1990

A SOURCE TAGGING THEORY FOR HETEROGENEOUS DATABASE SYSTEMS

Y. Richard Wang
Massachusetts Institute of Technology

Stuart E. Madnick
Massachusetts Institute of Technology

A SOURCE TAGGING THEORY FOR HETEROGENEOUS DATABASE SYSTEMS

Y. Richard Wang
Stuart E. Madnick
Composite Information Systems Laboratory
Sloan School of Management
Massachusetts Institute of Technology

ABSTRACT

Many important Management Support Systems require seamless access to and integration of multiple heterogeneous database systems. This paper studies heterogeneous database systems from the source perspective. It aims at addressing issues such as the following: (1) Where is the data from? (2) Which intermediate data sources were used to arrive at that data? Specifically, it presents a polygen model for resolving the Data Source Tagging and Intermediate Source Tagging problems. In addition, it presents the necessary and sufficient condition for source tagging. Source knowledge is important for many reasons. It enables users to apply their own judgment to the credibility of the information. It enables users to rationalize and reconcile data inconsistencies. It enables system designers to develop access charge systems. It enables an application user to adjust data. It enables a system to interpret data semantics more accurately. In sum, it justifies having source tagging capabilities as a required functionality for future heterogeneous database systems.

1. INTRODUCTION

The rapidly increasing complexity, interdependence, and competition in the global market has profoundly changed how corporations operate and how they align their information technology for competitive advantage in the marketplace. It has been argued (Madnick 1989) that improved communications capability and data accessibility will lead to systems integration both within and across organizational boundaries in the 1990s. This will lead to vastly improved group communications and, more importantly, the integration of business processes across traditional functional, product, and geographic lines. The integration of business processes, in turn, will accelerate demands for more effective Management Support Systems for product development, product delivery, and customer service and management (Rockart and Short 1989). Increasingly, many important Management Support Systems require seamless access to and integration of multiple heterogeneous database systems. These types of heterogeneous database systems have been referred to as *Composite Information Systems* (Wang and Madnick 1988; Madnick, Siegel and Wang 1990).

In this paper, we study heterogeneous database systems from the multiple source perspective. In particular, we address the following two issues: (1) Where is the data from? (2) Which intermediate data sources were used to arrive at that data?

It is interesting to note that these issues have not been directly addressed to date. Contemporary heterogeneous database systems strive to encapsulate the heterogeneity of the underlying databases in order to produce an illusion that all information originates from a single anonymous source. This illusion has been referred to as *location transparency* or *location independence* (Date 1990). In our field studies of actual needs, we have found that although users want the simplicity of *location transparency* for query formulation, they also want to know the source of each piece of data retrieved (e.g., Source: Corporate Customer Database). This source knowledge may be important to them for many reasons, as exemplified below:

- Source knowledge enables users to apply their own judgment to the credibility of the information. In our discussions with financial analysts, several exclaimed that data retrieved from heterogeneous systems would be totally useless to them unless they know its source.

- Source knowledge enables users to rationalize and reconcile data inconsistencies. For example, the attribute "Return on Equity" for Reuters Holdings PLC has different values when retrieved from I. P. Sharp's Disclosure database (based in Toronto) compared with Finsbury's Dataline database (based in London). It is likely that different accounting practices in Canada and the United Kingdom would explain the difference in values. Furthermore, since Reuters is a UK-based company, the Dataline database may be more appropriate. In short, knowing the data sources helps users to rationalize and reconcile the data inconsistencies as well as make their own judgment.

- Source knowledge enables system designers to develop access charge systems. In a major financial institution, analysts have access to multiple external commercial databases. With data source knowledge, system designers could develop access charge systems to help analysts select the optimal set of data with minimum cost for their work. Furthermore, this capability was specifically noted as an important requirement for internal charge back schemes. For example, different charges could be associated with data actually returned to the user versus intermediate data used in the query process.

- Source knowledge enables an application user to adjust data. In a manufacturing firm, production data was extracted from plants across the country in order to produce production reports on the hourly basis. With source knowledge, an application user could adjust production data due to time zone differences, particularly on the days when standard time is switched to daylight saving time and vice versa (Arizona, for instance, does not participate in the daylight saving time program).

- Source knowledge enables a system to interpret data semantics more accurately. In a Composite Information System where data from multiple sources are merged without human intervention, the system could use source knowledge to make reference to the right metadata dictionary in order to reconcile semantic heterogeneity of data from different sources.

Indeed, the importance to know the data source justifies it to be a required functionality for future heterogeneous database systems. It is this source knowledge that we focus on in this research. Our research contributions can be summarized as follows:

(1) We have developed a *polygen model*[1] to study heterogeneous database systems from the multiple (*poly*) source (*gen*) perspective. The polygen model provides a precise characterization of the source tagging problem and a solution including a polygen algebra, a data-driven query translation mechanism, and the necessary and sufficient condition for source tagging. A concrete example is also provided to illustrate the basic mechanism.

(2) We have developed the polygen model as a direct extension of the relational model to the multiple database setting with source tagging capabilities, thus the polygen model enjoys all of the strengths of the traditional relational model.

(3) We have established a theoretical foundation for resolving many other critical research issues. For example, the polygen algebra can be extended to address other basic attributes associated with data, such as the temporal aspect of data. Users normally want to know not only where the data is from but also when the data was collected and how they were collected. Furthermore, as we noted earlier, knowing the data source will enable a user or a query processor to interpret the data semantics more accurately, knowing data source credibility will enable the user or the query processor to further resolve potential conflicts amongst the data retrieved from different sources, and knowing data access cost will enable system designers to develop access charge systems.

1.1 Source Tagging Example

In preparing a special report on the top ten graduate programs in Information Systems (Computerworld, October 30, 1989), a member of the Computerworld staff called MIT's Sloan School to get the names of CEOs who graduated from Sloan School with an MBA degree. Suppose that the following SQL polygen query

```
SELECT ONAME, CEO
FROM PORGANIZATION, PALUMNUS
WHERE CEO = ANAME AND DEGREE = "MBA"
```

was created given a polygen schema derived from the MIT Alumni Database and Company Database below. For expository purposes, the prefix "P" is used to denote a *polygen scheme* in the polygen schema.

Polygen Schema	Alumni Database (AD): Alumni Schema	Company Database (CD): Company Schema
PORGANIZATION(ONAME, INDUSTRY, CEO, HEADQUARTERS)	BUSINESS(BNAME, IND)	FIRM(FNAME, CEO, HQ)
PFINANCE (ONAME, YEAR, PROFIT)		FINANCE(FNAME, YR, PROFIT)
PALUMNUS(AID#, ANAME, DEGREE, MAJOR)	ALUMNUS(AID#, ANAME, DEG, MAJ)	
PCAREER (AID#, ONAME)	CAREER(AID#, BNAME)	

In the table above, a firm in the Company Database has a name, a CEO, and is headquartered in a city. It discloses yearly financial information on profit. Each alumnus in the Alumni Database is uniquely identified through an alumnus identification number (AID#). Associated with each alumnus is a name, a degree, and a major. An alumnus may have careers in many businesses, and each business is associated with an industry. Attribute mapping relationships between the polygen schema and the Alumni Database and the Company Database are shown in Section 2.

The query result contains only the names of the CEOs which originated from the Company Database, but the query processor also needs to access the Alumni Database (an intermediate source) in order to select those CEOs who received an MBA degree. Moreover, the query processor needs to "know" that it has to merge the BUSINESS and the FIRM relations first before joining the CEO attribute with the ANAME attribute. As such, the challenge is to develop not only a polygen model but also a polygen algebra and the algorithms for a polygen query processor capable of resolving the data and intermediate source tagging problems for any arbitrary polygen query. Tagging the Company Database name accurately to the result is referred to as the *Data Source Tagging* problem. Tagging the intermediate use of the Alumni Database accurately is referred to as the *Intermediate Source Tagging* problem.

1.2 Research Issues and Goals

The data and intermediate source tagging problems have not been specifically addressed in the past. We have reviewed a broad range of literature and examined various research prototypes of heterogeneous distributed database systems, for example MULTIBASE in the United States (Smith et al. 1981), PRECI* in England (Deen, Amin, and Taylor 1987a, 1987b), and MRDSM in France (Litwin et al. 1982, 1986). In addition, we have surveyed more than forty U.S. commercial systems offering partial solutions to the heterogeneous distributed database problem, including Data Integration's MERMAID, Cincom's SUPRA, Metaphor's DIS, and TRW's Data Integration Engine (Gupta et al. 1989). To the best of our knowledge, none of these systems have dealt with these source tagging problems.

Two related issues, among others, need to be addressed in source tagging: (1) What kind of polygen model should be created in order to tag multiple sources explicitly? (2) What is the relationship between the polygen model and the polygen query processing facility?

Most heterogeneous distributed database systems adopt one of the following four data models (Hull and King 1987; Peckham and Maryanski 1988): the Relational Model, the Functional Data Model, the Semantic Database Model, or the Entity Relationship Model. Each data model has merits for its intended purposes.[2] We selected the relational model. Based on the relational model, we define a *polygen model* for resolving the data and intermediate source tagging problems.

One of the key activities in formulating composite information is to translate a polygen query into a set of local queries, which in turn are routed to the corresponding local databases. Query translation has been approached through view definition in most heterogeneous distributed database systems. A symbolic query transformation technique has also been proposed (Rusinkiewicz and Czejdo 1985; Czejdo, Rusinkiewicz and Embley 1987) in which a syntax-directed parser converts a polygen query and transformation rules[3] into multiway trees. Through subtree matching, these multiway trees are further translated into local queries, given the specific source and target language syntax descriptions.

As we will discuss later, our query translation mechanism differs from the above mentioned techniques in two important aspects:

(1) Instead of the view definition approach which encodes the procedure for translating a polygen query into the corresponding local queries, our mechanism separates the mapping algorithm from the mapping data. As a result, adding a new database to the existing system does not require modifying the existing procedural view definitions.

(2) Instead of the symbolic query transformation technique which tackles a broad range of nodal query languages at a higher level, our mechanism focuses on the mapping between a polygen algebraic expression and the corresponding local operations, permitting entities (and attributes) in local databases to overlap one another.

1.3 Research Background and Assumptions

At the MIT Sloan School's Composite Information Systems Laboratory (CISL), we have developed a heterogeneous database system which has access to three internal databases (the Alumni Database, the Placement Database, and the Student Database) and three external commercial databases (Finsbury's Dataline and I.P. Sharp's Disclosure and Currency). The query processor architecture is depicted in Figure 1. Briefly, the Application Query Processor translates an end-user query into a polygen query for the Polygen Query Processor (PQP) based on the user's application schema. The PQP in turn translates the polygen query into a set of local queries based on the corresponding polygen schema, and routes them to the Local Query Processors (LQP). The details of the mapping and communication mechanisms between an LQP and its local databases is encapsulated in the LQP. To the PQP, each LQP behaves as a local relational system. Upon return from the LQPs, the retrieved data are further processed by the PQP in order to produce the desired composite information.

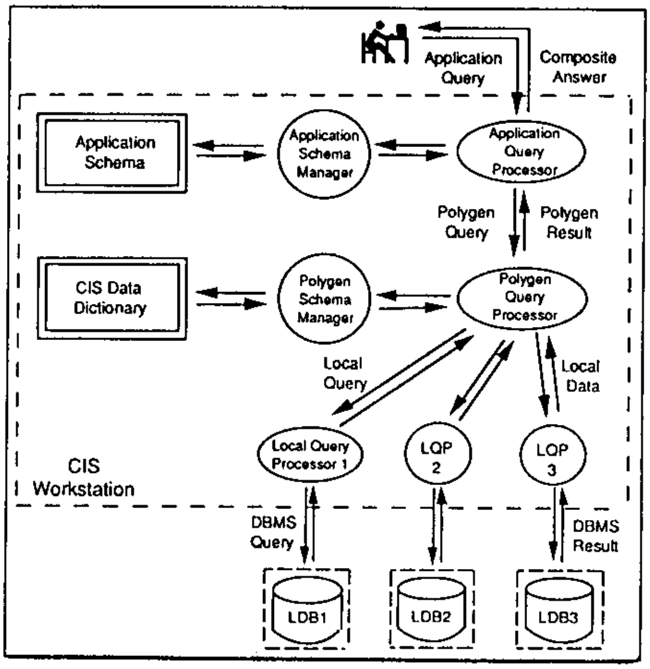

Figure 1. The Query Processor Architecture

Many critical problems need to be resolved in order to provide a seamless solution to the end-user. These problems include source tagging, query translation, schema integration (Batini, Lenzirini and Navathe 1986; Elmasri, Larson and Navathe 1987), inter-database instance matching (Wang and Madnick 1989), domain mapping (Shin 1988; DeMichiel 1989), and semantic reconciliation (Wang and Madnick 1989). We focus on the first two problems and make the following assumptions in this paper:

- The local schemata and the polygen schema are all based on the relational model.
- Sources are tagged after data has been retrieved from each database.
- Schema integration has been performed, and the attribute mapping information is stored in the polygen schema.
- The inter-database instance identifier mismatching problem (e.g., IBM versus I.B.M or social security identification number versus employee identification number) has been resolved and the information is available for the PQP to use.
- The domain mismatch problem such as unit ($ versus ¥), scale (in billions versus in millions), and description interpretation ("expensive" versus "$$$"; "Chinese Cuisine" versus "Hunan or Cantonese") has been resolved during schema integration and the information is also available to the PQP.

Section 2 defines the polygen model. Polygen query translation is presented in Section 3. Section 4 provides a detailed example of the basic polygen query processing mechanism. The necessary and sufficient condition of source tagging is presented in Section 5. Finally, concluding remarks are made in Section 6.

2. THE POLYGEN MODEL

To present the polygen model more concretely, we first exemplify the attribute mapping relationships between the polygen schema and their corresponding local schemata in the form {(database, relation, attribute),...} for the source tagging example described in Section 1.

The PORGANIZATION Polygen Scheme

ONAME	INDUSTRY	CEO	HEADQUARTERS
{(AD, BUSINESS, BNAME), (CD, FIRM, FNAME)}	{(AD, BUSINESS, IND)}	{(CD, FIRM, CEO)}	{(CD, FIRM, HQ)}

The PFINANCE Polygen Scheme

ONAME	YEAR	PROFIT
{(CD, FINANCE, FNAME)}	{(CD, FINANCE, YR)}	{(CD, FINANCE, PROFIT)}

The PALUMNUS Polygen Scheme

AID#	ANAME	DEGREE	MAJOR
{(AD, ALUMNUS, AID#)}	{(AD, ALUMNUS, ANAME)}	{(AD, ALUMNUS, DEG)}	{(AD, ALUMNUS, MAJ)}

The PCAREER Polygen Scheme

AID#	ONAME
{(AD, CAREER, AID#)}	{(AD, CAREER, BNAME)}

We now define the polygen model. Let PA be a polygen attribute in a polygen scheme P, LS a local scheme in a local database LD, and LA a local attribute in LS. For example, ONAME is a polygen attribute in the polygen scheme PORGANIZATION, BUSINESS a local scheme in the local database AD, and BNAME a local attribute in the local scheme BUSINESS.

Let MA be the set of local attributes corresponding to a PA, i.e.,

MA = {(LD, LS, LA) | (LD, LS, LA) denotes a local attribute to the corresponding PA}.

For ONAME in the PORGANIZATION polygen scheme,

MA = {(AD, BUSINESS, BNAME), (CD, FIRM, FNAME)}.

A polygen scheme P is defined as

$P = \langle (PA_1, MA_1), \ldots, (PA_n, MA_n) \rangle$ where n is the number of attributes in P.

For the polygen scheme PORGANIZATION in the above scenario,

PORGANIZATION = ⟨(ONAME, {(AD, BUSINESS, BNAME), (CD, FIRM, FNAME)}), (INDUSTRY, {(AD, BUSINESS, IND)}), (CEO, {(CD, FIRM, CEO)}), (HEADQUARTERS, {(CD, FIRM, HQ)})⟩

A polygen schema is defined as a set $\{P_1, \ldots, P_N\}$ of N polygen schemes. In the above scenario, the polygen schema consists of the following schemes:

{PORGANIZATION, PFINANCE,PALUMNUS, PCAREER}

A polygen domain is defined as a set of ordered triplets. Each triplet consists of three elements: the first is a datum drawn from a simple domain in an LQP. The second is a set of LDs denoting the local databases from which the datum originates. The third is a set of LDs denoting the intermediate local databases whose data led to the selection of the datum.

A polygen relation p of degree n is a finite set of time-varying n-tuples, each n-tuple having the same set of attributes drawing values from the corresponding polygen domains. A cell in a polygen relation is an ordered triplet $c=(c\langle d\rangle, c\langle o\rangle, c\langle i\rangle)$ where $c\langle d\rangle$ denotes the datum portion, $c\langle o\rangle$ the originating portion, and $c\langle i\rangle$ the intermediate source portion. Two polygen relations are union-compatible if their corresponding attributes are defined on the same polygen domain.

Note that P contains the mapping information between a polygen scheme and the corresponding local relational schemes. In contrast, p contains the actual time-varying data and their originating sources. A polygen scheme P and a polygen relation p may be used synonymously without confusion. The data and intermediate source tags for p are updated along the way as polygen algebraic operations are performed.

2.1 The Polygen Algebra

Let attrs(p) denote the set of attributes in p. For each tuple t in a polygen relation p, let $t\langle d\rangle$ denote the data portion, $t\langle o\rangle$ the originating source portion, and $t\langle i\rangle$ the intermediate source portion. If $x \in$ attrs(p), $X = \{x_1 \ldots, x_j \ldots, x_J\}$ is a sublist of attrs(p), then let p[x] be the column in p corresponding to attribute x, let p[X] be the columns in p corresponding the sublist of attributes X, let t[x] be the cell in t corresponding to attribute x, and let t[X] be the cells in t corresponding to the sublist of attributes X. As such, $p[x]\langle o\rangle$ denotes the originating source portion of the column corresponding to attribute x in polygen relation p while $t[X]\langle i\rangle$ denotes the intermediate source portion of the cells corresponding to the sublist of attributes X in tuple t. On the other hand, p[x] denotes the column corresponding to attribute x in polygen relation p inclusive of the data, originating source, and intermediate source portions while t[X] denotes the cells corresponding to the sublist of attributes X in tuple t inclusive of the data, originating source, and intermediate source portions. Note that the "[ ]" notation in *project* p[x] should not be confused with the operation p[x=y].

The five orthogonal algebraic operators in the polygen model are defined as follows:

Project. If p is a polygen relation, and $X = \{x_1 \ldots, x_j \ldots, x_J\}$ is a sublist of attrs(p), then

$$p[X] = \{t' \mid t' = t[X] \text{ if } t \in p \wedge t[X]\langle d\rangle \text{ is unique};$$

$$t'\langle d\rangle = t_i[X]\langle d\rangle, t'[x_j]\langle o\rangle = t_i[x_j]\langle o\rangle \cup \ldots \cup t_k[x_j]\langle o\rangle \; \forall x_j \in X, t'[x_j]\langle i\rangle = t_i[x_j]\langle i\rangle \cup \ldots \cup t_k[x_j]\langle i\rangle \forall x_j \in X$$

$$\text{if } t_i, \ldots, t_k \in p \wedge t_i[X]\langle d\rangle = \ldots = t_k[X]\langle d\rangle\}.$$

Cartesian product. If p_1 and p_2 are two polygen relations, then

$(p_1 \times p_2) = \{t_1 \circ t_2 \mid t_1 \in p_1$ and $t_2 \in p_2$ where $\circ$ denotes concatenation$\}$.

Restrict. If p is a polygen relation, $x \in$ attrs(p), $y \in$ attrs(p), and θ is a binary relation, then

$$p[x\,\theta\,y] = \{t' \mid t'\langle d\rangle = t\langle d\rangle, t'\langle o\rangle = t\langle o\rangle, t'[w]\langle i\rangle = t[w]\langle i\rangle \cup t[x]\langle o\rangle \cup t[y]\langle o\rangle \; \forall\, w \in \text{attrs}(p),$$

$$\text{if } t \in p \wedge t[x]\langle d\rangle \,\theta\, t[y]\langle d\rangle\}.$$

Union. If p_1 and p_2 are two polygen relations and both have degree n, $t_1 \in p_1$, $t_2 \in p_2$, then

$$(p_1 \cup p_2) = \{t' \mid t' = t_1 \text{ if } t_1\langle d\rangle \in p_1 \wedge t_1\langle d\rangle \notin p_2;$$

$$t' = t_2 \text{ if } t_2\langle d\rangle \notin p_1 \wedge t_2\langle d\rangle \in p_2;$$

$$t'\langle d\rangle = t_1\langle d\rangle, t'\langle o\rangle = t_1\langle o\rangle \cup t_2\langle o\rangle, t'i = t_1\langle i\rangle \cup t_2\langle i\rangle \text{ if } t_1\langle d\rangle = t_2\langle d\rangle\}$$

Difference. Let $p\langle o\rangle$ denote the union of all the $t\langle o\rangle$ sets in p, and $p\langle i\rangle$ denote the union of all the $t\langle i\rangle$ sets in p. If p_1 and p_2 are two polygen relations and both have degree *n*, then

$$(p_1 - p_2) = \{t' \mid t'\langle d\rangle = t\langle d\rangle,\ t'\langle o\rangle = t\langle o\rangle,\ t'[w]\langle i\rangle = t[w]\langle i\rangle \cup p_2\langle o\rangle \cup p_2\langle i\rangle\ \forall\, w \in attrs(p), \text{ if } t \in p_1 \text{ and } t\langle d\rangle \notin p_2\}.$$

The intermediate source portion, $t\langle i\rangle$, is updated by *Restrict* and *Difference*. The *Restrict* operation selects the tuples in a polygen relation which satisfies the $[x\ \theta\ y]$ condition. As such, the originating local databases of the x and y attribute values are added to the $t\langle i\rangle$ set in order to signify their mediating role. Since *Select* and *Join* are defined through *Restrict*, they also update $t\langle i\rangle$.

Difference selects a tuple in p_1 to be a tuple in $(p_1 - p_2)$ if the data portion of the tuple in p_1 is not identical to those of the tuples in p_2. Since each tuple in p_1 needs to be compared with all the tuples in p_2, it follows that all the originating sources of the data in p_2 should be included in the intermediate source set of $(p_1 - p_2)$, as $t'\langle i\rangle = t\langle i\rangle \cup p_2\langle o\rangle \cup p_2\langle i\rangle$ denotes.

In contrast, *Project, Cartesian Product,* and *Union* do not involve intermediate local databases as the mediating sources. Other traditional operators can be defined in terms of the above five operators. The most common are *Join*, *Select*, and *Intersection*. *Join* and *Select* are defined as the restriction of a *Cartesian product*. *Intersection* is defined as the project of a join over all the attributes in each of the relations involved in the Intersection.

In order to process a polygen query, we also need to introduce the following new operators to the polygen model: *Retrieve, Coalesce, Outer Natural Primary Join, Outer Natural Total Join,* and *Merge*.

A local database relation needs to be retrieved from a local database to the PQP first before it is considered as a *PQP base relation*. This is required in the polygen model because a polygen operation may require data from multiple local databases. Although a *PQP base relation* can be materialized dynamically like a view in the conventional database system, for conceptual purposes, we define it to reside physically in the PQP.[4] The *Retrieve* operation is defined as an LQP *Restrict* operation without any restricting condition.

Coalesce and *Outer Natural Join* have been informally introduced by Date to handle a surprising number of practical applications. *Coalesce* takes two columns as input, and coalesces them into one column. An *Outer Natural Join* is an outer join with the join attributes coalesced (Date 1983).

We define an *Outer Natural Primary Join* as an *Outer Natural Join* on the primary key of a polygen relation. For example, the *Outer Natural Primary Join* for PORGANIZATION is an *Outer Natural Join* on ONAME. An *Outer Natural Total Join* is an *Outer Natural Primary Join* with all the other polygen attributes in the polygen relation coalesced as well. In the PORGANIZATION example, an *Outer Natural Total Join* would perform an *Outer Natural Primary Join* on ONAME followed by a number of *Coalesce* operations on INDUSTRY, CEO, and HEADQUARTERS. *Merge* extends *Outer Natural Total Join* to include more than two polygen relations. It can be shown that the order in which *Outer Natural Total Join* is performed over a set of polygen relations in a *Merge* is immaterial.

Since *Coalesce* can be used in conjunction with the other polygen algebraic operators to define the *Outer Natural Primary Join*, *Outer Natural Total Join*, and *Merge*, we define *Coalesce* as the sixth orthogonal primitive of the polygen model.

Coalesce. Let $\bullet$ denote the coalesce operator. If p is a polygen relation, $x \in attrs(p)$, $y \in attrs(p)$, $z = attrs(p) - \{x, y\}$, and w is the coalesced attribute of x and y, then

$$p[x \bullet y{:}w] =$$

$$\begin{aligned}\{t' \mid\ & t'[z]=t[z],\ t'[w]\langle d\rangle=t[x]\langle d\rangle,\ t'[w]\langle o\rangle=t[x]\langle o\rangle \cup t[y]\langle o\rangle,\\ & t'[w]\langle i\rangle=t[x]\langle i\rangle \cup t[y]\langle i\rangle, \text{ if } t[x]\langle d\rangle=t[y]\langle d\rangle;\\ & t'[z]=t[z],\ t'[w]\langle d\rangle=t[x]\langle d\rangle,\ t'[w]\langle o\rangle=t[x]\langle o\rangle,\\ & t'[w]\langle i\rangle=t[x]\langle i\rangle, \text{ if } t[y]\langle d\rangle=nil;\\ & t'[z]=t[z],\ t'[w]\langle d\rangle=t[y]\langle d\rangle,\ t'[w]\langle o\rangle=t[y]\langle o\rangle,\\ & t'[w]\langle i\rangle=t[y]\langle i\rangle, \text{ if } t[x]\langle d\rangle=nil\}.\end{aligned}$$

Note that in a heterogeneous distributed environment, the values to be coalesced may be inconsistent. That issue is beyond the scope of this paper; we have assumed that inter-database instance mismatching problems will be resolved before the coalesce operation is performed (Wang and Madnick 1989).

We have presented the polygen model and the polygen algebra. The algebra will be used in Section 4 to compose information with *data source tags* and *intermediate source tags*. In order to do that, it is necessary to know the process of translating a polygen query into a query execution plan. This process is presented below.

3. POLYGEN QUERY TRANSLATION

For the SQL polygen query presented in Section 1, a corresponding polygen algebraic expression for the SQL polygen query is as follows:

Table 1. The Polygen Operation Matrix for the Example Polygen Algebraic Expression

PR	OP	LHR	LHA	θ	RHA	RHR
R(1)	Select	PALUMNUS	DEGREE	=	"MBA"	nil
R(2)	Join	R(1)	ANAME	=	CEO	PORGANIZATION
R(3)	Project	R(2)	ONAME, CEO	nil	nil	nil

Table 2. A Half-Processed IOM Generated by Pass One of the POI Algorithm

PR	OP	LHR	LHA	θ	RHA	RHR	EL
R(1)	Select	ALUMNUS	DEG	=	"MBA"	nil	AD
R(2)	Join	R(1)	ANAME	=	CEO	PORGANIZATION	PQP
R(3)	Project	R(2)	ONAME, CEO	nil	nil	nil	PQP

((PALUMNUS [DEGREE = "MBA"])[ANAME =CEO] PORGANIZATION)[ONAME, CEO]

In general, the PQP takes a polygen algebraic expression as an input and produces a query execution plan for retrieving data from the local databases and formulating composite information. Three components are involved in this process: the *Algebraic Analyzer*, the *Polygen Operation Interpreter*, and the *Query Optimizer*, as shown in Figure 2.

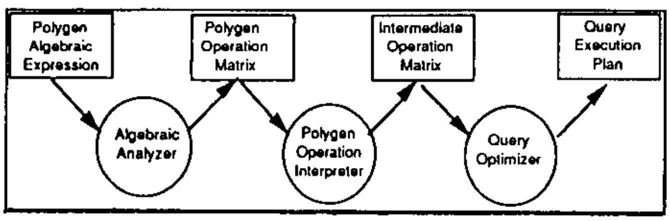

Figure 2. The Polygen Query Translation Process

The *Algebraic Analyzer* parses a polygen algebraic expression and generates a *Polygen Operation Matrix*. For example, the *Polygen Operation Matrix* for the example polygen algebraic expression is presented in Table 1. The first row indicates that a *Select* operation should be performed on the Left-Hand Relation (LHR) PALUMNUS using the θ relation "=" between the Left-Hand Attribute (LHA) DEGREE and the Right-Hand Attribute (RHA) "MBA." In this case, there is no need for a Right-Hand Relation (RHR). The result is denoted by R(1), a Polygen Relation (PR). Details of the *Algebraic Analyzer* is beyond the scope of this paper.

Next the *Polygen Operation Interpreter* expands the *Polygen Operation Matrix* and generates an *Intermediate Operation Matrix*. In addition to the *Polygen Operation Matrix*, the *Polygen Operation Interpreter* takes the polygen schema as an input in order to produce the *Intermediate Operation Matrix*. A two-pass *Polygen Operation Interpreter* algorithm, pass one dealing with the left-hand side and pass two the right-hand side of polygen operations has been developed (Wang and Madnick, 1990). We illustrate the algorithm below.

The input to pass one is a *Polygen Operation Matrix* as Table 1 exemplifies and an empty *Intermediate Operation Matrix*. The output from pass one (and input to pass two) is a half-processed *Intermediate Operation Matrix*, as shown in Table 2. The *execution location* (EL) of an operation depends on where the data resides. Note that when the execution location is an LQP (e.g., AD in the first row of Table 2), it is also used as the *originating source tag* for each of the cell, c(o), of the polygen base relation (R(1) in this case).

In this example, pass one recognizes that the first row of Table 1 contains the polygen relation PALUMNUS whose attribute DEGREE corresponds to {(AD,ALUMNUS,DEG)}. Thus, LS=ALUMNUS, LA=DEG, LD=AD, and the tuple ⟨R(1), Select, ALUMNUS, DEG, =, "MBA", nil, AD⟩ is inserted into the first row of Table 2 which is empty initially. The second and third row of Table 1 are mapped into Table 2 without any change, and the PQP is assigned as the execution location because the left-hand relations, R(1) and R(2), reside in the PQP.

Table 3. An Intermediate Operation Matrix for the Example Polygen Algebraic Expression

PR	OP	LHR	LHA	q	RHA	RHR	EL
R(1)	Select	ALUMNUS	DEG	=	"MBA"	nil	AD
R(2)	Retrieve	BUSINESS	nil	nil	nil	nil	AD
R(3)	Retrieve	FIRM	nil	nil	nil	nil	CD
R(4)	Merge	R(2), R(3)	nil	nil	nil	nil	PQP
R(5)	Join	R(1)	ANAME	=	CEO	R(4)	PQP
R(6)	Project	R(5)	ONAME, CEO	nil	nil	nil	PQP

In general, the *left-hand relation* is either a relation defined by the polygen schema or a R(#) denoting a polygen base relation (or a polygen relation derived from other polygen base relations). In the first case, the *left-hand relation* may correspond to either one or multiple local relations. If only one local relation exists, then the polygen operation is mapped into the local operation, and the corresponding LQP is assigned as the execution location. If multiple local relations exist, then these relations are retrieved and merged first before the requested operation is performed by the PQP. The second case involves an update of the R(#) from the Polygen Operation Matrix to the corresponding R(#) in the half-processed Intermediate Operation Matrix. In addition, the PQP is assigned as the execution location because R(#) resides in the PQP.

Continuing with the example, pass two processes the right-hand side of Table 2 and produces Table 3.

The first row of Table 2 is copied over to Table 3 directly because the *right-hand relation* is non-existent (nil) and no other mapping is required. The second row of Table 2 is a *Join* between R(1) and PORGANIZATION which corresponds to the BUSINESS and FIRM local relations. As such, these two relations are retrieved (Rows 2-3, Table 3), merged (Row 4, Table 3), and followed by a *Join* with R(1) of Table 2 — mapping to R(1) of Table 3. Finally, the third row of Table 2 maps to the sixth row of Table 3.

In general, three possibilities exist for the *right-hand relation*: (1) a relation defined by the polygen schema, (2) a R(#) denoting a polygen base relation or a polygen relation derived from other polygen base relations, and (3) non-existent (nil). The second and third cases follow the second case of pass one closely. The first case is also similar to pass one unless both the left- and right-hand sides require LQP operations. That being the condition, separate LQP operations need to be performed first before the requested polygen operation is performed. In addition, the polygen to local attribute mapping assigned in pass one needs to be reversed.

Finally, the *Query Optimizer* examines the *Intermediate Operation Matrix* and generates a query execution plan. Details of the *Query Optimizer* is also beyond the scope of this paper. Note also that the local database systems will most likely have their own high-level query languages, such as SQL, with their own optimization methods. As such, the algebraic expressions could be synthesized before sending to the corresponding local database systems.

4. EXAMPLE SOURCE TAGGING IN THE PQP

We now illustrate the processing of the example polygen query assuming the following local relations using Table 3 as a query execution plan.

The Alumnus Relation (AD)

AID#	ANAME	DEG	MAJ
012	John McCauley	MBA	IS
123	Bob Swanson	MBA	MGT
345	James Yao	BS	EECS
456	Dave Horton	MBA	IS
567	John Reed	MBA	MGT
678	Bob Horton	SF	MGT
789	Ken Olsen	MS	EE

The Career Relation (AD)

AID#	BNAME
012	Citicorp
123	Genentech
345	Oracle
456	Ford
567	Citicorp
678	BP
789	DEC

The Business Relation (AD)

BNAME	IND
IBM	High Tech
CitiCorp	Banking
Oracle	High Tech
Ford	Automobile
DEC	High Tech
BP	Energy
Genentech	High Tech

The Firm Relation (CD)

FNAME	CEO	HQ
AT&T	Robert Allen	NY, NY
Banker's Trust	Charles Sanford	NY, NY
CitiCorp	John Reed	NY, NY
Ford	Donald Peterson	Dearborn, MI
IBM	John Ackers	Armonk, NY
Apple	John Sculley	Cupertino, CA
Oracle	Lawrence Ellison	Belmont, CA
DEC	Ken Olsen	Maynard, MA
Genentech	Bob Swanson	So. San Francisco, CA

The Finance Relation (CD)

FNAME	YR	PROFIT
AT&T	1989	-1.7 bil
Banker's Trust	1989	648 mil
CitiCorp	1989	1.7 bil
Ford	1989	5.3 bil
IBM	1989	5.5 bil
Apple	1989	400 mil
Oracle	1989	43 mil
DEC	1989	1.3 bil
Genentech	1989	21 mil

The first row of Table 3 indicates that the operation ALUMNUS[DEG = "MBA"] should be executed by the Alumni Database LQP and the result is shown in Table 4. Note that the data source cell is the set {AD} which is taken directly from the EL column in Table 3. The intermediate source is an empty set.

Table 4. Results of the Operation of Row 1, Table 3

AID#	ANAME	DEG	MAJ
012, {AD}, {}	John McCauley, {AD}, {}	MBA, {AD}, {}	IS, {AD}, {}
123, {AD}, {}	Bob Swanson, {AD}, {}	MBA, {AD}, {}	MGT {AD}, {}
456, {AD}, {}	Dave Horton, {AD}, {}	MBA, {AD}, {}	IS, {AD}, {}
567, {AD}, {}	John Reed, {AD}, {}	MBA, {AD}, {}	MIT, {AD}, {}

Table 5. Results of the Operation of Row 2 through Row 4, Table 3

ONAME	INDUSTRY	CEO	HEADQUARTERS
IBM, {AD, CD},{AD, CD}	High Tech, {AD},{AD, CD}	John Ackers, {{CD},{AD, CD}	NY, {CD},{AD, CD}
CitiCorp, {AD, CD},{AD, CD}	Banking, {AD},{AD, CD}	John Reed, {CD},{AD, CD}	NY, {CD},{AD, CD}
Oracle, {AD, CD},{AD, CD}	High Tech, {AD},{AD, CD}	Lawrence Ellison, {CD},{AD, CD}	CA, {CD},{AD, CD}
Ford, {AD, CD},{AD, CD}	Automobile, {AD},{AD}	Donald Peterson, {CD},{AD, CD}	MI, {CD},{AD, CD}
DEC, {AD, CD},{AD, CD}	High Tech, {AD},{AD, CD}	Ken Olsen, {CD},{AD, CD}	MA, {CD},{AD, CD}
BP, {AD},{AD}	Energy, {AD},{AD}	nil, {}, {AD}	nil, {}, {AD}
Genentech, {AD, CD},{AD, CD}	High Tech, {AD},{AD}	Bob Swanson, {CD},{AD, CD}	CA, {CD},{AD, CD}
AT&T, {CD},{CD}	nil, {}, {CD}	Robert Allen, {CD},{CD}	NY, {CD},{CD}
Banker's Trust, {CD},{CD}	nil, {}, {CD}	Charles Sanford, {CD},{CD}	NY, {CD},{CD}
Apple, {CD},{CD}	nil, {}, {CD}	John Sculley, {CD},{CD}	CA {CD},{}

Next the BUSINESS and FIRM relations are retrieved from the Alumni Database and the Company Database respectively, then merged in the PQP. The result is shown in Table 5. The *Outer Natural Primary Join, Outer Natural Total Join,* and *Coalesce* operations for generating Table 5 is shown in Appendix A.

The PQP now joins Table 4 with Table 5 and produces Table 6.

Finally, Table 6 is projected to form Table 7 which contains only those organizations and their CEOs who graduated from MIT's Management School with an MBA degree.

Several observations can be made from the example:

(1) The information of Genentech is from the Alumni Database and Company Database.

(2) The information that Genentech's CEO is Bob Swanson comes from the Company Database, and the Alumni Database has served as an intermediate source in obtaining the information.

(3) The polygen query processor can derive the information that Genentech is from the Alumni Database's BNAME relation and Company Database's FNAME relation from the polygen schema and the information of (ONAME, {AD, CD}). This information can be shown to the user upon request with a simple mapping.

In this simple example, the data source information can be obtained by inspection from the polygen schema. The intermediate source information is not observable from the polygen schema. In a federated database system with hundreds of databases in which a polygen query is optimized to select only the relevant databases for information retrieval, the data source information observed from the polygen schema is a superset of the result obtained by the PQP. We now turn our attention to other theoretical issues of source tagging.

V. THE NECESSARY AND SUFFICIENT CONDITION OF SOURCE TAGGING

The polygen model in presented in Section 2 is based on the assumption that sources are tagged at the cell level. Two fundamental issues are addressed here: (1) Does the *closure* property hold for the polygen algebra? (That is, does a polygen operation over a set of polygen relations always produce a polygen relation?) (2) How many other potential approaches exist for source tagging? We address these two issues through the following lemma and theorem. Specifically, we show that although there are four conceivable ways to tag sources, the closure property holds if and only if sources are tagged by cell.

⟨Lemma⟩ **In extending the Relational Model to a polygen model, there exists four ways to source tagging: by cell, by tuple, by attribute, and by relation.**

Table 6. Results of the Operation of Row 5, Table 3

AID#	ANAME	DEG	MAJ	ONAME	INDUSTRY	CEO	HEADQUARTERS
123, {AD}, {AD, CD}	Bob Swanson, {AD}, {AD, CD}	MBA, {AD}, {AD, CD}	MGT, {AD}, {AD, CD}	Genentech, {AD, CD}, {AD, CD}	High Tech, {AD}, {AD, CD}	Bob Swanson, {CD}, {AD, CD}	CA, {CD}, {AD, CD}
567, {AD}, {AD, CD}	John Reed, {AD}, {AD, CD}	MBA, {AD}, {AD, CD}	MIT, {AD}, {AD, CD}	Citicorp, {AD, CD}, {AD, CD}	Banking, {AD},{AD, CD}	John Reed, {CD}, {AD, CD}	NY, {CD}, {AD, CD}

Table 7. Results of the Operation of Row 6, Table 3

ONAME	CEO
Genentech, {AD, CD}, {AD, CD}	Bob Swanson, {CD},{AD, CD}
Citicorp, {AD, CD}, {AD, CD}	John Reed, {CD},{AD, CD}

Since the polygen model is based on the Relational Model, the granularity of a data object to be tagged cannot be coarser than a relation because a relation is the basic unit of an algebraic operation. On the other hand, the granularity cannot be finer than a cell because a cell is the smallest unit of a relation. In addition, source tags are deleted or updated by algebraic operators, all of them perform operations either by tuple (*Cartesian product, union, difference, and restrict*) or by attribute (*project, coalesce).* It follows that sources may be tagged by cell, by tuple, by attribute, or by relation.

⟨Theorem⟩ **The <u>closure property</u> holds if and only if source tagging is by cell.**

Let E denote the set of results obtained from all possible combinations of algebraic operations defined in a polygen model. Let e_1 and e_1' denote two base polygen relations. Let f denote an algebraic operation, $e_2 = f(e_1)$ if f is *project, restrict, or coalesce*; $e_2 = f(e_1, e_1')$ if f is *Cartesian product, union, or difference*. Similarly, let $e_{k+1} = f(e_k)$ for some $e_k \in$ E if f is *project, restrict, or coalesce*; $e_{k+1} = f(e_k, e_k')$ for some $e_k \in$ **E**, $e_k' \in$ **E**, if f is *Cartesian product, union, or difference*. We now show that, by induction, if source tagging is by cell, then the *closure property* holds, i.e, e is a polygen relation defined by the polygen model $\forall$ e $\in$ E. Only the originating source portion is shown below; the intermediate source portion can be shown by the same token. For consistency, we use the notations developed in Section 2.

⟨Proof⟩ **Part 1: The <u>closure property</u> holds → Source tagging is by cell.**

Suppose that the *closure property* holds and source tagging is not by cell. It follows, by the lemma, that there exists a polygen model in which source tagging is by relation, by attribute, or by tuple. If source tagging is by relation, then a relation in this polygen model can be expressed as (e, e⟨o⟩). Consider the Cartesian product of $(e_1, e_1\langle o\rangle) \times (e_2, e_2\langle o\rangle)$. By definition, the operation yields $\{t_1 \circ t_2\colon t_1 \in e_1 \text{ and } t_2 \in e_2$, where $\circ$ denotes concatenation$\}$. However, the result cannot be expressed in the form of (e, e⟨o⟩) because the originating source tags from t_1 may be different from t_2. It follows that source tagging by relation is not feasible. If source tagging is by attribute, then an attribute in this polygen model can be expressed as (e[x], e[x]⟨o⟩). Consider union. By definition, $t'\langle d\rangle=t_1\langle d\rangle$, $t'[x]\langle o\rangle=t_1[x]\langle o\rangle \cup t_2[x]\langle o\rangle$ if $t_1\langle d\rangle=t_2\langle d\rangle$. However, the result cannot be expressed in the form of (e[x], e[x]⟨o⟩) because $t_1[x]\langle o\rangle$ may be different from $t_2[x]\langle o\rangle$. It follows that source tagging by attribute is not feasible. If source tagging is by tuple, then a tuple in this polygen model can be expressed as (t, t⟨o⟩). Consider Cartesian product. By the similar argument, the result cannot be expressed in the form of (t, t⟨o⟩). It follows that source tagging by tuple is not feasible. By contradiction, we conclude that the proposition is true.

Part 2: Source tagging is by cell → The <u>closure property</u> holds.

The premise that source tagging is by cell justifies the usage of the polygen model presented in Section 2 in the following proof. By the model's definition, t⟨o⟩ is the set of the originating source $\forall$ t $\in e_1$, $\forall e_1 \in$ **E**, and the *closure property* holds $\forall e_1 \in$ **E**.

Assuming that the *closure property* holds for $\forall e_k \in$ **E**, we show that the *closure property* also holds $\forall e_{k+1} \in$ **E**.

For projection, $e_{k+1} = e_k[X]$. Two cases need to be considered: (1) t[X]⟨d⟩ is unique and (2) $t_i[X]\langle d\rangle=\ldots=t_k[X]\langle d\rangle$. In the first case, c⟨o⟩ remains the same for all c $\in t[x_j]\langle o\rangle$ $\forall x_j \in$ X. In the second case, $t'[x_j]\langle o\rangle=t_i[x_j]\langle o\rangle \cup\ldots\cup t_k[x_j]\langle o\rangle$ $\forall x_j \in$ X. Since the *closure property* holds for

$t_i[X]\langle o\rangle = ... = t_k[X]\langle o\rangle$, thus the *closure property* also holds for $t'[x_j]\langle o\rangle\ \forall\ x_j \in X$. It follows that the *closure property* holds for $e_{k+1} = e_k[X]$.

For *Cartesian product*, $e_{k+1} = (e_k \times e'_k) = \{t_1 \circ t_2 : t_1 \in e_k \text{ and } t_2 \in e'_k\}$, where $\circ$ denotes concatenation. For *difference*, $e_{k+1} = (e_k - e'_k) = \{t : t \in e_k, td \notin e'_k\}$. For *restrict*, $e_{k+1} = e_k(x\ \theta\ y) = \{\ t : t \in e_k\ \ (t[x]d\ \theta\ t[y]d)\}$. Since $t\langle o\rangle$ remains the same in *Cartesian product, difference, and restrict*, it follows that the *closure property* holds for $e_{k+1} = (e_k \times e'_k)$, $e_{k+1} = (e_k - e'_k)$ and $e_{k+1} = e_k(x\ \theta\ y)$. The *closure property* holds for *union* and *coalesce* by the same token. From the Principle of Mathematical Induction, we conclude that the proposition is true.

6. CONCLUDING REMARKS

We have presented a polygen model for resolving the *Data Source Tagging* and *Intermediate Source Tagging* problems. The *polygen model* research addresses issues in heterogeneous distributed database systems from the "*where*" perspective — a perspective that, to the best of our knowledge, has not been studied to date. Furthermore, we have presented a data-driven query translation mechanism for mapping a polygen algebraic expression into a set of intermediate polygen operations dynamically.

This research has provided us with a theoretical foundation for further investigation of many other critical research issues in heterogeneous distributed systems, for example the cardinality inconsistency problem which is inherent in heterogeneous database systems.[5] It also enables us to interpret information from different sources more accurately. By storing the metadata about each of the data sources in the PQP, many domain mismatch, semantic reconciliation, and data conflict problems could be resolved systematically using the data and intermediate source tags. Furthermore, other polygen models can be developed for heterogeneous distributed database systems based on the Entity Relationship Model, the Functional Data Model, and the more recent object-oriented models (Shaw and Zdonik 1990).

The data source and intermediate source information can be very valuable to the user as well as the polygen query processor in formulating cost-effective, customized, and credible composite information in a federated database environment. As more and more important applications require seamless access to and integration of multiple heterogeneous database systems both within and across organizational boundaries, the capabilities in formulating cost-effective, customized, and credible composite information will also become increasingly critical to corporations.

7. REFERENCES

Batini, C.; Lenzirini, M.; and Navathe, S. B. "A Comparative Analysis of Methodologies for Database Schema Integration." *ACM Computing Survey*. Volume 18, Number 4, December 1986, pp. 323-364.

Codd, E. F. "Extending the Database Relational Model to Capture More Meaning" *ACM Transactions on Database Systems*, Volume 4, Number 4, 1979, pp. 397-434.

Czejdo, B.; Rusinkiewicz, M.; and Embley, D. "An Approach to Schema Integration and Query Formulation in Federated Database Systems." In *Proceedings of the Third International Conference on Data Engineering*, Los Angeles, California, February, 1987, pp. 477-484.

Date, C. J. *An Introduction to Database Systems, 1*, Fifth Edition. Reading, Massachusetts: Addison Wesley, 1990, pp. 621-627

Date, C. J. "The Outer Join." In *Proceedings of the Second International Conference on Databases*, Cambridge, England, September, 1983, pp. 76-106.

Deen, S. M.; Amin, R. R.; and Taylor M. C. "Data Integration in Distributed Databases." *IEEE Transactions on Software Engineering*, SE-13, Volume 7, July 1987a, pp. 860-864.

Deen, S. M.; Amin, R. R.; and Taylor, M. C. "Implementation of a Prototype for PRECI*." *Computer Journal*, Volume 30, Number 2, 1987b, pp. 157-162.

DeMichiel, L. G. "Performing Operations Over Mismatched Domains." In *Proceedings of the Fifth International Conference on Data Engineering*, Los Angeles, California, February 1989.

Elmasri R.; Larson J.; and Navathe, S. "Schema Integration Algorithms for Federated Databases and Logical Database Design." Submitted for publication, 1987.

Gupta, A.; Madnick, S.; Poulsen, C.; and Wingfield, T. "An Architecture Comparison of Contemporary Approaches and Products for Integrating Heterogeneous Information Systems." Massachusetts Institute of Technology, Sloan School of Management, IFSRC # 110-89, 1989.

Hull, R., and King, R. "Semantic Database Modeling: Survey, Applications, and Research Issues." *ACM Computing Surveys*, Volume 19, Number 3, September 1987, pp. 201-259

Litwin, W.; Boudenant, J.; Esculier, C.; Ferrier, A.; Glorieux, A. M.; LaChimia, J.; Kabbaj, K.; Moulinoux, C.; Rolin, P.; and Strangret, C. "SIRIUS System for Distributed Data Management." In *Proceedings of International Symposium on Distributed Databases*, Berlin, West Germany, 1982, pp. 311-366.

Litwin, W., and Abdellatif, A. "Multidatabase Interoperability." *IEEE Computer*, December 1986, pp. 10-18.

Madnick, S. E. "Information Technology Platform for the 1990s." In *Management in the 1990s Research Program Final Report*. Cambridge, Massachusetts: Massachusetts Institute of Technology, Sloan School of Management, 1989, pp. 19-48.

Madnick, S. E.; Siegel, M.; and Wang, Y. R. "The Composite Information Systems Laboratory (CISL) Project at MIT." To appear in *IEEE Data Engineering*, June 1990.

Peckham, J., and Maryanski, F. "Semantic Data Models." *ACM Computing Surveys*, Volume 20, Number 3, 1988, pp. 153-189.

Rockart, J. F., and Short, J. E. "IT in the 1990s: Managing Organizational Interdependence." *Sloan Management Review*, Volume 30, Number 2, Winter 1989.

Rusinkiewicz, M., and Czejdo, B. "Query Transformation in Heterogeneous Distributed Database Systems." In *Proceedings of the Fifth International Conference on Distributed Computing*, Denver, Colorado, 1985.

Shaw, G., and Zdonik, S. B. "A Query Algebra for Object-Oriented Databases." In *Proceedings of the Sixth International Conference on Data Engineering*, February 1990.

Shin, D. G. "Semantics for Handling Queries with Missing Information." In J. I. DeGross and M. H. Olson (eds.), *Proceedings of the Ninth International Conference on Information Systems*, Minneapolis, Minnesota, 1988, pp. 161-167.

Smith, J. M.; Bernstein, P. A.; Dayal, U.; Goodman, N.; Landers, T.; Lin, K. W. T.; and Wong, E. *Multibase — Integrating Heterogeneous Distributed Database Systems*. 1981 National Computer Conference, 1981, pp. 487-499.

Templeton, M.; Brill, D.; Hwang, A.; Kameny, I.; and Lund, E. "An Overview of the MERMAID System — A Frontend to Heterogeneous Databases." In *Proceedings IEEE EASCON*, Washington, DC, September, 1983.

Wang, Y. R., and Madnick, S. "Connectivity Among Information Systems." *Composite Information Systems (CIS) Project 1*, 1988.

Wang, Y. R., and Madnick, S. "The Inter-Database Instance Identification Problem in Integrating Autonomous Systems." In *Proceedings of the Fifth International Conference on Data Engineering*, February 6-10, 1989.

Wang, Y. R., and Madnick, S. E. "A Polygen Model for Heterogeneous Database Systems: The Source Tagging Perspective." In *Proceedings of the Eleventh International Conference on VLDB*, Brisbane, Australia, August 1990.

8. ENDNOTES

1. To highlight the source tagging problems, the phrase "polygen model" will be used in the paper instead of the conventional "global model. By the same token, "polygen query" will be used instead of "global query," and so on, and so forth.

2. Both the Functional Data Model and the Semantic Database Model are rich in semantics and implemented in operational systems. The Entity Relationship Model is also rich in semantics and is widely accepted as the leading database design tool. The relational model lends itself to a simple structure and an elegant theoretical foundation and Relational Data Base Management Systems dominate database market today. Codd (1979) also extended the relational model to capture semantics such as generalization and aggregation.

3. Each transformation rule contains a source part and a target part. For example,

```
Source:  SELECT attribute-1   FROM relation-1
         WHERE condition;
Target:  Projection ((attribute-1), Selection (condition,
         (relation-1));
```

4. This approach simplifies the Polygen Operation Interpreter, to be presented in Section 3.

5. Under the relational assumption, the cardinality inconsistency problem exists in heterogeneous database systems because the referential integrity is not enforceable over multiple pre-existing databases which have been developed and administered independently and are likely to remain so.

Appendix A

The Operations that Generate Table 5

The second and third row of Table 3 indicates that the BUSINESS and FIRM relations should be retrieved from the Alumni Database and the Company Database respectively. As such, the corresponding data source cells are the set {AD} and {CD} respectively as shown in Table A1 and Table A2 below. The intermediate source is an empty set because no other data sources have been involved in obtaining these relations.

Table A1. The Business Relation

BNAME	IND
IBM, {AD},{}	High Tech, {AD},{}
CitiCorp, {AD},{}	Banking, {AD},{}
Oracle, {AD},{}	High Tech, {AD},{}
Ford, {AD},{}	Automobile, {AD},{}
DEC, {AD},{}	High Tech, {AD},{}
BP, {AD},{}	Energy, {AD},{}
Genentech, {AD},{}	High Tech, {AD},{}

Table A2. The Firm Relation

FNAME	CEO	HQ
AT&T, {CD},{}	Robert Allen, {CD},{}	NY, {CD},{}
Banker's Trust, {CD},{}	Charles Sanford, {CD},{}	NY, {CD},{}
CitiCorp, {CD},{}	John Reed, {CD},{}	NY, {CD},{}
Ford, {CD},{}	Donald Peterson, {CD},{}	MI, {CD},{}
IBM, {CD},{}	John Ackers, {CD},{}	NY, {CD},{}
Apple, {CD},{}	John Sculley, {CD},{}	CA, {CD},{}
Oracle, {CD},{}	Lawrence Ellison, {CD},{}	CA, {CD},{}
DEC, {CD},{}	Ken Olsen, {CD},{}	MA, {CD},{}
Genentech, {CD},{}	Bob Swanson, {CD},{}	CA, {CD},{}

Table A1 and Table A2 are merged together (see Row 4, Table 3) to generate Table 5. This process involves an *Outer Natural Total Join* (ONTJ) of Table A1 and Table A2. The *Outer Natural Total Join* consists of three steps:

(1) An *outer join* on BNAME and FNAME because they are the local attributes of the primary polygen attribute ONAME for PORGANIZATION. The result is shown in Table A3.

Table A3. The Outer Join of Table A1 and Table A2

BNAME	IND	FNAME	CEO	HQ
IBM, {AD},{AD, CD}	High Tech, AD},{AD, CD}	IBM, {CD},{AD, CD}	John Ackers, ({CD},{AD, CD}	NY, {CD},{AD, CD}
CitiCorp, {AD},{AD, CD}	Banking, {AD},{AD, CD}	Citicorp, {CD},{AD, CD}	John Reed, {CD},{AD, CD}	NY, {CD},{AD, CD}
Oracle, {AD},{AD, CD}	High Tech, {AD},{AD, CD}	Oracle, {CD},{AD, CD}	Lawrence Ellison, {CD},{AD, CD}	CA, {CD},{AD, CD}
Ford, {AD},{AD, CD}	Automobile, {AD},{AD, CD}	Ford, {CD}, {AD, CD}	Donald Peterson, {CD},{AD, CD}	MI, {CD},{AD, CD}
DEC, {AD},{AD, CD}	High Tech, {AD},{AD, CD}	DEC, {CD},{AD, CD}	Ken Olsen, {CD},{AD, CD}	MA, {CD},{AD, CD}
BP, {AD},{AD}	Energy, {AD},{AD}	nil, {}, {AD}	nil, {}, {AD}	nil, {}, {AD}
Genentech, {AD},{AD, CD}	High Tech, {AD},{AD, CD}	Genentech, {CD},{AD, CD}	Bob Swanson, {CD},{AD, CD}	CA, {CD},{AD, CD}
nil, {}, {CD}	nil, {}, {CD}	AT&T, {CD},{CD}	Robert Allen, {CD},{CD}	NY, {CD},{CD}
nil, {}, {CD}	nil, {}, {CD}	Banker's Trust, {CD},{CD}	Charles Sanford, {CD},{CD}	NY, {CD},{CD}
nil, {}, {CD}	nil, {}, {CD}	Apple, {CD},{CD}	John Sculley, {CD},{CD}	CA {CD},{}

(2) A *Coalesce* of the BNAME and FNAME columns into the ONAME column. The result is shown in Table A4. As we defined in Section 2, step one and two together are called an *Outer Natural Primary Join.*

Table A4. The Outer Natural Primary Join of Table A1 and Table A2

ONAME	IND	CEO	HQ
IBM, {AD, CD},{AD, CD}	High Tech, {AD},{AD, CD}	John Ackers, {{CD},{AD, CD}	NY, {CD},{AD, CD}
CitiCorp, {AD, CD},{AD, CD}	Banking, {AD},{AD, CD}	John Reed, {CD},{AD, CD}	NY, {CD},{AD, CD}
Oracle, {AD, CD},{AD, CD}	High Tech, {AD},{AD, CD}	Lawrence Ellison, {CD},{AD, CD}	CA, {CD},{AD, CD}
Ford, {AD, CD},{AD, CD}	Automobile, {AD},{AD CD}	Donald Peterson, {CD},{AD, CD}	MI, {CD},{AD, CD}
DEC, {AD, CD},{AD, CD}	High Tech, {AD},{AD, CD}	Ken Olsen, {CD},{AD, CD}	MA, {CD},{AD, CD}
BP, {AD},{AD}	Energy, {AD},{AD}	nil, {}, {AD}	nil, {}, {AD}
Genentech, {AD, CD},{AD, CD}	High Tech, {AD},{AD CD}	Bob Swanson, {CD},{AD, CD}	CA, {CD},{AD, CD}
AT&T, {CD},{CD}	nil, {}, {CD}	Robert Allen, {CD},{CD}	NY, {CD},{CD}
Banker's Trust, {CD},{CD}	nil, {}, {CD}	Charles Sanford, {CD},{CD}	NY, {CD},{CD}
Apple, {CD},{CD}	nil, {}, {CD}	John Sculley, {CD},{CD}	CA {CD},{}

(3) *Coalesces* of other local columns into the corresponding non-primary polygen columns. Since no other overlapping local columns exist in this simplified example, only the local attributes IND and HQ are changed to INDUSTRY and HEADQUARTERS. The result is shown as Table 5 in the body of the paper.