REPRESENTING ROLES IN UNIVERSAL SCHEME INTERFACES

D. MAIER, D. ROZENSHTEIN, & J. STEIN

Oregon Graduate Center

OGC Technical Report No. CS/E-83-01

REPRESENTING ROLES IN UNIVERSAL SCHEME INTERFACES

David Maier*
Oregon Graduate Center

David Rozenshtein†
Jacob Stein†
State University of New York at Stony Brook

ABSTRACT

In universal scheme interfaces to relational databases, an attribute name must represent a unique role in the database, so that the connection among a set of attributes is unambiguous. A drawback to this requirement is that several attributes can represent the same underlying class of of entities, but the relationship among those attributes is not captured in the database scheme. As our method for relating attributes uses natural joins, some semantically meaningful connections among attributes are lost. We take a step toward removing this drawback by explicitly storing role relationships between attributes. The connections we lose enforcing our unique role assumption are recovered via equijoins based on these role relationships. We give conditions on such role connections to ensure that they are semantically consistent with other connections, and show how to extend an existing universal scheme language, PIQUE, to use this role information.

1. Introduction

In the relational model the role of an attribute is unique within a single stored relation; however, when navigating between relations, ambiguities may arise concerning the role an attribute plays. Therefore, it is often necessary to specify access paths (relation names) in order to insure that an attribute plays a particular role in a derived relation. If the path is improperly specified, an attribute may play one or more roles other than the one intended However, it is unreasonable to expect the user to know the detailed logical structure of the database. This is especially so if the database's scheme is large or subject to restructuring.

A possible solution is the use of views. While these virtual relations hide the access paths from the user, he must still know the virtual relations' names and schemes. Since the number of views often exceeds the number of stored relations, this solution seems of little use in databases with non-trivial schemes.

A more promising approach is the use of *universal scheme interfaces*. A universal scheme interface is one where the database is accessed solely through its attributes. In

*Work supported by NSF grants IST 81-04834 and MCS 82-07216.
†Work supported by NSF grant IST 81-04834.

Section 2 we discuss how this is done. Essentially, all the semantics of the database are loaded onto the attributes. The assumption is that the attribute names correspond naturally to entities in the real world, and that the users have an intuitive understanding for these entities and their roles that is close to the semantics of the database. The user deals with but one relation scheme, namely, the *universal scheme* over the universe of attributes U.

We discuss two constraints on a database that are necessary for a universal scheme interface to present the database as a consistent semantic whole. In Section 2 we discuss the *universal relation scheme assumption*. In Section 3 we discuss the *unique role assumption*, which requires that an attribute play but a single role within the universal scheme, and its implications. One of these implications is that the size of U is often larger in a universal scheme interface than in a corresponding relational database. In Section 4 we provide several examples where this is the case.

The unique role assumption also prevents certain information, which is expressible in the relational model, from being expressed in universal scheme interfaces. In particular, in the relational model relation names are often used to disambiguate the role an attribute plays. Since there are no relation names in a universal scheme interface, they can not be used to disambiguate an attribute's role. As an example consider a simple relational database scheme with relations $employee(EMP\#, DEPT, SAL)$ and $manager(EMP\#, DEPT)$. In relation $employee$ attribute $EMP\#$ represents an employee. In relation $manager$ attribute $EMP\#$ also represents an employee, but in the role of a manager. The semantic meaning of $EMP\#$ is disambiguated by the name of the relation it appears in. In a corresponding universal scheme interface the attribute $MNGR\#$ might have to be introduced to play the role that $EMP\#$ does in the $manager$ relation. Nowhere in the universal scheme, however, would it be indicated that $MNGR\#$ and $EMP\#$ represent the same class of entities in different roles. Yet, such role information may be needed in order to find a manager's salary, and the burden to provide it would fall upon the user.

Thus, in order for universal scheme interfaces to truly free the user from the need to know the logical structure of the database, the role information must be incorporated into the interface itself. We shall examine the use of role hierarchies to incorporate the role information into universal scheme interfaces. In many cases the user will not have

to specify either relation names or access paths. In Section 5 we discuss role hierarchies, and provide a structure to represent them. In Section 6 we incorporate role hierarchies into the PIQUE universal scheme interface (MW) so that we may automatically generate certain relations, which would otherwise require the explicit specification of access paths. In Section 6 we also discuss incorporating these hierarchies into other universal scheme interfaces.

2. Universal Scheme Interfaces

In a universal scheme interface the database is accessed by specifying a set of attributes. For each database state and set of attributes X, contained in U, a relation over X is associated with X. This is done via a function $[X]$, called the *window for* X, which maps a database state to a relation on X. We shall denote the evaluation of $[X]$ on database state d as $[X](d)$, and call it the *window for* X *on* d. The window functions for most universal scheme interfaces differ from arbitrary views in that there is a uniform discipline that defines them based solely upon the database's scheme. In addition, as we shall see, the windows in a universal scheme interface, unlike an arbitrary set of views, are semantically connected. A *window generator*, denoted as $[\cdot]$, is a functional that maps a relation scheme (sets of attributes) to a function from databases to relations over the scheme. One may think of $[\cdot]$ as the set of window functions (one for each subset of U). We shall abuse the notation by writing $[X]$ to mean $[X](d)$ when d is understood and the context is clear.

For a universal scheme interface to a database to be conceivable, any attribute in U must correspond to the same class of entities wherever it appears; otherwise, there is no way to distinguish one class of entities from another when accessing a database. The attribute *NAME* cannot refer to part names in one place and peoples' names in another. If in an interface any attribute in U corresponds to the same class of entities everywhere it appears, the interfaces said to satisfy the *universal relation scheme assumption* (URSA). Henceforth, we assume universal scheme interfaces that satisfy the URSA. As we shall see in the next section, a universal relation scheme interface must satisfy the stronger *unique role assumption* in order for the window function to return semantically consistent relations.

Query processing in a universal scheme interface can be viewed as a two step operation. First, the set of attributes, say X, appearing in the query is determined. On the basis of the state of the database, a relation over X, specifically $[X](d)$, is generated. (If the query contains several tuple variables, then a relation is generated for the set of attributes associated with each tuple variable.) Second, further operations specified by the query, such as selections, projections, and joins, are performed on the generated relation(s) to produce the answer. These two steps are *binding* and *evaluation*.

Currently, there are several universal scheme interfaces at various stages of development; among them are: the interface of Shenk and Pinkert (SP); APPLE (CK); q (AK); System/U (Ko, KU, Ul); PIQUE (MRSSW, Ro, St); FIDL (Ba); Parafrase (KMRS, KS). Maier, Rozenshtein and Warren (MRW) provide a comparison of these systems. For the purpose of exposition, we shall use the PIQUE universal scheme interface.

A PIQUE database scheme consists of a set of *associations* **A** and a set of *objects* **O**. Each association and each object is a set of attributes. In a database on a PIQUE scheme, associations correspond to allowable units of update. Semantically, a tuple over an association represents a non-decomposable fact. In the current implementation (St) there is a stored relation over X for each association X. Associations that are subsets of other associations are allowed in PIQUE. Although it is possible to store heterogeneous tuples in a single relation through the use of placeholders, the set of associations may be thought of as the scheme of the stored database. The relation on an association X is denoted as $r(X)$. Since **A** is a set, there can be at most one such relation for each X in **A**.

Each object corresponds to a unit of retrieval. Semantically, a tuple over an object represents a fact in the sense of Sciore (Sc1). These facts are derived from the non-decomposable facts represented in the stored relations on associations. Let W be an element of **O**. The relation on W, denoted $r'(W)$, is defined as the join of the relations on all associations contained in W:

$$r'(W) = \bowtie_{X \in \mathbf{A}, X \subseteq W} r(X).$$

To insure that $r'(W)$ as computed is indeed a relation on W, that the facts are indeed about W, each object is assumed to be the union of some associations. The window on a set of attributes X is then defined as the union of the projections onto X of the relations on all objects that contain X:

$$[X]_{\mathbf{A}\,\mathbf{O}} = \bigcup_{W \in \mathbf{O}, W \supseteq X} \pi_X(r'(W)).$$

Example 2.1: Consider a database with the attributes *OWNER*, *PRODUCT*, *DEALER*. (In the examples, we shall often abbreviate attribute names to the first character.) Let the set of associations be $\mathbf{A} = \{P, OP, PD\}$, meaning that products exist, owners own them and dealers stock them. Let $\mathbf{O} = \mathbf{A} \cup \{OPD\}$. The window on *OWNER DEALER* is computed by joining $r(OP)$ with $r(PD)$ and $r(P)$, and then projecting onto *OWNER DEALER*. A tuple $<o,d>$ in $[OWNER\ DEALER]$ means that owner o owns some product that is stocked by dealer d. If it were desired to also store information about which dealer an owner purchased a product from, then the association *OWNER PRODUCT DEALER*, would be added to **A**. The window $[OWNER\ DEALER]$ would then be computed by joining $r(OPD)$ with $r(OP)$, $r(PD)$ and $r(P)$, and then projecting onto *OWNER DEALER*. A tuple $<o,d>$ in $[OWNER\ DEALER]$ would mean that owner o purchased some product from dealer d.

Consider a request for a list of those dealers and owners where the owner has purchased a smoke shifter from the dealer. In PIQUE this request is posed as

retrieve *OWNER*, *DEALER* **where** (*PRODUCT* = 'smoke shifter').

In evaluating this query the selection *PRODUCT* = 'smoke shifter' is performed on the evaluation of the window $[OWNER\ DEALER\ PRODUCT]$. ■

3. The Unique Role Assumption

While the URSA guarantees that any attribute in U refers to the same underlying class of entities wherever it appears, it does not guarantee that the role an attribute plays is unique. If attribute *DATE* plays the role of birthdate in $[SS\#\ NAME\ DATE]$, then it should not play the role of hiring date in $[SS\#\ DATE]$, nor the role of reservation date in $[NAME\ DATE\ ROOM]$. The *unique role assumption* (URA) requires not only that an attribute always represent the same class of entities, but also that it always represent that class of entities in the same role. Since an exact definition of the role an attribute plays is dependent on the semantics of the real world situation the scheme is modeling, determining satisfaction of the URA is a human task. However, one can insure, in a mechanistic manner, that the role an attribute plays with respect to other attributes is consistent throughout the database. We propose the following as necessary for a universal scheme interface to satisfy the URA.

If the set of attributes X is contained in Y, the roles played by the attributes of X in $[Y]$ should be consistent with their roles in $[X]$. Let t be a tuple in $[Y]$. The X-value of t is a fact about X that is an aspect of a fact about Y represented by t. Therefore, the X-value of t should be present in $[X]$.

Example 3.1: Consider a database where U = {*STUDENT*, *COURSE*, *FACULTY*}. A tuple $<s, c, f>$ in [*STUDENT COURSE FACULTY*] means that student s takes course c with faculty member f, and a tuple $<s, c>$ in [*STUDENT COURSE*] means that student s takes course c. It would be surprising if the (*STUDENT COURSE*)-value of a tuple in [*STUDENT COURSE FACULTY*] were not present in [*STUDENT COURSE*]. This would not be surprising if the meaning of $<s, c, f>$ were that student s is a teaching assistant for course c under faculty member f, but then the URA would be violated. ■

A window generator satisfies the *containment condition* if for all $X, Y \subseteq U$, such that X is contained in Y, $\pi_X[Y] \subseteq [X]$. While this condition does not insure that the roles played by attributes in X are everywhere the same, it does insure that, in all windows $[Y]$ on sets of attributes Y containing X, the roles of the attributes in X are consistent with their roles in $[X]$. In a sense this implies that the semantics of the database is derived from the bottom up. For example, the tuples in [*STUDENT COURSE*] define the role of *STUDENT* relative to *COURSE*. Therefore, the role of *STUDENT* in [*STUDENT COURSE FACULTY*] must be consistent with that definition.

It is unlikely that a database not designed with the URA in mind will satisfy the assumption. To get satisfaction of the URA, attributes will often have to be renamed. For example, the roles of *DATE* above could be distinguished by renaming to *BIRTHDATE*, *HIRING_DATE* and *RESERVATION_DATE*. It may not be apparent, after renaming, that newly introduced attributes represent the same class of entities.

Of the universal relation interfaces mentioned in Section 1, PIQUE, FIDL, Parafrase and System/U always have window generators that satisfy the containment condition. In PIQUE satisfaction follows directly from the definition of the window function.

While objects are explicitly defined only in PIQUE and SYSTEM/U, sets of attributes can be defined that behave analogously in other window functions that satisfy the containment condition. A set of attributes W is an *implicit object* for a window generator $[\cdot]$

if there is a state of the database where the inclusion

$$[W] \supseteq \bigcup_{V \supset W} \pi_W[V]$$

is strict.

The window on an implicit object W may contain W-values that are not present in any tuple from any window on a set of attributes strictly containing W. Semantically, a tuple from a window on an implicit object W represents a fact about W that need not be an aspect of a fact about any set of attributes that strictly contains W. A window on a set of attributes that is not a implicit object is in some sense incomplete; the window is dependent upon attributes not seen through the window.

The containment condition requires that the window functions be consistent with each other. The next condition we propose requires that a database's window functions be consistent with the contents of the stored database. Let X be the scheme of some stored relation and let $r(X)$ be that stored relation. It would be surprising if $r(X)$ were not contained in $[X]$; a tuple could be inserted into $r(X)$ but not be retrievable. It would also be surprising if what was gotten back was more than what was put in; the window on X should be contained in $r(X)$. A window generator is *faithful* if for any stored relation $r(R)$ on scheme R, $r(R) = [R]$. If a window generator is faithful, the scheme of each stored relation is an implicit object.

The above definition assumes that in the database no two relations have the same scheme. Since having two relations on the same scheme makes sense only if the roles of some of the attributes in the scheme are not unique, any database that satisfies the URA will have at most one stored relation on a given scheme. In PIQUE faithfulness is assured by requiring that the containment condition be satisfied and that each association also be an object.

Henceforth, we shall assume universal scheme interfaces with window generators that satisfy the containment condition and are faithful.

Lastly, we mention a condition that primarily applies to PIQUE window functions, although it can be extended to other window functions through implicit objects. The motivation behind this condition is that one should be able to deduce the meaning of the window on any subset of W knowing the semantics of only those associations contained in W. An object W is *integral* relative to a window generator, if for any subset X of W, $[X]$

can be computed from $\{r(R) \mid R \in \mathbf{A}, R \subseteq W\}$. By a theorem of Maier, Rozenshtein and Warren (MRW), all objects in $\mathbf{O}$ are *integral* if and only if $\mathbf{O}$ is closed under nonempty intersection.

Example 3.2: As an example of where integrity of objects fails, let U = {*PAINTING*, *OWNER*, *ARTIST*, *TELEPHONE*}, $\mathbf{A}$ = {*PO*, *PA*, *OT*, *AT*} and $\mathbf{O} = \mathbf{A} \cup$ {*POT*, *PAT*}. We are storing information on owners and artists of paintings, telephones of owners and artists, and making connections through owners and artists. In the object *POT*, the connection from painting to telephone is via owner. However, the object *PAT* can also add tuples to $[PT]$, so $[PT]$ cannot be computed from only the relations over those associations contained in *POT*. The danger here is that if a user knows that the database has information about paintings, owners and telephones, but does not know about artists, his assumption as to the meaning of $[PT]$ will be incorrect, since, in this case, the window $[PT]$ is really the combination of two different connections. ■

A window generator where the set of implicit objects is closed under nonempty intersection has useful computational properties. To compute the window for a set of attributes X, the projection of a single window over an implicit object onto X is all that is needed. The implicit object will be the smallest one containing X. Since the set of implicit objects is closed under nonempty intersection, this implicit object is unique, if it exists at all. In the case where there is no such implicit object, $[X]$ is the empty relation on X. If the set of implicit objects is known or can be computed, then a list, topologically sorted on the implicit objects, of these implicit objects and the expressions used in computing their windows can be stored. Computing the window on any set of attributes X can then be accomplished by scanning the list until a containing implicit object is found. The corresponding expression is then evaluated and projected onto X. If no containing implicit object is found, then the empty relation on X is returned.

4. Renaming and Loss of Information

As we have noted, in databases that do not satisfy the URA relation names are often used to disambiguate the role an attribute plays. In designing databases that satisfy the URA it is often necessary to rename attributes. The relationship between attributes that represent the same class of entities in different roles is then lost.

Example 4.1 Consider a relational database with the attributes *FACULTY*, *COURSE*, *STUDENT* and *OFFICE* and relations *teaches*(*FCS*), *advises*(*FS*) and *has*(*FO*). The role of *FACULTY* is advisor in *advises*, and teacher in *teaches*. Without the relation names the role of *FACULTY* would not be clear in either relation.

Consider a corresponding PIQUE database where the set of associations is {*FO*, *FS*, *FCS*}. There is no way to distinguish the role of *FACULTY* in the association *FS* from its role in the association *FCS*. On one hand, we know the roles to be different. On the other hand, the URA requires that the role of faculty be the same in both of these associations. A possible solution is renaming *FACULTY* to *ADVISOR* in the association *FS* or to *TEACHER* in the association *FCS*. The set of associations corresponding to one such renaming is {*FO*, *AS*, *FCS*}. Now there is no ambiguity concerning the roles of *ADVISOR* and *FACULTY*. However, the semantic relationship between teaching faculty and advisors, namely, that they are both faculty members, is lost. In particular, there is no indication in the scheme as how to find the offices of advisors. ■

Example 4.2: Consider a relational database where the relation schemes are *manages*(*EMPLOYEE DEPARTMENT*) and *works*(*EMPLOYEE DEPARTMENT*). The role of *EMPLOYEE* is determined by the relation in which it appears. In *manages* attribute *EMPLOYEE* has the role of manager. In *works* it has the role of employee. In a corresponding universal relation scheme database, the attribute *EMPLOYEE* will need to be renamed in order to distinguish its two different roles. The PIQUE scheme corresponding to the scheme of the relational database after renaming is depicted in Figure 4.1. Note that in figures of PIQUE schemes an association is indicated by enclosing its set of attributes within a solid line, and an object that is not an association is indicated by enclosing its set of attributes within a dashed line. ■

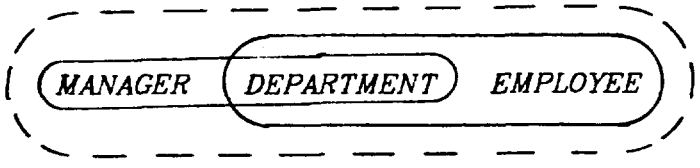

Figure 4.1

The renamings of examples 4.1 and 4.2 are obviously not unique. In Example 4.1 attribute *STUDENT* could have been renamed to *PUPIL* and *ADVISEE* in order to satisfy the URA. In Example 4.2 attribute *DEPARTMENT* could have been renamed to *MANAGED_DEPARTMENT* and *ASSIGNED_DEPARTMENT*. We believe the choices made in the examples are the more intuitive ones. For a discussion of renaming strategies see Sciore (Sc2).

It is sometimes possible to lose track of a renamed attribute altogether. In Example 4.1 attribute *FACULTY* was renamed in both the association where its role was that of teacher, and in the association where its role was that of advisor. In the example *FACULTY* was present in the PIQUE scheme after renaming. If the relation *has* had not been present in the original scheme, then the attribute that both *TEACHER* and *ADVISOR* were renamings of, *FACULTY*, would not have been in any of the PIQUE associations. In general, this is a rare situation; a pair of attributes that are introduced as renamings of a single attribute will probably have common properties. These common properties should be in an object containing the attribute that they are both renamings of. In any case, we assume that renaming is done in such a way that no attributes are lost, even if this requires associations whose only attribute is the one that has been renamed. If the relation *has* had not been present in the database of Example 4.2, we would require an association whose sole attribute is *FACULTY* to be included in the corresponding PIQUE scheme.

5. Representing Role Relationships

In this section we present a structure to store role relationships. In the next section we extend the PIQUE window function to incorporate this structure. The structure presented differs from that used by Beeri and Korth (BK) in that it is not restricted to a single tree. First, we must concern ourselves with defining what it means for two attributes to be role related.

Let A and B be distinct attributes. The *inclusion dependency* (CFP) $B \subseteq A$ is satisfied by a database if every tuple t in $[B]$ is also a tuple in $[A]$.

Consider a scheme containing the attributes *PERSON* and *EMPLOYEE*. Since every employee is a person, we assume that in any database on this scheme $[EMPLOYEE]$ is contained in $[PERSON]$; we assume that the inclusion dependency $EMPLOYEE \subseteq PERSON$

holds. Since every person need not be an employee, we do not assume that the inclusion dependency $PERSON \subseteq EMPLOYEE$ holds. Let A and B be two distinct attributes. If the inclusion dependency $B \subseteq A$ is known to always hold, then from a semantic point of view B is a *role* of A. Of course, this assumes some reasonable structure in the underlying domains of A and B. For example, consider a database scheme that contains attributes *HEIGHT* and *WEIGHT*. If height is measured in centimeters and weight is measured in pounds, it may be the case that the set of values used to store heights is contained in the set of values used to store weights. However, the domains are clearly distinct; comparing centimeters to pounds is like comparing strawberries to mangos.

We will denote role relationships by $A \twoheadrightarrow B$, where B is a role of A. The relation induced by $\twoheadrightarrow$ is transitive. In the next example, $PERSON \twoheadrightarrow FACULTY$ and $FACULTY \twoheadrightarrow ADVISOR$; therefore, $PERSON \twoheadrightarrow ADVISOR$.

Example 5.1: Consider the PIQUE scheme of Figure 5.1. In it the attributes are *PERSON*, *STUDENT*, *GPA*, *COURSE*, *FACULTY*, *OFFICE*, *TEACHER*, *ADVISOR*. The set of associations **A** is $\{PB, SG, SC, FO, TC, AS, S, C\}$. The facts represented by this scheme are: people have birthdates; students have grade-point averages and take courses; faculty members have offices; teachers teach courses; advisors advise students. The specified role relationships for this scheme are: $P \twoheadrightarrow S$; $P \twoheadrightarrow F$; $F \twoheadrightarrow T$; $F \twoheadrightarrow A$. ■

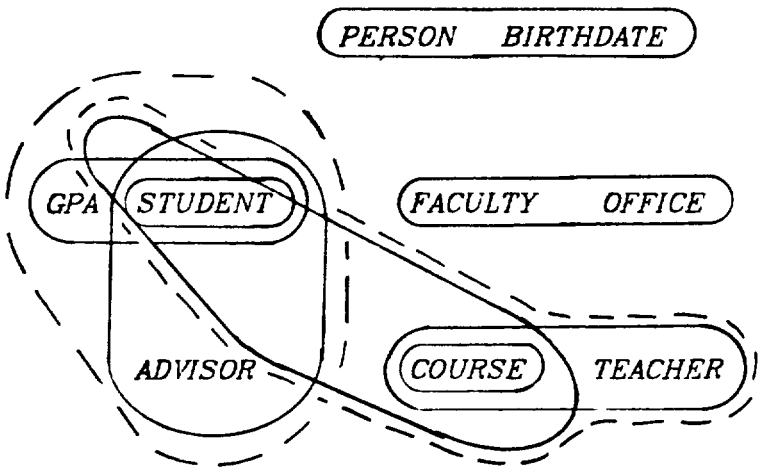

Figure 5.1

Role relationships for a database scheme are used to construct a directed graph, where the set of vertices is U and there is an edge from A to B if $A \twoheadrightarrow B$. Such a graph

is called a *role graph* or *R-graph* and its edges are called *R-edges*. Since the relationship induced by $\twoheadrightarrow_R$ is transitive, R-graphs are transitively closed. Intuitively, it can not be the case that A is a role of B, and B is a role of A. Therefore, we require that R-graphs be acyclic. A set of role relationships is *acyclic* if the corresponding R-graph is acyclic.

Example 5.2: The R-graph corresponding to the set of role relationships in Example 5.1 is depicted in Figure 5.2. ■

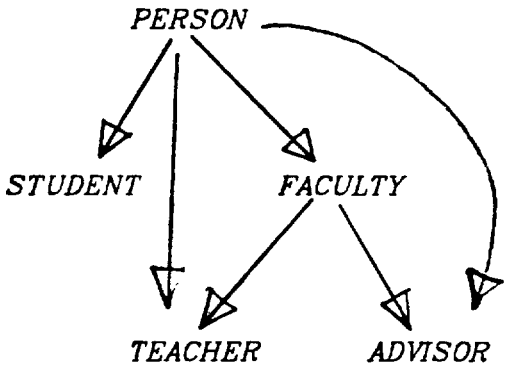

Figure 5.2

Example 5.3: Consider the PIQUE scheme of Figure 5.3. In it the attributes are *PERSON*, *ADDRESS*, *DEPARTMENT*, *EMPLOYEE*, *ITEM*, *REPAIR_DEPARTMENT*, *TESTING_DEPARTMENT*. The set of associations **A** is $\{PA, DE, RI, TI\}$. The facts represented by this scheme are: people have addresses; employees work in departments; repair departments repair items; testing departments test items. We assume that some departments only test items; some departments only repair items; some departments do both. The role relationships specified for the scheme are: $P \twoheadrightarrow_R E$; $D \twoheadrightarrow_R R$; $D \twoheadrightarrow_R T$. The corresponding R-graph has been drawn onto the scheme itself. Note that the R-graph has two connected components. ■

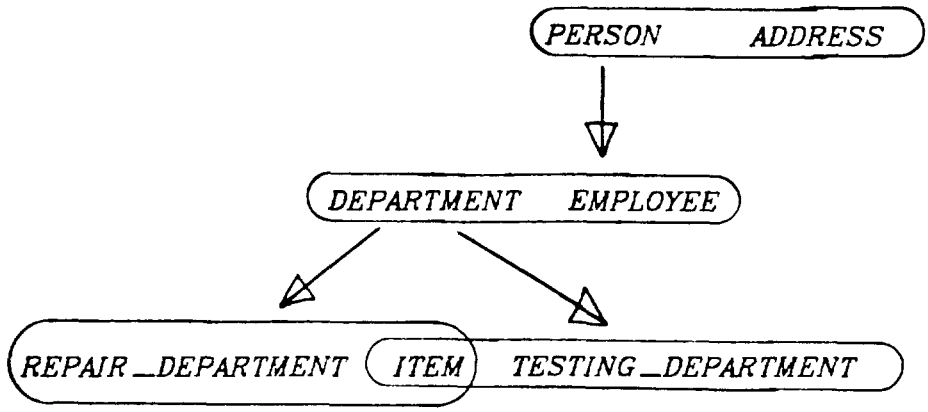

Figure 5.3

Example 5.4: Consider the PIQUE scheme and role relationships of Figure 5.4. The attributes are *GRADUATE_STUDENT*, *YEAR*, *INSTRUCTOR*, *WAGES*, *TEACHING_ASSISTANT*, *SUPERVISOR*. The set of associations **A** is $\{GY, IW, TS\}$. The facts represented by this scheme are: graduate students enter school in a particular year; instructors receive hourly wages; teaching assistants have supervisers. We assume that every teaching assistant is both a graduate student and an instructor. Note that this does not imply that every instructor or every graduate student is a teaching assistant. The role relationships specified for the scheme are: $I \twoheadrightarrow T$; $S \twoheadrightarrow T$. ■

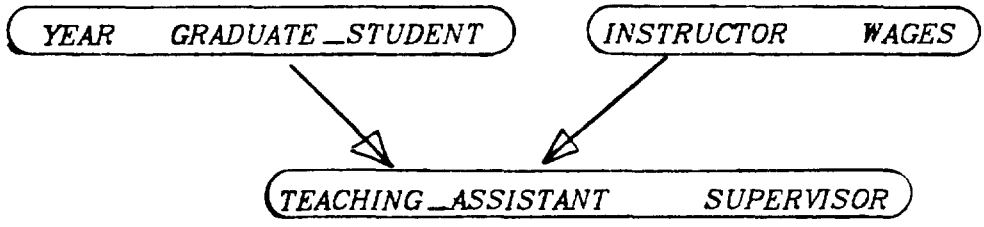

Figure 5.4

6. Incorporating Role Information into PIQUE

Before incorporating role hierarchies into PIQUE databases we shall first define the notion of *property* as it relates to pairs of attributes in U. We shall then discuss establishing connections between (computing relations over) pairs of attributes where the pair

is contained in no object of the database, yet the role relationships indicate a meaningful connection between them. We then expand the discussion to include establishing connections among larger sets of attributes. Finally, we shall extend the PIQUE window function to incorporate role relationships.

Let A and B be elements of U. If there is an object in $\mathbf{O}$ that contains both A and B, then A is an *immediate property* of B *with respect to* $\mathbf{O}$, and B is an *immediate property* of A *with respect to* $\mathbf{O}$. By immediate, we mean that a connection can be established without the use of any role information. Note that the definition of an immediate property is not dependent upon the role relationships that apply to a scheme.

Let Γ be the transitive closure of a set of role relationships specified for a PIQUE scheme $(\mathbf{A}, \mathbf{O})$. If there exists a sequence of ordered pairs of attributes (C_1, D_1), (C_2, D_2), ..., (C_n, D_n), where C_i and D_i need not be distinct, such that:

1. C_i is an immediate property of D_i with respect to $\mathbf{O}$, $1 \leq i \leq n$;
2. C_{i+1} is a role of D_i, $1 \leq i < n$;
3. $C_1 = A$ and $D_n \twoheadrightarrow_R B$,

then A is an *inherited property* of B *with respect* to $\mathbf{O}$. By inherited, we mean that a connection can be established using role information.

Example 6.1: Consider the PIQUE scheme of Figure 5.1, where the transitive closure of the specified set of role relationships is presented in Figure 5.2. Attribute *BIRTHDATE* is an inherited property of *FACULTY* with respect to $\mathbf{O}$, where the sequence of ordered pairs of attributes (*BIRTHDATE*, *PERSON*) satisfies the above definition. Consider the PIQUE scheme and role relationships of Figure 5.4. Attribute *ADDRESS* is an inherited property of *TESTING_DEPARTMENT* with respect to $\mathbf{O}$, where the sequence of ordered pairs of attributes (*ADDRESS*, *PERSON*), (*EMPLOYEE*, *DEPARTMENT*) satisfies the above definition. ■

Consider the PIQUE scheme of Figure 5.1. Let Γ be the set of role relationships in the R-graph of Figure 5.2. Attribute *OFFICE* is an inherited property of *TEACHER*. A connection between *OFFICE* and *TEACHER* can be computed by taking the equijoin $[OF] \overset{F=T}{\bowtie} [T]$ (which is the cross product of $[OF]$ and $[T]$ followed by the selection $F=T$) and projecting onto OT.

The next example shows that an attribute may be both an immediate and an inherited property of another attribute with respect to **O**. When an attribute B is both an immediate and an inherited property of an attribute A, preference is given to A as an immediate property when computing the connection between A and B.

Example 6.2: Consider the PIQUE scheme and role relationships of Figure 6.1. The scheme represents the following facts: employees work in departments; employees have salaries; managers manage departments. In this scheme *DEPARTMENT* is both an immediate and an inherited property of *MANAGER* with respect to **O**. Semantically, as an immediate property of *MANAGER*, *DEPARTMENT* represents a department that the manager manages. As an inherited property of *MANAGER*, *DEPARTMENT* represents a department the manager works in. The preferred connection between *DEPARTMENT* and *MANAGER* is $[DM]$, not $\pi_{DM}([ED] \overset{E=M}{\bowtie} [M])$. ■

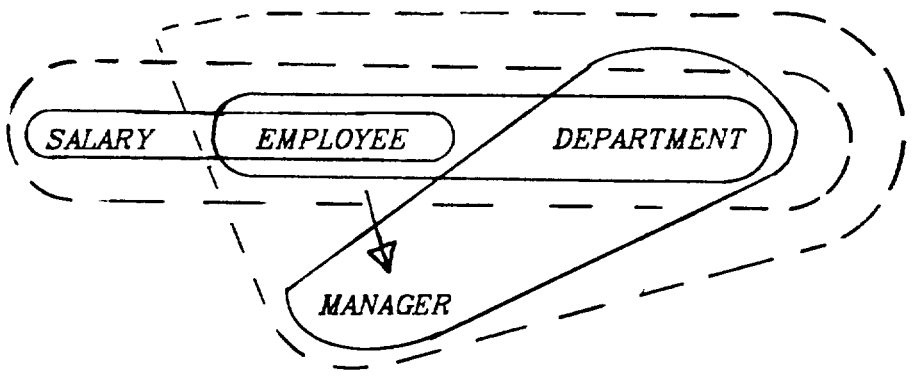

Figure 6.1

Consider computing the window for a set of attributes X in the presence of role information. If there exists an object containing X, then every attribute in X is an immediate property of every other attribute in X. Since a preference is given to immediate properties, $[X]$ is returned. If there is no object that contains X, then it may still be possible to construct a meaningful relation on X using role information. The role information is used to generate equijoins between objects to connect the attributes in X.

Example 6.3: Let **(A, O)** be as depicted in Figure 5.1, with the transitive closure of the specified set of role relationships presented in Figure 5.2. Consider the window on *ADVISOR OFFICE STUDENT*. There is no object containing the set of attributes named in the window. However, *OFFICE* is an inherited property of *ADVISOR*, and there is an

object that contains *STUDENT ADVISOR*. Recall, that the relation on an object W is denoted as $r'(W)$. By taking the equijoin $r'(OF) \overset{F=A}{\bowtie} r'(AS)$ and projecting onto *AOS* a semantically meaningful relation can be generated. A tuple $<a, o, s>$ in this relation means that advisor a, who has office o, advises student s.

■

Let Γ be the transitive closure of a set of role relationships specified for a scheme $(\mathbf{A}, \mathbf{O})$. Let W_1 and W_2 be objects in $\mathbf{O}$. Let $A \in W_1$ and $B \in W_2$ be two attributes such that $A \twoheadrightarrow B \in \Gamma$. The equijoin $r'(W_1) \overset{A=B}{\bowtie} (W_2)$ is a *role join* or *R-join*. A sequence of joins $r'(W_1) \overset{A_1=B_2}{\bowtie} r'(W_2) \overset{A_2=B_3}{\bowtie} \cdots \overset{A_{n-1}=B_n}{\bowtie} r'(W_n)$ is an R-join *sequence* if $r'(W_i) \overset{A_i=B_{i+1}}{\bowtie} (W_{i+1})$ is an R-join for $1 \leq i < n$.

As defined, not all R-join sequences are well-formed, in that objects participating in the join sequence may have common attributes. In addition, ill-formed R-join sequences do not have semantically consistent interpretations. In an ill-formed R-join sequence there will be at least one attribute, call it C, where C is present as both an immediate and an inherited property of an attribute D. An attribute C is an *immediate property* of D *with respect to* an R-join sequence $r'(W_1) \overset{A_1=B_2}{\bowtie} r'(W_2) \overset{A_2=B_3}{\bowtie} \cdots \overset{A_{n-1}=B_n}{\bowtie} r'(W_n)$ if for some i, $C, D \in W_i$. An attribute C is an *inherited property* of D *with respect to* an R-join sequence $r'(W_1) \overset{A_1=B_2}{\bowtie} r'(W_2) \overset{A_2=B_3}{\bowtie} \cdots \overset{A_{n-1}=B_n}{\bowtie} r'(W_n)$, if for some $1 \leq i < j \leq n$, $C \in W_i$ and $D = B_j$.

Example 6.4: Consider a database on the PIQUE scheme of Figure 6.1. The attribute *DEPARTMENT* is both a direct and inherited property of *MANAGER* with respect to the ill-formed R-join sequence $r'(SDE) \overset{E=M}{\bowtie} r'(MD)$. The meaning of *DEPARTMENT* is ambiguous with respect to this R-join sequence. As a direct property of *MANAGER* with respect to the above R-join sequence, *DEPARTMENT* represents the department a manager manages. As an inherited property of *MANAGER*, *DEPARTMENT* represents a department a manager works in as an employee. Therefore, unless managers are required to be employees in exactly those departments they manage, the meaning of *DEPARTMENT* is ambiguous in an evaluation of this R-join sequence.

■

Observation: Let $E = \tau'(W_1) \overset{A_1=B_2}{\bowtie} \tau'(W_2) \overset{A_2=B_3}{\bowtie} \cdots \overset{A_{n-1}=B_n}{\bowtie} \tau'(W_n)$ be an R-join sequence for $(\mathbf{A}, \mathbf{O})$. The expression E is well-formed if and only if there is no pair of attributes C and D, such that C is both an immediate and an inherited property of D with respect to E. ■

Henceforth, we shall assume R-join sequences that are well-formed. We now present some definitions necessary for further discussion. Let $E = \tau'(W_1) \overset{A_1=B_2}{\bowtie} \tau'(W_2) \overset{A_2=B_3}{\bowtie} \cdots \overset{A_{n-1}=B_n}{\bowtie} \tau'(W_n)$ be an R-join sequence. The set of attributes $\{C \mid C \in W_i, 1 \le i \le n\}$, denoted by $atts(E)$, is called a *role object*, or *R-object with respect to* E. A sequence of R-edges $A_1 \xrightarrow{R} B_2$, $A_2 \xrightarrow{R} B_3$, ..., $A_{n-1} \xrightarrow{R} B_n$ is called *productive* if there exists some R-join sequence $E = \tau'(W_1) \overset{A_1=B_2}{\bowtie} \tau'(W_2) \overset{A_2=B_3}{\bowtie} \cdots \overset{A_{n-1}=B_n}{\bowtie} \tau'(W_n)$. Let Γ be the set of R-edges appearing in E. Then E is an *R-join sequence based on* Γ.

Unfortunately, even a well-formed R-join sequence may produce semantically ambiguous results as the next example illustrates.

Example 6.5: Consider a database on the scheme of Figure 6.1 with the added association *MANAGER*. The R-join sequence $\tau'(SDE) \overset{E=M}{\bowtie} \tau'(M)$ is well-formed. In particular, *DEPARTMENT* is only an inherited property of *MANAGER* with respect to this R-join sequence. However, *DEPARTMENT* is an immediate property of *MANAGER* with respect to **O**. If one looks at the tuple $<s, d, e, m>$ from the evaluation of this R-join, without knowing how the relation was computed, it is not apparent whether d represents a department that employee m works in or a department that manager m manages. ■

In keeping with our preference to present an attribute A as an immediate property of an attribute B, we would like to allow an R-join sequence where A is an inherited property of B with respect to the R-join, only if A is not an immediate property of B with respect to **O**. However, in the presence of a global object over U, this would disallow all R-joins sequences, for the global object over U implies that for any A and B, A is an immediate property of B with respect to **O**. Even if there is no global object, this constraint will disallow an R-join sequence E if there is an object W, such that $atts(E)$ is contained in W.

Consider a productive sequence of R-edges $A_1 \xrightarrow{R} B_2$, $A_2 \xrightarrow{R} B_3$, ..., $A_{n-1} \xrightarrow{R} B_n$. Note that any prefix of this sequence is also productive. Consider an R-join sequence $\tau'(W_1) \overset{A_1=B_2}{\bowtie} \tau'(W_2)$. We call such an R-join sequence *allowable* if there is no object V, such that $B_2 \in V$, $W_1 \cap V \neq \phi$, and $A_1 \notin V$. In terms of properties, an R-join sequence $\tau'(W_1) \overset{A_1=B_2}{\bowtie} \tau'(W_2)$ is allowable if no attribute C in W_1 is an immediate property of B_1 with respect to $\mathbf{O} - \{W \mid W \in \mathbf{O} \text{ and } A_1, B_2 \in W\}$.

We now extend the definition of allowable to all R-join sequences. An R-join sequence $\tau'(W_1) \overset{A_1=B_2}{\bowtie} \tau'(W_2) \overset{A_2=B_3}{\bowtie} \cdots \overset{A_{n-1}=B_n}{\bowtie} \tau'(W_n)$ is *allowable* if for all $1 < i \leq n$ there is no object V such that:

1. B_i is an element of V;
2. $\bigcup_{j=1}^{j=i-1} W_j \cap V \neq \phi$;
3. A_{i-1} is not an element of V.

Henceforth, we shall consider only allowable R-join sequences. We now present some theorems that describe the behavior of R-objects. Proofs of these theorems are provided by Rozenshtein (Ro).

Theorem 6.1: Let $E = \tau'(W_1) \overset{A_1=B_2}{\bowtie} \tau'(W_2) \overset{A_2=B_3}{\bowtie} \cdots \overset{A_{n-1}=B_n}{\bowtie} \tau'(W_n)$ be an R-join sequence. There is no other R-join sequence $E' = \tau'(V_1) \overset{A_1=B_2}{\bowtie} \tau'(V_2) \overset{A_2=B_3}{\bowtie} \cdots \overset{A_{n-1}=B_n}{\bowtie} \tau'(V_n)$, such that $atts(E') = atts(E)$. ■

Theorem 6.2: Let Γ be a set of R-edges. If there exists a productive permutation of the elements of Γ, then it is unique. ■

It follows from theorems 6.1 and 6.2 that if E is an R-join sequence based on a set of R-edges Γ such that $atts(E) = P$, then there is no other R-join sequence E' based on Γ, such that, $atts(E') = P$. Recall, that in Section 3 we defined the notion of an object being integral relative to a window generator. Recall also, that in PIQUE schemes all of the objects in **O** are integral if and only if **O** is closed under nonempty intersection.

Theorem 6.3: Let **S** be the set of R-objects with respect to some set of R-edges Γ. If the set of objects is closed under nonempty intersection, then **S** is closed under nonempty intersection.

■

In PIQUE window generators always satisfy the containment condition. The relations corresponding to R-objects, that is the evaluations of R-joins, satisfy a restricted version of the containment condition.

Theorem 6.4: Let Γ be a set of R-edges. Let P and P' be distinct R-objects with respect to Γ. Let E and E' be the R-joins based on Γ, such that $\mathit{atts}(E) = P$ and $\mathit{atts}(E') = P'$. If P is contained in P', then the evaluation of E' projected onto P is contained in the evaluation of E.

■

We now formally extend the PIQUE window function to incorporate role information. We will treat objects as R-join sequences on the productive sequence of R-edges with length zero. Let d be a PIQUE database on **(A, O)**. Let Γ be the transitive closure of the set of role relationships specified for **(A, O)**. Let $RJ(\gamma)$, where $\gamma \subseteq \Gamma$, denote the set of R-join sequences based on γ.

Window functions are extended so that for each database state, set of attributes $X \subseteq U$ and set of role relationships $\varphi \subseteq \Gamma$ a relation over X is associated with X and φ. We shall denote the extended window function as $[X, \varphi]$. For the sake of clarity, we shall define the extended window function procedurally. Rozenshtein provides an equivalent non-procedural definition (Ro).

Algorithm 6.1

1. Let S_1 be the set of all $RJ(\gamma)$. Delete from S_1 those $RJ(\gamma)$ where γ does not contain φ. Call the remaining set of $RJ(\gamma)$ S_2.
2. Delete from S_2 those $RJ(\gamma)$ where there is no R-join sequence E in $RJ(\gamma)$, such that $\mathit{atts}(E)$ contains X. Call the remaining set of $RJ(\gamma)$ S_3.
3. Consider every pair $RJ(\gamma_i)$ and $RJ(\gamma_j)$ in S_3. If γ_i contains γ_j delete $RJ(\gamma_i)$ from S_3. This step is repeated until no further changes occur to S_3. Call the remaining set of $RJ(\gamma)$ S_4.
4. Consider every pair $RJ(\gamma_i)$ and $RJ(\gamma_j)$ in S_4. Let α_i and α_j be the productive

permutations corresponding to γ_i and γ_j respectively. (Theorem 6.2 guarantees their uniqueness.) If the R-edges in $\gamma_i - \gamma_j$ form a subsequence β_i of α_i, $A \twoheadrightarrow B, \ldots, C \twoheadrightarrow D$, and the R-edges in $\gamma_j - \gamma_i$ form a subsequence β_j of α_j, $A \twoheadrightarrow F, \ldots, G \twoheadrightarrow D$, and β_j is shorter than β_i delete $RJ(\gamma_i)$ from S_4. This step is repeated until no further changes occur to S_4. Call the remaining set of $RJ(\gamma)$ S_5.

5. If S_5 contains more than one element, return the empty relation on X. Else let **RJ** be the remaining set of R-join sequences in S_5. Return the union of the projections onto X of the evaluations of those R-join sequences E in **RJ**, where $atts(E)$ contain X.

■

Example 6.6: Consider a database scheme depicted in Figure 5.1 with the set of role relationships depicted in Figure 5.2. Suppose that the local draft board requests a list of students and their birthdates. Assuming that the university is willing to comply, this question would be posed in PIQUE by the query

retrieve *STUDENT*, *BIRTHDATE*.

The needed connection between STUDENT and BIRTHDATE will be computed automatically as $\pi_{SB}(\tau'(BP) \overset{P=S}{\bowtie} \tau'(S))$.

■

Incorporating role hierarchies into other universal scheme interfaces is straight forward. The universal scheme interface must have window generators that satisfy the containment condition. If a universal scheme interface has window generators that satisfy the containment condition, then implicit objects are defined for the interface. The results of this section hold when implicit objects are considered in place of PIQUE objects.

References

(AK) A.V. Aho, B.W. Kernighan. research!user/ava/q/README, 1980.

(Ba) E. Babb. Joined normal form: a storage encoding for relational databases, ACM TODS 7(4), December 1982, 588-614.

(CK) C.R. Carlson, R.S. Kaplan. A generalized access path model and its application to a relational database system, 1976 ACM SIGMOD Conf., June 1976, 143-154.

(Ko) System/U: a progress report, XP2 Workshop on Relational Database Theory, June 1981.

(KMRS) S.M. Kuck, D.A. McNabb, S.V. Rice, Y. Sagiv. The Parafrase database user's manual, Computer Science Technical Report 80-1046, Univ. of Illinois, December 1980.

(KS) S.M. Kuck, Y. Sagiv. A universal relation database system implemented via the network model, ACM Symp. on Principles of Database Systems, March 1982, 147-157.

(KU) H.F. Korth, J.D. Ullman. System/U: A database system based on the universal relation assumption, XP1 Workshop on Relational Database Theory, June-July 1980.

(MRSSW) D. Maier, D. Rozenshtein, S.C. Salveter, J.Stein, D.S. Warren. Toward logical data independence: A relational query language without relations, 1982 ACM SIGMOD Conf., June 1982, 51-60.

(MRW) D. Maier, D. Rozenshtein, D.S. Warren. Windows on the world, to appear 1982 ACM SIGMOD Conf., May 1983.

(MW) D. Maier, D.S. Warren. Specifying connections for a universal relation scheme database, 1982 ACM SIGMOD Conf., June 1982, 1-7.

(Ro) D. Rozenshtein. Query and role playing in the association-object data model, Doctoral dissertation in preparation, SUNY at Stony Brook, 1983.

(Sc1) E. Sciore. The universal instance and database design, Doctoral dissertation, Princeton Univ., October 1980.

(Sc2) Improving semantic specification in a relational database, 1979 ACM SIGMOD Conf., May-June 1979, 170-178.

(SP) K.L. Schenk, J.R. Pinkert. An algorithm for servicing multi-relational queries, 1977 ACM SIGMOD Conf., August 1977, 10-20.

(St) J. Stein, Data definition and update in the association-object data model, Doctoral dissertation in preparation, SUNY at Stony Brook, 1983.

(Ul) J.D. Ullman. The U.R. strikes back. ACM Symp. on Principles of Database Systems, March 1982, 10-22.