Meta Object Approach to Database Schema Integration *

J. Tan, A. Zaslavsky
School of Computer Science and Software Engineering
Monash University
Caulfield East, VIC 3145, Australia
Jefferson.Tan,Arkady.Zaslavsky@csse.monash.edu.au

A. Bond
Distributed Systems Technology Centre
The University of Queensland
Brisbane, QLD 4072, Australia
bond@dstc.edu.au

Abstract

Database schema integration is significant not only in building multidatabase systems but also in data warehousing. Metadata, which define schemas, are normally involved in the surrounding issues. And while many of these issues have been addressed in the past, unresolved issues remain. In this paper, we present an approach that not only uses metadata but also uses meta-meta information to make schema integration more possible. Our solution requires meta object facility that serves not only as a repository but also as a more feasible means of managing meta data. We also advocate the use of such a facility as part of an object-oriented middleware environment that provides an open interface standard and several useful services in distributed object management.

1. Introduction

In enterprises with multiple large databases which were developed or had evolved independent of one another, one of the major difficulties is how to process queries involving several autonomous databases. This is difficult because of the heterogeneity, whereas any purported solution must do more than provide a heap of data sources with individual interfaces [19, 40]. The collective data must be interoperable and integrated. *Interoperability* in this context refers to the ability to collectively manipulate multiple hetreogeneous data sources. This involves *homogenization*, a process which makes heterogeneous databases less or non-heterogeneous [33]. *Integration* refers to the process of combining data into some meaningful composite.

Many of the solutions in literature use the structures of component databases to form a super-structure, normally requiring *database schema integration*. *Schema* refers to the model that defines the database. Interoperability may well focus on schemas since this is where structure and some semantics are defined. *Schema integration* refers to the construction of an integrated schema by combining the schemas of the component databases to form a virtually cohesive system [1] which is often referred to as a *multidatabase system* [20, 33]. There are three major variants. *Global schema systems* are based on a single *global schema* that is constructed from the schemas of all component databases. Such systems are very complex to construct due to differences and conflicts between component databases that are difficult to resolve [2, 23], and the difficulty increases with the number of component databases involved. Complexity is also compounded by variations in the design logic and semantics of the component databases. *Federated database systems* use several integrated schemas based on some of the component database schemas. These smaller composites are known as *import schemas*, and are each designed for some applications or users only, just as views may be used in conventional database systems. Constructing several import schemas would be easier than constructing an all-encompassing global schema. It is common for import schemas to be based on *export schemas* of component databases, which are meant to restrict access to component databases by defining them on only a subset of the component database. *Multidatabase language systems* are very different and generally do not involve schema integration to form a composite schema. A language provides access to the databases on the network. This is a more flexible approach but leaves it up to the language user to formulate the integration logic. A common denominator among these variations is the use of expressions to map between one or more integrated

*Original paper appears in the Proc. of the 2nd International Symposium on Distributed Objects and Applications (DOA'00), Antwerp, Belgium, September 21-23, 2000.

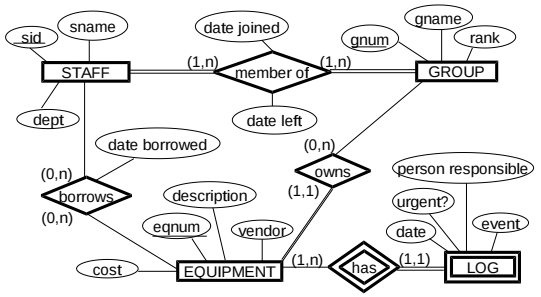

Figure 1. Conceptual schema of DLSU.

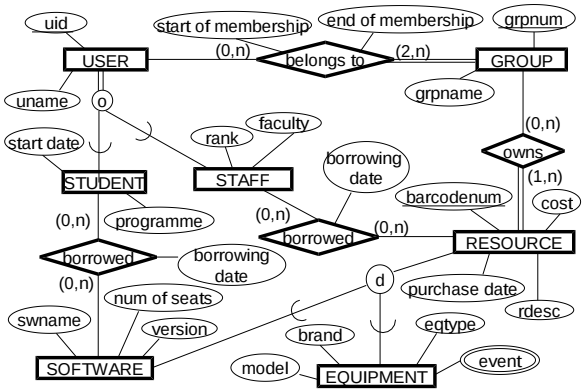

Figure 2. Conceptual schema of CSSE.

schemas and the component database schemas. In multidatabase language systems, these functions are invoked when access commands are invoked. In global schema and federated systems, these exressions are used to define global or import schemas.

Several methods of the schema integration process have been discussed in literature. *Visual tools* provide graphical facilities [9, 41]. *Language-based tools* provide the language and processor to define mappings [11, 23], similar to view definition facilities in relational database systems. *Automated tools* also exist which use reasoning techniques to infer associations that can be made among component databases [4], although interaction with a human user is still necessary to resolve uncertainties, ambiguities or other difficulties. There are many other methods discussed in literature over the last two decades. Research has been active but many issues have yet to be resolved completely.

2. Schema Integration Issues

Two sample databases, whose conceptual schemas appear in Figures 1 and 2, will be used to illustrate schema integration issues. Both schemas, expressed using the Entity-Relationship (ER) data model, are of inventory tracking databases for research organizations. DLSU is that of De La Salle University in Manila while CSSE is that of the School of Computer Science and Software Engineering at Monash University in Melbourne. Both databases are mainly concerned with who borrows what, and when.

Some schema integration issues in multidatabase systems have been identified in literature are illustrated below.

1. *Domain mismatch problems* [2]:
   - *Naming conflicts,* [2, 5, 7]. The entity type `Staff` in DLSU and `Staff` in CSSE are *homonyms* — given the same names but are not equivalent, since CSSE staff entities are distinct from students whereas this distinction is not made for DLSU staff entities. The relationship types `Member-of` in DLSU and `Belongs-to` in CSSE are *synonyms* — given different names but pertaining to the same concept: that of relating people with their groups.
   - *Format differences,* which include problems with data type [2, 5]. Date values in DLSU would be formatted as MM/DD/YYYY, this being the norm in Manila, while dates might be expressed in DD/MM/YYYY format in CSSE.
   - *Scale differences,* such as different measurement units or standards [2, 5, 7]. The DLSU database would express money in terms of Philippine pesos, e.g., `Equipment.cost`, while CSSE would use Australian dollars, e.g., `Resource.cost`.
   - *Domain differences* [7], referring to abstract domain differences rather than data type or implementation differences, e.g., `Staff` entities in DLSU includes students while this is not so in CSSE.
   - *Structural differences* [2, 5, 7], e.g., `Staff` entities in DLSU *must* belong to one group or more while this is not required for `Staff` entities in CSSE. Consider as well the difference between the attribute `Log.event` in DLSU and the multi-valued attribute `Equipment.event` in CSSE, given that `Log` is a different way of modeling the `Equipment.event` entity type.
   - *Missing or conflicting data* in which some entities or attributes exist on one database and does not on others, or values disagree where they should agree [2, 5].

gnum	gname	rank
01	neural networks	4
02	databases	2

Table 1. `Group` **entities in DLSU.**

grpnum	grpname
01	cryptography
02	neural networks

Table 2. `Group` **entities in CSSE.**

For example, assume that Tables 1 and 2 are the relational implementations of `DLSU.Group` and `CSSE.Group`, respectively, and Table 3 would be the integration of Tables 1 and 2. `Null` values occur in Table 3 because group entities from CSSE, appearing in rows 2 and 4, are do not have rank attributes. But `null` values are discouraged as they can be ambiguous. Conflict also occurs as the `grp_num` attribute, which should distinguish between groups, appears in two tuples that seem to pertain to the same group — the 'neural networks' group.

2. *Semantic problems* [14, 22, 34] since expression of semantics and similar information may be implemented differently. It is not easy writing software that can *comprehend* semantics to a degree that allows it to infer associations, although a few projects have been successful in literature [4, 39].

3. *Data model conflicts.* Different data models may be used to design the databases, posing a serious difficulty that must be resolved [5].

4. *Schematic discrepancy or heterogeneity.* This occurs when data, e.g., a value, on one schema is metadata, e.g., an attribute, on another schema [22, 26, 37, 39]. For example, while events occurring in DLSU are logged in detail as `Log` entities, the same events in CSSE are logged in vague terms in `Equipment.event`, a multi-valued attribute. DLSU uses a boolean or yes-no *attribute* `urgent` while it appears as a *value* within attribute `event` in CSSE, i.e., it is an urgent event if the substring 'URGENT!' appears.

grp_num	grp_name	rank
01	neural networks	4
01	cryptography	null
02	databases	2
02	neural networks	null

Table 3. Integrating Tables 1 and 2.

Literature also lists various approaches to solving the problems that must be resolved to successfully perform schema integration.

1. *Views.* Schema integration can be considered as the definition of views on top of several databases [1, 2, 28]. This approach makes it easier to treat the multidatabase as an "ordinary" database, and is a convenient method of restricting access to the multidatabase. This would require some language to define the mappings [11, 22, 36]. Defining such views is non-trivial. Note the problems shown earlier when Table 3 was derived, as with a view, over Tables 1 and 2.

2. *Meta-data comparison/analysis.* This provides an abstract level at which integration can be carried out [14, 15, 27, 43], perhaps using a semantic dictionary, or a type hierarchy approach to incorporate schemas from remote databases with local schemas. Meta-schemas and meta-knowledge can also enable homogenization [34], as can metamodeling, reverse engineering, using a semantic repository [15], metadata facilities and repositories within middleware environments [8, 29].

3. *Stronger semantics.* Resolving semantic heterogeneity requires the representation of semantics [14, 43] where metadata facilities may be involved [4, 14]. Specialization or generalization hierarchies and using a thesaurus can also enable the association of schema terms, e.g., to infer that `Staff` in CSSE could be equivalent to `Staff` in DLSU due to similarities in names or structure. A semantically rich common data model, such as the functional data model [28, 42] or the object-oriented data model, which appear several times in literature [13, 14, 16, 21, 34, 35]. can also enable easier conflict resolution.

4. *Discovering object equivalence.* Being able to determine object equivalence and discover sharing patterns between data objects [4, 14] would greatly enhance the process of schema integration. It is also possible to focus on the equivalence of terms found in schemas and queries, using dictionary and thesaurus lookups [3]. It is possible, for

example, that an imprecise query involving the name of a staff member can be mapped to both `DLSU.Staff.sname` and `CSSE.User.uname`.

5. *Virtual attributes and domain mapping.* As virtual attributes may range over different domains and partial values, and *maybe tuples* may be defined, domains may be mapped with others even when 1:1 mapping is impossible [12]. The previous examples of Tables 1, 2 and 3 illustrate the case of two relations that are only partially compatible. A virtual attribute may be defined for the attribute in question, `rank`, to make the two relations union-compatible.

   $\texttt{GRP}_{grpnum,grpname,rank} \leftarrow$
   $\sigma_{grpnum,grpname,NULL}(\texttt{CSSE.Group})$

   The expression above redefines `CSSE.Group` as a new relation `GRP` with a virtual attribute `rank` making it union-compatible with `DLSU.Group`.

6. *Higher order expressions and higher order views* [22]. These can range over data as well as metadata to define integrated views over participating databases. This is similar to the use of virtual attributes to homogenize originally incompatible relations.

   ```
   .dbH.grp(.grpnum=N,.grpname=M,+.rank=R) ←
   .CSSE.Group(.grpnum=N,.grpname=M,.rank=R)

   .dbH.grp(.grpnum=N,.grpname=M,.rank=R) ←
   .DLSU.Group(.gnum=N,.gname=M,.rank=R)
   ```

   The first expression above creates `dbH.grp`, a higher order view, to redefine `CSSE.Group` with a `rank` attribute. The next expression adds the tuples from `DLSU.Group` to `dbH.grp`, effectively combining `DLSU.Group` and `CSSE.Group`.

7. *Partial automation.* Automation in schema integration is very complex [1], and is perhaps impossible [33]. But it is desirable, so attempts have been made with some success [3, 4, 6]. At the very least it is possible to provide intelligent, assistant tools to make the task easier [13, 14, 17, 28, 41, 42, 43].

8. *Identifying target schemas,* since having a goal, i.e., the target schema, can minimize the complexity of integration [17]. Note that DLSU and CSSE in our example are strongly related, and a target schema should be very possible to conceptualize as in Figure 3.

9. *Middleware solutions.* Middleware backbones can enable interoperability [13, 17]. The trend has been to use object-oriented middleware to exploit the richness of the OO paradigm. Other benefits include portability across operating system and programming platforms and standardization of programming interfaces for easier development of new applications.

10. *Mediator systems and wrappers. Object wrappers* can hide the heterogeneity of component databases and make them conform to standard interfaces [25]. Intelligent facilities and a semantic framework further improve the technique [6, 25].

3. A Meta Object Approach

Semantic issues are obviously key issues, and, without an effective resolution to these issues, automation becomes impossible. But attempts have met with some success [3, 4, 6, 14, 39, 43]. While schema and semantics appear to be disjoint concepts, one may contend that they are linked. Take for example the ER data model, which is favored by database practitioners [31] because of its semantic content. One may consider that the major difference between an ER schema and a relational schema would be the fact that ER diagrams lend to better interpretation. Of course, some semantics cannot be expressed in relational schemas, e.g., cardinality. But some semantics can still be inferred from schema as shown in literature [32, 31, 15]. Meta objects such as names, types, structural relationships with other objects can be used to infer object equivalence or associations [31]. For this reason, we advocate a meta object approach. The use of ontologies, taxonomies, and intelligent systems should also be applied.

Several approaches advocate a metadata approach in schema integration [14, 29, 31, 34, 38] as well as in other areas such as in enterprise modeling and integration where *metadatabase systems* or *metadata repositories* are used [15, 18]. Distributed knowledge-based systems can make use of a *meta-layer* [38] which is important for cooperating systems dealing with heterogeneity. Data warehousing enterprises also require schema integration and can benefit from metadata, such that warehouse metadata interchange standards have been advocated [10, 30].

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) provides the middleware

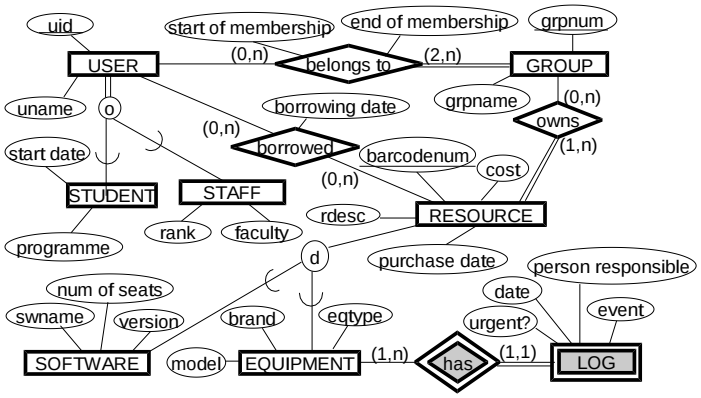

Figure 3. A possible target schema for DLSU and CSSE.

LEVEL	CONTENTS
meta-meta information	meta-types, meta-relations and meta-schemas
meta-information	types, type relations and type schemas
information	entities and relationships, grouped by domain

Table 4. Levels of the MOF abstract model.

backbone in the OMA. It relies on several Object Request Brokers (ORBs) running on participating machines to enable connectivity and interoperability within the distributed system. Within this framework, the Meta Object Facility (MOF) provides the repository for *meta-information*, i.e., information about information [8].

The abstract model of the MOF consists of three levels, shown in Table 4. At the information level, the universe of information consists of *entities* or "things" in a given domain. At the meta-information level, entities are classified under *types*, which may be involved in *type relations* with other types. A *type language* and a *type system* can be used to define *type schemas* for a specific domain system. At this level we are dealing with *meta-information*. The top level classifies types into *meta-types*, which may be involved in *meta-relations* and, using a *meta-type system*, may also define *meta-schemas*. We refer to this level that deals with *meta-meta-information* as the meta-meta-information level, or simply, the meta-meta level.

Applying concepts behind the MOF to database schemas, the first level of information consists of entities and relationships as they are understood in the ER data model. The information may be classified into entity types, relationship types and database schemas. Another database in the same domain may be defined perhaps in different terms, perhaps using the Relational data model, in terms of relations, relations with foreign keys and relational schemas. It is possible to define the similarity between the relational schema and the ER schema at the meta-meta level where we can define relations with foreign keys (in a relational schema) as being similar to relationship types (in an ER schema), for example.

MOF models expressed using the Unified Modeling Language (UML) or the Meta Object Definition Language (MODL) [8, 10] may be converted into CORBA Interface Definition Language (IDL) definitions. These can then be compiled using IDL compilers to create stubs and skeletons for client and server applications using several possible languages such as C or Java on top of several possible operating systems.

There are three major MOF building blocks. *MOF classes* define meta objects, the basic building block in a MOF system. *MOF associations* define classes of binary and directed links between one MOF object and another. *MOF packages* are modules of meta-information models, instances of which can be defined for schemas of related meta-information for a specific domain. Other MOF constructs include data types, constants, exceptions and constraints.

The OMG Common Warehouse Metamodel (CWM) includes metamodel packages used for metadata interchange [10]. One such package is the ER data model package. Such a package can also be used in schema integration. Domain-specific semantics may be encapsulated into a meta package for meta object comparison and analysis, perhaps with the help of semantic dictionaries, type hierarchies, and perhaps a meta knowledge system. Virtual attributes may be defined using meta objects. Higher order

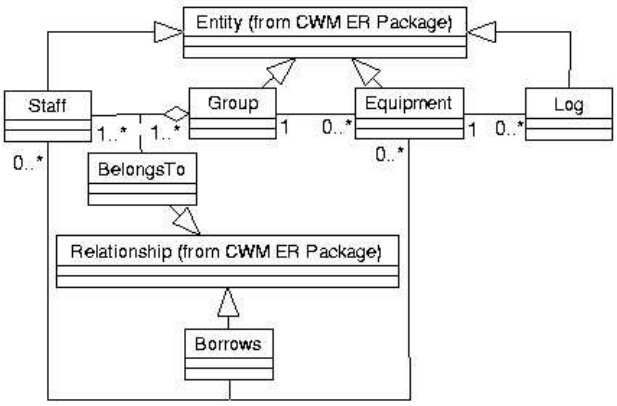

Figure 4. The DLSU Metamodel.

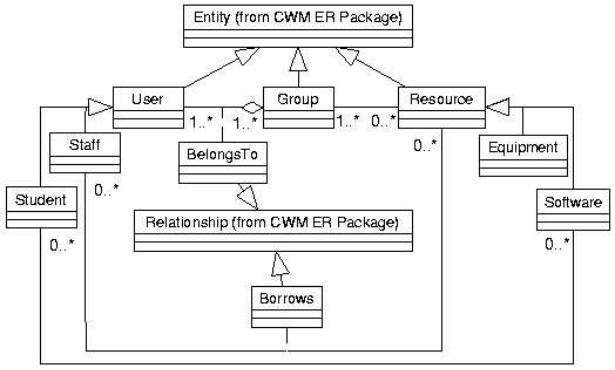

Figure 5. The CSSE Metamodel.

expressions are likewise possible in an environment where objects, e.g., data, may coexist with meta objects, e.g., metadata. Metamodels may also define target schemas.

Furthermore, middleware environments provide abstraction, the ability to use object wrappers, the use of standard facilities and services, e.g., for queries, transactions and naming, and the ability to hook up new applications and tools more conveniently since programming interfaces are standardized.

Figures 4 and 5 show the metamodels for DLSU and CSSE. The meta objects defined are derived from definitions in the ER metamodel package of CWM. The metamodels of Figures 4, 5 plus the CWM ER package can be used to imply the similarities and associations between DLSU and CSSE. The ER package helps identify meta objects in Figures 4 and 5 as attributes, entity types and relationship types. Of course, equating which ones are actually equivalent, e.g., `DLSU.Staff` and `CSSE.Staff`, is another matter, although it is possible. It is of further value to define a domain-specific metamodel

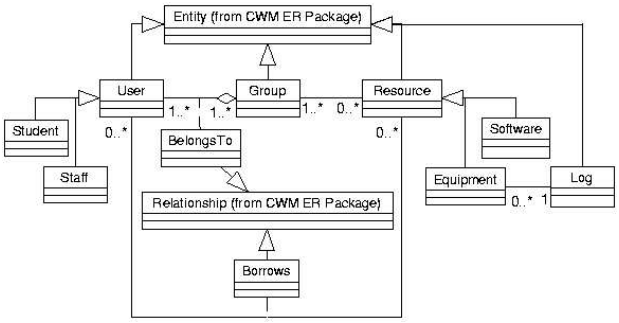

Figure 6. The target schema metamodel.

package to provide a target schema as shown in Figure 6, which is based on the target schema in Figure 3. This can be used to assimilate additional database schemas in the same domain. It should be possible for some expert system to detect possible (meta) object equivalence and associations between the DLSU and CSSE meta objects, e.g., `DLSU.Staff` and `CSSE.Staff`, `DLSU.Group` and `CSSE.Group`, and so on. And if these associations and possible equivalence relationships can be verified by an expert user, then the federated schema metamodel can be derived as shown in Figure 7.

3.2. MOF-Based Integration

A schema integration method is therefore proposed based on the OMG MOF, and the framework we are working on is shown in Figure 8. Database systems are wrapped as objects, and the system is federated rather than one that uses a global schema. It uses three repositories: the Export, Import and Auxiliary Schema Repositories. An *auxiliary schema* defines extensions to the existing schemas for entities in the multidatabase that do not exist in the component databases. Using repositories as one would use a data dictionary in a non-distributed database system provides many benefits including program-data independence and extensibility. The *schema integration facility* manipulates the meta objects to construct import schemas. Intelligent modules may be used to form *integration incubators* to facilitate the evolution of composite schemas based on inferences, user assistance and other clues leading to correct mappings. Query and transaction facilities provide collective access to component databases. Basic facilities for name services and communication are generally part of the middleware environment. An object discovery facility would also improve extensibility. A front-end schema integration tool must also be available to allow authorized users to compose

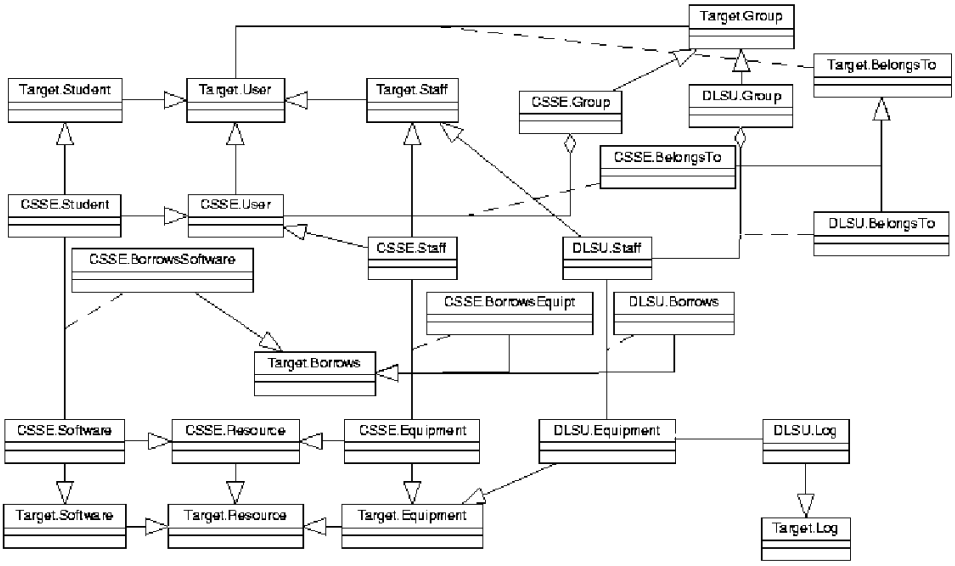

Figure 7. Mapping the import schema in Fig. 6 with export schemas in Fig. 4 and 5.

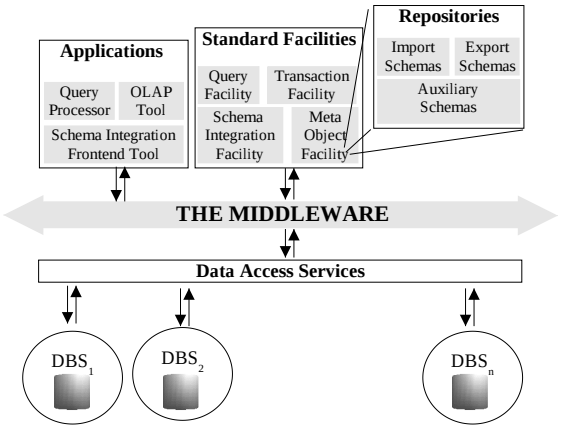

Figure 8. The general framework.

export, import and auxiliary schemas. It should be possible to write applications that range over the multidatabase system using a popular and full-featured DBMS which can interact with the middleware. An issue to resolve would be whether or not updates are allowed, although there are successful prototypes in literature that allow this [24, 32].

Homogenization at the meta information and meta-meta level is achieved through such a facility which allows equivalence and associations to be drawn among meta objects as defined with expressions such as those in Figure 9. Each expression implies the equivalence between meta objects of the import schema, i.e., Target, and those of the export schemas of DLSU and CSSE. It involves operations and other notations as defined below:

- $attr_i \Leftrightarrow attr_j$ signifies mapping
- $attr_i = const_x$ signifies that a constant is assigned
- $A.B_{a,b,c}$ refers to class B of schema A, and B has attributes a, b and c.
- $A.B \rightarrow C.D$ implies the derivation of class $C.D$ from $A.B$.
- π and $\cup$ are the relational algebraic operators for *project* and *union*.
- $'literal'$ refers to a literal constant, and $''$ refers to an empty constant.
- $ifsubstr()$ and $extractdate()$ are string functions which return boolean and date values, respectively.

These expressions map the export schemas in Figures 4 and 5 into the import schema of the multidatabase system defined in Figure 6. These expressions can be defined in one of two ways: a language-based front-end, perhaps using the Object Constraint Language (OCL), or a graphical front-end tool that can be parsed to generate those expressions.

In addition, the framework must also be object-oriented, to exploit the semantic richness of the paradigm, but it must provide relational DBMS functionality, this being desirable for many information system practicitioners. The framework must also provide open interface standards for tools and applications that may be added in the future.

1. $\pi_{uid \Leftrightarrow sid, uname \Leftrightarrow sname, utype='staff', startdate='', programme='', rank='', faculty \Leftrightarrow dept}(DLSU.Staff) \cup$
   $CSSE.User \rightarrow Target.User_{uid, uname, utype, startdate, programme, rank, faculty}$
2. $\pi_{grpnum \Leftrightarrow gnum, grpname \Leftrightarrow gname}(DLSU.Group) \cup CSSE.Group \rightarrow Target.Group_{grpnum, grpname}$
3. $\pi_{uid \Leftrightarrow sid, grpnum \Leftrightarrow gnum, startofmembership \Leftrightarrow datejoined, endofmembership \Leftrightarrow dateleft}(DLSU.MemberOf) \cup$
   $CSSE.BelongsTo \rightarrow Target.BelongsTo_{uid, grpnum, startofmembership, endofmembership}$
4. $\pi_{rnum \Leftrightarrow eqnum, rdesc \Leftrightarrow description, cost, purchasedate=''}(DLSU.Equipment) \cup$
   $\pi_{rnum \Leftrightarrow barcodenum, rdesc, cost, purchasedate}(CSSE.Resource) \rightarrow Target.Resource_{rnum, rdesc, cost, purchasedate}$
5. $\pi_{rnum \Leftrightarrow eqnum, eqtype='', brand \Leftrightarrow vendor, model=''}(DLSU.Equipment) \cup$
   $\pi_{rnum, eqtype, brand, model}(CSSE.Equipment) \rightarrow Target.Equipment_{rnum, eqtype, brand, model}$
6. $\pi_{rnum, swname, numofseats, version}(CSSE.Software) \rightarrow Target.Software_{rnum, swname, numofseats, version}$
7. $\pi_{uid \Leftrightarrow sid, rnum \Leftrightarrow eqnumborrowingdate \Leftrightarrow dateborrowed}(DLSU.Borrows) \cup$
   $\pi_{uid, rnum \Leftrightarrow barcodenum, borrowingdate}(CSSE.BorrowedSoftware) \cup$
   $\pi_{uid, rnum \Leftrightarrow barcodenum, borrowingdate}(CSSE.BorrowedEquipment) \rightarrow Target.Borrowed_{uid, rnum, borrowingdate}$
8. $\pi_{rnum \Leftrightarrow eqnum, urgent, logdate, personresp, logevent}(DLSU.Log) \cup$
   $\pi_{rnum, urgent=ifsubstr('urgent', event, 1, 0), logdate=extractdate(event), personresp='', logevent}(CSSE.Equipment) \rightarrow$
   $Target.Log_{rnum, urgent, logdate, personresp, logevent}$

Figure 9. Sample expressions relating meta objects in the meta model.

4. Conclusion and Future Work

This paper presented a meta object approach to schema integration that exploits a standard middleware and some solutions already devised literature. We contend that our approach can reasonably deal with semantics through meta object manipulation, object wrapping, mediation and similar concepts that can be exploited for higher levels of automation and feasibility for schema integration solutions. Furthermore, powerful features in mature middleware with meta object facilities such as OMG CORBA and Microsoft COM/DCOM provide more feasible solutions where there is no need to write everything from scratch.

There are, however, unresolved issues to tackle such as coming up with packaged metamodels for specific domains and the use of intelligent systems to assist in the process. Neither the schema integration facility nor its front-end tool exist just yet. The sample databases used are by necessity simple to begin with, and are both implemented on the same platform. We should now construct the complete environment as described in the framework including a novel feature involving schema incubation.

The approach discussed in this paper is clearly not the first to exploit metadata, repositories or middleware. We must stress, however, the need to exploit standard facilities in existing middleware standards to maximize the potential for interoperability, platform-neutrality, object re-use, and wide applicability in real-world enterprises.

Acknowledgement. The authors appreciate valuable advise and feedback from Dr. Stephen Crawley of the Distributed Systems Technology Centre and Dr. Catherine Ewald of Queensland Transport, Brisbane, Australia.

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.

References

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

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

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

[4] M. W. Bright, A. R. Hurson, S. Pakzad, and H. Sarma. The summary schemas model – an approach for handling multidatabases: Concept and performance analysis. In A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors, *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, chapter 4, pages 199–216. IEEE Computer Society Press, 1993.

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

[6] P. E. Cannata. The irresistible move towards interoperable database systems. In *Proc. First*

Int'l Workshop on Interoperability in Multidatabase Systems, pages 2–5, 1991.

[7] S. Ceri, B. Pernici, and G. Wiederhold. Distributed database design methodologies. In *Proc. IEEE*, pages 533–546, May 1987.

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

[9] S. Cua, G. Gaw, J. Kiok, and J. Lau. MDB1. Bachelor's Thesis, Software Technology Department, De La Salle University, 1999.

[10] Common Warehouse Metamodel (CWM) specification. OMG Document ad/2000-01-01, February 2000.

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

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

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

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

[15] N. Georgalas. Integrating distributed data over their semantic identity. In *Proc. Association for Information Systems 1997 Americas Conf. on Heterogeneous Database Interoperability & Data Warehousing*, Indianapolis, Indiana US, August 15-17 1997.

[16] P. M. D. Gray and N. W. P. K. G. Kulkarni. *Object-Oriented Databases: A Semantic Data Model Approach.* Prentice Hall International Series in Computer Science. Prentice Hall International (UK) Ltd, 1992.

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

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

[19] A. R. Hurson and M. W. Bright. Object-oriented multidatabase systems. In O. A. Bukhres and A. K. Elmagarmid, editors, *Object-oriented multidatabase systems: a solution for advanced applications*, chapter 1, pages 1–36. Prentice-Hall, 1996.

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

[21] W. Klas, P. Fankhauser, and P. Muth. Database integration using the open object-oriented database system VODAK. In O. Bukhres and A. K. Elmagarmid, editors, *Object Oriented Multidatabase Systems: A Solution for Advanced Applications.* Prentice Hall, Englewood Cliffs, N.J., 1996.

[22] R. Krishnamurthy, W. Litwin, and W. Kent. Interoperability of heterogeneous databases with schematic discrepancies. In *Proc. First Int'l Workshop on Interoperability in Multidatabase Systems*, pages 144–151, 1991.

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

[24] W. Litwin and A. Abdellatif. An overview of the multidatabase manipulation language MDSL. In *Proc. IEEE*, pages 621–632, May 1987.

[25] L. Liu, L. L. Yan, and M. Özsu. Interoperability in large-scale distributed information delivery systems. In A. D. et.al, editor, *Workflow Management Systems and Interoperability (NATO Asi Series. Series F, Computer and Systems Sciences, Vol 164).* Springer Verlag, 1997.

[26] R. J. Miller. Using schematically heterogeneous structures. In *Proc. ACM SIGMOD Int'l Conf. on the Management of Data*, 1998.

[27] Meta object facility (MOF) specification v1.3. Object Management Group (OMG), March 2000.

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

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

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

[31] S. Navathe and A. Savasere. A schema integration facility using object-oriented data model. In O. A. Bukhres and A. K. Elmagarmid, editors, *Object-oriented multidatabase systems: a solution for advanced applications*, chapter 4, pages 105–128. Prentice-Hall, 1996.

[32] M. H. Nodine and S. B. Zdonik. The impact of transaction management on object-oriented multidatabase views. In O. A. Bukhres and A. K. Elmagarmid, editors, *Object-oriented multidatabase systems: a solution for advanced applications*, chapter 1, pages 57–104. Prentice-Hall, 1996.

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

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

[35] A. Rafii, R. Ahmed, P. DeSmedt, B. Kent, M. Ketabchi, W. Litwin, and M.-C. Shan. Multidatabase management in pegasus. In *Proc. First Int'l Workshop on Interoperability in Multidatabase Systems*, pages 166–173, 1991.

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

[37] F. Saltor, M. G. Castellanos, and M. Garcia-Solaco. Overcoming schematic discrepancies in interoperable databases. In D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, editors, *IFIP DS-5 Semantics of Interoperable Database Systems*, volume 1, pages 184–198, Nov. 1992.

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

[39] A. Sheth and V. Kashyap. So far (schematically) yet so near (semantically). In D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, editors, *IFIP DS-5 Semantics of Interoperable Database Systems*, volume 1, pages 282–301, Nov. 1992.

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

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

[42] J. M. Smith, P. A. Bernstein, U. Dayal, N. Goodman, T. Landers, K. W. T. Lin, and E. Wong. Multibase – integrating heterogeneous distributed database systems. In *Proc. of AFIPS*, pages 487–499, 1981.

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