Facilitating the Interoperability of Heterogeneous Databases with a Meta Object Facility

J. Tan, A. Zaslavsky
Sch. of Computer Science
& Software Engineering
Monash University, Australia
jtan@csse.monash.edu.au
azaslavs@csse.monash.edu.au

C. Ewald
Queensland Transport
Australia
catherine.a.ewald@transport.qld.gov.au

A. Bond
Distributed Systems
& Technology Centre
Australia
bond@dstc.edu.au

ABSTRACT

Metadata management is undoubtedly crucial in facilitating interoperability among heterogeneous databases. There is an obvious need for a strong metadata facility that captures important structure as well as semantics. This paper presents our approach to doing this using the Object Management Group's Meta Object Facility (OMG MOF). Its reflective feature and multilevel approach to metativity is very much suited to resolving conflicts among heterogeneous databases. While metadata management and reflection are old concepts, the authors feel that their potential has yet to be fully realized. We provide an example of how the OMG MOF may be applied.

Keywords: global query system, metadata, middleware, multidatabases, schema integration.

1. INTRODUCTION

To make a number of autonomous, heterogeneous database systems interoperable, one of the major difficulties is how to provide global query access. There must be more to it than providing individual interfaces to the databases [15, 31]. Such a solution is generally possible but does not provide enough database management system (DBMS) functionality.

Older solutions in literature combine the structures or *schemas* of component databases to form a super-structure through a *database schema integration* process. A database schema refers to the model that defines the database. This model may be conceptual or may model the actual implementation. However, in schema integration solutions, the model in mind must be conceptual to capture sufficient semantics. Schema integration is the process of constructing an integrated schema to form a virtually cohesive system [1] which is often referred to as a *multidatabase system* [16, 25]. There are three major variants. *Global schema systems* are based on a single *global schema* that combines the schemas of all component databases. This has been proven to be difficult given the complexity of resolving differences and even conflicts among the schemas to combine [18]. *Federated database systems* consist of several integrated *import schemas* constructed from some, not all, *export schemas* defined over individual component databases. This simplifies the integration process since fewer component databases are actually combined for each import schema. *Multidatabase language systems* do not require schema integration. Instead, an access language over the component databases is provided for flexible access. On the other hand, it may be unfair to expect users to deal with the complexity involved in such a method of access [24] since this leaves it up to the users to formulate how to access the component databases individually prior to combining these individual results.

A common denominator among these variations is the process of mapping between the global system and component databases. In global schema and federated systems, mapping expressions are used to define global or import schemas over component database schemas. In multidatabase language systems, mapping functions are invoked when access commands are invoked. Strong metadata support is needed to facilitate the accessibility to individual databases through those mappings. Furthermore, strong metadata support is needed to resolve the semantic issues that also arise over and beyond schematic issues.

Whereas schema integration has been frequently visited in literature, we attack the problem at its core: the lack of strong metadata support, which is a proven requirement to enable interoperability among heterogeneous database systems. With sufficient metadata management features, it becomes more possible to tackle the structural, infrastructural and semantic obstacles.

2. RELATED WORKS

Several approaches are known to facilitate the interoperability of heterogeneous, autonomous databases on a network. We discuss a few of them here:

1. *A global query system.* Using online intelligence and assistance supported by a *metadatabase* facility, users are allowed to browse through the metadata of the databases to formulate global queries [4].

2. *Views.* The abstraction that brings the heterogenous databases together in a single system may also be

Appeared in the Proceedings of the Philippine Computing Science Congress (PCSC 2000), Manila, Philippines, November 29 to December 2, 2000.

through views defined on top of them [1, 2, 19]. The multidatabase effectively appears as an "ordinary" database, and views are widely accepted means of providing access to actual data. It is also a convenient method of restricting access to the multidatabase. Some definition language must exist to define the mappings between the views and the actual data sources [7, 17, 27]. Defining such views is non-trivial, however, due to the heterogeneity among the databases.

3. *Stronger semantics.* Semantic heterogeneity is probably the biggest concern in bridging the heterogeneity among databases, and requires good representation of semantics to resolve [3, 11, 24, 33]. Specialization or generalization hierarchies, ontologies or using a thesaurus can help, as can using a semantically rich common data model.

4. *Virtual attributes and domain mapping.* Since the databases are heterogeneous, straightforward mapping is probably not possible. Virtual attributes may be defined to range over several domains. Partial values and *maybe tuples* may also be defined [8].

5. *Higher order expressions and higher order views* [17]. These may range over data as well as metadata to resolve discrepancies such as attribute-value conflicts. Higher order views can range over the instances as well as the metadata of participating databases.

6. *Middleware solutions.* Middleware backbones enable interoperability by providing the basic requirements such as the communication backbone [10, 13]. Object-oriented middleware exploits the rich object-oriented paradigm and provides strong abstraction mechanisms. Most middleware products are also portable across operating system and programming platforms and make it easier to expand the system with subsequent applications and services.

3. APPLYING THE OMG MOF

Several approaches attack the problems of interoperability at the metadata level [6, 11, 12, 14, 20, 21, 22, 26, 29] not only for databases but also in enterprise modeling and data warehousing. We follow through on the same strategy for our approach using a good metadata facility. We also use good language facilities for expressing metadata as well as repository features for reuse and sharing.

3.1 The OMG MOF

In the Object Management Architecture (OMA) reference model of the Object Management Group (OMG) the Common Object Request Broker Architecture (CORBA) is the communication backbone that links participating systems via Object Request Brokers (ORBs). It is supported by several services and facilities such as the Meta Object Facility (MOF).

The abstract model of the MOF may be viewed as having four levels, shown in Table 1. At the lowest level, M0, the universe of information for some domain consists of *entities* and the *relationships* they may have with one another. This is similar to how entities and relationships are understood in the Entity-Relationship data model, which is commonly used for conceptual database design. At the M1 level, entities are classified into *types*, which may be involved in *type relations* with other types and may define *type schemas* using some *type language* and *type system*. This is similar to the concept of entity types, relationship types and database schemas in the ER data model. The M2 level classifies types into *meta-types*, which may be involved in *meta-relations* and, using a *meta-type system*, define *meta-schemas*. We can simply refer to this level as the meta-meta level. The ER data model would not have an equivalent meta-meta level since it can only define metadata. On the other hand, the MOF may, at the meta-meta level, define the metamodel of data models. For instance, the OMG common warehouse interchange standard specifies metamodel packages for database schema interchange [6]. These metamodels are used to associate similar concepts between data models, e.g., entity types are implemented as relations, the concept of attributes in both data models are very similar, and so on. At the topmost level, M3, the MOF model defines the language to define metamodels. The MOF model is also used to define itself, which exhibits the *reflective* feature of the MOF.

Table 1: Levels of the MOF abstract model.

LEVEL	CONTENTS
M3: the MOF model	defines metamodels
M2: metamodels, meta-meta information	meta-types, meta-relations and meta-schemas
M1: meta-information	types, type relations and type schemas
M0: information	entities and relationships, grouped by domain

The MOF defines meta objects at the M2 level. However, the MOF standard also specifies the means by which M1 objects maybe instantated from M2 definitions. For example, the dMOF package from the Distributed Systems and Technology Centre automatically generates factory objects to perform this instantiation [9]. Also, unlike UML, MOF meta objects may be implemented in a system [9]. This ability to associate actual data with metadata is crucial. Many issues of *schematic discrepancy* [13, 17, 28, 30] can be resolved if data and metadata can be made to coexist in the same plane.

MOF models may be expressed using the Unified Modeling Language (UML) or the Meta Object Definition Language (MODL) [5, 6]. MODL definitions may be converted into CORBA Interface Definition Language (IDL) definitions, which can then be compiled and implemented in the middleware framework as client objects and server objects. Using the UML for conceptual metamodels is a significant advantage given UML's widespread acceptance as a standard OO modelling language. It is also a prime choice for model browsing for the articulation of global queries or views. Model browsing in model-assisted systems [4] promises a more feasible approach than schema integration and is part of the plans for our project as well. We take this further by using the MOF not only to specify the metamodels in UML but we also use the MOF repository features to store and share the metamodels and the metadata generated thereby.

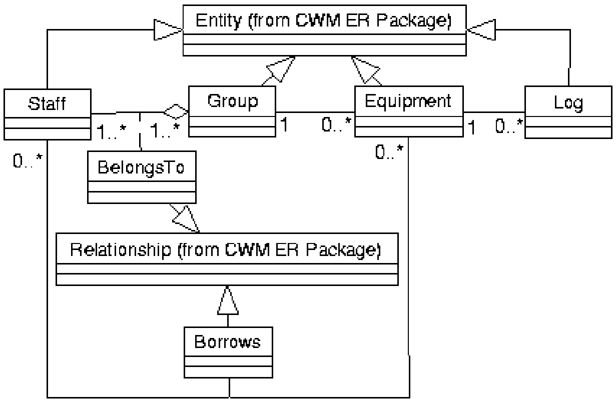

Figure 1: The DLSU Metamodel.

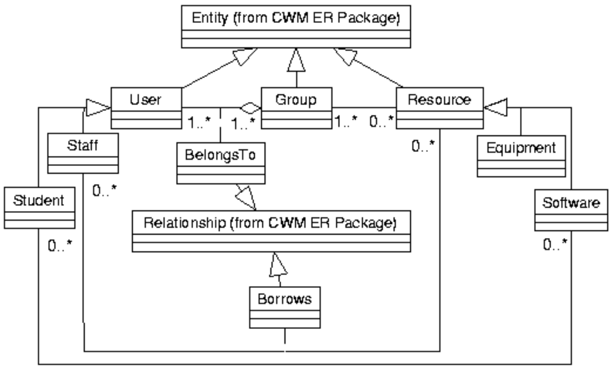

Figure 2: The CSSE Metamodel.

3.2 MOF Metamodels

We now illustrate the use of MOF metamodels to implement several solutions already discussed in literature to enable multidatabase interoperability.

Figures 1 and 2 show the MOF metamodels for DLSU and CSSE in UML notation. The meta objects defined are derived from definitions in the ER metamodel package of OMG Common Warehouse Metamodel (CWM). The CWM defines a standard for metadata interchange for purposes of managing data warehouses, which consist of several databases. For example, the CWM specification includes the metamodel for the ER data model, normally used for high-level conceptual design of databases, and as well as another metamodel for the Relational data model, the basis for most modern relational DBMS products. Database metamodels such as those in Figures 1 and 2 can be analyzed with the ER, relational and other metamdoel packages, to infer similarities and associations between the DLSU and CSSE databases schemas. If an Object-Oriented (OO) database were to enter the picture, and we had an OO metamodel package to define how OO databases are modelled, then it should be possible to draw associations between related data structures given the fact that

- the OO metamodel defines how OO schemas are defined,
- the ER metamodel defines how ER schemas are defined,
- the relational metamodel defines how relational schemas are defined, and
- the MOF model defines how the OO metamodel, the ER metamodel and the relational metamodel were defined.

If these associations and possible equivalence relationships can be verified by an expert system or an expert user, then a federated schema metamodel can be derived as shown in Figure 3. On the other hand, users may instead construct smaller composites after browsing and analyzing pertinent schemas [4]. These composites may be view definitions or query expressions.

The metamodel in Figure 3 glues together the global system metadata with the component database metadata. Supporting mapping/associative constraints are given in Figure 4 using notations based on tuple relational calculus. Each expression defines the mapping constraints between meta objects. The UML normally uses Object Constraints Language (OCL) expressions to specify complex constraints, but we use a relational query notation here since we are dealing with relational data. The relational calculus expressions also take up less space.

`x.A`: the attribute A of tuple object (variable) x.

`x.A = y.B`: equivalence between the two operands.

`x.function()`: returns the result of the `function` method.

`x = y`: equivalence for all attributes between the two operand objects.

3.3 Domain Mapping and Higher Order Expressions

Figure 4 shows how domain mapping becomes possible even when it is not straightforward, as it is not in these cases:

- DLSU.Staff and CSSE.User do not completely share the same domain. Students as users are not modeled in the DLSU database as it is in the CSSE database.
- In the case of CSSE.Resource and DLSU.Equipment, note that the DLSU database does not include software resources in its inventory system.
- There is also discrepancy between DLSU.Log and CSSE.Equipment.event. The latter is a multi-valued attribute that haphazardly describes events concerning equipment, whereas the former is a structured record of similar events.

Likewise, higher order expressions are possible using the MOF metamodel, effectively addressing schematic discrepancies. DLSU.Log.urgent is an attribute that indicates the urgency of a given event. The same information can be extracted from CSSE.Equipment by examining the contents of the CSSE.Equipment.event attribute of a given equipment object. This illustrates a schematic discrepancy known as an attribute-value conflict. But since MOF meta objects in a CORBA system are allowed to exit or be instantiated along with data objects, the discrepancy can be solved.

3.4 Views or Queries?

In our approach, it hardly matters if the resulting system is intended to be a global query system [4] or a multidatabase

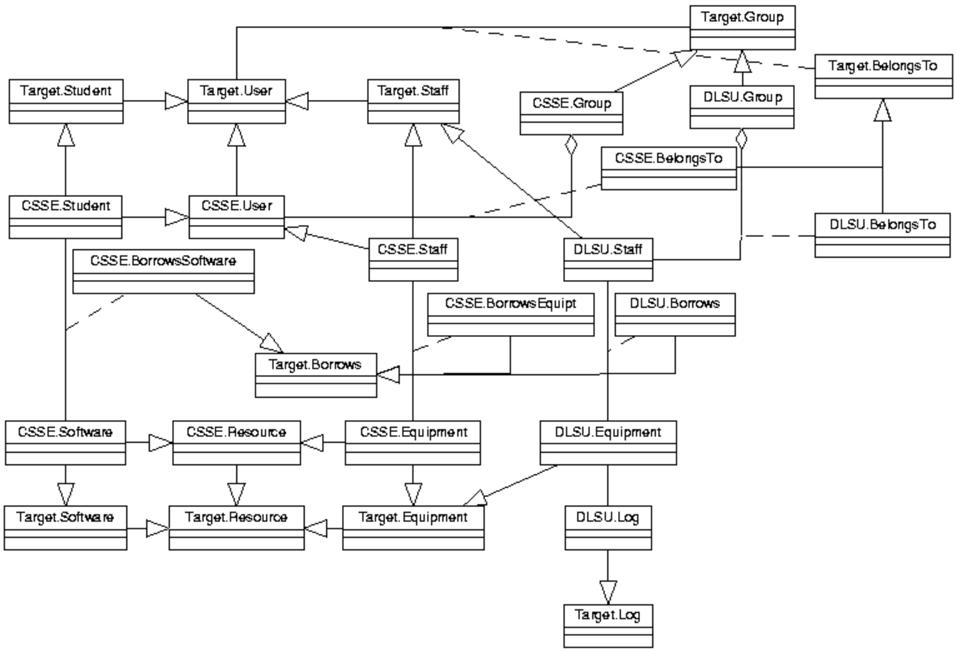

Figure 3: The defined "integrated" metamodel.

Global.Group

$\forall(x)(x \in \mathrm{Global.Group}) \Rightarrow \exists\,(y)\,\exists\,(z)(y \in \mathrm{CSSE.Group} \wedge z \in \mathrm{DLSU.Group} \wedge (x.num{=}y.num \wedge x.name{=}y.name) \vee (x.num{=}z.num \wedge x.name{=}z.name))$

Global.User

$\forall(x)(x \in \mathrm{Global.User}) \Rightarrow \exists(y)\,\exists(z)((y \in \mathrm{CSSE.User} \wedge x{=}y) \vee (z \in \mathrm{DLSU.Staff} \wedge x.uid{=}z.sid \wedge x.uname{=}z.sname \wedge x.utype{=}\text{'staff'} \wedge x.startdate{=}\text{''} \wedge x.programme{=}\text{''} \wedge x.rank{=}\text{''} \wedge x.faculty{=}z.dept))$

Global.BelongsTo

$\forall(x)(x \in \mathrm{Global.BelongsTo}) \Rightarrow \exists(y)\,\exists(z)((y \in \mathrm{CSSE.BelongsTo} \wedge x{=}y) \vee (z \in \mathrm{DLSU.BelongsTo} \wedge x.uid{=}z.sid \wedge x.grpnum{=}z.gnum \wedge x.startofmembership{=}z.datejoined \wedge x.endofmembership{=}z.dateleft))$

Global.Resource

$\forall(x)(x \in \mathrm{Global.Resource}) \Rightarrow \exists(y)\,\exists(z)((y \in \mathrm{CSSE.Resource} \wedge x.rnum{=}y.barcodenum \wedge x.rdesc{=}y.description \wedge x.cost{=}y.cost \wedge x.purchasedate{=}y.purchasedate) \vee (z \in \mathrm{DLSU.Equipment} \wedge x.rnum{=}z.eqnum \wedge x.rdesc{=}z.description \wedge x.cost{=}z.cost \wedge x.purchasedate{=}\text{''}))$

Global.Equipment

$\forall(x)(x \in \mathrm{Global.Equipment}) \Rightarrow \exists(y)\,\exists(z)((y \in \mathrm{CSSE.Equipment} \wedge x.rnum{=}y.barcodenum \wedge x.eqtype{=}y.eqtype \wedge x.brand{=}y.brand \wedge x.model{=}y.model) \vee (z \in \mathrm{DLSU.Equipment} \wedge x.rnum{=}z.eqnum \wedge x.eqtype{=}\text{''} \wedge x.brand{=}z.vendor \wedge x.model{=}\text{''}))$

Global.Software

$\forall(x)(x \in \mathrm{Global.Software}) \Rightarrow \exists(y)\,\exists(z)((y \in \mathrm{CSSE.Equipment} \wedge x{=}y))$

Global.Borrowed

$\forall(x)(x \in \mathrm{Global.Borrowed}) \Rightarrow \exists(w)\,\exists(y)\exists(z)((w \in \mathrm{CSSE.BorrowedEquipment} \wedge x.uid{=}w.uid \wedge x.rnum{=}w.barcodenum \wedge x.borrowingdate{=}w.borrowingdate) \vee (y \in \mathrm{CSSE.BorrowedSoftware} \wedge x.uid{=}y.uid \wedge x.rnum{=}y.barcodenum \wedge x.borrowingdate{=}y.borrowingdate) \vee (z \in \mathrm{DLSU.Borrows} \wedge x.uid{=}z.sid \wedge x.rnum{=}z.eqnum \wedge x.borrowingdate{=}z.dateborrowed))$

Global.Log

$\forall(x)(x \in \mathrm{Global.Log}) \Rightarrow \exists(y)\,\exists(z)((y \in \mathrm{CSSE.Equipment} \wedge x.rnum{=}y.barcodenum \wedge x.urgent{=}y.IsThisUrgent() \wedge x.logdate{=}y.getDate() \wedge x.personresp{=}\text{''} \wedge x.event{=}y.event) \vee (z \in \mathrm{DLSU.Log} \wedge x.rnum{=}z.eqnum \wedge x.urgent{=}z.urgent \wedge x.logdate{=}z.logdate \wedge x.personresp{=}z.personresp \wedge x.event{=}z.event))$

Figure 4: Sample expressions relating meta objects in the meta model.

system based on views [13, 23]. As a matter of fact, multidatabase systems using views is hardly different from using schema integration: literature shows several schema integration techniques that use the view approach [1, 2, 19]. The metamodel definitions in MOF can be used as stored query definitions or as view definitions. In many relational DBMSs, the primary if not exclusive manipulation and definition language is based on the Structured Query Language (SQL). SQL is used to express queries as well as to define views.

Moreover, it should also be possible to use pre-defined metamodels [32] to assist in the articulation of queries [4]. The limitation of this approach is that updates are not generally possible, as the abstraction provided is only a view or query result.

4. CONCLUSION AND FUTURE WORK

This paper presented our approach to enabling the interoperability among heterogeneous databases using the OMG MOF. Using the MOF conventions for metamodeling, we are able to capture metadata at several levels, disabling conflicts arising from schematic discrepancies. We illustrated how the MOF implements some solutions already devised in literature. The approach shown also exhibits flexibility since using metamodels as we illustrated supports a multidatabase system that provides global access whether the access is by defining views or by defining global queries. These metamodels may also be considered as integrated schemas as required by federated database systems or global schema multidatabase systems. Our approach also exploits the CORBA middleware which provides strong semantics and abstraction mechanisms. It is also packaged with other facilities which can extend the usefulness of the MOF meta objects. For example, the CORBA Trader service enables the discovery of objects for use in subsequent applications. Other services we exploit are the concurrency control, transaction and query services.

Further development of our approach involves the completion of our prototype. While the existing MODL-to-IDL compiler provides the means of introducing meta objects into the system, we must also provide a front-end tool for articulating views or global queries based on the metamodels defined. Another future project is to develop automation modules that use AI techniques, ontologies, schema browsing and assisted query/view definition to fully or semi-automatically *incubate* meta schemas.

5. ACKNOWLEDGEMENTS

The work reported in this paper has been funded in part by the Co-operative Research Centre Programme through the Australian Government's Department of the Industry, Science and Resources.

6. REFERENCES

[1] Batini, C., Lenzerini, M., and Navathe, S. B. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys 18*, 4 (December 1986).

[2] Breitbart, Y. Multidatabase interoperability. *SIGMOD Record 19*, 3 (September 1990), 53–60.

[3] Bright, M. W., Hurson, A. R., and Pakzad, S. Automated resolution of semantic heterogeneity in multidatabases. *ACM Trans. on Database Systems 19*, 2 (June 1994), 212–253.

[4] Cheung, W., and Hsu, C. The model-assisted global query system for multiple databases in distributd databases. *ACM Transactions on Information Systems 14*, 4 (1996), 421–470.

[5] Crawley, S., Davis, S., Indulska, J., McBride, S., and Raymond, K. Meta-meta is better-better! In *IFIP WG 6.1 International Working Conference on Distributed Applications and interoperable Systems (DAIS'97)* (1997).

[6] Common warehouse metamodel (CWM) specification. OMG Document ad/2000-01-01, February 2000. URL: http://www.omg.org/cgi-bin/doc?ad/98-09-02.

[7] Deen, S. M., Amin, R. R., and Taylor, M. C. Data integration in distributed databases. *IEEE Trans. on Software Eng. SE-13*, 7 (July 1987), 860–864.

[8] DeMichiel, L. G. Performing operations over mismatched domains. In *Proc. Fifth Int'l Conf. on Data Eng.* (1993), pp. 36–45.

[9] Distributed Systems Technology Centre. *dMOF 1.0 User Guide*. University of Queensland, Australia, 2000.

[10] Dogac, A., Dengi, C., Kilic, E., Ozhan, G., Ozcan, F., Nural, S., Evrendilek, C., Halici, U., Arpinar, B., Koksal, P., Kesim, N., and Mancuhan, S. METU interoperable database system. Tech. rep., Software Research and Development Center, Scientific and Technical Research Council of Turkiye, Middle East Technical University, June 1995. Technical Report 6-1.

[11] Fang, D., Hammer, J., and McLeod, D. The identification and resolution of semantic heterogeneity in multidatabase systems. In *Proc. First Int'l Workshop on Interoperability in Multidatabase Systems* (1991), pp. 136–143.

[12] Georgalas, N. Integrating distributed data over their semantic identity. In *Proceedings of the Association for Information Systems 1997 Americas Conference on Heterogeneous Database Interoperability & Data Warehousing* (Indianapolis, Indiana US, August 15-17 1997).

[13] Haas, L. M., Miller, R. J., Niswonger, B., Roth, M. T., Schwarz, P. M., and Wimmers, E. L. Transforming heterogeneous data with database middleware: Beyond integration. *IEEE Data Engineering Bulletin 22*, 1 (1999), 31–36.

[14] Hsu, C., Ed. *Enterprise integration and modeling: the metadatabase approach.* Kluwer Academic Publishers, 1996.

[15] Hurson, A. R., and Bright, M. W. Object-oriented multidatabase systems. In *Object-oriented multidatabase systems: a solution for advanced applications*, O. A. Bukhres and A. K. Elmagarmid, Eds. Prentice-Hall, Englewood Cliffs, New Jersey, 1996, ch. 1, pp. 1–36.

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

[17] Krishnamurthy, R., Litwin, W., and Kent, W. Interoperability of heterogeneous databases with

schematic discrepancies. In *Proc. First Int'l Workshop on Interoperability in Multidatabase Systems* (1991), pp. 144–151.

[18] Litwin, W., and Abdellatif, A. Multidatabase interoperability. *IEEE Computer 19*, 2 (December 1986), 10–18.

[19] Motro, A. Supverviews: Virtual integration of multiple databases. *IEEE Trans. on Software Eng. SE-13*, 7 (July 1987), 785–798.

[20] Microsoft repository. http://msdn.microsoft.com/repository/, June 2000.

[21] Microsoft repository in data warehousing. http://msdn.microsoft.com/repository/scenarios/dw.asp, June 2000.

[22] Navathe, S., and Savasere, A. A schema integration facility using object-oriented data model. In *Object-oriented multidatabase systems: a solution for advanced applications*, O. A. Bukhres and A. K. Elmagarmid, Eds. Prentice-Hall, Englewood Cliffs, New Jersey, 1996, ch. 1, pp. 105–128.

[23] Nodine, M. H., and Zdonik, S. B. The impact of transaction management on object-oriented multidatabase views. In *Object-oriented multidatabase systems: a solution for advanced applications*, O. A. Bukhres and A. K. Elmagarmid, Eds. Prentice-Hall, Englewood Cliffs, New Jersey, 1996, ch. 1, pp. 57–104.

[24] Ouksel, A., and Ahmed, I. Coordinating knowledge elicitation to support context construction in cooperative information systems. In *Proc. First IFCIS International Conference on Cooperative Information Systems (CoopIS'96)* (June 1996).

[25] Özsu, M. T., and Valduriez, P. *Principles of Distributed Database Systems, 2nd edition.* Prentice-Hall, 1999.

[26] Papazoglou, M. P., Marinos, L., and Bourbakis, N. G. Distributed heterogeneous information systems and schema integration. In *Proc. Int'l Conf. on Databases, Parallel Archtectures, and Their Applications (PARABASE-90)* (1990), pp. 388–397.

[27] Robertson, P. Integrating legacy systems with modern corporate applications. *Communications of the ACM 40*, 5 (May 1997), 39–46.

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

[29] Schwartz, D. G. *Cooperating Heterogeneous Systems.* Kluwer Academic Publishers, 1995.

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

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

[32] Tan, J., Zaslavsky, A., and Bond, A. Meta object approach to database schema integration. In *2nd International Symposium on Distributed Objects and Applications (DOA'00)* (September 1999).

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