Composite Information Systems: Resolving Semantic Heterogeneities

Michael Siegel
Stuart Madnick
Amar Gupta

March 1991

WP #3265-91-MSA
WP # CIS-91-21

Composite Information Systems: Resolving Semantic Heterogeneities

Michael Siegel
Stuart Madnick
Amar Gupta

March 1991

WP #
WP # CIS-91-21

Composite Information Systems: Resolving Semantic Heterogeneities

Michael Siegel
Stuart Madnick
Amar Gupta
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

The Composite Information System/Tool Kit (CIS/TK) is a prototype implementation strategy for accessing and integrating data from multiple disparate database systems. This prototype focuses on data access and schema integration, and in particular on the integration of disparate data semantics. In this paper we describe the CIS/TK prototype as well as research important for the further development of semantically integrated disparate database systems.

1 Introduction

The Composite Information Systems/Tool Kit (CIS/TK) is a framework for integrating heterogeneous database systems. Like other frameworks [LR82,LMR90,Te87,Tem89,SL90], CIS/TK provides access through an integrated schema defined over a portion of the underlying components. However, CIS/TK is different from other frameworks because of its emphasis on semantic connectivity [WM88] and its capability to integrate systems with disparate data semantics [BLN86].

CIS/TK was designed to assist in accessing data and integrating information from disparate systems to meet ad hoc needs and to answer "what if...?"questions encountered in decision support systems. As such, issues of portability, extensibility, and practicality were assigned higher importance than issues of updates and optimization. The main goal was to derive solutions to queries where the data reside on multiple disparate systems. The endeavor was to provide unintrusive access, allowing component systems to continue to operate autonomously. CIS/TK provides transparent access to multiple disparate databases through an integrated schema represented using an extension of the entity-relationship [Che76] model.

It should be emphasized that decision support systems are more dependent on efficient data access (e.g., query language and data models) and information-oriented operations (e.g., schema integration) than other types of computer based systems, such as transaction oriented systems. Various approaches to data access and schema integration have been described in the literature [BLN86,CRE87,She87,SG89,RS91,Tem89,YSDK90]. But these approaches inadequately address many of the semantic integration aspects of heterogeneous database systems. For example, it has been suggested by several researchers [LR82,SG89,Tem89] that translation routines be used to

resolve semantic differences among component systems; however, the application of these techniques to systems with changing component database semantics is questionable. Static schema integration will work for systems that do not change, but dynamic systems require special capabilities to accommodate evolution of component databases.

Further, in decision support system it is especially important to provide effective mechanisms for the representation of data semantics (i.e., metadata) and the detection and resolution of semantic conflicts (i.e., *semantic reconciliation*). These mechanisms can be utilized to reduce the dependency on translation routines, to simplify integrated system development, and to provide for more efficient and accurate systems operation.

This paper is organized in four sections. In the next section, salient features of the CIS/TK prototype are described. This description includes the architecture and the operations that provide the capabilities for query processing. Section 3 deals with research efforts in the area of semantic integration. The final section includes our conclusions and a discussion of open problems in semantic connectivity.

2 CIS/TK: The Implementation

The Composite Information Systems/Tool Kit is a repertoire of tools for overcoming the idiosyncrasies of heterogeneous database systems, developed at the Composite Information Systems Laboratory of MIT. Systems developed with this toolkit provide access to multiple disparate systems through a global representation of component database systems. The global representation is provided by a common data model and a common retrieval language. The global data model provides for *transparent* access to any number of component database systems.

CIS/TK utilizes a three-schema architecture [LR82] as shown in Figure 1. At the lowest level are the local database schemata. The interface between each local database and the global data model is provided by a Local Query Processor (LQP). Given a portion of a global query, the LQP translates both the semantics and the syntax of that query into the local database retrieval language. The data model for integrating the desired components of information, called the MERGE Data Model includes the global schema, the Global Retrieval Language (GRL), and the data catalogs (i.e., for data semantics and synonym translations). The Global Query Processor (GQP) utilizes the mappings between the local database schema and the global schema to parse a global query and route the subqueries to the appropriate LQPs. At the top layer is the application schema, the application's view of the global schema. This includes parts of the global schema and data not explicitly represented in the global schema but derivable from the data defined in the global schema. At this level, the Application Query Processor (AQP) provides aggregate operations and application domain specific methods for semantic integration.

2.1 The Local Query Processor

The GQP parses the global query into subqueries which are sent as messages to the appropriate local query processor. The design envisages one local query processor (LQP) for each database management system. The LQP provides syntactic and semantic translations of messages into the corresponding local database language. As depicted in Figure 1, the LQP receives a portion of the query presented by the application. The message sent to the LQP is of the following form:

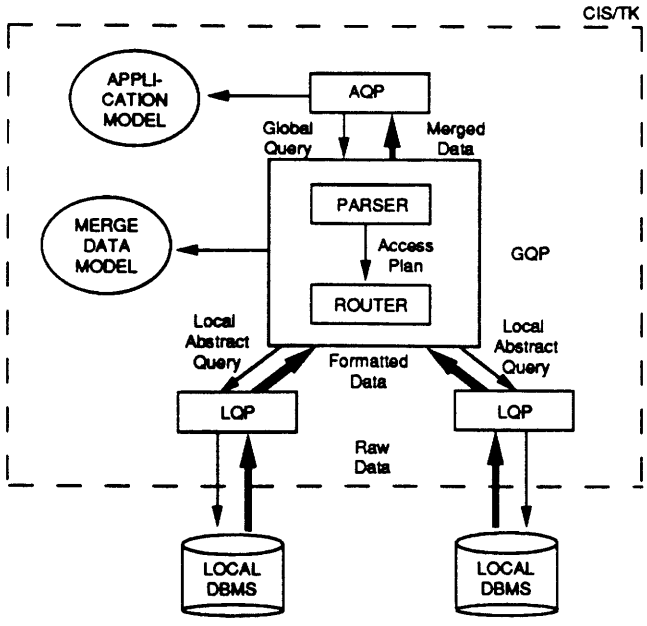

Figure 1: CIS/TK Three-Schema Architecture

(send-message *lqp* :get-data *data type catalog* (*table* (att_1 att_2 ... att_n)) *conditions*)

The message, *get-data*, is sent to the LQP object named in the message (i.e., *lqp*). As an example of the syntactic translation that occurs, assume that the local DBMS uses an SQL query language; this message will then be transformed into the query:

select (att_1 att_2 ... att_n)
from *table*
where *conditions*

Frequently additional syntactic and semantic transformation will be required to resolve complex tables references. The *data type catalog* defines the global data semantics. The LQP compares this catalog with its own to perform semantic translation (e.g., conversion of units). Synonym translations are also handled as data catalogs [Won89] in a manner similar to the use of an auxiliary database proposed in [LR82]. Both these semantic translation techniques will be discussed in greater detail later in Section 2.4.3.

After the query is executed by the local DBMS, data are returned to the LQP. The LQP must apply semantic translations so that the data conforms to the global representation specified in the data type catalog. The data are then returned by the LQP in the form:

((att_1 att_2 ... att_n) ((val_1 val_2 ... val_n)...(val_1 val_2 ... val_n))

which contains the list of attribute names followed by the set of tuple values associated with these attributes.

A LQP object is created for each logical table defined over the local database. The group of LQPs for a given database defines the logical table structure for that database. Each LQP must support four operations: *get-data, get-info, get-table, get-columns.* The *get-info* returns a textual description of the table, the *get-table* returns the table name and *get-columns* returns the column names. All of these operations are defined by the LQP. In the case of non-table databases (e.g., menu-based, network, data files), a LQP specifies the means for accessing the data defined by the logical table structure.

In addition to the standard commands, the LQP provides operations not supported by the local database system (e.g., arithmetic comparison operators other than equivalence such as $\geq$). The set of operators is defined by the GRL and supported through the use of an auxiliary database management system.

While all multidatabase systems provide some method for query and data translation over local DBMS one of the major problems is the large development time required in building new LQPs to accommodate new or modified local database management systems, menu-driven data interfaces, data files, etc. In CIS/TK, this process is relatively simple because most of the LQP code can be reused. Ultimately, information about the local database language and functional capabilities as inputs could be used to automatically generate an executable LQP.

2.2 The MERGE Data Model

The MERGE data model provides the conceptual view of the distributed database system through an integrated view of the underlying data. The three components of the data model are the global schema, the data catalog and the Global Retrieval Language (GRL). The global schema provides an integrated view of the local databases and defines the structural mappings between the global attributes and the local database components. The data catalogs define data semantics (e.g., synonyms, units, formats) and their use in query execution. The GRL provides access to the local databases through the global schema.

2.2.1 The Global Schema

The global schema defines the structural abstraction from the local schema to the global data model. The global data model representation has been built using an extended version of the entity-relationship model [Che76]. The extension permits the aggregation of data from multiple sources to produce single entities and relationships. Figure 2 shows an example of a global schema created to represent data available from two sources: a *Recruiting* database and an *Alumni* database. The global schema contains two entities and one relationship. The *Alumni* entity represents data about alumni of the Sloan School of Management, and the *Company* entity contains information about companies that are recruiting students from the Sloan School. The *works_for* relationship represents alumni that work for companies that are recruiting at the Sloan School.

The structural relationship between the underlying databases, as defined by the LQPs (i.e., the databases are single tables each with its own LQP) are shown along with the global entities. The link between the two entities is provided by the *name* attribute of the *Company* entity and the *company* attribute of the *Alumni* entity.

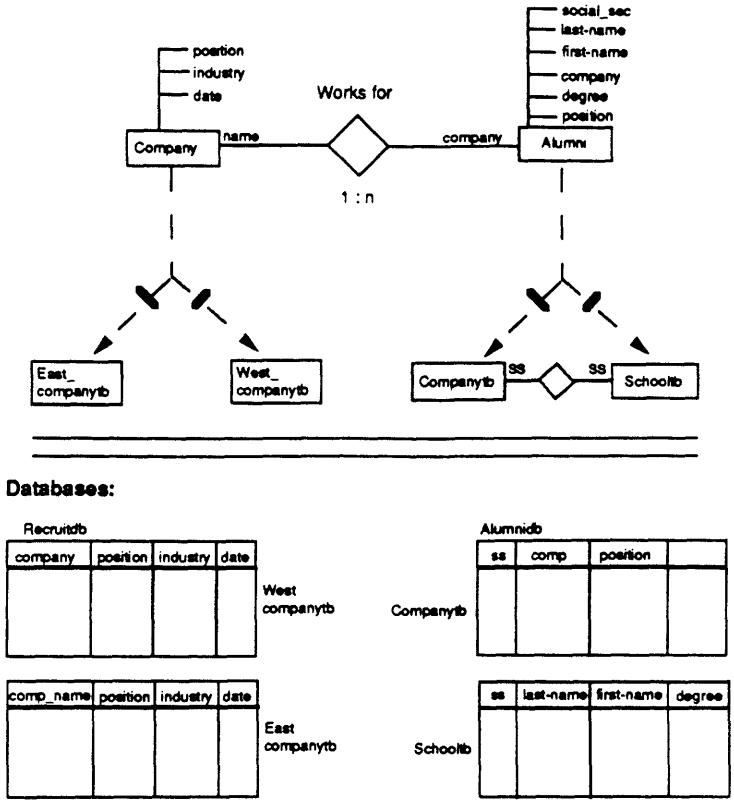

Figure 2: A Sample Global Schema

According to other researchers [BLN86,SL90,SP90,Te90], a number of *structural integration* issues must be resolved in order to provide a global schema based on a set of component databases. Structural integration includes solutions to problems in attribute naming and data fragmentation. Attribute name mappings are defined as simple expressions. For example, the *company* attribute of the *Alumni* entity is declared equal to the *name* attribute of the *Company* entity by the expression:

(= (alumni company) (company name))

Data fragmentation may be either horizontal or vertical. An example of horizontal fragmentation is the *West_companytb* and *East_companytb* relations representing the west coast companies and the east coast companies respectively. Integrating these relations requires their *concatenation* according to the statement:

(concatenate (recruitdb West_companytb) (recruitdb East_companytb))

This defines the union of the two relations. Vertical integration or merging (i.e., joining) of relations is defined for the Companytb and the Schooltb relations as follows:

(merge (alumnidb schooltb) (alumnidb companytb)
on (= (alumnidb schoolth ss) (alumnidb companytb ss)))

Merge and concatenate are used as operator names (rather than join and union) to emphasize that

(create-entity *name*
:attributes ((*global-attr*$_1$ (*lqp*$_1$ *table*$_1$ *attr*$_1$)
...
(*lqp*$_n$ *table*$_n$ *attr*$_n$))
global-attr$_1$ (*lqp*$_1$ *table*$_1$ *attr*$_1$)
...
(*lqp*$_n$ *table*$_n$ *attr*$_n$)))
table-relations ((merge ((*lqp*$_1$ *table*$_1$) ... *lqp*$_n$ *table*$_n$)
on *cond*)
(concatenate ((*lqp*$_1$ *table*$_1$) ... *lqp*$_n$ *table*$_n$))

(create-relation *name*
:entity-from *entity*$_1$
:entity-to *entity*$_2$
:join (= (*entity*$_1$ *attr*$_1$) (*entity*$_2$ *attr*$_2$))

Figure 3: Creating Global Entities and Relationships

these operations construct entities and relationships. The above implementation approach can also support other table-combining operations such as outer join and outer union. Additional examples of structural integration capabilities of CIS/TK are described in [Won89].

2.2.2 Schema Definition Language

Entities and relationships are viewed as objects. The format for entity and relationships declarations is shown in Figure 3. The *:attribute* slot is used to assign global names to attributes found in the local databases. The *:table-relations* slot is used to define relationships between tables that must be combined to form a global entity. Relationships are formed using the *create-relation* command with slots for two entities and the join condition defining the relationship between attributes of those entities.

2.2.3 Global Retrieval Language

The Global Retrieval Language (GRL) provides a common query language for access to the component databases represented in the global schema. The GRL is an SQL-like language[1] that can easily be mapped to more common relational query languages (e.g., SQL and, QUEL). One example of a GRL query is shown in Figure 4. Aggregates and other higher-order operators not defined in the GRL are implemented by the application query processor. Also shown in the figure is the output schema and an example data stream.

[1]The BNF can be found in [Won89].

Query:
"Find the positions and names of alumni who work for the Ford Company and the recruiting dates for that company."

GRL:
```
(join (select company (position date)
                      where (= name "Ford"))
(select alumni (last-name first-name degree)
on works_for)
```

Result:
```
((( company position) (company date) (alumni last-name) (alumni first-name) (alumni degree))
("engineer", "10/23/90", "John", "Smith", "MS 80"))
```

Figure 4: Sample GRL Query

2.3 The Global Query Processor

The Global Query Processor is the engine for executing global queries. The major functions of the GQP are query parsing, query routing, and construction of query solutions. In some cases, the GQP may need to access the data catalog to transform values in the query to those used by the database. Data returned by the LQPs are assembled by the GQP to provide the answer to the query. In the next section, the architecture and algorithms that define the GQP are described and data translations methods are examined.

2.4 The GQP Architecture

The major components of the GQP are the query parser and the query router. The parser decomposes the global query into subqueries over the local databases. The parser accepts a GRL query and creates an operator parse tree. The router executes the parse tree. The modules that relate to the parser and the router are shown in Figure 5.

2.4.1 The Query Parser

The parser begins with an error checking phase looking for syntactical and lexical errors. During the lexical checking phase, objects specified in the query (e.g., entities, attributes, and relationships) are checked against the global schema. Correct queries are expanded to include join conditions implied by the relationships in the query. Figure 6 shows the expanded version of the query shown in Figure 4. The *works_for* relationship in the original query is replaced with a join between the *name* attribute in the *Company* entity and the *company* attribute in the *Alumni* entity as specified in the schema definition (see Figure 3). Global attributes are mapped to local attributes. Merge and concatenate operations are added to the query as defined in the schema specification. Based on the specification shown in Figure 3, the query in Figure 4 was expanded to include the concatenate operation between the *East_companytb* and the *West_companytb* relations and the merge operation

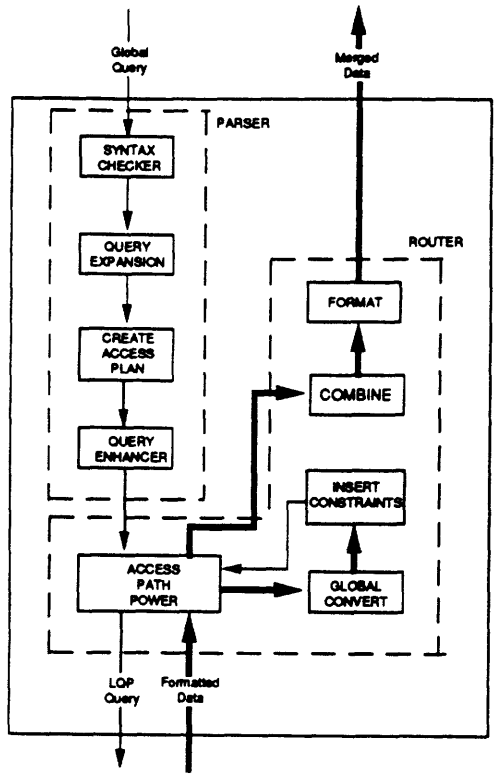

Figure 5: The GQP Architecture

between the *Companytb* and *Schooltb* relations (using the social security number as the merge attribute). The get-table function is supported by each LQP.

Because there may be overlapping and replicated data, the number of access plans may be very large. The selection of the optimal access plan involves minimizing the number of local database tables (i.e., LQPs) required to answer the query. The parser creates a query parse tree where the leaves are LQP names and the nodes of the tree define the operations for combining the data streams retrieved from the component databases.

By selecting an access plan on the basis of minimizing the number of tables (i.e., LQPs) needed to resolve the query, other optimization criteria are neglected. No weight is given to important conditions such as data accuracy, delivery costs, delivery time, and availability of data sources. It is shown later how the LQP Manager utilizes such selection conditions to choose the data source for responding to the LQP request.

2.4.2 The Query Router

The router executes the query by a depth-first reduction of the query parse tree. The first subquery, which is the left-most branch, is sent to the LQP and executed over the local database. The results from the execution of the first subquery are used as constraints in the next subquery. In other words, the query is modified using the results of previous subqueries as constraints in the next query. This process continues through the query parse tree. The router recognizes the three operations of *get-table, merge, and concatenate*; join operations are handled like merge operations but are defined between global entities. Required translations of data are handled based on the procedure described in Section 2.4.3.

```
(join (concatenate (get-table recruitdb wes_companytb (position date company)
                        where (= name "Ford"))
                   (get-table recruitdb east_companytb (position date comp_name)
                        where (= name "Ford"))
      (merge (get-table alumnidb alumnitb (last-name first-name position company ss))
             (get-table alumnidb schooltb (degree ss)
             on (= alumnidb alumnitb ss)
                (alumnidb schooltb ss))
      on (= (company name) (alumni company)))
```

Figure 6: Expanded Query

The combination of data streams is handled by the router. This query processing strategy does not take advantage of intradatabase binary operators (e.g., join and union) available at local database sites. When processing large amounts of data, this approach may significantly increase communication costs and query processing times. The use of component database capabilities to mitigate this problem is being investigated. Other approaches to query processing in similar environments are described in [Gup89,LR82,Te87].

2.4.3 Data Translation in CIS/TK

Data translation is required whenever there are discrepancies among local database data representations or between a local database and the global data representation. Data translations includes instance name translations, also called synonyms translation (e.g., from "IBM" to "International Business Machines"), data format translation (e.g., from EBCDIC to ASCII), unit translations (e.g., currency conversion), and scale translation (e.g., from thousands of units to units). Data translations must be performed at two distinct moments of the query routing algorithm: (i) on the way to; and (ii) on the way from the local database. Data translations are invoked through an object-based message passing mechanism.

Each LQP has its own query parser, whose function is two-fold: (i) to remove from the subquery any predicate that cannot be applied at the remote site; and (ii) to convert, when possible, any value in the query condition whose current GQP data representation (i.e., as defined in the *data type catalog*) is different from the representation at the local database. A predicate may not be applicable at the database remote site if the predicate (e.g., >, $\neq$, max) is not supported by the local system or the predicate value cannot be translated into the representation understood by the local database. For example, consider the following subquery (in the GRL) received by LQP, *DB1*:

(select *DB1* (Revenue Sales Date)
(and (= Company "Honda") (> Sales "20,000,000")))

which requests revenue, sales, and dates on which the company HONDA had sales greater than 20 billion. The GQP data representation includes a format with commas to mark thousands, a U.S. Dollar unit, and a scale of 1000. Local database *DB1* may have data in Japanese Yen with

a different scale and format. The translator has stored procedures to translate some of the data aspects such as the format and scale. However, it may not be able to translate all of the data aspects. It cannot, for example, translate the data currency because such translation requires additional information such as the date at which the exchange rate will be chosen and the kind of exchange rate that will be used.

The LQP Query Parser outputs a query that involves removing from the original subquery the predicates that cannot be applied at the remote site, and converting those values that are convertible. The query used in the above example would be parsed into:

(select DB1 (Revenue Sales Country Data)
(and (= Company "Honda Motor Co.")))

where "Honda" has been translated into the name used by the local system and the predicate, (Sales > "20,000,000"), has been removed because the local database does not support the > operation. The column, Country, was added by the translator because they are required to determine the currency of the data in DB1.

If no error occurs in obtaining the data from the local database, then the LQP invokes the Data Translator to convert the data obtained into the representation understood by the GQP. Finally, the LQP applies to the data any predicate of the original subquery that could not be performed by the local database (e.g., the > operation on the Sales attribute). This is achieved by the LQP downloading the data into a local auxiliary relational database management system.

The Synonyms Translation [Won89] facility is based on a GQP-maintained global synonym table, and a set of local synonym tables, one table for each LQP. The CIS/TK Data Translator [Rig90] classifies data by types, models the representation of a data type by a set of representation facets, and performs data representation translations. External function calls are used when a function is needed to describe the methods for obtaining the value of the representation facet (e.g., currency is determined by the Country of incorporation).

The synonym translation facility of CIS/TK is extensible; new data types can be easily created when new data types are added to the system. Data types can range from simple ones as integer representation to more complex types such as ones involving graphics, as long as a standard representation exists for each data type. Indeed, the use of a standard representation minimizes the number of additional translation routines required when a new type is added. The data type and the representation taxonomy are implemented through objects.

A data type catalog is associated with the global schema, as well as for each LQP. A LQP data type catalog describes the data format conventions adopted by the local system, and the GQP data type catalog reflects the data semantics assumed by the GQP. This latter catalog is passed as argument when sending a message to a LQP.

2.5 The Application Query Processor

The application processor interacts with the application by supplying a view of the global schema that consists of objects in the global schema plus objects derived from the global schema and knowledge provided by the application. At this level, the applications' request for data are processed

using the global schema and concepts inferred in the application domain. Aggregate and higher-order operators may be defined over the mappings from the global schema to the application schema.

One important issue in schema integration is the ability to join information that may come from different sources or from different data structures inside a single source. In conventional database management systems, these joins are performed using a primary or foreign key relationship. However, this relationship may not exist in a heterogeneous database environment. A technique called *concept inferencing* [WM88] is introduced at the application level for joining information from disparate databases even though a primary or foreign key join is not available. Concept inferencing provides application specific reasoning that identifies joinable attributes in such relations.

Concept inferencing has been developed in CIS/TK to assist in the joining of data from different tables where there is no obvious way to logically join the data. That is, there is no primary or foreign key relation between the data in the two tables. We have developed two application level approaches to inferring the logical connections between data in two tables. First, rules can be used to specify non-key relationships that imply logical connectivity. Secondly, we allow application specific rules for defining *derived attributes.* These derived attributes can then be used to compare data between systems. Examples of this methods of inferring join relationships are described in [Scu91].

2.6 The Communication Server

An integrated system may be composed of hundreds of systems each requiring a communication link from CIS/TK. However, an individual query may require access only to a limited number of these sources. The CIS/TK Communication Server [Mak90] facilitates access to all systems but creates and maintains links to sources for durations as short as the execution time of a single query. Within a single query, many independent requests may be made for data from a given source. The communication server maintains active links to these sources for the duration of the query while inactive sources are disconnected to save costs.

Each LQP, through the UNIX Interprocess Communication (IPC) facility, places a message in a queue which is constantly read by the servers, presently one server per source. All servers are identical to each other except for the source specific scripts [JPL*88] describing the interactions required for logon, error handling, data access, and disconnection. Data are returned through the communication server in a file. The file is read by the LQP, the header information is removed, additional data filtering is performed, and data are converted to match the data semantics described in the GQP data type catalog.

If an error occurs during communication, the Communication Server places an error message in the file. The LQP forwards the error message to the LQP Manager which then attempts to locate an alternative data source. For example, there may be two LQPs that provide equivalent sets of data. One may reside on the local machine while the other may be on a remote system. The administrator can specify that if the local database is unavailable, then the LQP for the remote database should be selected. If an error message is returned by the local database, then the LQP Manager automatically selects the remote database LQP.

2.7 Database Selection - The LQP Manager

The LQP Manager controls the selection of LQPs. Because data may be available from multiple sources, the LQP Manager was designed to permit the database administrator to exercise control over which database should be selected. This selection may depend on cost, reliability, availability, and performance of the data source.

The present version of CIS/TK allows the administrator to group LQPs into LQP sets [Tun90]. Each LQP within a set are structurally identical. For example, a LQP set may contain two LQPs, one for access to data at a remote source and one for access to local data. The LQP Manager maintains a selection order for these two LQPs. The LQP Manager might select the local database. Should an error message be returned by the LQP, the LQP Manager would select the remote database LQP. In addition to controlling the selection of LQPs, the LQP Manager can keep track of which LQPs are active, and the data requests handled by active LQPs.

2.8 Pseudo-Databases: Data Caching

Data caching involves local storage of portions of remote databases. The cached data may be sufficient for answering most queries and are more likely to be available than remote source data. The present implementation of data caching provides for the specification of *pseudo databases* that mimics the structure of the remote source and thus can be included in the same LQP set. Data retrieved from the remote source are automatically loaded into the pseudo database during normal query execution. Should the remote source be inaccessible, the LQP Manager can select the pseudo database.

There are several limitations in the present implementation of data caching. First, there is no mechanism to guarantee that data in the pseudo database will remain current during updates to the remote database; this problem is partially overcome by providing recent data through periodically purging the pseudo database. Second, there is no guarantee that data needed by a query were cached in during previous query processing operations. Finally, the user is not alerted to the fact that the results of a query were partially based on cached data. In spite of these limitations, the concept of data caching in heterogeneous database systems has been demonstrated.

2.9 Implementation Status

CIS/TK, Version 3.1 was completed in August 1990. It runs on an AT&T 3B2/500 computer under UNIX Systems V and is implemented in LISP with data structures specified using an object-oriented rule-based extension of Common Lisp, called Knowledge-Oriented Representation Language (KO-REL). The Communication Server is implemented in C and UNIX Shell.

The operational capabilities of CIS/TK have been demonstrated using six disparate database systems including Oracle SQL and Informix database management systems and several menu-based interfaces. This technology is being utilized for integration of multiple systems at a sponsor sites.

3 Research Highlights

In addition to refinements to the existing prototype, current research efforts are directed towards enhancing the capabilities of this integration framework. Research in the area of semantic integra-

tion examines the use of a metadata approach to resolving semantic conflicts, the identification of the data source in multidatabase systems, and the development of an extensible schema integration tool. We briefly describe these issues and results obtained so far.

3.1 A Metadata Approach to Resolving Semantic Conflicts

In traditional data processing technology, much of the meaning of data is expressed in the names of tables, fields, and variables, and in written documentation. For correct results, a program or programmer must ensure that, for data utilized, appropriate meaning exist in the context of their use. Metadata provides one mechanism to achieve this objective [SM89a,SM89b,SM91]. Data type descriptions in a normal database catalog are typically extended to include information about the different semantics of each type of data. This information may range from units of measure (e.g., price represented as a percent or a fractional quantity) to the interdependency relationships among data. A preliminary approach to data semantic representation and manipulation was described in Section 2.4.3. This research has developed a more general approach involving a rule-based language that allows system designers to specify the data semantics for data sources and the semantic requirements for their application (i.e., the application semantic view [SM91]).

Algorithms have also been developed for the comparisons of these data semantic specifications. These algorithms [SM91] can be used to determine if a data source can supply semantically meaningful data to an application. These methods can be used in dynamic system environments where the data semantics may change; they can also be used to resolve semantic conflicts that occur, and to assist the user in understanding the solution to a query. These methods are non-intrusive on the existing applications and data sources.

3.2 Identifying the Data Source in Multidatabase Systems

As systems are integrated, the characteristics of the individual sources are unified to provide transparent access to the federation. Unfortunately, the user of such a multidatabase system may not be able to obtain information about the source of the data (i.e., which of the component databases supplied the data). This information is important when making judgments concerning data quality and accuracy. A method that provides tags or metadata defining the source of the data should be included in the integrated system. This is particularly important and challenging if the data goes through various intermediate processing steps and databases. A data-source tagging methodology has been developed to provide metadata containing information about both the source of data and the intermediate sources used in selecting data[WM90]. This data-source tagging capability is based on a data-source algebra, which is an extension of the relational algebra.

A prototype implementation of a the data-source tagging facility has been developed but has not been integrated in the present version of CIS/TK. This prototype is described in detail in [Yua90].

3.3 Development of an Extensible Schema Integration Tool

Information systems integration requires sound pragmatic algorithms for schema integration and schema extensibility[RS91]. Schema integration and extensibility must address all types of descriptive information, not just the structural information expressed in a schema (e.g., entities, relationships, attributes, constraints, generalization). Changes or extensions to the component's

schema might include new data types for defining accuracy, versioning policy, security level, timeliness, charge-for-use, and person-responsible. Current research addresses requirements for four types of schema flexibility: *scheduling flexibility* which allows integration to be done in different stages and along different levels of expertise; *schema evolution* allowing for incremental development and evolution of the component schema and the integrated schema; *model extensibility*, which captures descriptive information that is not standardized across the industry; and *method extensibility* adding new or site-specific integration methods.

4 Conclusions and Future Research

CIS/TK is an integration framework that provides connectivity among disparate information systems. It provides a practical approach for building extensible systems in which changes can be made in the components of the federation and the autonomy of local database is fully preserved. CIS/TK is unintrusive, and does not require not requiring modification of any of the constituent systems. By providing new methods for integration of semantically disparate data sources and by supporting transparent access to data in heterogeneous environments, CIS/TK is geared towards query-oriented decision support applications.

Multidatabase systems will be dynamic in nature and the current version of CIS/TK is incorporated with the ability to adapt to changes in the component systems. Automatic methods for schema integration and LQP generation are being explored and an architecture for an extensible schema integration tool is begin created. Our work in semantic reconciliation strategies is leading to the development of new representations for data semantics, as well as methods for manipulating these semantics between component systems, the global schema, and the application.

Since the use of conventional query processing methods in large, distributed, and heterogeneous environments may involve long processing times, options for optimization are being investigated. One option relies on cost statistics for the creating the query access plan, the ordering algorithm for the query parse tree, and the optimal strategy for using local database operations. Improved performance may also come from the use of data caching. For this, one needs to develop better methods for operating on the pseudo databases.

The use of application-dependent rules for integration, as described in Section 2.5, must be complemented with new methods of knowledge representation and application dependent reasoning. The refinement of the application query processor to support larger number of application views and more extensive application-driven integration methods is also being examined.

One of the short-term goals is to utilize the CIS/TK framework to integrate spreadsheets and database management systems running in the PC environment. The prototype described in this paper has provided a practical platform for testing these and other new approaches to schema integration and the use of connectivity in large evolving information system environments.

References

[BLN86] C. Batini, M. Lenzerini, and S. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[Che76] P. Chen. The entity relationship mode - toward a unified view of data. *Transactions on Database Systems*, 1(1), 1976.

[CRE87] B. Czejdo, M. Rusinkiewicz, and D. Embley. An approach to schema integration and query formulation in federated database systems. In *Proceedings of the Third International Conference on Data Engineering*, February, 1987.

[Gup89] A. Gupta, editor. *Integration of Information Systems: Bridging Heterogeneous Database Systems.* IEEE Press, N.Y., N.Y., 1989.

[JPL*88] G. Jakobson, G. Piatetsky-Shapiro, C. Lafond, M. Rajinikanth, and J. Hernandez. CALIDA: a system for integrated retrieval from multiple heterogeneous databases. In *Proceedings of the Third International Conference on Data and Knowledge Engineering*, pages 3–18, Jerusalem, Israel, 1988.

[LMR90] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, September 1990.

[LR82] T. Landers and R. Rosenberg. An overview of multibase. In *Distributed Data Bases*, pages 153–183, North Holland, 1982.

[Mak90] B. Makatiani. *Enhancement of Communication Server in the Composite Infromation System/Tool Kit (CIS/TK) Prototype.* Technical Report #CIS-90-18, Composite Information Systems Laboratory, Sloan School of Management, Massachusetts Institute of Technology, 1990.

[Rig90] B. Rigaldies. *Technologies and Policies for the Development of Composite Information Systems in Decentralized Organizations.* Technical Report #CIS-90-05, Composite Information Systems Laboratory, Sloan School of Management, Massachusetts Institute of Technology, 1990.

[RS91] A. Rosenthal and M. Siegel. An architecture for practical metadata integration. In *Submittion to The 17th Conference on Very Large Data Bases*, 1991.

[Scu91] B. Scurlock. *An Intelligent Mechanism for Non-Primary Foreign Key Joins in the CIS/TK System.* Technical Report, Composite Information Systems Laboratory, Sloan School of Management, Massachusetts Institute of Technology, 1991.

[SG89] A. Sheth and S. Gala. Attribute relationships: an impediment in automating schema integration. In *Position Papers: NSF Workshop on Heterogeneous Databases*, December 11-13, 1989.

[She87] A. Sheth. Heterogeneous distributed databases: Issues in integration, Tutorial on heterogeneous databases. In *Proceedings of the Conference on Data Engineering*, 1987.

[SL90] A. Sheth and J. Larson. Federated databases: architectures and integration. *ACM Computing Surveys*, September 1990.

[SM89a] M. Siegel and S. Madnick. *Identification and Resolution of Semantic Conflicts Using Metadata.* Technical Report #3102-89-MSA, Sloan School of Management, Massachusetts Institute of Technology, 1989.

[SM89b] M. Siegel and S. Madnick. *Schema Integration Using Metadata.* Technical Report #3092-89-MS, Sloan School of Management, Massachusetts Institute of Technology, (Also NSF Workshop on Heterogeneous Database Systems, 1989), 1989.

[SM91] M. Siegel and S. Madnick. A metadata approach to resolving semantic conflicts. In *Submittion to The 17th Conference on Very Large Data Bases*, 1991.

[SP90] S. Spaccapietra and C. Parent. *View Integration: A Step Forward in Solving Structural Conflicts.* Technical Report, INRIA, 1990.

[Te87] M. Templeton and et al. Mermaid - a front-end to distributed heterogeneous databases. In *Proceedings of the IEEE*, pages 695–708, 1987.

[Te90] G. Thomas and et al. Heterogeneous distributed database systems for production use. *ACM Computing Surveys*, September 1990.

[Tem89] M. Templeton. Schema integration in Mermaid. In *Position Papers: NSF Workshop on Heterogeneous Databases*, December 11-13, 1989.

[Tun90] M. Tung. *Local Query Processor Manager for the Composite Information System/Tool Kit.* Technical Report #CIS-90-18, Composite Information Systems Laboratory, Sloan School of Management, Massachusetts Institute of Technology, 1990.

[WM88] R. Wang and S. Madnick, editors. *Connectivity Among Information Systems.* Volume 1 of *Composite Information Systems (CIS) Project*, Massachusetts Institute of Technology, 1988.

[WM90] R. Wang and S. Madnick. Data-source tagging. In *Proceeding from the Very Large Database Conference*, 1990.

[Won89] T. Wong. *Data Connectivity for the Composite Information Systems/Tool Kit.* Technical Report #CIS-89-03, Composite Information Systems Laboratory, Sloan School of Management, Massachusetts Institute of Technology, June 1989.

[YSDK90] C. Yu, W. Sun, S. Dao, and D. Keirsey. Determining relationships among attributes for interoperability of multi-database systems. In *Position Papers: Workshop on Multidatabases and Semantic Interoperability*, November 2-4, 1990.

[Yua90] Y. Yuan. *The Design and Implementation of System P: A Polygen Database Management System.* Technical Report #CIS-90-07, Composite Information Systems Laboratory, Sloan School of Management, Massachusetts Institute of Technology, 1990.