A SOURCE TAGGING THEORY FOR HETEROGENEOUS DATABASE SYSTEMS

Y. Richard Wang  
Massachusetts Institute of Technology

Stuart E. Madnick  
Massachusetts Institute of Technology

Follow this and additional works at: http://aisel.aisnet.org/icis1990

Recommended Citation
http://aisel.aisnet.org/icis1990/42

This material is brought to you by the International Conference on Information Systems (ICIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ICIS 1990 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
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 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 FORGORGANIZATION, 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.

<table>
<thead>
<tr>
<th>Polygen Schema</th>
<th>Alumni Database INDB</th>
<th>Company Database IDDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORGORGNAME, IND, CEO, INDUSTRY</td>
<td>BUSINESSNAME, IND</td>
<td>FIRMNAME, CEO, HQ</td>
</tr>
<tr>
<td>PALUMNUSNAME, ANAME, DEGREE, MAJOR</td>
<td>ALUMNINAME, ANAME, DEGRE, MAJOR</td>
<td>FINANCEGNAME, YR, PROFIT</td>
</tr>
<tr>
<td>TCAREER, GQDD, DDNAME</td>
<td>CAREER, GQDD, DDNAME</td>
<td></td>
</tr>
</tbody>
</table>

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 polygon 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 polygon model but also a polygon algebra and the algorithms for a polygon query processor capable of resolving the data and intermediate source tagging problems for any arbitrary polygon 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 MERMAD, Cincos'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 polygon model should be created in order to tag multiple sources explicitly? (2) What is the relationship between the polygon model and the polygon 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 Data-Base Model, or the Entity Relationship Model. Each data model has merits for its intended purposes. We selected the relational model. Based on the relational model, we define a polygon model for resolving the data and intermediate source tagging problems.

One of the key activities in formulating composite information is to translate a polygon 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 polygon query and transformation rules 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 polygon 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 polygon 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 polygon query for the Polygon 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.

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 Naumche 1986; Elmasri, Larson and Naumche 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") have 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 \{\text{database, relation, attribute}\}... for the source tagging example described in Section 1.

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.,

\[
\text{MA} = \{(LD, LS, LA) | (LD, LS, LA) \text{ denotes a local attribute to the corresponding PA}\}
\]

For ONAME in the PORGANIZATION polygen scheme,

\[
\text{MA} = \{(AD, BUSINESS, BNAME), (CD, FIRM, FNAME)\}
\]
A polygen scheme \( P \) is defined as

\[
P = (\langle P_{A1}, M_{A1} \rangle, \ldots, (P_{An}, M_{An})) \quad \text{where} \quad n \quad \text{is the number of attributes in} \quad P.
\]

For the polygen scheme PORGANIZATION in the above scenario,

\[
PORGANIZATION = (\langle \text{ONAME}, \langle \text{AD, BUSINESS, BNAME} \rangle, \langle \text{CD, FIRM, FNAME} \rangle \rangle, \langle \text{INDUSTRY}, \langle \text{AD, BUSINESS, IND} \rangle \rangle, \langle \text{CEO}, \langle \text{CD, FIRM, CEO} \rangle \rangle, \langle \text{HEADQUARTERS}, \langle \text{CD, FIRM, HQ} \rangle \rangle)
\]

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:

\[
\{\text{PORGANIZATION, PFINANCE, PALUMNUS, FCAREER}\}
\]

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 defining values from the corresponding polygen domains. A cell in a polygen relation is an ordered triplet \( c = \langle (d), (o), (g) \rangle \) where \( d \) denotes the datum portion, \( o \) the originating portion, and \( g \) 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 \( \text{attrs}(p) \) denote the set of attributes in \( p \). For each tuple \( t \) in a polygen relation \( p \), let \( t(d) \) denote the data portion, \( t(o) \) the originating source portion, and \( t(g) \) the intermediate source portion. If \( X = \{x_1, \ldots, x_n\} \) is a sublist of \( \text{attrs}(p) \), then \( p[X] \) be the column in \( p \) corresponding to attribute \( x_i \), let \( t[X] \) be the cells in \( t \) corresponding to the sublist of attributes \( X \). As such, \( p[x](o) \) denotes the originating source portion of the column corresponding to attribute \( x \) in polygen relation \( p \) while \( t[X](o) \) 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 \( X = \{x_1, \ldots, x_n\} \) is a sublist of \( \text{attrs}(p) \), then

\[
p[X] = \{t'[.] | t' = t[X] \quad \text{if} \quad t \in p \land t[X](d) \quad \text{is unique;}
\]

\[
t'[d] = t[X](d), t'[x](o) = t[x](o) \quad \text{and} \quad t'[x](g) = t[x](g)
\]

\[
\forall x \in X, t'[x](g) \iff t[x](g)
\]

\[
\text{if} \quad t_0, \ldots, t_k \in p \land t[X](d) = \ldots = t_k[X](d).
\]

**Cartesian product.** If \( p_1 \) and \( p_2 \) are two polygen relations, then

\[
(p_1 \times p_2) = \{t_1 \circ t_2 | t_1 \in p_1 \land t_2 \in p_2 \quad \text{where} \quad \circ \quad \text{denotes concatenation}\}
\]

**Restrict.** If \( p \) is a polygen relation, \( x \in \text{attrs}(p) \), \( y \in \text{atts}(p) \), and \( \theta \) is a binary relation, then

\[
p[x \theta y] = \{t'[.] | t' = t[d], t'[o] = t[o],
\]

\[
t'[w](i) = t[w](i) \quad \text{or} \quad t[x](o) \quad \text{or} \quad t[y](o) \forall w \in \text{atts}(p),
\]

\[
\text{if} \quad t \in p \land t[X](d) \neq t[Y](d).
\]

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

\[
(p_1 \cup p_2) = \{t'[.] | t' = t_i \quad \text{if} \quad t_i(d) \in p_1 \land t_i(d) \notin p_2;
\]

\[
t' = t_2 \quad \text{if} \quad t_2(d) \notin p_1 \land t_2(d) \in p_2;
\]

\[
t'[d] = t_i(d), t'[o] = t_i(o) \quad \text{or} \quad t_2(o), t'[g] = t_i(g) \quad \text{or} \quad t_2(g) \quad \text{if} \quad t_i(d) = t_2(d)
\]
Difference. Let \( p(o) \) denote the union of all the \( t(o) \) sets in \( p \) and \( p(\overline{o}) \) denote the union of all the \( t(\overline{o}) \) sets in \( p \). If \( p_1 \) and \( p_2 \) are two polygon relations and both have degree \( n \), then

\[
(p_1 - p_2) = \{t'| t'(d) = t(d), t'(o) = t(o), t'[w][s] = [w][s] \land p_2(s) \land p_1(s) \land w \in \text{attrs}(p), \text{if } t \in p_1 \text{ and } (t)(p_2) \}.
\]

The intermediate source portion, \( t(\overline{o}) \), is updated by Restrict and Difference. The Restrict operation selects the tuples in a polygon relation which satisfies the \( [x \theta y] \) condition. As such, the originating local databases of \( x \) and \( y \) attributes values are added to the \( t(\overline{o}) \) set in order to signify their mediating role. Since Select and Join are defined through Restrict, they also update \( t(\overline{o}) \).

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 \( t'(\overline{o}) = t(\overline{o}) \lor p_2(o) \lor p_1(o)(\overline{o}) \) 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 polygon query, we also need to introduce the following new operators to the polygon 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 polygon model because a polygon 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. 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 polygon 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 polygon attributes in the polygon 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 polygon relations. It can be shown that the order in which Outer Natural Total Join is performed over a set of polygon relations in a Merge is immaterial.

Since Coalesce can be used in conjunction with the other polygon 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 polygon model.

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

\[
p(x \bullet y)w = \\
\{t'| t'(x) = t(x), t'(y) = t(y), t'(z) = t(z), t'(w)(o) = t(x)(o) \land y(o), t'[w](x) = [w][x]_o, a[y](y), \text{if } t'(x)(d) = t'(y)(d); \\
t'[x] = t[x], t'[y] = t[y], t'[y] = t[y], t'[w](x) = t'[x](o), \\
t'[w] = t[x], t'[y] = t[y], t'[y] = t[y], t'[w](o) = t'[x](o), \\
t'[w] = t[x], t'[y] = t[y], t'[y] = t[y], t'[w](o) = t'[x](o), \\
t'[w] = t[x], t'[y] = t[y], t'[y] = t[y], t'[w](o) = t'[x](o), \\
t'[w] = t[x], t'[y] = t[y], t'[y] = t[y], t'[w](o) = t'[x](o)
\}
\]

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-matching problems will be resolved before the coalesce operation is performed (Wang and Madnick 1989).

We have presented the polygon model and the polygon 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 polygon query into a query execution plan. This process is presented below.

3. POLYGEN QUERY TRANSLATION

For the SQL polygon query presented in Section 1, a corresponding polygon algebraic expression for the SQL polygon query is as follows:
Algebraic Attribute Hand
The PALUMNUS first polygen example, in Operation composite retrieving
In Algebraic Polygon Figure general, an input
Alg.b,aic PORGANIZATION)
(R(1)
Right-Hand row and
Figure Alg.bic Analyzer process:
R(2)
R(3)
Project R(2) ONAME, CEO nil nil nil

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

<table>
<thead>
<tr>
<th>PR</th>
<th>OP</th>
<th>LHR</th>
<th>LHA</th>
<th>0</th>
<th>RHA</th>
<th>RHR</th>
</tr>
</thead>
<tbody>
<tr>
<td>R(1)</td>
<td>Select</td>
<td>PALUMNUS</td>
<td>DEGREE</td>
<td>&quot;MBA&quot;</td>
<td>nil</td>
<td></td>
</tr>
<tr>
<td>R(2)</td>
<td>Join</td>
<td>R(1)</td>
<td>ANAME</td>
<td>CEO</td>
<td>PORGANIZATION</td>
<td>nil</td>
</tr>
<tr>
<td>R(3)</td>
<td>Project</td>
<td>R(2)</td>
<td>ONAME, CEO</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>PR</th>
<th>OP</th>
<th>LHR</th>
<th>LHA</th>
<th>0</th>
<th>RHA</th>
<th>RHR</th>
<th>EL</th>
</tr>
</thead>
<tbody>
<tr>
<td>R(1)</td>
<td>Select</td>
<td>ALUMNUS</td>
<td>DEG</td>
<td>&quot;MBA&quot;</td>
<td>nil</td>
<td></td>
<td>AD</td>
</tr>
<tr>
<td>R(2)</td>
<td>Join</td>
<td>R(1)</td>
<td>ANAME</td>
<td>CEO</td>
<td>PORGANIZATION</td>
<td>nil</td>
<td>POP</td>
</tr>
<tr>
<td>R(3)</td>
<td>Project</td>
<td>R(2)</td>
<td>ONAME, CEO</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>POP</td>
</tr>
</tbody>
</table>

((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.

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 0 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 LOP (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.
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 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

<table>
<thead>
<tr>
<th>AID#</th>
<th>ANAME</th>
<th>DEG</th>
<th>MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>012.</td>
<td>John McCauley</td>
<td>MBA,</td>
<td>IS</td>
</tr>
<tr>
<td></td>
<td>[AD], [AD]</td>
<td>[AD],</td>
<td>[AD]</td>
</tr>
<tr>
<td>123.</td>
<td>Bob Swanson</td>
<td>MBA,</td>
<td>MGT</td>
</tr>
<tr>
<td></td>
<td>[AD], [AD]</td>
<td>[AD],</td>
<td>[AD]</td>
</tr>
<tr>
<td>456.</td>
<td>Dave Horton</td>
<td>MBA,</td>
<td>IS</td>
</tr>
<tr>
<td></td>
<td>[AD], [AD]</td>
<td>[AD],</td>
<td>[AD]</td>
</tr>
<tr>
<td>567.</td>
<td>John Reed</td>
<td>MBA,</td>
<td>MIT</td>
</tr>
<tr>
<td></td>
<td>[AD], [AD]</td>
<td>[AD],</td>
<td>[AD]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ONAME</th>
<th>INDUSTRY</th>
<th>CEO</th>
<th>HEADQUARTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM, [AD],</td>
<td>High Tech,</td>
<td>John Akers, [CD],</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td></td>
<td>[AD], [AD]</td>
<td>[AD], [AD]</td>
<td>[AD], [CD]</td>
</tr>
<tr>
<td>CitiCorp,</td>
<td>Banking,</td>
<td>John Reed, [CD],</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td>[AD], [AD]</td>
<td>[AD], [AD]</td>
<td>[AD], [AD]</td>
<td>[AD], [CD]</td>
</tr>
<tr>
<td>Oracle,</td>
<td>High Tech,</td>
<td>Lawrence Ellison,</td>
<td>CA, [CD],</td>
</tr>
<tr>
<td>[AD], [AD]</td>
<td>[AD], [AD]</td>
<td>[CD], [AD]</td>
<td>[AD], [CD]</td>
</tr>
<tr>
<td>Ford,</td>
<td>Automobile,</td>
<td>Donald Peterson,</td>
<td>MI, [CD],</td>
</tr>
<tr>
<td>[AD], [AD]</td>
<td>[AD], [AD]</td>
<td>[CD], [AD]</td>
<td>[AD], [CD]</td>
</tr>
<tr>
<td>DEC,</td>
<td>High Tech,</td>
<td>Ken Olsen, [CD],</td>
<td>MA, [CD],</td>
</tr>
<tr>
<td>[AD], [AD]</td>
<td>[AD], [AD]</td>
<td>[CD], [AD]</td>
<td>[AD], [CD]</td>
</tr>
<tr>
<td>BP,</td>
<td>Energy,</td>
<td>nil, [AD]</td>
<td>nil, [AD]</td>
</tr>
<tr>
<td>[AD], [AD]</td>
<td>[AD],</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genentech,</td>
<td>High Tech,</td>
<td>Bob Swanson, [CD],</td>
<td>CA, [CD],</td>
</tr>
<tr>
<td>[AD], [AD]</td>
<td>[AD], [AD]</td>
<td>[CD], [AD]</td>
<td>[AD], [CD]</td>
</tr>
<tr>
<td>AT&amp;T,</td>
<td>nil, [CD]</td>
<td>Robert Allen, [CD],</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td>[CD], [CD]</td>
<td></td>
<td>[CD], [CD]</td>
<td>[CD], [CD]</td>
</tr>
<tr>
<td>Banker’s</td>
<td>nil, [CD]</td>
<td>Charles Sanford,</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td>Trust,</td>
<td></td>
<td>[CD], [CD]</td>
<td>[CD], [CD]</td>
</tr>
<tr>
<td>[CD], [CD]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple,</td>
<td>nil, [CD]</td>
<td>John Sculley, [CD],</td>
<td>CA, [CD],</td>
</tr>
<tr>
<td>[CD], [CD]</td>
<td></td>
<td>[CD], [CD]</td>
<td>[CD], [CD]</td>
</tr>
</tbody>
</table>

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.

\[\textbf{Lemma}\] 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.
Since the polygon 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.

\(\text{Theorem}\) The closure property 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 polygon model. Let \(e_i\) and \(e_i'\) denote two base polygon relations. Let \(f\) denote an algebraic operation, \(e_i = f(e_i)\) if \(f\) is project, restrict, or coalesce; \(e_i = f(e_i, e_i')\) if \(f\) is Cartesian product, union, or difference. Similarly, let \(e_{i+1} = f(e_i)\) for some \(e_i \in E\) if \(f\) is project, restrict, or coalesce; \(e_{i+1} = f(e_i, e_i')\) for some \(e_i \in E, e_i' \in E\), if \(f\) is Cartesian product, union, or difference. We now show that, if source tagging is by cell, then the closure property holds, i.e., \(e\) is a polygon relation defined by the polygon 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.

\(\text{Proof}\) Part 1: The closure property 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 polygon model in which source tagging is by relation, by attribute, or by tuple. If source tagging is by relation, then a relation in this polygon model can be expressed as \((c, e(o))\). Consider the Cartesian product of \((e_1, e_2(o)) \times (e_2, e_3(o))\). By definition, the operation yields \(t_1 \ast t_2\), \(t_1 \in e_1\) and \(t_2 \in e_2\), where \(\ast\) denotes concatenation. However, the result cannot be expressed in the form of \((e, e(o))\) because the originating source tags \(t_1\) 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 polygon model can be expressed as \((e[x], e(x)(o))\). Consider union. By definition, \(t'(d) = t(d), t'[x](o) = t_1[x](o) \cup t_2[x](o)\) if \(t_1(d) = t_2(d)\). However, the result cannot be expressed in the form of \((e[x], e(x)(o))\) because \(t_1[x](o)\) may be different from \(t_2[x](o)\). It follows that source tagging by attribute is not feasible. If source tagging is by tuple, then a tuple in this polygon 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 closure property holds.

The premise that source tagging is by cell justifies the usage of the polygon model presented in Section 2 in the following proof. By the model’s definition, \(t(o)\) is the set of the originating source \(\forall t \in e_i, \forall e_i \in E, \text{ and the closure property holds}\ \forall e_i \in E\).

Assuming that the closure property holds for \(\forall e_i \in E\), we show that the closure property also holds \(\forall e_{i+1} \in E\).

For projection, \(e_{i+1} = e_i[X]\). Two cases need to be considered: (1) \(t[X](d)\) is unique and (2) \(t[X](d)\) remains the same for all \(c \in t[X](o)\). In the first case, \(c(o)\) remains the same for all \(c \in t[X](o)\) and \(x_i \in X\). In the second case, \(t[D](o) = t[X](o) \cup \cup t[X](o) \forall x_i \in X\). Since the closure property holds for
For Cartesian product, \( c_{k+1} = (c_k \times c'_k) = \{ t, t' : t \in e_k, t' \in e'_k \} \), where \( * \) denotes concatenation. For difference, \( e_{k+1} = (c_k - e'_k) = \{ t : t \in e_k \text{ and } t \notin e'_k \}. \) For restrict, \( c_{k+1} = c_k(x \theta y) = \{ t : t \in c_k \text{ and } (t[x] \theta y(t[y]))) \). Since \( t(o) \) remains the same in Cartesian product, difference, and restrict, it follows that the closure property holds for \( c_{k+1} = (c_k \times c'_k), e_{k+1} = (c_k - c'_k), \) 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 polygon algebraic expression into a set of intermediate polygon 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. It also enables us to interpret information from different sources more accurately. By storing the metadata about each of the data sources in the POP, many domain mismatch, semantic reconciliation, and data conflict problems could be resolved systematically using the data and intermediate source tags. Furthermore, other polygon 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 polygon 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


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 Database 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

<table>
<thead>
<tr>
<th>BNAME</th>
<th>IND</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM, [AD],</td>
<td>High Tech, [AD],</td>
</tr>
<tr>
<td>CitCorp, [AD],</td>
<td>Banking, [AD],</td>
</tr>
<tr>
<td>Oracle, [AD],</td>
<td>High Tech, [AD],</td>
</tr>
<tr>
<td>Ford, [AD],</td>
<td>Automobile, [AD],</td>
</tr>
<tr>
<td>DEC, [AD],</td>
<td>High Tech, [AD],</td>
</tr>
<tr>
<td>BP, [AD],</td>
<td>Energy, [AD],</td>
</tr>
<tr>
<td>Genentech, [AD],</td>
<td>High Tech, [AD],</td>
</tr>
</tbody>
</table>

Table A2. The Firm Relation

<table>
<thead>
<tr>
<th>FNAME</th>
<th>CEO</th>
<th>HQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT&amp;T, [CD],</td>
<td>Robert Allen, [CD],</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td>Banker's Trust, [CD],</td>
<td>Charles Sanford, [CD],</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td>CitCorp, [CD],</td>
<td>John Reed, [CD],</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td>Ford, [CD],</td>
<td>Donald Peterson, [CD],</td>
<td>MI, [CD],</td>
</tr>
<tr>
<td>IBM, [CD],</td>
<td>John Ackers, [CD],</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td>Apple, [CD],</td>
<td>John Sculley, [CD],</td>
<td>CA, [CD],</td>
</tr>
<tr>
<td>Oracle, [CD],</td>
<td>Lawrence Ellison, [CD],</td>
<td>CA, [CD],</td>
</tr>
<tr>
<td>DEC, [CD],</td>
<td>Ken Olsen, [CD],</td>
<td>MA, [CD],</td>
</tr>
<tr>
<td>Genentech, [CD],</td>
<td>Bob Swanson, [CD],</td>
<td>CA, [CD],</td>
</tr>
</tbody>
</table>

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 polygon attribute ONAME for FORGANIZATION. The result is shown in Table A3.

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

<table>
<thead>
<tr>
<th>BNAME</th>
<th>IND</th>
<th>FNAME</th>
<th>CEO</th>
<th>HQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM, [AD],</td>
<td>High Tech, [AD],</td>
<td>IBM, [CD],</td>
<td>John Ackers, [CD],</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td>(AD),[AD, CD]</td>
<td>(CD),[CD]</td>
<td>(AD),[AD, CD]</td>
<td>(CD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
</tr>
<tr>
<td>CitCorp, (AD),</td>
<td>Banking, (AD),</td>
<td>Citcorp, (CD),</td>
<td>John Reed, (CD),</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
</tr>
<tr>
<td>Oracle, (AD),</td>
<td>High Tech, (AD),</td>
<td>Oracle, (CD),</td>
<td>Lawrence Ellison, (CD),</td>
<td>CA, [CD],</td>
</tr>
<tr>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
</tr>
<tr>
<td>Ford, (AD),</td>
<td>Automobile, (AD),</td>
<td>Ford, (CD),</td>
<td>Donald Peterson, (CD),</td>
<td>MI, [CD],</td>
</tr>
<tr>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
</tr>
<tr>
<td>DEC, (AD),</td>
<td>High Tech, (AD),</td>
<td>DEC, (CD),</td>
<td>Ken Olsen, (CD),</td>
<td>MA, [CD],</td>
</tr>
<tr>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
</tr>
<tr>
<td>BP, [AD],</td>
<td>Energy, [AD],</td>
<td>nil, [AD],</td>
<td>nil, [AD],</td>
<td>nil, [AD],</td>
</tr>
<tr>
<td>(AD)</td>
<td>(AD)</td>
<td>(AD)</td>
<td>(AD)</td>
<td>(AD)</td>
</tr>
<tr>
<td>Genentech, (AD),</td>
<td>High Tech, (AD),</td>
<td>Genentech, (CD),</td>
<td>Bob Swanson, (CD),</td>
<td>CA, [CD],</td>
</tr>
<tr>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
<td>(AD),[AD, CD]</td>
</tr>
<tr>
<td>nil, [AD]</td>
<td>nil, [AD]</td>
<td>AT&amp;T, [CD],</td>
<td>Robert Allen, (CD),</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td>[CD]</td>
<td>[CD]</td>
<td>(CD)</td>
<td>(CD),</td>
<td>(CD),</td>
</tr>
<tr>
<td>nil, [CD]</td>
<td>nil, [CD]</td>
<td>Banker's Trust, (CD),</td>
<td>Charles Sanford, (CD),</td>
<td>NY, [CD],</td>
</tr>
<tr>
<td>[CD]</td>
<td>[CD]</td>
<td>(CD)</td>
<td>(CD),</td>
<td>(CD),</td>
</tr>
<tr>
<td>nil, [CD]</td>
<td>nil, [CD]</td>
<td>Apple, (CD),</td>
<td>John Sculley, (CD),</td>
<td>CA, [CD],</td>
</tr>
<tr>
<td>[CD]</td>
<td>[CD]</td>
<td>(CD)</td>
<td>(CD),</td>
<td>(CD),</td>
</tr>
</tbody>
</table>
(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

<table>
<thead>
<tr>
<th>ONAME</th>
<th>IND</th>
<th>CEO</th>
<th>HQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM, (AD, CD), (AD, CD)</td>
<td>High Tech, (AD), (AD, CD)</td>
<td>John Ackers, (CD), (AD, CD)</td>
<td>NY, (CD), (AD, CD)</td>
</tr>
<tr>
<td>Citcorp, (AD, CD), (AD, CD)</td>
<td>Banking, (AD), (AD, CD)</td>
<td>John Reed, (CD), (AD, CD)</td>
<td>NY, (CD), (AD, CD)</td>
</tr>
<tr>
<td>Oracle, (AD, CD), (AD, CD)</td>
<td>High Tech, (AD), (AD, CD)</td>
<td>Lawrence Ellison, (CD), (AD, CD)</td>
<td>CA, (CD), (AD, CD)</td>
</tr>
<tr>
<td>Ford, (AD, CD), (AD, CD)</td>
<td>Automobile, (AD), (AD, CD)</td>
<td>Donald Peterson, (CD), (AD, CD)</td>
<td>MA, (CD), (AD, CD)</td>
</tr>
<tr>
<td>DEC, (AD, CD), (AD, CD)</td>
<td>High Tech, (AD), (AD, CD)</td>
<td>Ken Olsen, (CD), (AD, CD)</td>
<td>MA, (CD), (AD, CD)</td>
</tr>
<tr>
<td>BP, (AD), (AD)</td>
<td>Energy, (AD), (AD)</td>
<td>nil, {CD}, (AD)</td>
<td>nil, {AD}</td>
</tr>
<tr>
<td>Genentech, (AD, CD), (AD, CD)</td>
<td>High Tech, (AD), (AD, CD)</td>
<td>Bob Swanson, (CD), (AD, CD)</td>
<td>CA, (CD), (AD, CD)</td>
</tr>
<tr>
<td>AT&amp;T, (CD), (CD)</td>
<td>nil, {CD}</td>
<td>Robert Allen, (CD), (CD)</td>
<td>NY, (CD), (CD)</td>
</tr>
<tr>
<td>Banker's Trust, (CD), (CD)</td>
<td>nil, {CD}</td>
<td>Charles Sanford, (CD), (CD)</td>
<td>NY, (CD), (CD)</td>
</tr>
<tr>
<td>Apple, (CD), (CD)</td>
<td>nil, {CD}</td>
<td>John Sculley, (CD), (CD)</td>
<td>CA, (CD), (CD)</td>
</tr>
</tbody>
</table>

(3) Coalesce 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.