The Inter-Database Instance Identification Problem in Integrating Autonomous Systems

Y. Richard Wang
Stuart E. Madnick
SLOAN SCHOOL OF MANAGEMENT, E53-320
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MA 02139
(617) 253-6671

I. INTRODUCTION

The combination of a turbulent global economy with increased competition and recent advances in computer and communications technologies have generated significant interest in, and successful example of, using such information technology (IT) for strategic applications [2, 5, 21]. A key concern facing corporations today is how to make most effective use of IT to meet their needs [3, 13, 20]. For example, an on-going dialogue exists between the MIT Sloan School of Management and the sponsors of the *Management in the 1990*'s research program. In a recent meeting with the sponsors[1] to assess their information technology requirements for 1995, a critical need was identified for developing systems that can provide access to, and integration of their corporations' numerous information systems, as depicted in Figure 1.

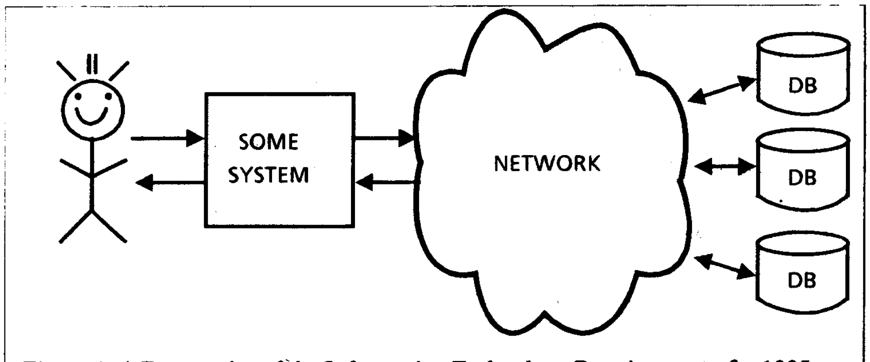


Figure 1 A Perspective of the Information Technology Requirements for 1995

It has become increasingly evident that many important applications require such access to and integration of multiple disparate databases both within and across organizational boundaries in order to increase productivity [4, 10]. We refer to this type of systems as *Composite Information Systems* (CIS) [12, 17, 23, 24].

This paper examines the issue of joining information about the same instance across disparate databases in a CIS environment. A technique called *inter-database instance identification* is presented. It employs a combination of database management systems and artificial intelligence techniques. Common attributes in the disparate databases[2] are applied first to reduce the number of potential candidates for the same instance. Other attributes in these databases, auxiliary databases, and inferencing rules are exploited next to identify the same instance. A detailed example of the *inter-database instance identification technique* is also presented using an operational research prototype.

Section II presents a CIS application scenario where composite information is produced from five information systems in four organizations. Furthermore, connectivity issues involved in facilitating the CIS application such as instance identification are revealed. Section III presents an example of a professor and his teaching assistant engaging in the process of identifying a student in their class. Section IV then presents the technique for inter-database instance identification using the Professor-TA example. It is presented in the context of a Tool Kit for Composite Information Systems (CIS/TK)[2] -- a *knowledge and information delivery system* which has four functional components: knowledge processing,

[1] The meeting was held at the MIT Sloan School in the late April, 1988. The participants were IT executives from American Express, Arthur Young & Co , Bell South, British Petroleum, Cigna Insurance, Digital Equipment Corporation (DEC), International Computers, Ltd,, Kodak, IRS, and U.S. Army.

[2] The CIS/TK ensemble is a research prototype being developed at the MIT Sloan School of Management for the development of CIS applications. An operational prototype. is being implemented in the UNIX environment both to take advantage of its portability across disparate hardware and its multi-programming and communications capabilities to enable accessing multiple disparate remote databases in concert.

Best Copy

CH2695-5/89/0000/0046$01.00 © 1989 IEEE

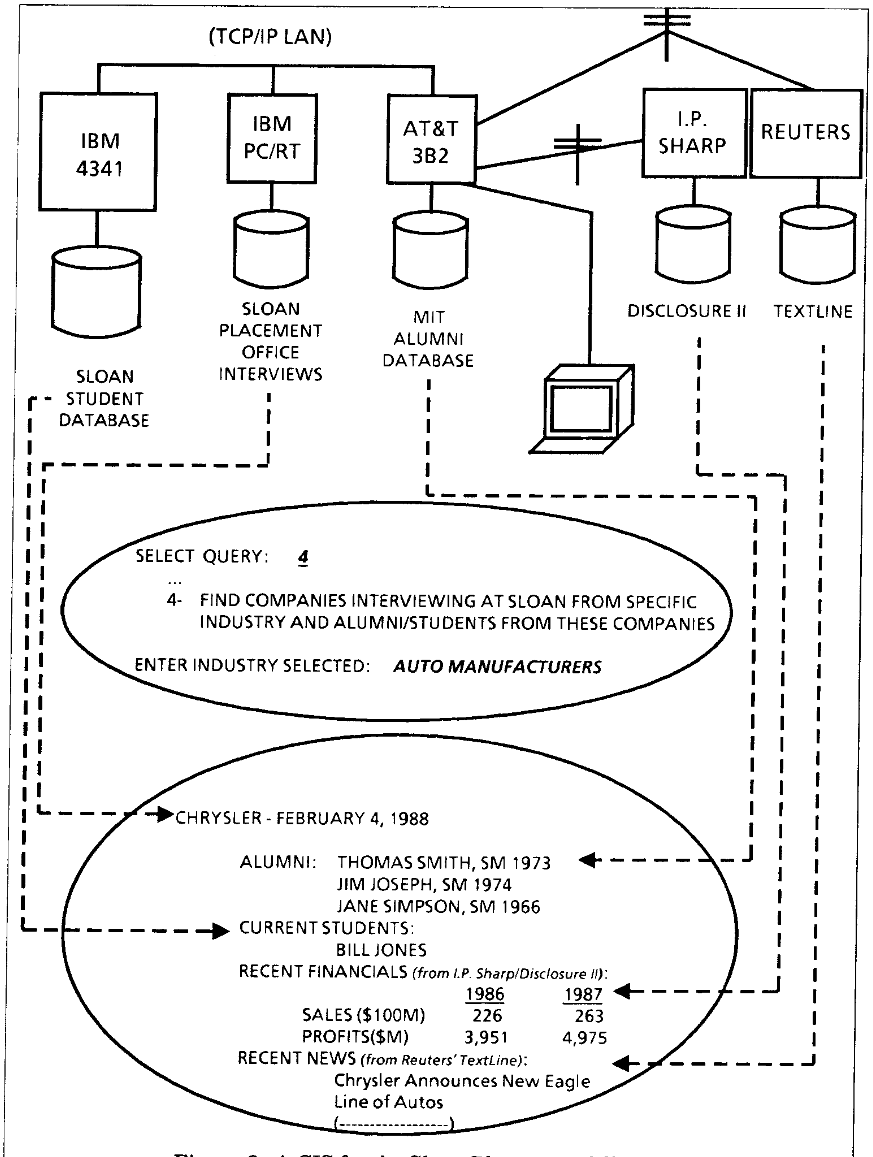
information processing, physical and logical connectivity, and user interfaces. Finally, concluding remarks are made in section V.

II. CIS APPLICATION: PAS

Consider the Placement Assistant System (PAS), depicted in Figure 2, which is being developed for the MIT Sloan Placement Office. PAS spans five information systems in four organizations: (1) the **student database** and the **interview schedule database** are located in the Sloan School; (2) the **alumni database** is located in the MIT alumni office; (3) the *recent news* is accessed by dialing into Reuter's **Textline database**; and (4) the *recent financial information* is accessed through I.P. Sharp's **Disclosure II database**.

An interesting query for PAS to handle would be to "find companies interviewing at Sloan"

- that are *auto manufacturers*,
- the *students* from these companies,
- the *alumni* from these companies,
- recent *financial information*, and
- recent *news* about these companies.



```
SELECT QUERY:  4
...
4-  FIND COMPANIES INTERVIEWING AT SLOAN FROM SPECIFIC
    INDUSTRY AND ALUMNI/STUDENTS FROM THESE COMPANIES

ENTER INDUSTRY SELECTED:  AUTO MANUFACTURERS
```

```
CHRYSLER - FEBRUARY 4, 1988

ALUMNI:    THOMAS SMITH, SM 1973
           JIM JOSEPH, SM 1974
           JANE SIMPSON, SM 1966
CURRENT STUDENTS:
           BILL JONES
RECENT FINANCIALS (from I.P. Sharp/Disclosure II):
                        1986      1987
     SALES ($100M)       226       263
     PROFITS($M)       3,951     4,975
RECENT NEWS (from Reuters' TextLine):
           Chrysler Announces New Eagle
           Line of Autos
           (-----------------)
```

Figure 2 A CIS for the Sloan Placement Office

This information would be very valuable to a student interested in the automobile industry. Current students from these companies can offer first-hand insider information. The alumni from these companies may be able to "put in a good word" on his behalf. Recent financial information indicates the economic environment at that company as well as providing information that may be helpful during an interview. Finally, recent news will keep the student abreast of what is going on in that company as well as to be well prepared for the interview with the recruiters.

Figure 2 depicts a partial menu for the query and its corresponding answer in the case of *Chrysler*. Many research problems need to be solved in order to obtain the composite information for this query [1, 7, 8, 9, 15, 23, 24]. These connectivity problems can be categorized into first- and second-order issues, as discussed below.

Connectivity Issues in CIS

The first-order issues are encountered immediately when attempting to provide access to and integration of multiple information resources:

- multi-vendor machines (IBM PC/RT, IBM 4341, AT&T 3B2, etc.)
- physical connection (Ethernet, wide-area net, etc.)
- different databases (ORACLE__SQL, IBM's SQL/DS, flat files)
- information composition (formating)

The issues of multiple vendor machines and physical communication are inherent as long as information resources are dispersed across geographic locations, be they intra- or inter-organizational. For example, the Sloan recruiting database is implemented in an *IBM PC/RT* computer whereas the Sloan alumni database is stored in an *AT&T 3B2* computer. Communication protocols need to be established (e.g., TCP/IP LAN) between different machines for encapsulating the machine idiosyncrasies.

Assuming that hardware idiosyncrasies and networking problems are resolved, the next hurdle is the idiosyncrasies of different databases. For example, the recruiting database is developed in the ORACLE relational database, thus accessed through SQL type queries; whereas I.P. Sharp's Disclosure II financial database is accessed through a menu driven interface. Different query commands and the corresponding skill are required in order to obtain the information available from these various information resources.

Suppose that one is able to resolve the above problems, he will nevertheless encounter the information composition task which abounds with second-order issues such as:

- database navigation (where is the data for alumni position, base salary, etc.)
- attribute naming (*company* attribute vs. *comp__name* attribute)
- simple domain value mapping ($, ¥, and £)
- instance identification problem (*IBM Corp* in one database vs. *IBM* in another database)

Database navigation is needed in order to determine which database to access to get the required information. Furthermore, on a menu-driven database, e.g., Reuter's Textline, it is important to know which menu path to access in order to save not only time but also access cost. Similarly, in a relational database system, it is necessary to know in which tables the required data is located (e.g., alumni position, company name) so that appropriate SQL queries can be formulated.

Entity and attribute names may be termed differently among databases, such as company vs. comp__name. This type of issues have been referred as the schema-level integration problem [1, 9, 15, 24]. In addition to the schema level integration, it is necessary to perform mapping at the instance level. For example, sales may be reported in $100 millions, but revenue in $millions. Furthermore, in a multi-national environment, financial data may be recorded in $, ¥, or £ depending on the subsidiary.

The *instance identification* problem becomes critical when multiple independently developed and administered information systems are involved because different identifiers may be used in different databases, e.g., *IBM Corp* vs. *IBM*. In the more complicated cases, no common key identifiers are available for joining the data across databases for the same instance. It is the inter-database instance identification problem that we focus in the rest of the paper, as the example in the following section manifests.

III. INTER-DATABASE INSTANCE IDENTIFICATION

Imagine a professor and his teaching assistant (TA) discussing the performance of one of their students. We can view each of them as maintaining a database containing various types of information on the same group of students. A conversation will typically begin by the professor identifying one of the students by *name*, following which both will volunteer information about that student (e.g. grades, performance). This is an example of joining information from two databases by means of a primary-foreign key join -- in this case, using *student name* as that key.

Suppose, however, that while the professor knows the students by *name*, the TA identifies them by means of *nicknames* that he has attached to them (e.g. Sleepy, Dopey). This would cause a real problem of making sure that they are even talking about the same person because the mapping between names and nicknames isn't captured -- there is no longer a primary-foreign key relationship. However, they are likely to pursue other ways of mutually identifying the student, as the following discussion manifests:

(Professor):	Do you know who TK Wong is ?
(TA):	No. Does he come to the morning class ?
(Professor):	Yes, when he comes at all.
(TA):	How well is he doing in the class ?
(Professor):	Not well. He's always falling asleep.
(TA):	Is he quiet ?
(Professor):	No ! He keeps complaining about our LISP compiler.
(TA):	Oh, sure ! I call that guy Big Mouth.

So, even though there is no common unique key, there may be a way of using other shared (non-unique) attributes (e.g., attendance, performance) which can be used to eliminate all other possibilities. This technique we call attribute subsetting [23] or more precisely, *inter-database instance identification.*

At first glance, this may seem like nothing more than searching for a common unique multiple-key identifier. There are two reasons why *inter-database instance identification* is more than just searching for a common unique multiple-key identifier [6]. First, there may be no way to eliminate all the possibilities, as opposed to the common unique multiple-key case. For example, the professor and TA may at best reduce the possibilities to three. At that point they may pick up the student directory and look at pictures of the three students for final identification. By the same token, in a database environment, while the process can help identify the same instance across databases, some (hopefully small) degree of user input will also be required as well. However, this will be much less painstaking than checking through the pictures of each class member (or each instance in the database).

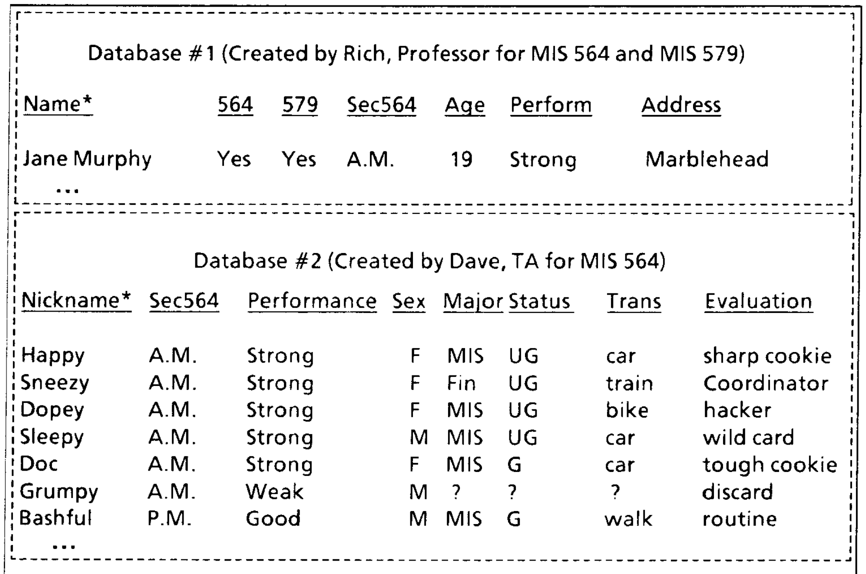
The second reason is more interesting. As well as comparing shared attributes, the professor and the TA may also be able to make inferences that can help them in the identification process. As an example, consider the same type of professor-TA example as above, this time as depicted in Figure 3. Suppose that Rich, the professor for *Management Information Technology* (MIS 564) and *Communication and Connectivity* (MIS 579), has a database of students who take 564 and 579; while Dave, the TA for 564, has a database for the 564 students. In preparation for final grading, Rich and Dave need to pool information about all the students. In this case Dave is trying to identify someone Rich calls *Jane Murphy*. There are two common attributes in the two database, i.e., sec564 and performance. By applying these two attributes, the candidate students that correspond to *Jane Murphy* are reduced from the entire database to 5 (i.e., those who attend the A.M. section of 564 with strong performance, as shown in the first five rows of the TA's database.)

Using the other attributes in these databases, plus auxiliary databases and inferencing rules, one may come to the conclusion that Jane Murphy is "*Happy*." The logic goes as follows:

- Jane is 19 years old; therefore, the status is most likely "UG" (undergraduate) [this eliminates "*Doc*"].
- Assuming the availability of a database of typical male and female names, we can conclude that Jane Murphy is a female [this eliminates "*Sleepy*"].
- Jane lives in Marblehead. Assuming a distance database of locations of New England exists, we determine that Marblehead is 27 miles from Cambridge and therefore, it is unlikely that the transportation type is bike [this eliminates "*Dopey*"].
- Jane takes 564 and 579 which are the core courses for MIS major; therefore, it is more logical to conclude that Jane Murphy is majoring in MIS [this eliminates "*Sneezy*"].

Therefore, Jane Murphy is "*Happy*" who is a *sharp cookie.* Note that this analysis requires a combination of database and artificial intelligence techniques.

Thus, even though only a few attributes are common to both databases, further comparisons can be made because of the relationships between the data. These kinds of relationships are likely to occur in a CIS environment precisely because of the heterogeneity: fragmentation of information is frequently caused by the fact that separate organizations are interested in different attributes of the same entity. For example, the registrar's office is likely to be concerned about a student's course schedule and home address, the bursar's office is likely to be concerned about his



Database #1 (Created by Rich, Professor for MIS 564 and MIS 579)

Name*	564	579	Sec564	Age	Perform	Address
Jane Murphy	Yes	Yes	A.M.	19	Strong	Marblehead
...						

Database #2 (Created by Dave, TA for MIS 564)

Nickname*	Sec564	Performance	Sex	Major	Status	Trans	Evaluation
Happy	A.M.	Strong	F	MIS	UG	car	sharp cookie
Sneezy	A.M.	Strong	F	Fin	UG	train	Coordinator
Dopey	A.M.	Strong	F	MIS	UG	bike	hacker
Sleepy	A.M.	Strong	M	MIS	UG	car	wild card
Doc	A.M.	Strong	F	MIS	G	car	tough cookie
Grumpy	A.M.	Weak	M	?	?	?	discard
Bashful	P.M.	Good	M	MIS	G	walk	routine
...							

Figure 3 Student Databases Without Common Key Identifier

financial status (tuition and fines owed), while the campus police is concerned whether he has been issued a parking sticker. In such a system there may be little opportunity to directly compare data between the different databases. Using heuristics, though, we may be able to make further comparisons, as discussed below.

IV. INTER-DATABASE INSTANCE IDENTIFICATION IN CIS/TK

The preceeding example displays the process of *inter-database instance identification*, but it also raises several questions. For example, how is the knowledge that a bike is an inappropriate form of transportation from Marblehead to school stored? Part of it is knowing that the distance between Marblehead and school is 27 miles (distance), part is knowing what constitutes an acceptable commute for a student (in terms of time), and part is knowing which types of transportation can satisfy those distance and time constraints. The rules which determine this choice should represent as much of this knowledge as possible so that the system is both understandable and flexible. Thus, *a rule such as :*

IF address = "Marblehead"

THEN transport = "Car" or "Train"

would be unacceptable because it ignores much of the chain of logic. Furthermore, the system would require one such rule for each possible town, which obscures the simplicity of the underlying logic that walking and biking are unacceptable modes of transportation for long commmutes.

These issues must be solved before the inter-database instance identification process can be effectively applied. The knowledge and information processing capabilities we have developed for the CIS/TK ensemble can accommodate the *inter-database instance identification* technique naturally, as outlined below.

Knowledge and Information Processing Capabilities in the CIS/TK

The *knowledge processing capability* is built on an enhanced version of a Knowledge Object Representation Language. An object-oriented approach [16, 18, 19] is employed, whereby the entities in an application model are represented as objects and their attributes are represented as slots. Message passing is used as the communication mechanism between objects [22]. Heuristics act through the rule sets in the rule facets of the relevant objects in the application model. Rules are fired to infer either a value for an attribute (setting the VALUE facet of the attribute's slot) or a set of values for an attribute (setting the CHOICES facet of the attribute's slot). Two instance objects can then be compared to see if they are identical by either comparing the VALUE facets of each slot or by checking that the value in one's VALUE facet is among the set of values in the other's CHOICES facet.

By comparing each instance in a table with all the instances in the other table in this fashion, the same instance across databases may be identified and the information joined. This evaluation of each of the Cartesian products of the tables is equivalent to the procedure used for relational joins. The difference is that the information is embedded in the instance objects retrieved across databases instead of tuples of relations in the same database, and that inferencing is utilized.

Central to the CIS/TK *information processing capability* is a query processor architecture as shown in Figure 4 [12]. The architecture consists of an Application Query Processor (AQP), a Global Query Processor (GQP), and a Local Query Processor (LQP) to interface with the query command processor (e.g. a DBMS) for each database in the CIS.

The AQP converts an application model query, defined by an application developer, into a sequence of global schema queries, passes them on to the GQP, and receives the results.

The primary query processor is the GQP. It converts a global schema query into abstract local queries, sends them to the appropriate LQPs, and joins the results before passing them back to the AQP. The GQP must know where to get the data, how to map global schema attribute names to column names, and how to join results from different tables.

The LQP establishes the physical connection between the host and the appropriate remote machines where information is stored, transforms the abstract local query into the appropriate executable query commands for the remote system, sends the executable query commands to the actual processor, receives the results, and transforms the results to the standard GQP format.

Equally important in CIS/TK are the global schema and application models. The Global Schema is an integrated schema [1, 7, 8, 9, 23] that models the data and relationships which are available in a set of underlying disparate databases. Thus, its sole concern is the *available* data, and it may only partially reveal the richness of the underlying reality. On the other hand, the application model is best thought of as a mental model of a set of entities, which completely models their inter-relationships and exists independently of whether there is a lot, a little, or no data available.

Condition for Inter-Database Instance Identification

The heuristics that are used to supplement the inter-database instance identification technique reside in the application model environment. Queries are handled first by the AQP, which interacts with the application model, and then by the GQP which interacts with the global schema. The GQP is responsible for performing "simple" instance joins -- those which use a traditional primary-foreign key approach. If no such join is possible (as in the Professor-TA example) then the inter-database instance identification technique is employed by the AQP at the application model level. Figure 5 shows the global schema and an application model for the Professor-TA example, including the heuristics which are part of the application model.

When the AQP sends a query on to the GQP it typically receives a single table in return. If, however, the GQP was unable to join all the instances, then the AQP will receive more than one table in response to its query. Thus, the GQP always responds to a query request with an argument list which contains, first, the number of tables being returned, followed by that number of tables.

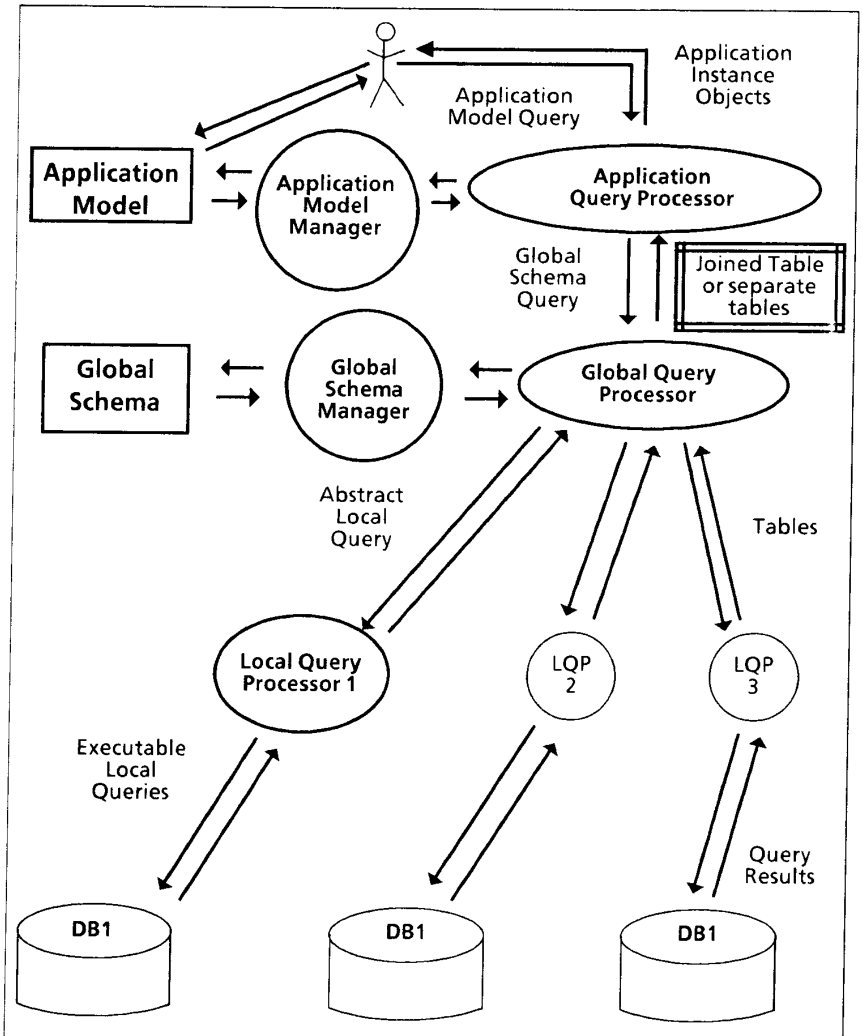


Figure 4 The CIS/TK Query Processor Architecture

If only a single table is returned, of course, inter-database instance identification isn't necessary and there are no interesting complications. The results can be simply returned to the end-user. If, however, more than one table is returned, then inter-database instance identification becomes necessary.

The Professor-TA Example Revisited

With the CIS/TK information and knowledge processing capabilities and the condition for inter-database instance identification in place, we illustrate the Inter-Database Instance Identification Algorithm (IDIIA)[3] using the Professor-TA example discussed earlier.

The initial objective is the same: to gather all the available information about Jane Murphy. The process proceeds as follows:

A. The AQP sends the following message to the GQP requesting data about Jane Murphy:

(send-message 'GQP :query '(student (Name Sex 564 579 Section Age Perform Major Status Transport Evaluation) (= Name "Jane Murphy")))

Note that the syntax used above is only for the application developers. An end-user would either be provided with a menu-based front end or use SQL type 4th generation language (4GL) to interact with the CIS/TK [24]. For example, an equivalent SQL query for the above message is shown below:

[3] Pronounced as "idea".

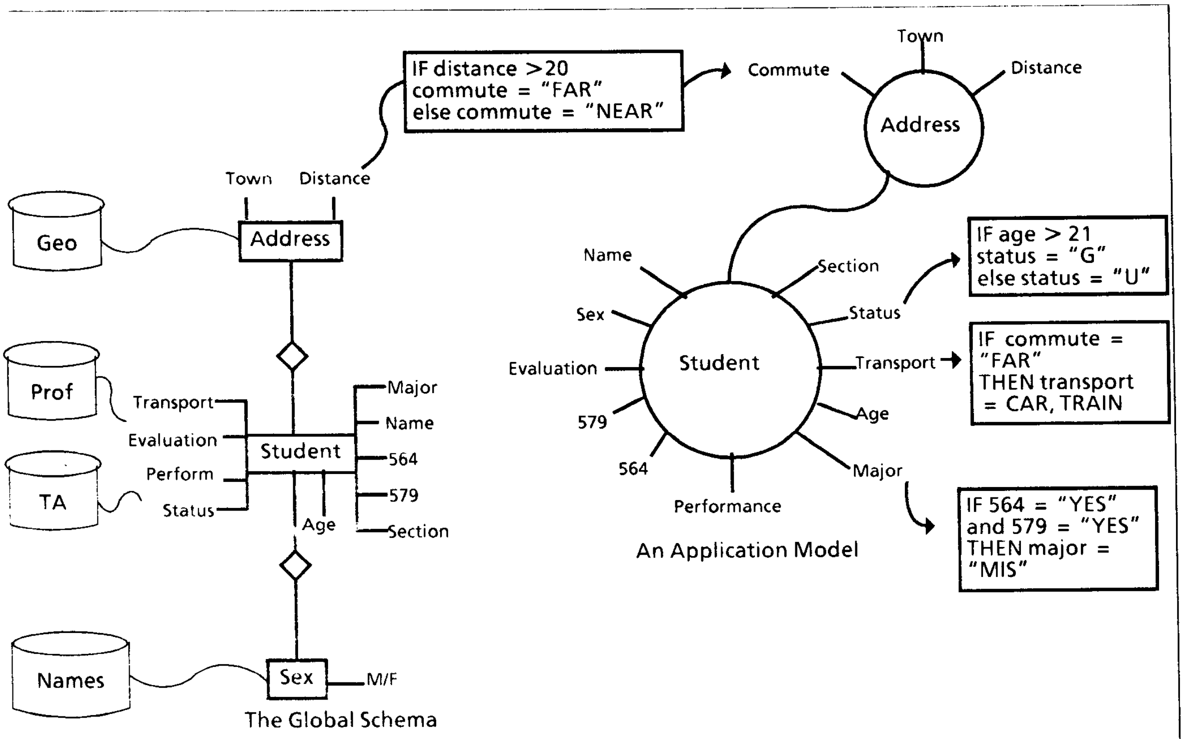


Figure 5 The Global Schema and An Application Model for the Professor-TA Example

```
Select   name, sex, 564, 579, section, age,
         perform, major, status, Transport
         Evaluation
From     student
Where    name = "Jane Murphy";
```

B. Because there is no primary-foreign key relationship to join the entities in the Professor and TA database, the GQP is unable to perform the join. Therefore, the GQP returns two tables (as "2" in the beginning of the returned list indicates):

```
(2
(( Name          Sex   564   579   Section   Age
("Jane Murphy"F        Yes   Yes   A.M.  19

Perform)
Strong))

((Nickname  Section   Perform Sex  Major    Status
(Happy      A.M.      Strong   F   MIS      UG
(Sneezy     A.M.      Strong   F   Fin      UG
(Dopey      A.M.      Strong   F   MIS      UG
(Sleepy     A.M.      Strong   M   MIS      UG
(Doc        A.M.      Strong   F   MIS      G
(Grumpy     A.M.      Weak     M   ?        ?
(Bashful    P.M.      Good     M   MIS      G

Transport   Evaluation)
car         sharp cookie)
train       Coordinator)
bike        hacker)
car         wild card)
car         tough cookie)
?           discard)
walk        routine)
   . . . . . . . . . . . . . . . . . . . . . . )))
```

At this point, The AQP invokes[4] the Inter-Database Identification Algorithm (IDIIA) which in turn initiates the following process:

1. Use the common attributes (section and perform) to reduce the potential candidates in the TA's database from the entire table (could be hundreds or thousands of instances depending on the application) to 5 [i.e., Happy, Sneezy, Dopey, Sleepy, and Doc].
2. Instantiates the 6 instances in the student object class [i.e., Jane Murphy, Happy, Sneezy, Dopey, Sleepy, and Doc].

[4] The IDIIA can be implemented as an object; in which case message passing will be used as the communication mechanism between the IDIIA and AQP as well as GQP.

3. Identify all the RULE facets in the student object class. There are three slots which have rule sets attached to the corresponding RULE facets: the *transport* rule set, the *status* rule set and the *major* rule set, as shown in Figure 6. The transportation rule set determines the type of commute depending on the distance. Similarly, the rule sets for major and status determines a student's major and status depending on the instance values[5].
4. For each rule set, the IDIIA first checks each of the 6 instances to see if its corresponding VALUE facet has been instantiated. If the answer is yes, then the IDIIA moves on to the next instance because there is no need to infer a value; otherwise, it examines each rule in the rule set against the instance in a backward chaining fashion to see what data is needed to infer the value. So for the transportation rule set, the IDIIA requests only for those instances which currently have no value for transport; in this case, only Jane Murphy [Happy, Sneezy, Dopey, Sleepy, and Doc all have transport value].

Following the notion of backward chaining, the IDIIA recognizes the need to get information about address [from rule 3 in the transportation rule set]. In order to formulate the appropriate condition, it sends a message to the GQP requesting a key which can be used to join the student and the address entities:

> *(send-message 'GQP :get-shared-key (student address))*

In response to the message, the GQP returns (as "-->" indicates) name in the student entity as the key to join student and address.

> *--> (student name)*

The IDIIA then sends the following message to get the distance information:

> *(send-message 'GQP :query '(address (town distance commute)*

The rule set for transportation

```
1.  (IF (> = (<address) distance 20)
        (THEN (= (>address) commute "FAR")))*
2.  (IF (< (<address) distance 20)
        (THEN (= (>address) commute "NEAR")))
3.  (IF (= (<student address) commute "FAR")
        (THEN (= (>student) transport CHOICES ("Car" "Train")))
4.  (IF (= (<student address) commute "NEAR")
        (THEN (>student) transport CHOICES ("Bike" "Walk" "Car"
        "Train")))
```

The rule set for major

```
1.  (IF (= (<student) 564 "YES") and (= (>student) 579 "YES")
        (THEN (= (>student) major "MIS")
```

The rule set for status

```
1.  (IF (> = (<student) age "21")
        (THEN (= (>student) status "U")))
2.  (IF (< (<student) age "21")
        THEN (= (>student) status "G"))
```

* The rule reads as follows: IF the distance for the address is greater or equal to 20 (miles), THEN bind the commute value of the address instance to FAR.

The symbol ">" is used to mean "greater" and "unbound variable" depending on the position in the rule; similarly "<" means either "smaller" or "bind variable".

Figure 6 Rule Sets for the Professor-TA Scenario

[5] Two observations can be made here: (a) The concept of "far" and "near" is somewhat subjective. After all, that is one reason why it is represented as heuristic rules to begin with so the rationale can be checked through the "why" mechanism in the inference engine. (b) These rules are by no means absolute. They can be further refined to reflect the details.

(= (student name) "Jane Murphy")))

In response to the message, the GQP returns 1 table with 3 columns: town, distance, and commute. One instance is retrieved, i.e., (Marblehead 25 nil).

--> (1 ((town distance commute) (Marblehead 25 nil)))

The address object is instantiated with the data and linked to Jane Murphy. Note that there is no data available for *commute* in the database. Therefore, nil is returned and the value will be inferred through the transportation rule set.

The process of backward chaining may continue on depending on the situation. In this case, no further database requests are necessary because the address object does not request additional information from other objects so the backward chaining process terminates at this point.

By the same token, the rule sets for status and major are examined as follows: The IDIIA first checks the major VALUE facet, and finds that all the instances except Jane Murphy have a major ["MIS" or "FIN" or "?"]. Next the IDIIA parses the *major rule set* for *Jane Murphy* and sees that it requires information about 564 and 579 in order to infer values about major. Since the 564 and 579 information for Jane Murphy already exists [as the initial condition to IDIIA], no additional data need to be requested from the GQP.

Similarly, the IDIIA first checks the status VALUE facet, and finds that all the instances except Jane Murphy have a status ["UG" or "G"]. Next the IDIIA parses the *status rule set* for *Jane Murphy* and sees that it requires information about age in order to infer the value about status. Since the age information for Jane Murphy already exists [as the initial condition to IDIIA], no additional data need to be brought in from the GQP.

5. Now the IDIIA is ready to use the heuristics to infer additional information about the students. For each of the student attributes which currently have no value in the VALUE facet, but for which heuristics exist, the associated student information is placed into working memory and the rule set forward-chained. Thus, the transport, status and major heuristics for Jane Murphy are tested and the results placed in the instance:

   transport: CHOICES ("Car" "Train")

   status : "U"

   major: "MIS"

6. Now the comparison of instances can proceed. Each instance from the first table (in this case just "Jane Murphy") is compared with the 5 instances [Happy, Sneezy, Dopey, Sleepy, and Doc] in the second table to see if they match. Comparisons are performed on a slot-by-slot basis for any matching slots which both contain data in either the VALUE or CHOICES facet. As before, we find that Jane Murphy matches only with "Happy".

Note that knowledge is represented both in heuristics and in database format [14]. The knowledge of the distance between Marblehead and Cambridge, for instance, was retrieved from a geographical database, while the concept of "FAR" and "NEAR" commutes and the appropriate mode of transport for each is represented by heuristic rules. Likewise, the knowledge of which first names are (typically) male and which female was also retrieved from a database containing potential names for infants. It is appropriate to capture this knowledge in a database because a substantially greater number of rules would be needed if this information were to be represented by heuristic rules.

V. DISCUSSION AND CONCLUDING REMARKS

We have presented the inter-database instance identification technique in this paper. It has provided a solid base for further optimization and extension of the identification problem. Work is in progress to formalize the inter-database instance identification technique as an algorithm. Furthermore, inter-database instance identification under uncertainty as well as partial matching techniques are being developed to tackle the even more complicated situations where deterministic inferencing is not sufficient.

Another closely related research issue that we are addressing is a more elegant representation of the rule sets currently attached to the RULE facet of an object slot. We are actively designing and testing the "*concept agent*" which behave as an autonomous object. Each *concept agent* is tasked with a single goal and adheres to a well defined specification for rule syntax and communication protocols. Each rule set may be encapsulated in a concept agent which has reasoning capabilities based on the rule set as well as other internal functionalities. For example, a major concept agent will be able to determine a student's major, and major only, given the right protocol. The major concept agent may in turn call another two concept agents: the core concept agent and the elective concept agent. With a number of concept agents made available, we will be able to draw inferences based on these concept agents -- a task we call *concept inferencing*. A *concept processor* is also being developed in our research to enable concept agents to respond to messages from objects in the CIS/TK such as the AQP, the GQP, and other concept agents.

Our focus is on real, exciting, and nontrivial research problems. We are actively researching inter-database instance identification problems in life databases. For instance, Reuters' Textline, Dataline, and Newsline databases as well as its I.P. Sharp subsidiary's databases have been applied as a testbed for interesting research issues. We believe that this effort will not only contribute to the academic research frontier but also benefit the business community in the foreseeable future.

REFERENCES

1. Batini, C. Lenzirini, M. and Navathe, S.B. A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys, Vol. 18, No. 4, (December 1986), pp. 323 - 363.

2. Benjamin, R.I., Rockart, J.F., Scott Morton, M.S., and Wyman, J. Information technology: a strategic opportunity. Sloan Management Review, Vol. 25, No. 3, (Spring 1985), p. 3-10.

3. Benson, R.J. and Parker, M. M. Enterprise-Wide Information Management: An Introduction to the Concepts. IBM Los Angeles Scientific Center, G320-2768, (May 1985).

4. Cash, J. I., and Konsynski, B.R. IS Redraws Competitive Boundaries. Harvard Business Review, (March-April 1985), 134-142.

5. Clemons, E.K. and McFarlan, F.W., Telecom: Hook Up or Lose Out. Harvard Business Review, (July-August, 1986).

6. Date, C. J. An Introduction to Database Systems Third Ed., Addison-Wesley Publishing Company, (1981)

7. Dayal, U. and Hwang, K. View Definition and Generalization for Database Integration in Multidatabase System. IEEE Transactions on Software Engineering, Vol. SE-10, No. 6, November 1984, pp. 628-644.

8. Deen, S. M., Amin, R.R., and Taylor M.C. Data integration in distributed databases. IEEE Transactions on Software Engineering, Vol. SE-13, No. 7, (July 1987) pp. 860-864.

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

10. Frank, W.F., Madnick, S.E., and Wang, Y.R. A Conceptual Model for Integrated Autonomous Processing: An International Bank's Experience with Large Databases. Proceedings of the 8th Annual International Conference on Information Systems (ICIS), (December 1987), pp. 219-231.

11. Goldhirsch, D., Landers, T., Rosenberg, R., and Yedwab, L. MULTIBASE: System Administrator's Guide. Computer Corporation of America, Cambridge, MA, (November 1984).

12. Horton, D.C. An Object-Oriented Approach Towards Enhancing Logical Connectivity in a Distributed Database Environment. Master's Thesis, Sloan School of Management, MIT, (May 1988).

13. Ives, B. and Learmonth, G.P., The Information Systems as a Competitive Weapon. Communications of the ACM, Vol. 27(12), (December 1984), pp. 1193-1201.

14. Kerschberg, L. Ed. Expert Database Systems, Proceedings from the First International Workshop. The Benjamin/Cummings Publishing Company (1986).

15. Litwin, W. and Abdellatif, A. "Multidatabase Interoperability," IEEE, Computer, (December 1986).

16. Lyngbaek, P. and McLeod D. An approach to object sharing in distributed database systems. The Proceedings of the 9th International Conf. on VLDB, (October, 1983).

17. Madnick, S.E. and Wang, Y.R. Evolution Towards Strategic Applications of Databases Through Composite Information Systems. To Appear in the Journal of Management Information Systems, (Fall, 1988).

18. Manona, F. and Dayal, U. PDM: An Object-Oriented Data Model Proceedings International Workshop on Object-Oriented Database Systems. Pacific Grove, CA. (September 1986) pp. 18 - 25.

19. Ontologic Inc. Vbase Integrated Object System. 47 Manning Rd., Billerica, MA 01821 (November 1987).

20. Porter, M. and Millar, V.E., How information gives you competitive advantages. Harvard Business Review (July-August 1985) p. 149-160.

21. Rockart, J.F. and Scott Morton, M.S., Implications of changes in information technology for corporate strategy. Interfaces, Vol. 14, No. 1 (January-February, 1984), pp. 84-95.

22. Stefik, M. and Bobrow, D.G. Object-Oriented Programming: Themes and Variations. The AI Magazine, Vol. 6, No. 4, (Winter 1986), pp. 40 - 62.

23. Wang, Y.R. and Madnick, S.E. Facilitating Connectivity in Composite Information Systems. To Appear in ACM Database.

24. Wang, Y.R. and Madnick, S.E. Connectivity Among Information Systems. WP# 2025-88, Sloan School of Management, MIT (June 1988).