Knowledge-Based Query Formulation for Integrated Information Systems

Ralph D. Semmel, Elisabeth A. Immer, David P. Silberberg, and Robert P. Winkler

The U.S. Army, like many large organizations, relies upon heterogeneous and distributed information systems for managing massive quantities of data. Formulating queries within these systems has been difficult because of the underlying structural complexity of the associated databases. Formulating queries across these systems has been impossible because of the additional lack of integration and interoperability. Building on work done by the Milton S. Eisenhower Research and Technology Development Center in artificial intelligence and database systems, the Applied Physics Laboratory is collaborating with the U.S. Army Research Laboratory to develop advanced capabilities to support automated query formulation, conceptual modeling, and systems integration. In addition, a framework based on intelligent agents and distributed objects has been developed that will facilitate the fusing of new and legacy systems, interoperability, and the creation of intelligent interfaces.
(Keywords: Artificial intelligence, Database systems, Query formulation, Systems integration.)

INTRODUCTION

In collaboration with the U.S. Army Research Laboratory (ARL), APL's Milton S. Eisenhower Research and Technology Development Center has been developing approaches that facilitate information access over complex systems. The effort evolved from a well-established program in artificial intelligence and database systems that the Research and Technology Development Center has been involved in for more than a decade. Existing techniques developed by APL researchers have been extended, and new approaches have been devised that facilitate advanced conceptual design, support automated query formulation over new and legacy systems, and provide a basis for information systems integration and interoperability.

The program with ARL is motivated by the U.S. Army's increasing dependence on advanced informa-

tion systems for both combat and support operations. As these systems have become more complex, accessing relevant information efficiently has become more difficult partly because of the nature of modern information systems, which tend to rely on vast quantities of stored data with a high degree of structural complexity. In addition, such systems often rely on geographically separated but conceptually related information sources. U.S. Army information systems, in particular, contain terabytes of data spread across thousands of attributes dispersed among hundreds of tables or objects. In turn, the tables and objects reside in heterogeneous and distributed repositories.

Existing information systems typically rely on the capabilities of an underlying database management system for information access. However, most database management systems provide relatively low-level data retrieval and manipulation support that requires users to be familiar with the underlying structure of the database. As a result, query language requirements tend to be complex. Moreover, few standards for representing similar data objects exist, and little support exists for integrating legacy systems. Consequently, interoperability is limited, costly customized interfaces are required, and user needs that were not anticipated at the time of the initial design cannot be satisfied.

The approach being explored with ARL to facilitate access is loosely based on the universal relation (UR) model.[1,2] In the UR approach, all attributes are perceived as belonging to a single relation, thus eliminating the need for specific knowledge regarding relations, objects, or complex associations. Although the UR approach can dramatically reduce the complexity of an interface, the level at which reasoning should occur is still an open issue.[3] For example, with a relational database, where data are stored in tables, there is insufficient knowledge for inferring queries from high-level requests because of the sparsity of navigational information that would indicate how tables can be correctly combined. Similarly, with conceptual modeling schemes based on basic entity-relationship (ER) constructs, where data are represented by classes and relationships among those classes, there is insufficient knowledge to infer semantically reasonable queries over complex domains.[4]

To counteract problems with traditional UR approaches, APL has developed a system known as QUICK (QUICK is a Universal Interface with Conceptual Knowledge), which uses an extended set of ER constructs based on semantic data modeling, object-oriented, and knowledge representation schemes.[5,6] Although initially developed as a query formulation tool, QUICK has evolved over the years to support many information system life-cycle activities. In addition, QUICK's reverse engineering and reengineering capabilities support high-level integration of legacy systems.

AN ARL EXAMPLE

The ultimate goal of the program with ARL is to facilitate retrieval of information across heterogeneous U.S. Army database systems. The initial focus of the program has been on two databases. The first is the ARL-ELINT database, which is used for gathering and correlating low-level electronic and imaging information. The second is the MIIDS/IDB database, which is used to support war planning and fighting at multiple echelons. It is anticipated that other databases will eventually be integrated as well (e.g., SORTS, STACCS, and PROBE), although issues such as how different logical-level implementations (e.g., relational and object oriented) can be fused must be resolved. ARL would also like to facilitate the development of intelligent interfaces to provide ad hoc access without semantic or structural knowledge and to direct attention and reasoning under uncertainty, both of which are goals related to work that the Research and Technology Development Center has done in collaboration with APL's Submarine Technology Department on intelligent systems interfaces.[7]

Achieving the program goals requires that database design knowledge be available to support reasoning. Such knowledge can be found in the conceptual schema, which is a high-level representation of a database system that is created early in the information system life cycle and is independent of the final logical-level implementation. For example, Fig. 1 shows a small portion of the ARL-ELINT database using basic ER constructs. The diagram shows two entities, represented by rectangles, associated by a many-to-one (*N*:1) relationship, represented by a diamond. Attributes are shown as ovals and, in this case, are associated with entities only. For clarity, only a few of the attributes are shown. In actuality, entities typically have between 10 and 100 attributes. Similarly, real-world databases often have on the order of 100 entities and relationships.

As the de facto standard for modeling large and complex database systems, the ER model is easy to use and understand, and supports a straightforward transformation to the relational and other implementation models. For example, to create the table descriptors corresponding to the simple ER diagram in Fig. 1, the entities are represented as relations, with entity attributes included as fields in the corresponding tables. The *key* of each table, which is used to uniquely identify any record, is set to the identifier in the corresponding entity. Note that in both the ER diagram and in the table descriptors below, key attributes are underlined. Then, it can be recognized that the *N*:1 relationship corresponds to a functional association from the *N*-side entity (i.e., ELINT_RF) to the 1-side entity (i.e., ELINT_Report). Thus, each tuple in the ELINT_RF relation will be associated with no more than one tuple in the ELINT_Report relation. As a result,

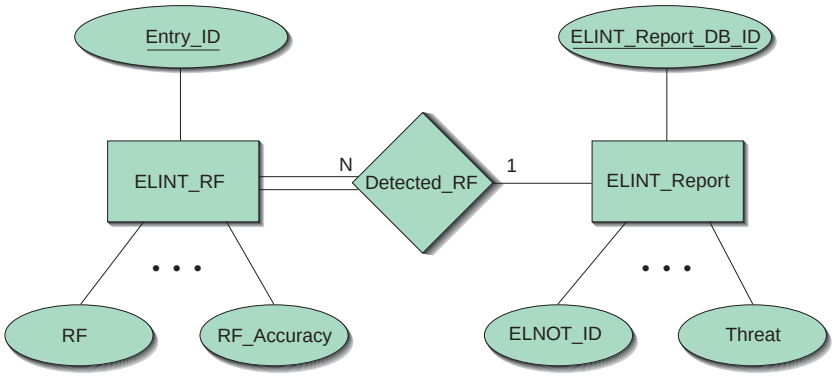

Figure 1. Small portion of ARL-ELINT ER design.

ELINT_RF is augmented with *foreign key* attributes that correspond to the key of ELINT_Report (i.e., ELINT_Report_DB_ID). The transformation is now complete, and the diagram in Fig. 1 is represented by the following table descriptors:

ELINT_RF(Entry_ID,RF,…,RF_Accuracy,ELINT_Report_DB_ID)

ELINT_Report(ELINT_Report_DB_ID,ELNOT_ID,…,Threat)

Significantly, knowledge has been lost in mapping from the ER representation to the sparser relational representation. For example, mandatory participation is present on the *N*-side entity, as indicated by the double-line edge in the ER diagram, which is not indicated in the relational representation. Similarly, the fact that a relationship exists between the relations must be inferred from the fact that identical character strings appear in the two descriptors. However, in practice, different strings are often used to represent the same attribute in different descriptors (e.g., the attribute ELINT_Report_DB_ID in the table ELINT_Report could have been called EDB_ID). In such cases, even simple syntactically based inferencing would not be possible as representation of the relationship would be obscured.

INTELLIGENT DATABASE PROCESSING

Although there are significant limits to the reasoning that can be done with implementation-level models, it is possible to use ER-level knowledge and mapping information to dramatically enhance intelligent database processing capabilities. In particular, it is possible to automate retrieval tasks for which relational-level knowledge would not be sufficient. Figure 2 illustrates the ER representation of a larger portion of the ARL-ELINT database. To avoid clutter, the diagram does not include attributes. However, it does present what is referred to as a context, which is illustrated by the enclosing polygon.

Conceptually, a context identifies a maximal set of objects at the ER level that are in some way strongly associated.[8] Unfortunately, the notion of strong association is based on being able to infer user intent, which is not always possible. However, given a well-designed database, likely intent is often implicit in the conceptual schema and can be inferred.

As a starting point for determining whether ER objects are strongly associated, it is possible first to consider whether corresponding relational-level tables are strongly associated. At the relational level, strong association is based on the semantics of the design and the ability to compose, or join, tables to produce meaningful results. One formal design approach initially considers all attributes as part of a single relation. Then, anomalies that could arise as a result of redundancy are identified, and the single relation is decomposed into two or more relations that can be recomposed. If the resulting recomposition is guaranteed not to contain spurious tuples, as illustrated in the following example, then the join is meaningful and is said to be lossless. Each decomposed relation is then evaluated for potential anomalies and recursively decomposed until every resultant relation is an appropriate normal form that avoids specific anomalies.

As an example, Fig. 3 shows a piece of a relation for suppliers. Figure 3a highlights some of the anomalies that can occur when storing all data in a single relation. For instance, for each supplier entry the city must be repeated. If a city were changed, but only one entry were modified to reflect this fact, the database would become inconsistent. Figure 3b resolves the problem by decomposing the single relation into two relations that can be joined on the attribute S_Name to produce the original relation. Under the decomposition schema, only one city entry per supplier exists, so introducing inconsistency is not possible. In addition to modification anomalies, insertion and deletion anomalies are also present. For instance, in the single-relation representation, it is not possible to represent a supplier who does not supply at least one item unless null values are allowed for Item and Quantity. With the decomposed relations, this is not an issue, as the upper relation of Fig. 3b can be used to store the desired data.

Unfortunately, it is possible to decompose relations in an inappropriate manner. For example, consider Fig.

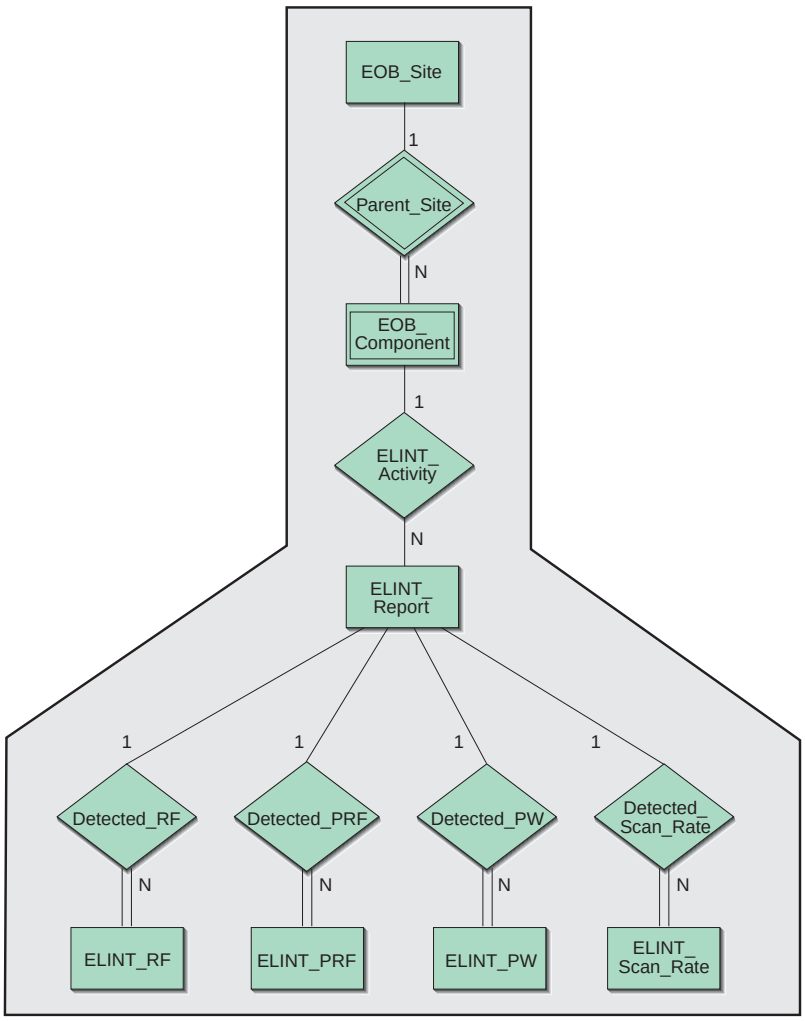

Figure 2. Larger portion of ARL-ELINT ER design with context.

(a)

S_Name	City	Item	Quantity
Acme	Baltimore	Bolt	10
EMG	Columbia	Widget	20
EMG	Columbia	Bolt	15
Acme	Baltimore	Cog	20

(b)

S_Name	City
Acme	Baltimore
EMG	Columbia

S_Name	Item	Quantity
Acme	Bolt	10
EMG	Widget	20
EMG	Bolt	15
Acme	Cog	20

Figure 3. A simplified supply domain (a) represented as a single relation, and (b) represented by a lossless-join decomposition.

4a, which decomposes the relation of Fig. 3a by sharing the attribute Quantity instead of S_Name. The result of recomposing the resultant tables using the shared attribute Quantity is the introduction of the spurious tuples shown below the red line of Fig. 4b. To prevent these types of decompositions, database developers use results from dependency theory to determine appropriate designs that guarantee lossless joins. For instance, the two relations in Fig. 3b are said to be in Boyce-Codd Normal Form (BCNF), as each attribute is functionally dependent on the key of the relation in which the attribute resides (S_Name is the key for the upper relation, and the composite S_Name and Item is the key for the lower relation). Similarly, the relation in Fig. 3a is not in BCNF as BCNF requires functional dependence on the complete key, and City is functionally dependent only on S_Name, which is only part of the key consisting of S_Name and Item.

Whereas a significant body of work exists regarding formal database design and dependency theory, it is recognized that the process cannot be completely mechanized.[9] In particular, identification of all dependencies (e.g., functional, multivalued, and join) in a real-world system is difficult, and dependency analysis can be computationally expensive. As a result, more intuitive models such as the ER and IDEF1X models have gained in popularity and have become the de facto methods used for database design.[10] Formal methods are then used to augment and support targeted portions of a design.

With ARL, we have abstracted the notion of a lossless join at the relational level to the ER level to ensure that appropriate aggregations of ER objects can be identified. Then, given a knowledge-rich conceptual schema, contexts can be formulated automatically and used as the basis for intelligent database processing. For basic ER constructs, the process can be defined inductively. First, all relationships and their participating entities may be considered contexts. This follows from the fact that relationship sets are formally defined as subsets of the Cartesian product of their participating entity sets. As a result, a given relationship instance contains the keys to each entity and thus functionally determines its participating entity instances. A theorem from dependency theory then enables us to infer that a lossless join exists among the relations corresponding to the ER-level objects and that the participating objects are strongly associated.[11] The inductive step entails extending a context by inferring localized functional dependencies that exist from the entities to neighboring vertices in the ER graph. In particular, this is done by

(a)

S_Name	City	Quantity
Acme	Baltimore	10
EMG	Columbia	20
EMG	Columbia	15
Acme	Baltimore	20

Item	Quantity
Bolt	10
Widget	20
Bolt	15
Cog	20

(b)

S_Name	City	Item	Quantity
Acme	Baltimore	Bolt	10
EMG	Columbia	Widget	20
EMG	Columbia	Bolt	15
Acme	Baltimore	Cog	20
EMG	Columbia	Cog	20
Acme	Baltimore	Widget	20

Figure 4. A poorly designed supply database: (a) inappropriate decomposition based on the attribute Quantity; and (b) lossy join that results in spurious tuples (shown below red line).

following *N*:1 and 1:1 associations that do not introduce cycles in a given context. Again, this is justified by the fact that the corresponding relations will be associated by a functional dependency, which ensures that a lossless join will result. Extension rules also exist for object-oriented generalization associations, recursive relationship types, and multivalued dependencies that can be inferred from implicit hypergraph connectivity, where edges correspond to arbitrary nonempty sets of vertices rather than to sets containing two vertices.

Given context, ER, and mapping knowledge, it is possible to formulate queries automatically in response to high-level requests over real-world databases. For example, consider a request over the ARL-ELINT database to list the pulse repetition frequency, the radio frequency, and the pulse width of electronic device ABC1234:

```
SELECT
   prf,
   rf,
   pw
WHERE
   elnot_id = 'ABC1234'
```

The preceding request notation is referred to as USQL; it resembles the standard query language SQL, but is adapted for use with universal relation interfaces by not requiring a FROM clause or join criteria in the WHERE clause.[12] The request is, in some sense, close to what a user with domain knowledge, but without conceptual schema knowledge, would generate in response to an operational need.

Typically, USQL is generated by a higher-level graphical or direct-manipulation interface. In turn, the USQL request is passed to the knowledge-based query formulator in QUICK, which uses its knowledge of contexts, specific ER characteristics, and relational mapping information to generate the final SQL query. In particular, QUICK first identifies each ER object that contains a requested attribute, and designates those objects as *sacred*. Then, it finds each context that contains every sacred object and continually prunes each context until all leaves are sacred; this pruning can be done in O(*N*) time, where *N* corresponds to the number of entities and relationships in the unpruned context. Finally, QUICK uses its ER-to-relational mapping information, similar to that described for transforming the ER graph in Fig. 1 to a set of relations, to generate the final query:

```
SELECT
   elint.elint_prf.prf,
   elint.elint_rf.rf,
   elint.elint_pw.pw
FROM
   elint.elint_report,
   elint.elint_rf,
   elint.elint_prf,
   elint.elint_pw
WHERE
   elint.elint_report.elnot_id =
    'ABC1234' AND
   elint.elint_rf.elint_report_db_id =
    elint.elint_report.elint_report_db_id AND
   elint.elint_prf.elint_report_db_id =
    elint.elint_report.elint_report_db_id AND
   elint.elint_pw.elint_report_db_id =
    elint.elint_report.elint_report_db_id
```

This final query could not likely be generated by the typical operational user.

INTEGRATING MULTIPLE DATABASES

As illustrated by the previous example, it can be difficult to produce SQL queries without automated support. In particular, detailed knowledge of the underlying logical structure is required, and clauses cannot be left out without changing the semantics of the final result. Yet, database systems typically provide little automated support, thus making stored information difficult to retrieve and use. Moreover, there is typically no support for the more difficult problems of database integration and interoperability. One approach that APL is exploring with ARL with respect to integration is based on conceptual gateways. This approach entails

explicitly identifying the conceptual connections that exist among the ER conceptual schemas of related database systems and establishing gateway relationships to indicate specific associations. For example, Fig. 5 illustrates a gateway relationship between the ARL-ELINT and MIIDS/IDB database systems. In particular, the gateway explicitly identifies the entities that are conceptually associated between the two systems as well as the means to relate them.

The ER model and contexts can be extended in a straightforward manner to work with gateways, thus enabling the development of high-level representations of integrated database systems.[13] For example, Fig. 6 shows the integration of the ARL-ELINT and MIIDS IDB systems (as before, attributes are not shown to reduce clutter). One of several contexts is shown for the MIIDS/IDB system, and the single context previously identified for ARL-ELINT is shown. Moreover, the two contexts are explicitly associated by the gateway relationship from Fig. 5. QUICK can now infer that a larger *virtual* context exists that subsumes the two independent contexts and the gateway relationship.

Given an extended ER model, a set of virtual contexts, and a common implementation model, query formulation can be adapted in a straightforward manner. For example, suppose a user wanted to list the last-known activities, mobility status, and equipment types and quantities of any known enemy unit associated with an emitter site where a pulse repetition frequency of 50 Hz has been detected. In USQL, this could be expressed as follows:

```
SELECT
   site_name,
   unit_name,
   unit_role,
   echelon,
   activity_name,
   mobility_status,
   equip_code,
   equip_quantity
WHERE
   allegiance = 'ENEMY' And
   prf = 50
```

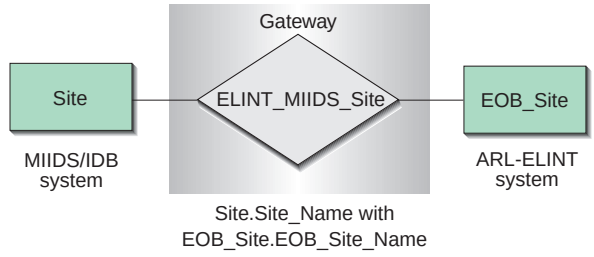

Figure 5. Gateway relationship connecting two databases.

Note that the USQL request does not indicate from which database the attributes are to be retrieved. Instead, that task is left to the enhanced automated formulator. As before, the enhanced formulator identifies relevant contexts, which are now selected from the intersystem virtual contexts. One virtual context is found, and it is pruned until all leaves are sacred (i.e., cover requested attributes). Finally, ER-to-relational mappings are used to produce the final SQL query:

```
SELECT
   miids.site.site_name,
   miids.unit.unit_name,
   miids.unit.unit_role,
   miids.unit.echelon,
   miids.unit.activity_name,
   miids.unit.mobility_status,
   miids.equipment_type.equip_code,
   miids.equipment.equip_quantity
FROM
   miids.equipment,
   miids.unit,
   miids.site,
   elint.eob_site,
   elint.eob_component,
   elint.elint_report,
   elint.elint_prf,
   miids.equipment_type
WHERE
   (miids.unit.allegiance =
    'ENEMY' AND
    elint.elint_prf.prf =
    50) AND
   miids.equipment.unit_name =
    miids.unit.unit_name AND
   miids.equipment.unit_role =
    miids.unit.unit_role AND
   miids.equipment.echelon =
    miids.unit.echelon AND
   miids.equipment.site_name =
    miids.site.site_name AND
   elint.eob_site.eob_site_name =
    miids.site.site_name AND
   elint.eob_component.eob_site_db_id =
    elint.eob_site.eob_site_db_id AND
   elint.elint_report.elnot_id =
    elint.eob_component.elnot_id AND
   elint.elint_prf.elint_report_db_id =
    elint.elint_report.elint_report_db_id AND
   miids.equipment.equip_code =
    miids.equipment_type.equip_code
```

GENERALIZING THE SOLUTION TO SUPPORT INTEROPERABILITY

The SQL query in the previous example is complex. It mixes attributes from two databases, as indicated by the prefixes MIIDS and ELINT, and it requires complex join criteria in the WHERE clause. Furthermore, it

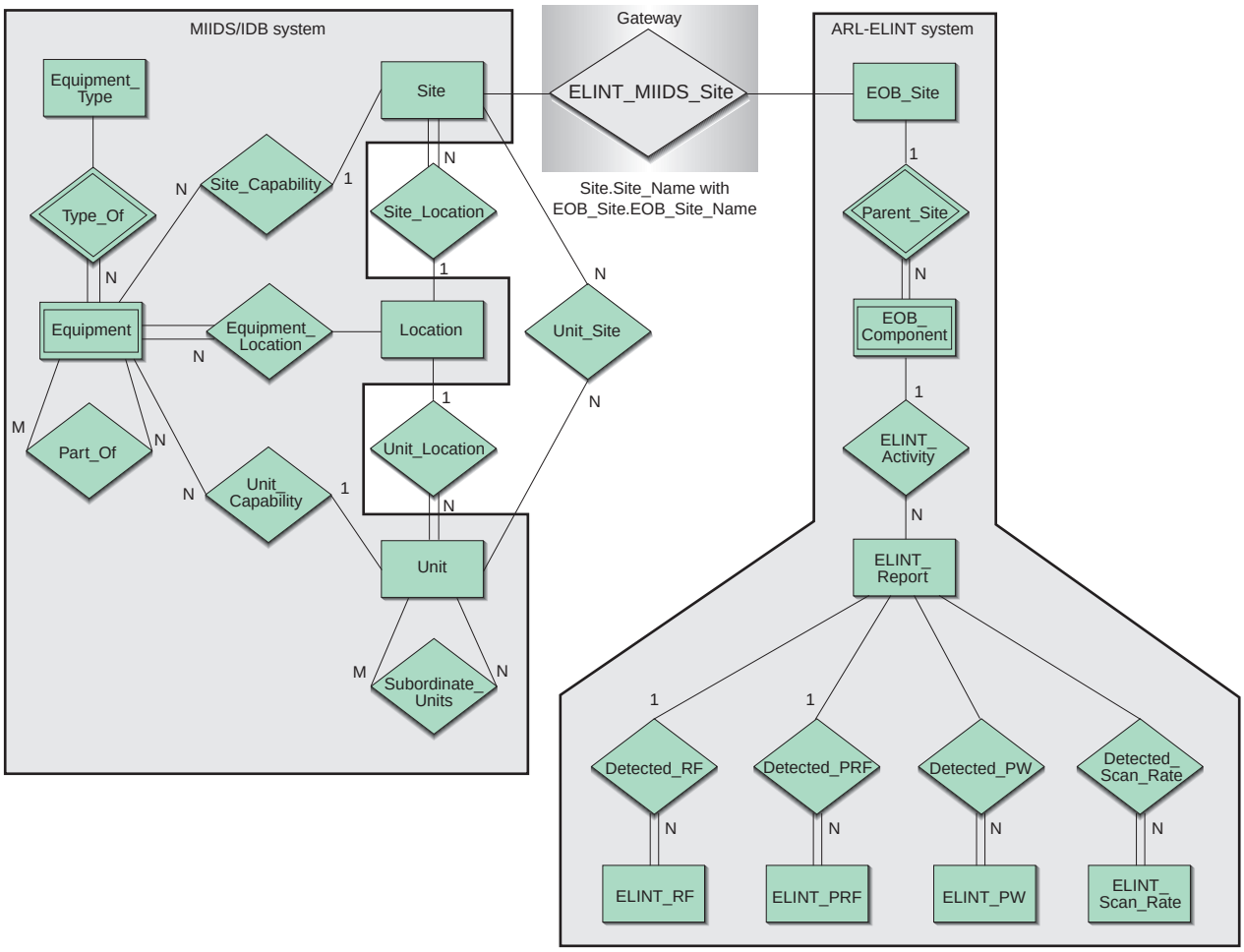

Figure 6. Integrated database systems with selected contexts and a gateway relationship.

is based upon knowledge of the structure of two databases that many users do not possess. The automated approach described to this point eliminates the need for detailed user knowledge of the databases and is extremely fast, generating semantically reasonable queries in less than one second. However, it supports only preliminary techniques for integrating systems, which must be based on the relational model. The approach does not solve the more general interoperability problem for heterogeneous systems, nor does it support the dynamic addition or removal of databases, or the reconfiguration of complex information systems.

A more comprehensive solution requires more sophisticated techniques. In this regard, the Research and Technology Development Center is exploring the use of intelligent agents based on a distributed object implementation using the Common Object Request Broker Architecture (CORBA).[14,15] Consider the functional diagram for an intelligent systems interface in Fig. 7. The ISI uses an intelligent user interface component that contains an information manager implemented as a set of software agents. Actions of the agents are coordinated by an information manager facilitator agent. When a request is posed, the information manager request agent passes a processed request to a facilitator agent in the intelligent user interface manager of a middleware module; this module is used to balance system load and coordinate the activities of multiple intelligent user interfaces. In turn, the intelligent user interface manager facilitator broadcasts a request for assistance to QUICK automated query formulation agents associated with each underlying system. Based on how the QUICK agents respond, information will be integrated by the intelligent user interface manager facilitator and passed back to the requesting intelligent user interface, which will then format and present the information to the user or requesting process in an appropriate manner.

An advantage of the approach is the ease with which new database systems can be incorporated. In particular, when a new or additional legacy system is to be integrated, a QUICK agent is created, and that agent's

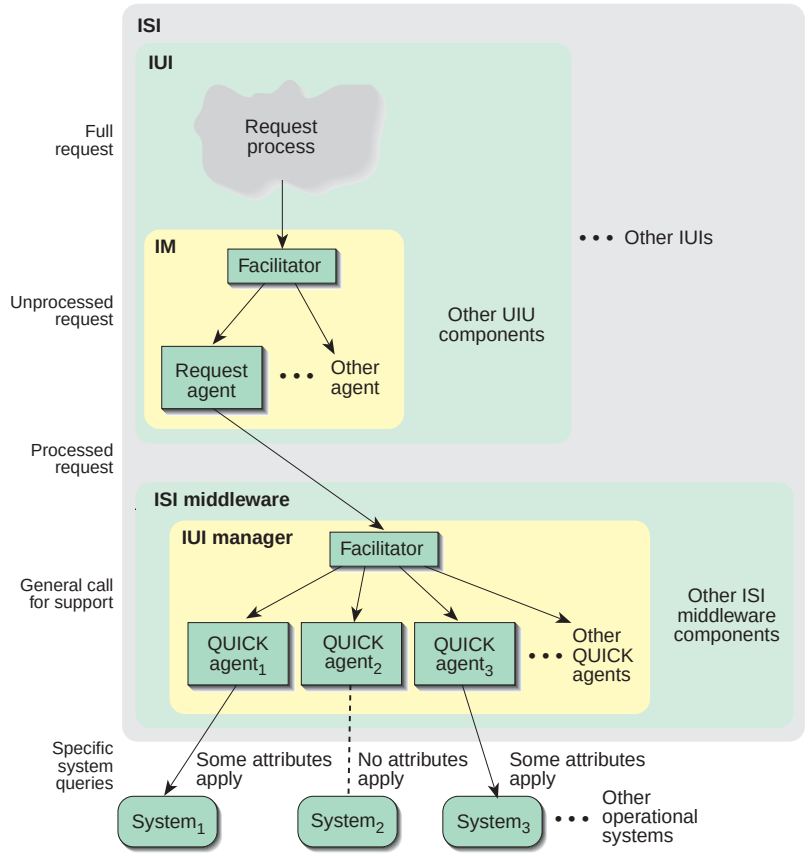

Figure 7. Agent-based approach to an intelligent systems interface (ISI). IUI = intelligent user interface, IM = information manager.

capabilities are made apparent to the intelligent user interface manager facilitator. The facilitator is made aware of the gateway associations between the added and existing systems, and wrapper-based techniques are used to hide specific system implementation details. Then, information processing proceeds in the manner described previously.

CONCLUSIONS

Automated database access, integration, and interoperability are important, though difficult problems. From APL's standpoint, a clear need to develop and extend technology to support both ongoing and potential work exists. In the logistics domain, for example, the Defense Advanced Research Projects Agency is sponsoring a major program that requires the development of decision aids that integrate numerous legacy systems through the use of object-oriented and agent-based technologies. Similarly, the U.S. Army Office of the Director of Information Systems for Command, Control, Communications, and Computers (DISC4) has a requirement to integrate database systems that contain potentially inconsistent information. Finally, ARL has a continuing need to integrate and maintain disparate information systems that are subject to catastrophic failures.

With respect to QUICK, plans include the development of richer conceptual constructs to support more complex domains, enhancing contexts to support more sophisticated knowledge-based processing, implementing roles and tuple variables for ER-based structural disambiguation, and developing data fusion capabilities to support heterogeneous implementations. In addition, research and development is being considered to support automatic interface generation, employ user models to guide query processing, and provide consistency maintenance via specified conceptual-level constraints.

With substantial capabilities already in place in automated query formulation, distributed object computing, and intelligent systems interfaces, APL's Research and Technology Development Center is prepared to initiate and support significant programs in intelligent information processing. Moreover, with the continuing expansion of the Internet and the World Wide Web, and the concomitant need for ad hoc access to complex information systems, these in-house capabilities will become increasingly important to APL activities.

REFERENCES

[1]Maier, D., Ullman, J. D., and Vardi, M. Y., "On the Foundations of the Universal Relation Model," *ACM Trans. Database Syst.* **9**(2), 283–308 (1984).
[2]Vardi, M. Y., "The Universal-Relation Data Model for Logical Independence," *IEEE Software*, 80–85 (Mar 1988).
[3]Leymann, F., "A Survey of the Universal Relation Model," *Data Knowl. Eng.* **4**(4), 305–320 (1989).
[4]Markowitz, V. M., and Shoshani, A., "Abbreviated Query Interpretation in Extended Entity-Relationship Oriented Databases," in *Entity-Relationship Approach to Database Design and Querying*, F. H. Lochovsky (ed.), North-Holland, Amsterdam, pp. 325–343 (1990).
[5]Hull, R., and King, R., "Semantic Database Modeling: Survey, Applications, and Research Issues," ACM *Comput. Surv.* **19**(3), 201–260 (1987).
[6]Semmel, R. D., and Silberberg, D. P., "An Extended Entity-Relationship Model for Automated Query Generation," *Telematics and Informatics* **10**(3), 301–317 (1993).
[7]Coury, B. G., and Semmel, R. D., "Supervisory Control and the Design of Intelligent User Interfaces for Supervisory Control," in *Automation and Human Performance: Theory and Applications*, R. Parasuraman and M. Mouloua (eds.), Lawrence Erlbaum, Mahwah, NJ, pp. 221–242 (1996).
[8]Semmel, R. D., "Discovering Context in an Entity-Relationship Conceptual Schema," *J. Comput. Software Eng.* **2**(1), 47–63 (1994).
[9]Vardi, M. Y., "Fundamentals of Dependency Theory," in *Trends in Theoretical Computer Science*, E. Borger (ed.), Computer Science Press, Rockville, MD, pp. 171–224 (1988).

[10]Teorey, T. J., *Database Modeling and Design: The Entity-Relationship Approach*, Morgan Kaufmann, San Mateo, CA (1990).
[11]Ullman, J. D., *Principles of Database and Knowledge-based Systems*, Vol. 1. Computer Science Press, Rockville, MD (1988).
[12]Diamond, S. D., *The Universal SQL Processor—version 1.1*, Research Center Internal Report RSI-96-002, JHU/APL, Laurel, MD (1996).
[13]Semmel, R. D., and Winkler, R. P., "Integrating Reengineered Databases to Support Data Fusion," *J. Syst. Software* **30**, 127–135 (1995).
[14]Cattell, R. G. G., *Object Data Management: Object-Oriented and Extended Relational Database Systems*, Revised Ed., Addison Wesley, Reading, MA (1994).
[15]Genesereth, M. R., and Ketchpel, S. P., "Software Agents," *Commun. ACM*, **37**(7), 48–53 and 147 (1994).

THE AUTHORS

RALPH D. SEMMEL is a member of APL's Principal Professional Staff and Supervisor of the Advanced Signal and Information Processing Group of the Milton S. Eisenhower Research and Technology Development Center. He received a B.S. in engineering from the U.S. Military Academy in 1978, an M.S. in systems management from the University of Southern California in 1981, an M.S. in computer science from The Johns Hopkins University in 1985, and a Ph.D. in computer science from the University of Maryland, Baltimore, in 1992. He has published more than 40 papers in the areas of database systems, artificial intelligence, and software engineering, and is currently investigating automated query formulation over distributed and heterogeneous systems. Dr. Semmel also serves as Chair of both the computer science and the information systems and technology programs in the Whiting School of Engineering Part-Time Programs in Engineering and Applied Science. His e-mail address is Ralph.Semmel@jhuapl.edu.

ELISABETH A. IMMER is a senior computer scientist in the Advanced Signal and Information Processing Group of the Milton S. Eisenhower Research and Technology Development Center at APL. She received a B.A. in psychology from The Johns Hopkins University in 1982, a B.S. in computer science from the University of Maryland Baltimore County in 1985, and an M.S. in computer science from The Johns Hopkins University in 1993. Before joining APL in 1994, Ms. Immer worked for General Physics Corporation, where she participated in the design and development of real-time distributed processing systems. Her current work focuses on automated query formulation, database reengineering, and legacy systems integration. Her e-mail address is Lis.Immer@jhuapl.edu.

DAVID P. SILBERBERG is a senior computer scientist in the Advanced Signal and Information Processing Group of the Milton S. Eisenhower Research and Technology Development Center at APL. He received a B.S. in computer science and an M.S. in electrical engineering and computer science from the Massachusetts Institute of Technology in 1981. He is currently pursuing research in the areas of intelligent database systems and distributed intelligent information systems with emphasis on agent-based architectures. He is also an architect of the next-generation DSCS Operations Centers for MILSATCOM. He has extensive experience with large-scale user interface systems, advanced object-oriented and relational database systems, client-server systems over wide- and local-area networks, expert systems, and functional programming systems at the Space Telescope Science Institute, NASA, Bell Communications Research, and the IBM Research Labs. His e-mail address is David.Silberberg@jhuapl.edu.

ROBERT P. WINKLER is a computer engineer in the Information Sciences and Technology Directorate of the U.S. Army Research Laboratory. He graduated magna cum laude from the University of Maryland, College Park, with a B.S. in computer science in 1988, and received an M.S. in computer science from The Johns Hopkins University in 1995. His current work focuses on automated query formulation, object-oriented programming, and database modeling and re-engineering techniques. His e-mail address is winkler@arl.mil.