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
Email:{chgoh,smadnick,msiegel}@mit.edu

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. This paper highlights the problem of *receiver heterogeneity*, *scalability*, and *evolution* which have received little attention in the literature, provides an overview of the *Context Interchange* approach to interoperability, illustrates why this is able to better circumvent the problems identified, and forges the connections to other works by suggesting how the context interchange framework differs from other integration approaches in the literature.

1 Introduction

Within the next decade, we will witness an increasing number of organizational forms (e.g., adhocracies and virtual corporations) which are critically dependent on the ability to share information across functional and organizational boundaries [13]. 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" [15], "federated database systems" [20], and "multidatabase systems" [3]. 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.

Despite the large body of research literature on interoperable databases, the bulk of the work has focused on how *schematic* and *semantic* conflicts can be resolved. A key tenet of this paper is that a viable integration strategy must take into account additional issues brought about by large-scale and dynamic systems. For example, in the case of the Integrated Weapons Systems Data Base (IWSDB) [24], more than 50 databases (containing information on technical specifications, design, manufacturing, and operational logistics) have been identified as of 1993 and many more are expected to be added over a period of five decades. This has three major implications: (1) frequent entry and exit of interoperating data sources and receivers renders "frozen" interfaces (e.g., shared schemas) impractical and moreover results in greater diversity of data requirements, (2) scalability of the integration strategy becomes an issue, and (3) the longer life-span of such systems demands that the integration strategy be able to deal with the evolution of the system as a whole as well as the semantics of data contained therein.

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 various strategies for addressing them. Section 3 presents

*This research is supported in part by the grant IRI-90-2189, Advanced Research Projects Agency and USAF/Rome Laboratory under USAF Contract F30602-93-C-0160NSF, the Productivity From Information Technology (PROFIT) project, and the International Financial Services Research Center at MIT. A more detailed report of the same title is available as Sloan School of Management Working Paper #3658-94.

†Funding from the National University of Singapore is gratefully acknowledged.

Database 1

(Based on actual realtime feed from a financial services provider)

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	OTC	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 2

(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 1: Examples of conflicts arising from semantic heterogeneity

the issues pertinent to large-scale and dynamic interoperable systems. 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 progress.

2 Interoperable Database Systems: Current Issues and Strategies

2.1 Schematic and Semantic Heterogeneities

Conflicts arising from *schematic heterogeneity* have been extensively documented in Kim and Seo [8]. 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. *Structural conflicts* come about because the same piece of information may be modeled as a relation name, an attribute name, or a value in a tuple. An excellent discussion of structural conflicts can be found in [9].

Conflicts arising from *semantic heterogeneity* are more interesting and are less well understood [19]. Figure 1 illustrates some examples of semantic conflicts. The reader is encouraged to consider what knowledge is needed to determine how much money the investor (Database 2) 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 1 might associate a different stock code with the same stock (e.g., SAPG.F versus SAP AG). *Measurement conflicts* arise from data being represented in different units or scales: for example, stock prices may be reported in different currencies (e.g., US Dollars 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 1 (e.g., 5\05 on the OTC 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 "Earnings", it is not always clear what interpretation of "Price" and "Earnings" are used. (Interestingly, this information provider does not report the PE-value when "Earnings" are negative.) *Confounding conflicts* may result from having different 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

Interoperation of disparate information sources has received considerable attention in the database research community in the recent years. We provide a brief tour of this literature by highlighting two distinct approaches for achieving logical connectivity [20].

1. In the *tight-coupling* approach, conflicts between multiple database systems are reconciled *a priori* in one or more shared (*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*. In this framework, users are only allowed to interact with one or more federated schemas which mediate access to the underlying component databases. One early example of systems adopting this strategy is Multibase [10]. More recent implementations advocating the adoption of an object-oriented data model as a basis for integration include Pegasus [1], and several others (see [4] for a recent survey).

2. In the *loose-coupling* approach, users interact with constituent databases directly instead of being constrained to querying shared schemas exclusively. Semantic interoperability is accomplished in this framework with the provision of a *multi-database manipulation language*, rich enough so that users may easily devise ways of circumventing conflicts inherent in multiple databases. For example, MRDSM [11] allows users to include in their query, specifications (e.g., conversion rules) which dictate how data can be mapped from one representation to another.

From one perspective, the tradeoff between the tight- and loose-coupling approaches can be understood as that of ease-of-use versus flexibility. By reconciling semantic conflicts up-front and encapsulating these in a shared schema, the tight-coupling approach allows data to be delivered to users without them being ever aware of its disparate origins. Unfortunately, this also leave users with little prerogative in determining which databases should be queried and how data should be transformed. Moreover, changes to the shared schema is likely to be unwieldy. The loose-coupling approach, on the other hand, requires no pre-integration and allows users unrestrained access to all available information sources. Unfortunately, it provides little or no support for identifying semantic conflicts and delegates all responsibilities for conflict resolution to the users. Thus, neither of these approaches constitutes a viable strategy for large-scale interoperable systems. The inherent complexity of schema integration suggests that adoption of the tight-coupling strategy is likely to result in systems which are unresponsive to changes, whether due to changes in user requirements or data sources. The loose-coupling approach on the other hand places too much of a burden on users to understand semantic differences: this is an unrealistic expectation given the large number of data sources currently available and the often subtle semantic differences.

More recently, a new approach for database integration using a *knowledge representation (KR)* approach has been suggested. Two such attempts are the Carnot project [5] which employs the Cyc knowledge base as the underlying knowledge substrate, and the SIMS project [2] which uses Loom (a derivative of KL-ONE) 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 and hence is analogous to global schema systems, except that the richness of the representation language allows not just the schematic details but the underlying semantics to be represented. One advantage of this approach is that data integration can be achieved on a piece-meal basis by "hooking" up each component system independently to the semantic model, thus circumventing the combinatorial complexity inherent in schema integration.

3 Beyond Schematic and Semantic Heterogeneities

In this section, we examine the issues of *receiver heterogeneity*, *scalability*, and *evolution* which have received comparatively little attention in the literature. We posit that these are important considerations in formulating a viable integration strategy for large-scale, dynamically evolving interoperable systems.

3.1 Receiver Heterogeneity

Previous research in interoperable database systems is largely motivated by the constraint that *data sources* (databases, data feeds, or even applications furnishing data) in such a system are autonomous (i.e., sovereign) and any strategy for achieving interoperability must be *non-intrusive* [15]: i.e., interoperability must be accomplished 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 sovereignty of *data receivers* (i.e., users and applications retrieving data from one or more source databases). In actual fact, receivers 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[1]. 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 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

[1]evident in the adoption of *external views* in the ANSI/SPARC DBMS architecture.

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.

Consider the global schema approach as an illustration of why receiver heterogeneity is a real concern. With the global schema approach, disparate data representations are reconciled in a shared schema. For instance, consider the following granularity conflict for student grades reported by two databases [1]: the first database represents student grades using letter grades, and the second represents the same as points in the range of 0 to 100. In the Pegasus system, this integration is accomplished by introducing a supertype which subsumes the two `student` types, and allowing all attributes of the subtypes to be "upward inherited", while providing the necessary conversion functions (in this case, `Map1` and `Map2`) to effect the translation to a "canonical" representation:

```
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;
```

The problem with this approach is that it implicitly assumes that all data receivers are perfectly happy with the canonical representation. If this is not the case, they will then have to further convert the canonical representation into their preferred interpretation. This is not only inefficient (e.g., if the original letter grade representation is what a user wanted) but poses problem when conversion to the canonical form is not lossless (e.g., if the canonical form were "letter grades" and a user had wanted the raw score).

3.2 Scale

A large-scale interoperable database environment (e.g., one with three hundred component databases as opposed to three) presents additional 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. The federated systems approach attempts to mitigate this by having multiple shared schemas (presumably each encompassing a smaller set of data sources). Notwithstanding this, designing a shared schema involving n different systems entails reconciling an order of n^2 possibly conflicting representations. Such activities are time consuming and are aggravated by the idiosyncratic nature of different database schemas which provide little opportunity for abstraction or reuse of previous integration efforts.

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 on 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 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 may lead to performance degradation. 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 the network organization (i.e., when a new component database is added or an old one removed from the system) and changes in semantics (of a component database or receiver).

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 [23] 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* [18], are abstraction mechanisms which allow us to cope with the complexities of life. For example, in the US, a date string such is "12/1/94" is unambiguously interpreted to mean "December 1, 1994" and not "January 12, 1994" (and certainly not "December 1, 2094"). 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.

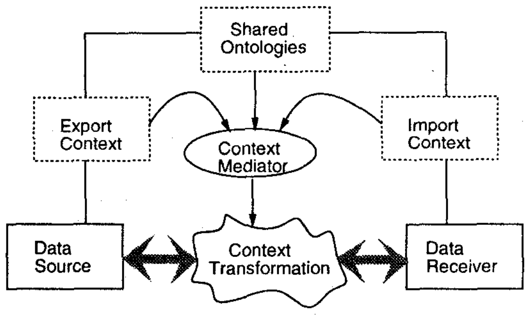

Figure 2: Context mediation in a simple source-receiver system.

4.1 Context Interchange in Source-Receiver Systems

We will first exemplify the key notions underlying the context interchange strategy as depicted in Figure 2 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) [21,22]. To allow data exchange between the data source and data receiver to be meaningful, data 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 collection of *shared ontologies* [6] which constitutes a shared vocabulary for context definition. Intuitively, the shared ontologies are 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 NYS 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 3 illustrates what the architecture of such a system might look like.

This architecture differs from the source-receiver framework presented in Figure 2 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 a 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.

4.3 Advantages Over Existing Integration Strategies

We posit that a distinguishing feature of the context interchange approach is its focus on the *representation* of disparate data semantics as opposed to the *a priori resolution* of semantic conflicts (a characteristic of tightly-coupled systems). This allows conflict resolution to be (1) automated by a context mediator, and (2) deferred to the time when data is actually retrieved. We claim that this brings about a number of features which are novel and advantageous over existing integration strategies.

Feature 1: *Explicit representation of and access to semantics in data source*

As we have illustrated via the example in Section 3.1, the tight-coupling approach achieves integration by encapsulating the semantics of data from disparate sources in their mappings to the shared schema. One disadvantage of this is that the "original" meaning of the data is often unavailable to a user. For example, the question "how are student scores reported in the source databases?" cannot be easily ascertained. Embedding data semantics implicitly in this way poses a number of problems. First, as was noted earlier, this assumes a commitment to a canonical representation. Second, because data semantics are not readily available, conflict resolution can only be done statically as opposed to being performed on demand (see Feature 4). Third, since data semantics are defined with respect to some shared schema, knowledge of data semantics cannot be reused readily. For example, the mapping function (e.g., Map1) created for one federated schema will only be useful if the same exact canonical representation is chosen for a second federated schema involving this database. Finally, this approach suggests that the shared schema needs to be constantly modified in response to changes in the system (as new systems come on-line and existing ones are retired) as well as changes in underlying data semantics (e.g., when one system changes its representation of student grades). The last two points will be further elaborated in the discussion under Feature 6.

Feature 2: *Explicit representation of and access to semantics underlying heterogeneous receivers*

Another important distinction of the context interchange approach lies with its commitment to *receiver heterogeneity*. We recognize that both data sources and data receivers are autonomous and sovereign agents and hence any strategy for achieving interoperability must be *non-intrusive* with respect to *both*. Hence, instead of requiring data receivers to commit to a canonical representation (as in classical integration strategies), we provide data receivers with the flexibility of capturing the "routine" interpretation of data in an *import* context. Unlike the use of external views, the import context can be altered by users in the query process and is not immutable. For example, the import context of the aforementioned stock

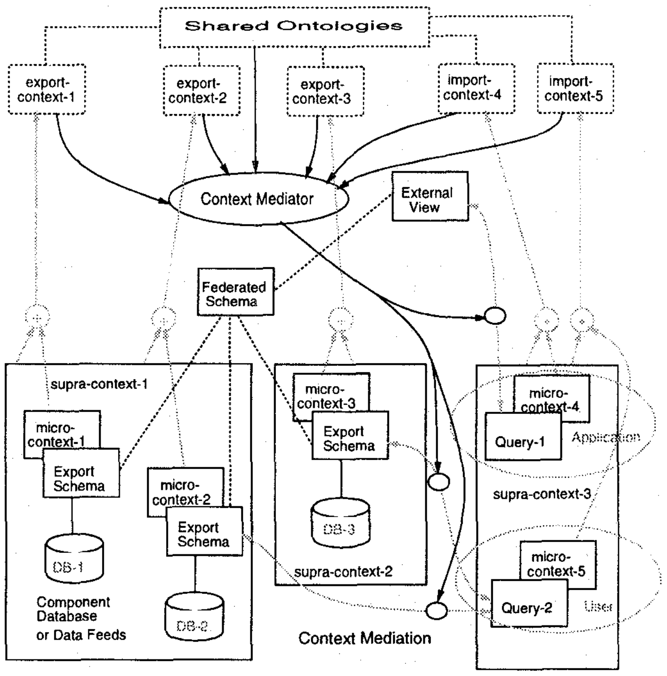

Figure 3: Context interchange with multiple data sources and data receivers.

broker can alter the import context of his query to report in US Dollars using an extension of SQL (called *Context-SQL* [16]:

select stkCode
from NYS
where stkPrice > 10
incontext stkPrice.currency = "US Dollars";

Such a query might be useful if the stockbroker is interested in seeing the raw data instead of the processed data (which might introduce errors or unwarranted assumptions). More generally, this ability to alter one's assumptions about how data should be interpreted is often useful in response for complex decision processes.

Feature 3: *Support for both tight- and loose-coupling integration strategies*

Given that users frequently differ in their judgment and preferences for data, the use of shared schemas is not always appropriate since this presumes that users' requirements can be "frozen" in a schema. The truth is, users rarely commit to a single conception 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. These observations suggest that users need to retain the prerogative in selecting data sources from which data are to be retrieved. On the other hand, shared schemas are sometimes advantageous in that it helps achieve resource transparency. The context interchange strategy resolves this dilemma by supporting both the tight- and loose-coupling approaches: hence, a query can be formulated against a shared schema or against multiple component schemas.

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

A principal goal of the context interchange strategy is to promote *context transparency*: i.e., data receivers are able to retrieve data from multiple data sources situated in different contexts without having to worry about how conflicts are resolved. Through the explicit representation of data semantics in the export and import contexts, detection and resolution of semantic conflicts can be automated by a context mediator which supervises *context mediation* 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. A more in-depth discussion on how data conversions take place is found in [17].

Feature 5: *Improved query optimization with conversion considerations*

As a natural consequence of "lazy evaluation" of data conflicts, it becomes feasible to consider how a query might be optimized by taking into consideration different processing costs resulting from different sequencing of conversion functions. Following the stock example in Section 4.1, the query

select stkCode
from NYS **where** stkPrice > 1000;

issued by a Tokyo trader may be evaluated in two ways[2]: (1) convert all prices in the NYS database to Yen before performing the selection; or (2) convert 1000 Yen to its equivalent in US Dollars such that the query can be directly evaluated by the DBMS at NYS The latter query plan is obviously far more superior. (Once again, a simple example is chosen here for pedagogical reason; more complex conversion sequences can be easily visualized.) The point is that performance gains through such optimization is likely to be pronounced.

Feature 6: *Improved scalability with loosely-coupled system components and reusable semantic knowledge*

While both shared schemas and shared ontologies play similar roles in providing a frame of reference for defining data semantics, the latter represents a much better approach for a number of reasons. First, shared schemas are often designed around the idiosyncratic data requirements of specific tasks. The shared schema designed for a particular integration tasks will often not be reusable in a different context. Shared ontologies, on the other hand, are intended to be domain-dependent but context-independent: e.g., an ontology for the world of "stock trades" is a good basis for understanding the meanings of trade data in a large variety of systems. Second, while comprehensive shared ontologies might be more difficult to construct, being sharable and reusable means that exerting a greater level of effort in its design is probably justifiable. This notion is consistent with the endeavor of the Knowledge Sharing Effort [6,14], where it is envisioned that opportunities exist for establishing a library of shared ontologies which will allow existing systems to interoperate in a meaningful way. Third, by capturing the semantics of data in local data contexts in a dispersed manner (as opposed to doing so centrally in the definition of a shared schema), component systems remain only loosely-coupled to one another, suggesting that system evolution is less of a problem. Finally, the use of the "context" mechanism allows semantic definitions to be shared: for example, the reporting of calendar dates in the US follows a MM-DD-YY format by convention. By capturing and representing shared assumptions such as these, (1) adding a new system or data element becomes easier since they can simply inherit previous specifications (say, in the supra-context), and (2) changes are easily accommodated when they occur since only the assertion in the relevant supra-context needs to be changed.

[2]Assuming that stock prices in the NYS database is represented in US Dollars and the Tokyo trader is expecting prices in Yen

Table 1 presents a comparison of the context interchange approach with other existing dominant strategies along with the features we have identified above. 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 all of which are well described in the literature.

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 suite of shared ontologies. 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. 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 [7]. 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 new challenges for identifying new query optimization strategies which are qualitatively different from classical distributed query optimization [12]. Another challenge lies with the design of a suitable language for representing and querying data and knowledge embodied in the system. Context-SQL [16] 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.

A prototype of a context interchange system is currently being implemented in pursuit of these issues. The current implementation uses Loom for the representation of ontological and contextual knowledge, and includes a conversion-based query optimizer. This prototype is currently being applied to the integration of ten (real-world) financial databases which provide a rich array of test cases.

Table 1: Features of the Context Interchange strategy and how others measure up.

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. Use of export schemas are inadequate since semantic assumptions are frequently not obvious from the database schemas.	Limited support. 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. Through 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 to use	No support. Access is restricted to via pre-established (federated) schemas and ad hoc access to component databases is not permitted.	Good support. 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 although it is not clear how conflicts are resolved in this case.	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.	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. (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. 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
6 Improved scalability with loosely coupled system components and reusable semantic knowledge.	No support. Shared schemas provide little opportunity for sharing and reuse; schema integration and subsequent evolution is costly.	Limited support. Lack of support for conflict detection and resolution constitute a bottleneck for usability	Limited support The comment for federated database systems applies here.	Reasonably good support Lack of attention to receiver heterogeneity creates problem when user requirements are diverse.

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., 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] Gruber, T. R. Toward principles for the design of ontologies used for knowledge sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation*, N. Guarino and R. Poli, Eds. Kluwer Academic, 1994. To appear.

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

[9] 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.

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

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

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

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

[14] 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.

[15] 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.

[16] 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.

[17] Sciore, E., Siegel, M., and Rosenthal, A. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems 19*, 2 (June 1994), pp. 254–290.

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

[19] 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.

[20] 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.

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

[22] 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.

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

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