another is an expensive operation and it pays handsomely if we can apply knowledge of data semantics in optimizing these queries.
- Another advantage of abstaining from a pre-established resolution strategy (i.e., conversion function) is that it allows data receivers to define (at a much later time) the conversion which they wish to use. An example we gave earlier in Section 3 involves users having different exchange rates for currency conversion. In this instance, there is no one "right" conversion rate: by defining the exchange rate they wish to use in their import context, data receivers can customize the behavior of conversion functions.
- Finally, the independent specification of data semantics in semantic domains provide a level of abstraction which is more amenable to sharing and reuse. Sharing can take place at two levels. First, semantic domains can be shared by different attributes in a single data source. For example, a financial database may contain data on trade price, issue price, exercise price, etc, all of which refer to some monetary amount. Instead of reconciling data elements individually with their counterparts (e.g., stock price of a stock in Tokyo stock exchange and that in New York stock exchange), these attributes can be associated with a single semantic domain ("monetary amount <currency: Yen, unit: thousands>") in the export context. Second, context definitions may also be shared between different data sources which are situated in the same *social context*. For example, both the commodity exchange and stock exchange at Tokyo

are likely to have similar semantics for all things "japanese" (e.g., currency. date format, reporting convention). Instead of redefining a conflict resolution strategy for each database, their commonality can be captured by allowing them to share overlapping portions of their export context. As mentioned earlier, we refer to the portion of the context definition peculiar to the individual data source as the *micro-context*, and the sharable part of the context definition, the *supra-context*.

2. *Explicit source selection subjected to users' preferences.*

   Instead of constraining data receivers to interacting with shared schemas (and hence predetermined sets of data sources), we consider it imperative that users and applications should (if desired) retain the prerogative in determining the databases which are to be interrogated to find the information they need. Data receivers in a context interchange system therefore have the flexibility of querying a *federated schema* (providing transparent access to some predefined set of databases) *or* querying the data sources directly. The two different scenarios are described below.

   As in the case of Sheth and Larson's federated database systems [33], a federated schema may be shared by multiple data receivers. The federated schema can also be tailored to the needs of individual groups of users through the definition of *external views*. We make no commitment to representing or reconciling semantic heterogeneity in these schemas: schemas exists merely as descriptors for database structure and we *do not* define how semantic conflicts are to be resolved. The shared schema exists as *syntactic glue* which allow data elements in disparate databases to be compared (e.g., stock price in NYSE database versus stock price in Tokyo stock exchange database) but no attempt should be made at defining how semantic disparities should be reconciled. Instead, semantic reconciliation is performed by the *context mediator* at the time a query is formulated against the shared schema. A detailed discussion of how this is accomplished can be found later in this section.

   Direct access to data sources in the context interchange system is facilitated by making available the *export schema* of each data source for browsing. Once again, this export schema corresponds directly to that in the five-level schema architecture [33], and depicts only data structure without any attempt at defining the data semantics. In order to query data sources directly, trivial extensions to the query language Context-SQL [29] allowing direct naming of databases would also be necessary. Once again, the context mediator provides for semantic reconciliation by identifying the conflict and initiating the necessary data conversion. This approach therefore provides the flexibility of multidatabase language systems without requiring users to have intimate knowledge of underlying semantics of disparate databases.

3. *Explicit receiver heterogeneity: representation of and access to user semantics.*

   Data receivers in the context interchange framework are accorded with the same degree of autonomy and sovereignty as data sources. In other words, receivers are allowed to define

their interpretation of data in an import context. As is the case with export contexts, commonality among groups of users can be captured in a supra-context (see Figure 4).

Compared with export contexts, the import context of a user may be more "dynamic": i.e., the needs of a single user can change rapidly over time or for a particular occasion; thus, different assumptions for interpreting data may be warranted. This can be accomplished with Context-SQL [29] which allows existing context definitions to be augmented or overwritten in a query. For example, a stock broker in US may choose to view stock prices in Tokyo as reported in Yen. The default context definition (which states that prices are to be reported in USD) can be overwritten in the query with the **incontext** clause:

select stkCode
from Tokyo-stock-exchange.trade-rel
where stkPrice > 1000
incontext stkPrice.currency = "Yen";

This query therefore requests for stock codes of those stocks priced above 1000 Yen.

4. *Automatic recognition and resolution (e.g., conversion) of semantic conflicts.*

   One of the goals of the context interchange architecture is to promote *context transparency*: i.e., data receivers should be able to retrieve data from multiple sources situated in different contexts without having to worry about how conflicts can be resolved. Unlike classical integration approaches, semantic heterogeneity are not resolved a priori. Instead, detection and resolution of semantic conflicts take place only *on demand*: i.e., in response to a query. This form of *lazy evaluation* provides data receivers with greater flexibility in defining their import context (i.e., the meaning of data they expect, and even conversion functions which are to be applied to resolve conflicts) and facilitates more efficient query evaluation.

   Conflicts between two attributes can be easily detected by virtue of the fact that semantic domains corresponding to both attributes can be traced to some common concept in the shared ontology. Consider the query issued by the US stock broker to the Tokyo stock exchange database. The attribute `stkPrice` in the database is associated with a semantic domain "monetary amount <Currency=Yen, Unit=thousands>" which can be traced to a concept in the shared ontology called "monetary amount". The import context of the broker would have defined some assumptions pertaining to monetary amounts, e.g., "monetary amount <Currency=USD>". By comparing semantic domains rooted in a common concept in the shared ontology, discrepancies can be readily identified. The conversion functions for effecting context conversions can similarly be defined as part of the shared ontology. In our example above, the concept "monetary amount" will have a function which, given an exchange rate, converts an amount from one currency to another, and a different conversion function for achieving translation between units. A more in-depth discussion of conversion functions in a context interchange environment can be found in [30].

5. *Conversion considerations in query optimization.*

   As mentioned earlier, data contexts allow disparate data semantics to be represented independently of the conflict resolution mechanism, thus facilitating more intelligent query processing. (An example of which is presented earlier in this section.) Hence, apart from conflict detection and reconciliation, the context mediator can also play an important role as a query optimizer which takes into account the vastly different processing costs depending on the sequence of context conversions chosen. The problem here is sufficiently different from query optimization in distributed databases and warrants further research (see [19] for an overview of the pertinent issues). As we have explained in Section 3, the performance gains from optimization is likely to be more pronounced in large-scale systems. By making data semantics explicitly available, the context interchange architecture establishes a good framework within which the optimizer can function.

6. *Extensive scalability.*

   The efficacy with which the context interchange architecture can be scaled-up is a primary concern and forms an important motivation for our work. Our claim that the context interchange approach provides a more scalable alternative (to most of the other strategies) is perhaps best supported by contrasting it with the others.

   In integration approaches relying on shared schemas (e.g., federated database systems), designing a shared schema involving n systems requires the resolution of an order of at least n^2 conflicts (assuming that each data source brings only one conflict to the federated schema). In many instances, the growth is even exponential since each data source has the potential of adding many more conflicts than n. This problem is also evident in the more recent *intelligent mediation* approaches proposed in [26] and [27] even though these do not rely on shared schemas. We contend that these integration approaches are impractical when applied to large-scale systems simply because the complexity of the task is way beyond what is reasonably manageable.

   The context interchange approach takes on greater similarity with KR multidatabases in this aspect of system scalability. Hence, instead of comparing and merging individual databases one with another, we advocate that individual databases need only to define semantic domains corresponding to each data element, and relating these domains to concepts in the shared ontology (and if necessary, augmenting the latter). This process is also facilitated by the ease with which context definitions can be shared (e.g., sharing of semantic domains between disparate data elements in a database, and sharing of supra-contexts across system boundaries). This ability to share semantic specifications means that adding a new system or data element can become incrementally easier by reusing specifications which are defined earlier.

7. *Incremental system evolution in response to domain evolution.*

   The context interchange architecture is predicated on the declarative representation of data semantics. Hence, when data semantics changes, it is merely necessary to change the

corresponding declaration. Sweeping changes in data semantics (e.g., all French companies change from reporting in French francs to ECUs) may require changes to one or more micro-contexts. However, in some instances, a single change in the semantic domain or supra-context may be sufficient. Whereas in other approaches such changes would require explicit changes to code, here all changes are declarative. Moreover, the context mediator automatically selects the conversion as appropriate (see point #4).

8. *Sharable and reusable semantic knowledge.*

   Given that tremendous amount of work in system integration goes into understanding and documenting the underlying domain (i.e., defining the shared ontology), semantic knowledge gained from these endeavors should be sharable and reusable. This notion of constructing a library of reusable ontologies has been a major endeavor of the Knowledge Sharing Effort [24,25] as well as the Cyc Project [15,16][6]. The context interchange approach is well poised to both take advantage of this endeavor and to contribute to it by being a participant in the construction of ontologies for reuse. Given sufficient time, these endeavors can result in sufficient critical mass which allow interoperable systems to be constructed with minimal effort by reusing and combining libraries of shared ontologies.

4.3 Comparison with Existing Integration Strategies

We provide a comparison of the context interchange approach with other dominant integration approaches by way of Table 1. We should point out that the assessment of each strategy is based on adherence to the *spirit* of the integration approach and thus are not sweeping statements about particular implementations. This is inevitable since there is a wide spectrum of implementations in each category of systems, not of all which are well described in the literature.

[6]The Cyc Project takes on a slightly different flavor by advocating the construction of a single global ontology representing the "consensus reality" of a well-informed adult.

Table 1: Comparison between the Context Interchange Framework and other existing integration strategies

Features	Global Schema/Federated Db Systems (e g , Multibase)	Multidatabase Language Systems (e.g., MRDSM)	Object-oriented Systems (e.g., Pegasus)	Knowledge Rep Systems (e.g., Carnot)
1. Explicit representation of and access to underlying data semantics. Semantics of data in underlying databases are explicitly (declaratively) represented and accessible by users.	No support.	No support. The export schema available to users might provide some hint to the semantics of underlying data This however is inadequate since semantic assumptions are frequently not obvious from the database schema.	Limited support. The object hierarchy provides some support for inferring the relationships of data in different databases. However, a substantial amount of the data semantics are embedded in (the code of) methods which provide for the translation from one representation to another.	Good support. This conceivably, is possible by examining the mappings (called articulation axioms in Carnot) between individual databases and the global ontology.
2. Explicit source selection subjected to users' preferences. Users retain the prerogative in determining the databases which are to be interrogated to find the desired information.	No support. Access is restricted to via pre-established (federated) schemas and ad hoc access to component databases is not permitted	Good support. This is a feature of multidatabase language systems. As a result, the export schema of each component database is generally available to users for perusal and to facilitate query formulation.	Implementation dependent. Pegasus for instance allow access to component databases directly. This however is not a native feature of the object-oriented integration strategy.	No support. Existing KR systems subscribe strongly to the need for a global semantic model and do not normally provide direct access to component databases.
3. Explicit receiver heterogeneity: representation of and access to user semantics. Receivers state their assumptions as to how data should be represented or interpreted instead of being coerced into a single unified world view.	Limited support. Via the definition of external schemas, which really only provides for different ways of structuring the data.	Limited support. By having users define the conversions needed to comply with the assumptions	Limited support. Possibly by adding an additional layer of schema objects and writing the code for realizing the conversion, or redefining the shared schema altogether.	Limited support. A possible strategy for realizing this is again to define external views on the global semantic model and having users write conversion functions to do the necessary conversion.
4. Automatic recognition and resolution (e.g., conversion) of semantic conflicts. The system recognizes semantic conflicts between component databases and users issuing the query, and performs the necessary resolution. This may entail converting from one representation to another, or simply annotating the answers to signal semantic incompatibility when no conversion methods are known.	No support. Resolutions are established by the system administrator, a priori, in the underlying shared schemas. This method works well when shared schema exists but breaks down in its absence.	No support. Users are left to their own devices in detecting semantic conflicts and defining the steps needed to convert from one semantic representation to another.	No support. This is the same as in federated database systems	Limited support. The focus on semantic heterogeneity has been limited and largely involves mapping from heterogeneous representations into a common interpretation in the global ontology.
5. Conversion considerations in query optimization. Increase system throughput by considering how a query plan can be restructured to take into account the cost of converting from one semantic representation to another.	No support. Always require component databases to map to a canonical representation prior to any further conversion.	Limited support. Users need to determine when conversion should take place by specifying these in the query. It is not clear what latitude exists for query optimization.	Limited support. Methods encapsulated in subtypes provide for conversion from disparate representations in component databases to a unified representation. Procedural definition of data semantics however cannot support the inferences needed for query optimization.	Limited support. The same comment for object-oriented multidatabase systems apply here.

Table 1: (cont'd)

Features	Global Schema/Federated Db Systems (e g., Multibase)	Multidatabase Language Systems (e g., MRDSM)	Object-oriented Systems (e.g., Pegasus)	Knowledge Rep Systems (e.g., Carnot)
6. Extensive scalability. Efficacy of the integration strategy when applied to large-scale systems.	Limited support. Designing a shared schema involving n systems entails resolving $O(n^2)$ conflicting representations. (This may be mitigated by having hierarchies of federated systems, in which a federated system constitutes one of the component database in another.) The system is also likely to suffer from performance degradation since larger systems tend to be more diverse, making (the lack of) conversion-based query optimization more critical.	Limited support. Large-scale system implementation appears viable since each component database system is only loosely coupled to the network. We claim however that these systems will not operate effectively on a large-scale due to the lack of semantic explication in these systems that would create difficulties for users in understanding conflicts in data semantics across systems.	Limited support. The comment for federated database systems applies here.	Reasonably good support. Large-scale system implementation is simplified by allowing individual database systems to be added to the system via the articulation axioms which bind the given database to the global ontology: i.e., this requires only $O(n)$ sets of conflicts. The performance of KR systems are however likely to degrade in large-scale implementations as a result of the lack of conversion-based query optimization.
7. Incremental system evolution in response to domain evolution. Changes to data semantics (in underlying component databases) result in incremental changes to the system.	This is irrelevant here since little if any of the underlying data semantics is ever captured.	The same comment for federated database systems applies here.	Limited support. The conflict resolution strategy of circumventing heterogeneity by encapsulating these in a common supertype may be a bug here since changes entail modifications to the underlying methods.	Good support. Once again, this is due to the fact that each component database is merged independently in the shared ontology using a set of articulation axioms. Changes in underlying semantics simply entails changing the relevant axioms.
8. Sharable and reusable semantic knowledge. Promotes sharing and reuse of knowledge (of data semantics) needed to perform the integration. (Supports large-scale system implementations, but emphasizes the task of knowledge acquisition.)	No support.	No support.	Limited support through the use of complex data types and abstraction hierarchy.	Relatively good support, since the knowledge of the underlying domain is being captured in a global semantic model.

5 Conclusion

We have presented a new approach to interoperable database systems based on the notion of context interchange. Unlike existing integration strategies which focus on resolving schematic and semantic conflicts, we have suggested that conflict resolution should be secondary to the explicit representation of disparate data semantics. This is achieved in our framework by representing the assumptions underlying the interpretations (attributed to data) in both export and import contexts described with reference to a shared ontology. By decoupling the resolution of heterogeneity from its representation, we now have a more flexible framework for efficient system implementations and graceful evolution especially critical in large-scale dynamic environments.

The richness of this integration model has opened up a wide range of research opportunities. We highlight below some of our current endeavors. First, we recognize that the design of a shared ontology is a complex task and there is a need for a well-defined methodology for accomplishing this [10]. This problem manifests itself in other ways even after we have a stock of ontologies for different domains. For example, we would need to consider how different ontologies can be additively combined and how conflicts should be resolved. Second, the deferment of conflict resolution to the time when a query is submitted (as opposed to resolving this a priori in some shared schema) presents great opportunities for query optimization. Since transformation of data from one context to another can conceivably be a very expensive process, the gains from a carefully crafted query optimizer can be immense. This, coupled with other issues pertaining to multidatabase query optimization, constitute a difficult problem [19]. Another challenge lies with the design of a suitable language for querying both the intensional and extensional components of the system. Context-SQL provides an excellent vehicle for users who might be interested in modifying their import contexts dynamically as queries are formulated. More sophisticated languages and interactions are needed to support users in querying the ontology and in identifying the information resources needed.

References

[1] Ahmed, R., Smedt, P. D., Du, W., Kent, W., Ketabchi, M. A., Litwin, W. A., Raffi, A., and Shan, M.-C. The Pegasus heterogeneous multidatabase system. *IEEE Computer 24*, 12 (1991), pp. 19–27.

[2] Arens, Y., Chee, C. Y., Hsu, C.-N., and Knoblock, C. A. Retrieving and integrating data from multiple information sources. Tech. Rep. ISI-RR-93-308, USC/Information Sciences Institute, March 1993. To appear in the International Journal on Intelligent and Cooperative Information Systems.

[3] Arens, Y., and Knoblock, C. A. Planning and reformulating queries for semantically-modeled multidatabase systems. In *Proceedings of the 1st International Conference on Information and Knowledge Management* (1992), pp. 92–101.

[4] Bertino, E., Gagliardi, R., Negri, M., Pelagatti, G., and Sbattella, L. The comandos integration system: an object oriented approach to the interconnection of heterogeneous applications. In *Proceedings of the 2nd International Workshop on Object-Oriented Database Systems* (Sept 1988), K. R. Dittrich, Ed., LNCS 334, Springer-Verlag, pp. 213–218.

[5] Bright, M., Hurson, A., and Pakzad, S. A taxonomy and current issues in multidatabase systems. *IEEE Computer 25*, 3 (1992), pp. 50–60.

[6] Buchmann, A., Özsu, M., Hornick, M., and Georgakopoulos, D. A transaction model for active distributed object systems. In *Database Transaction Models for Advanced Applications* (1992), A. K. Elmagarmid, Ed., Morgan Kaufmann.

[7] Bukhres, O. A., Elmagarmid, A. K., and Mullen, J. G. Object-oriented multidatabases: systems and research overview. In *Proceedings of the 1st International Conference on Information and Knowledge Management* (1992), pp. 27–34.

[8] Collet, C., Huhns, M. N., and Shen, W.-M. Resource integration using a large knowledge base in Carnot. *IEEE Computer 24*, 12 (Dec 1991), pp. 55–63.

[9] Dayal, U., and Hwang, H.-Y. View definition and generalization for database integration in a multidatabase system. *IEEE Software Engineering 10*, 6 (1984), pp. 628–645.

[10] Gruber, T. R. Toward principles for the design of ontologies used for knowledge sharing. Available via ftp from ksl.stanford.edu.

[11] Gruber, T. R. The role of common ontology in achieving sharable, reusable knowledge bases. In *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference* (Cambridge, MA, 1991), J. A. Allen, R. Files, and E. Sandewall, Eds., Morgan Kaufmann, pp. 601–602.

[12] Kim, W., and Seo, J. Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer 24*, 12 (1991), pp. 12–18.

[13] Krishnamurthy, R., Litwin, W., and Kent, W. Language features for interoperability of databases with schematic discrepancies. In *Proceedings of the ACM SIGMOD Conference* (1991), pp. 40–49.

[14] Landers, T., and Rosenberg, R. An overview of Multibase. In *Proceedings 2nd International Symposium for Distributed Databases* (1982), pp. 153–183.

[15] Lenat, D. B. Ontological versus knowledge engineering. *IEEE Transactions on Knowledge and Data Engineering 1*, 1 (March 1989), pp. 84–88.

[16] Lenat, D. B., and Guha, R. *Building large knowledge-based systems: representation and inference in the Cyc project.* Addison-Wesley Publishing Co., Inc., 1989.

[17] Litwin, W., and Abdellatif, A. An overview of the multi-database manipulation language MDSL. *Proceedings of the IEEE 75*, 5 (1987), pp. 621–632.

[18] Litwin, W., Mark, L., and Roussopoulos, N. Interoperability of multiple autonomous databases. *ACM Computing Surveys 22*, 3 (1990), pp. 267–293.

[19] Lu, H., Ooi, B.-C., and Goh, C. H. On global multidatabase query optimization. *ACM SIGMOD Record 20*, 4 (1992), pp. 6–11.

[20] MacGregor, R., and Bates, R. The Loom knowledge representation language. In *Proceedings of the Knowledge-Based Systems Workshop* (1987).

[21] Malone, T. W., and Rockart, J. F. Computers, networks, and the corporation. *Scientific American 265*, 3 (September 1991), pp. 128–136.

[22] Motro, A., and Buneman, P. Constructing superviews. In *Proceedings of the ACM SIGMOD Conference* (1981), pp. 56–64.

[23] Mullen, J. G. Fbase: a federated objectbase system. *International Journal of Computer Systems Science and Engineering 7*, 2 (April 1992), pp. 91–99.

[24] Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., and Swartout, W. R. Enabling technology for knowledge sharing. *AI Magazine 12*, 3 (1991), pp. 36–56.

[25] Patil, R. S., Fikes, R. E., Patel-Schneider, P. F., Mckay, D., Finin, T., Gruber, T., and Neches, R. The DARPA knowledge sharing effort: Progress report. Available via ftp from ksl.stanford.edu.

[26] Qian, X., and Lunt, T. F. Semantic interoperability via intelligent mediation: architecture and query mediation. Tech. Rep. CDRL A004, SRI International, 333 Ravenswood Avenue, Menlo Park CA 94025-3493, Nov 1993.

[27] Raschid, L., Chang, Y., and Dorr, B. J. Query mapping and transformation techniques for problem solving with multiple knowledge bases. In *Proceedings of the 2nd International Conference on Information and Knowledge Management* (1993).

[28] Scheuermann, P., Yu, C., Elmagarmid, A., Garcia-Molina, H., Manola, F., McLeod, D., Rosenthal, A., and Templeton, M. Report on the workshop on heterogeneous database systems. *ACM SIGMOD RECORD 19*, 4 (Dec 1990), pp. 23–31. Held at Northwestern University, Evanston, Illinois, Dec 11–13, 1989. Sponsored by NSF.

[29] Sciore, E., Siegel, M., and Rosenthal, A. Context interchange using meta-attributes. In *Proceedings of the 1st International Conference on Information and Knowledge Management* (1992), pp. 377–386.

[30] Sciore, E., Siegel, M., and Rosenthal, A. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems* (1992). To appear.

[31] Shank, R. C., and Abelson, R. P. *Scripts, Plans, Goals, and Understanding.* Lawrence, Erlbaum, Hillsdale, 1977.

[32] Sheth, A., and Kashyap, V. So far (schematically) yet so near (semantically). In *Proceedings of the IFIP WG2.6 Database Semantics Conference on Interoperable Database Systems (DS-5)* (Lorne, Victoria, Australis, Nov 1992), D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, Eds., North-Holland, pp. 283–312.

[33] Sheth, A. P., and Larson, J. A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys 22*, 3 (1990), pp. 183–236.

[34] Siegel, M., and Madnick, S. Context interchange: sharing the meaning of data. *ACM SIGMOD Record 20*, 4 (1991), pp. 77–78.

[35] Siegel, M., and Madnick, S. A metadata approach to solving semantic conflicts. In *Proceedings of the 17th International Conference on Very Large Data Bases* (1991), pp. 133–145.

[36] Templeton, M., et al. Mermaid — a front end to distributed heterogeneous databases. *Proceedings of the IEEE 75*, 5 (1987), pp. 695–708.

[37] Ventrone, V., and Heiler, S. Semantic heterogeneity as a result of domain evolution. *ACM SIGMOD Record 20*, 4 (Dec 1991), pp. 16–20.

[38] Wang, R., and Madnick, S. The inter-database instance identification problem in integrating autonomous systems. In *Proceedings of the 5th International Conference on Data Engineering* (1989), pp. 46–55.

[39] Wang, R. Y., and Madnick, S. E. A polygen model for heterogeneous database system: the source tagging perspective. In *Proceedings of the 16th Conference on Very Large Data Bases* (1990), pp. 519–538.

[40] Wiederhold, G. Mediators in the architecture of future information systems. *IEEE Computer 25*, 3 (March 1992), pp. 38–49.

[41] Wiederhold, G. Intelligent integration of information. In *Proceedings of the ACM SIGMOD Conference* (May 1993), pp. 434–437.

MIT LIBRARIES
3 9080 00856764 3

3248 037

Date Due

OCT. 3

Lib-26-67