# INTER-DATABASE INSTANCE IDENTIFICATION IN COMPOSITE INFORMATION SYSTEMS

Y. Richard Wang
Stuart E. Madnick
David C. Horton

JUNE 1988 WP # 2029-88

# *Inter-Database Instance Identification in Composite Information Systems*

Y. Richard Wang
Stuart E. Madnick
Sloan School of Management, E53-320
Massachusetts Institute of Technology
Cambridge, MA 02139

David C. Horton
Ford Motor Company
The American Road
Dearborn, Michigan 48121

**ABSTRACT** Many important information systems applications require multiple disparate databases to work together within and across organizational boundaries. These systems have been referred to as Integrated Information Systems, Federated Systems, or *Composite Information Systems* (CIS). 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 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.

**KEY WORDS AND PHRASES**: database management systems, composite information systems, strategic computing, systems development.

**ACKNOWLEDGEMENTS** Work reported herein has been supported, in part, by Reuters, Ford Motor Company, the National Computer Board of Singapore, and the MIT Center for Information Systems Research.

# I. INTRODUCTION

Many strategic information systems require multiple disparate systems which were developed and administered independently to work together within and across organizational boundaries. These systems have been referred to as Federated Systems [14], Heterogeneous Distributed Database Systems [9], Integrated Information Systems [8], Organizational Information Systems [2], Strategic Information Systems [3, 12], or *Composite Information Systems* (**CIS**) [15, 16, 18, 19]. CIS-type applications are increasingly being deployed by corporations to produce composite information from existing operational systems to support line managers' decisions without major rewrites of current systems. Many research issues such as inter-system inconsistency, ambiguity, and contradiction need to be resolved in order to produce composite information [1, 5, 6, 7, 18].

A critical issue involved in CIS is the ability to join information about a particular instance[1] from disparate databases. In conventional homogeneous centralized and distributed data base management systems, these joins are performed using a primary-foreign key relationship [4]. However, this type of join may not always be possible in a CIS environment because the primary key identifiers may not be compatible across databases. As a result, they would not be applicable for joining information. We have found this phenomenon ubiquitous in the CIS environment.

In the simpler case, a common unique key exists, but is coded ambiguously. For example, *IBM* is coded as "IBM Corp" in The MIT Sloan School's alumni database but "IBM Corporation" in its placement Database [20]. To join information about

---

1. The term "instance" and "instance object" are used throughout the paper to mean an instantiation of an entity or object (class).

IBM from both the alumni and placement databases, it is necessary to realize that "IBM Corp" and "IBM Corporation" both refer to the same corporation.

In the more complicated case, a common unique key does not exist (e.g. students may be coded by *name* in one database and *nickname* in another; as a result, no direct mapping exists). A technique called *inter-database instance identification*, or *attribute subsetting* [18], is presented in this paper to deal with the more complicated case. It employs a combination of database management systems and artificial intelligence techniques to identify the same instance across databases, and optionally, retain this mapping in an *Inter-Database Instance Identification Table* (IDIIT) for later use. Table 1 exemplifies an IDIIT for corporation instances.

Table 1 An IDIIT for the alumni and placement Databases

| MIT Alumni Database | MIT Sloan Placement Database |
|---|---|
| IBM Corp | IBM Corporation |

Section II presents a scenario of a professor and his teaching assistant (TA) engaging in the process of identifying a student in their class. The concept of inter-database instance identification is manifested in the scenario. Section III presents the algorithm 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, information processing, physical and logical connectivity, and user interfaces. Concluding remarks are made in section IV.

---

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.

## II. AN INTER-DATABASE INSTANCE IDENTIFICATION SCENARIO

Imagine a professor and his 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[3]. 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

---

3. A primary key is the attribute in a relation which uniquely identifies a tuple. A foreign key is an attribute in a relation which is also the primary key in another relation. Primary and foreign keys provide a means of representing relationships between tuples [4, pp. 87 - 91].

used to eliminate all other possibilities. This technique we call *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 [4]. 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 1. 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*"].

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 1 Student Databases Without Common Key Identifier

- 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 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.

## III. 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.

## An Overview of the CIS/TK Knowledge and Information Processing Capabilities

The *knowledge processing capability* is built on an enhanced version of a Knowledge Object Representation Language [13]. An object-oriented approach 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 [17]. 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 2 [10, 11]. 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,

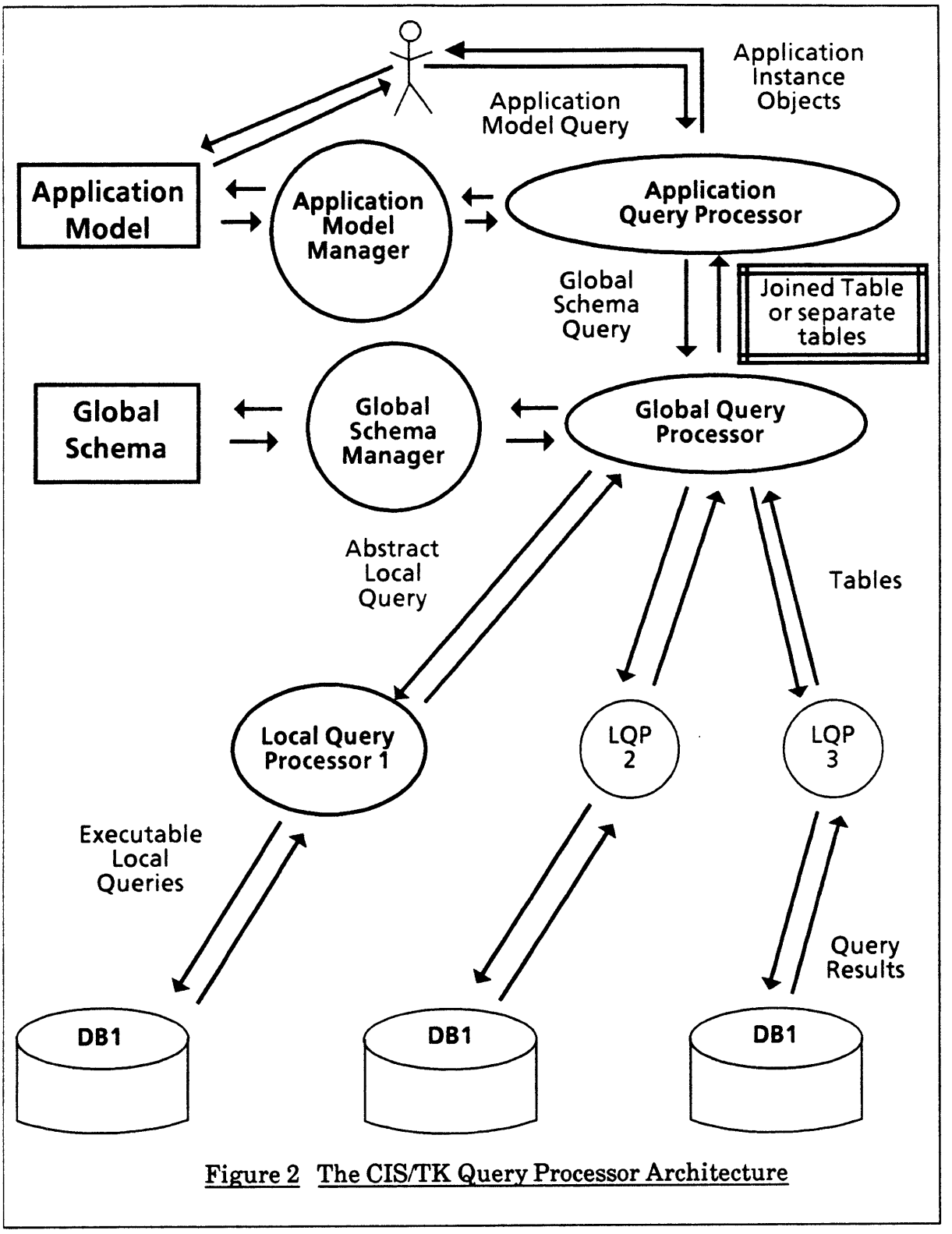


Figure 2 The CIS/TK Query Processor Architecture

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, 5, 6, 7, 18] 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 3 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

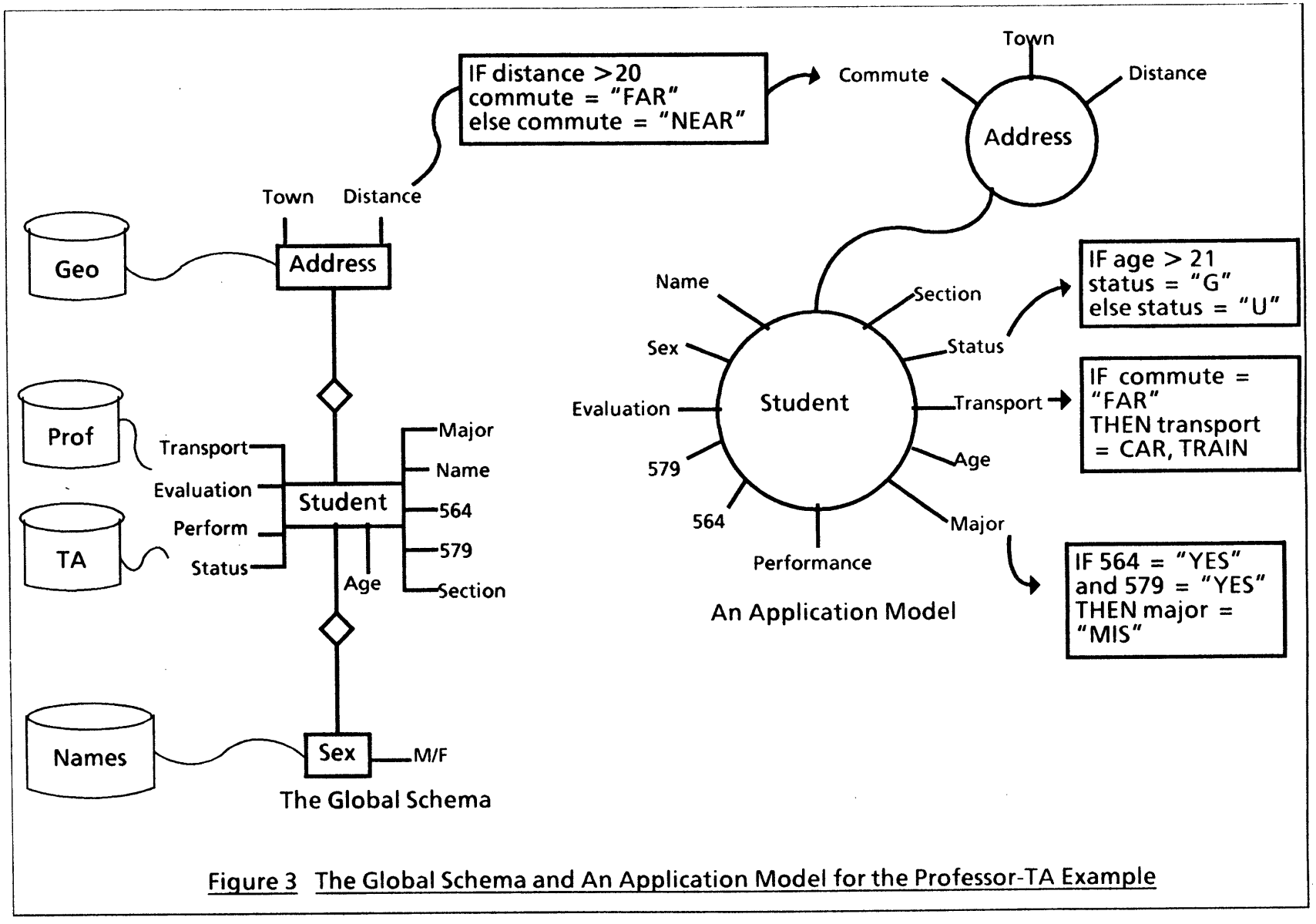


Figure 3 The Global Schema and An Application Model for the Professor-TA Example

to a query request with an argument list which contains, first, the number of tables being returned, followed by that number of tables.

*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)[5] 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 [19]. For example, an equivalent SQL query for the above message is shown below:

---

5. Pronounced as "idea".

*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 Perform)*
*("Jane Murphy" F Yes Yes A.M. 19 Strong))*

| *((Nickname* | *Section* | *Perform* | *Sex* | *Major* | *Status* | *Transport* | *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)* |

. . . . . ))) 

At this point, The AQP invokes[6] 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].

---

6. 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.

2. Instantiates the 6 instances in the student object class [i.e., Jane Murphy, Happy, Sneezy, Dopey, Sleepy, and Doc].

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 4. 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[7].

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))*

---

7. 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.

**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 4 Rule Sets for the Professor-TA Scenario

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)*

*(= (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. 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.

## VI. 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. Cash, J. I., and Konsynski, B.R. IS Redraws Competitive Boundaries. Harvard Business Review, (March-April 1985), 134-142.
3. Clemons, E.K. and McFarlan, F.W., Telecom: Hook Up or Lose Out. Harvard Business Review, (July-August, 1986).
4. Date, C. J. An Introduction to Database Systems Third Ed., Addison-Wesley Publishing Company, (1981)
5. 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.
6. 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.
7. Elmasri R., Larson J. and Navathe, S. Schema Integration Algorithms for Federated Databases and Logical Database Design. Submitted for Publication, 1987.
8. 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.
9. Goldhirsch, D., Landers, T., Rosenberg, R., and Yedwab, L. MULTIBASE: System Administrator's Guide. Computer Corporation of America, Cambridge, MA, (November 1984).
10. Horton, D.C., Madnick, S.E., Wang, Y.R., and Wong, T.K. The Design and Implementation of the CIS/TK Query Processor Architecture. Technical Report # CIS-88-02, Sloan School of Management, MIT, (April 1988).
11. Horton, D.C., Madnick, S.E., Wang, Y.R., and Wong, T.K. The Translation Facility for the CIS/TK Query Processor Architecture. Technical Report # CIS-88-03, Sloan School of Management, MIT, (April 1988).
12. 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.
13. Levine, S., Interfacing Objects and Database. Master's Thesis, Electrical Engineering and Computer Science, MIT, (May 1987).
14. 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).
15. Madnick, S.E. and Wang, Y.R. A Framework of Composite Information Systems for Strategic Advantage. Proceedings of the 21st Hawaii International Conference on Systems Sciences (January 1988) pp. 35 - 43.

16. 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).

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

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

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

20. Wang, Y.R., Banks, A. D., Kooper, L. S. and Flamburis, A. Placement Assistant System. Technical Report # CIS-88-05, Sloan School of Management, MIT, (May 1988).