Title

Name Assignment Techniques for Relational Schemas Representing Extended Entity-Relationship Schemas

Permalink

https://escholarship.org/uc/item/8520j745

Authors

Markowitz, V M
Shoshani, A

Publication Date

1989-08-01

LBL-27070

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

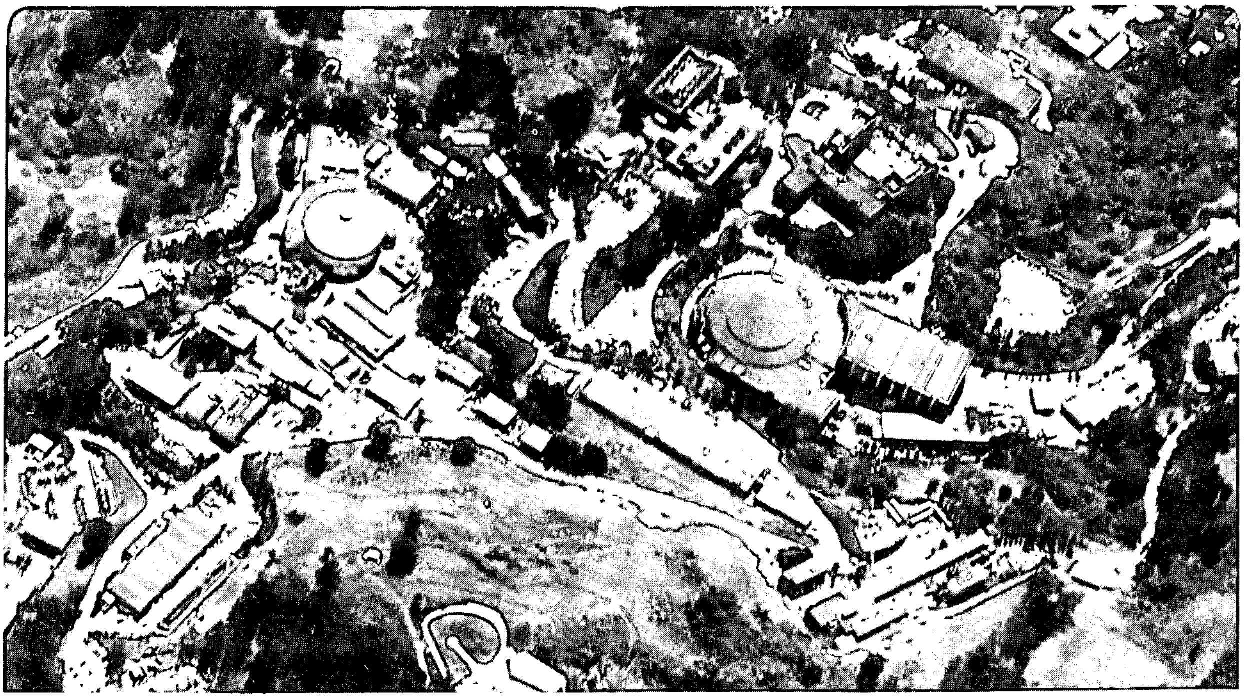
Information and Computing Sciences Division

To be presented at the Eighth International Conference on Entity-Relationship Approach, Toronto, Canada, October 18–20, 1989, and to be published in the Proceedings

Name Assignment Techniques for Relational Schemas Representing Extended Entity-Relationship Schemas

V.M. Markowitz and A. Shoshani

August 1989



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

DISCLAIMER

LBL-27070

NAME ASSIGNMENT TECHNIQUES FOR RELATIONAL SCHEMAS REPRESENTING EXTENDED ENTITY-RELATIONSHIP SCHEMAS

Victor M. Markowitz and Arie Shoshani

Computing Science Research & Development
Information & Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, California 94720

August 1989

To appear in Proceedings of the 8th International Conference on Entity-Relationship Approach, Toronto, Canada, October 18-20, 1989.

This work was supported by the Office of Health and Environmental Research Program and the Director, Office of Energy Research, Applied Mathematical Sciences Research Program, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

NAME ASSIGNMENT TECHNIQUES FOR RELATIONAL SCHEMAS REPRESENTING EXTENDED ENTITY–RELATIONSHIP SCHEMAS *

Victor M. Markowitz and Arie Shoshani

Computer Science Research Department
Information and Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road, Berkeley, CA 94720

The mapping of *Extended Entity–Relationship* (*EER*) schemas into relational schemas involves the assignment of names to relational attributes. We propose several criteria for such name assignments, such as brevity and clarity. These name assignments must satisfy certain assumptions underlying relational normalization. We develop two name assignment algorithms that satisfy these assumptions and follow the criteria mentioned above. The first algorithm is intended for regular relational interfaces where users know both relations and attributes. The second algorithm is designed for special relational (Universal Relation) interfaces where users interact at the attribute level only and are not expected to know how attributes are grouped into relations. Both algorithms employ well known graph algorithms in order to devise appropriate name assignment strategies.

1. INTRODUCTION

Designing a relational database schema can be a difficult and confusing task for users not familiar with relational concepts. The difficulties are even greater when the goal is to produce normalized schemas. Normalization methodologies assume that all the semantics are captured by dependencies expressed over a universal set of attributes, irrespective of how the attributes might be grouped into relations. Accordingly, the assignment of names to relational attributes is excessively complex and constitutes the critical part of these methodologies. Furthermore, it has been shown that inter-relation constraints (e.g. referential integrity) must be specified in the relational model in order to insure proper integrity of the database and such constraints are disregarded by normalization. As a result of these difficulties, the *Entity–Relationship* (ER) model [2]

---

* This work was supported by the Office of Health and Environmental Research Program and the Applied Mathematical Sciences Research Program, of the Office of Energy Research, U.S. Department of Energy, under Contract DE-AC03-76SF00098.

and various versions of extended ER (EER) models (e.g. [9]) have been proposed as a tool for relational schema design. Typically, some schema design tool (e.g. graphical) helps the user to specify the ER or EER schema, and then that schema is mapped into a normalized relational schema. We have shown in a previous paper [7] that, while there were many proposals for such mappings, they are imprecise and not compatible with relational normalization. Consequently, we have defined in [7] the *correctness* criteria for such mappings. Informally, a correct mapping should preserve the structural semantics of the ER or EER schema. For example, given a relationship-set between several entity-sets of an ER schema, the corresponding relational representation for that relationship-set must reflect the existence dependency of the relationship-set on the involved entity-sets. Similarly, the semantics of EER structures, such as generalization and aggregation, must be captured correctly by the corresponding relational schema.

Another aspect of mapping ER or EER schemas into relational schemas involves the assignment of names to relational attributes. Note that the issue of assigning names to relational attribute has not been addressed by the EER-oriented design methodologies (e.g. [9]). Typically, an ad-hoc assignment of names is used which often leads to incorrect conclusions, because normalization depends on the names assigned to relational attributes. For example, [4] reached the conclusion that *Boyce-Codd Normal Form* (BCNF) is achievable for relational schemas that represent ER schemas, only if the corresponding ER diagrams have no cycles. Thus, in order to ensure BCNF, structures such as the simple example shown in figure 1(i) would be disallowed. This conclusion was reached because, as shown in [7], the attribute name assignment in the mapping of ER schemas into relational schemas of [4], did not take into account the assumptions underlying normalization. In [7] we have shown that ER schema restrictions such as those of [4] are unnecessary, and that any unrestricted EER schema can be mapped into a BCNF schema, provided a proper name assignment is used.

Our approach to the mapping of EER schemas into relational schemas is to treat separately the different aspects of the mapping. Specifically, we have identified in this mapping the following three parts: (i) the canonical mapping of EER schemas into relational schemas, (ii) the normalization of relational schemas representing EER schemas, and (iii) the assignment of names to relational attributes. In [7] we have specified a provably correct canonical mapping that is independent of normalization and attribute name assignment. We have shown that an EER schema can be represented correctly by a relational schema of the form $(R, I \cup F)$, where R is a set of relation-schemes, I is a set of inclusion dependencies, and F is a set of functional dependencies. Further, we have specified a normalization mapping for such schemas into BCNF schemas that, while taking into account the assumptions underlying normalization, avoids being dependent on a specific name assignment for relational attributes. In order to keep the canonical and normalization mappings independent of a specific attribute name assignment, we employed (internal) symbolic names for the representation of relational attributes (e.g. A_{3_2} represents the 2nd attribute of the 3rd relation).

This paper is concerned with the third aspect mentioned above, that is, assigning names to for relational attributes. Given that the mappings mentioned above are followed, we can generate from an EER schema a (BCNF) relational schema that has symbolic names for representing attributes. At this point we have several options:

(i) Interface at the EER level. If we were only interested to access the database through an EER level interface, then we can stop here and use the symbolic names for a relational implementation of the database.

(ii) Interface at the relational level. Quite often EER schemas are used only as a design tool. Once the corresponding relational schema is generated, the user as well as the application programs access the database through a relational interface (e.g. SQL). In that case, meaningful, rather than symbolic, names for relational attributes are appropriate.

(iii) Interface using attribute names only. There is a school of thought that claims that the user interface should deal only with attribute names. In this case the relations are invisible to the user and the attribute names must have a global independent meaning. This is one of the goals of the *Universal Relation* (UR) interfaces [6]. In that case, a name assignment that generates as few as possible global names is needed.

We present in this paper two attribute name assignment algorithms: the first one is an algorithm for relational interfaces, and the second one is a more complex algorithm for UR interfaces. Note that UR interfaces have to rely on globally meaningful attribute names that convey the semantics of the database. It is well known that the difficulties in selecting these global names is one of the serious drawbacks of the UR approach [1]. We see the second name assignment algorithm proposed in this paper as providing the solution to this problem. Both name assignment algorithms must comply with the normalization assumptions mentioned above, because a careless name assignment can still lead to an incorrect design.

In general, the name assignment depends on the goals of the schema design. For example, one may wish to assign relational attributes names that are as brief as possible, even at the cost of some loss of meaning. Our choice is more of a compromise. We discuss in section 2 the principles that we chose to follow and give an informal description of the two name assignment algorithms. In section 3 we review the EER model, the canonical relational representation for EER structures, and the assumptions underlying normalization and UR interfaces. A name assignment algorithm for regular relational interfaces is proposed in section 4. In section 5 we present a name assignment algorithm for UR-oriented relational interfaces. We conclude with a brief discussion of the results. The relational and graph-theoretical concepts used in this paper are briefly reviewed in the appendix. Any textbook on databases (e.g. [10]) and on graph theory (e.g. [3]) can provide the necessary background.

2. ATTRIBUTE NAME ASSIGNMENT: CRITERIA AND OUTLINES

2.1. Criteria for Name Assignment to Relational Attributes.

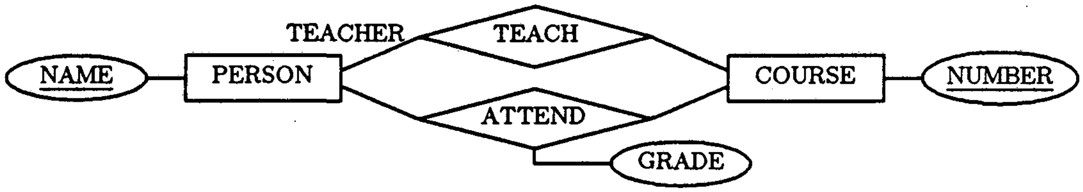
Most of the considerations for attribute name assignment stem from the need to select names for distinct relational attributes that correspond to the same EER attribute. Such attributes are called foreign attributes. More precisely, if R_i is a relation that corresponds to object-set O_i, then a ***foreign attribute*** of R_i is an attribute that corresponds to an EER attribute of an object-set other than O_i. Consider the simple EER diagram of figure 1(i), where entity-sets PERSON and COURSE have identifiers NAME and NUMBER, respectively. The obvious thing to do is to assign the names of these EER attributes to the corresponding (non-foreign) relational attributes in the relations that correspond to these entity-sets. However, these identifiers also have corresponding (foreign) relational attributes in the relations that correspond to relationship-sets TEACH and ATTEND. Using the same attribute name assignment for foreign attributes is problematic. For example, such an assignment can violate the condition that all the attributes in the a relation have distinct names. Thus, if the identifier for PERSON was NUMBER, then, following the name assignment above, two relational attributes would be assigned the same name in the relations that correspond to ATTEND and TEACH. Note that foreign attributes appear in relations corresponding to relationship-sets, weak entity-sets, and entity-sets involved in generalizations, and can propagate through a chain of relations corresponding to EER structures that involve relationship-sets, weak entity-sets, and generalizations. There are some common sense principles that one can follow in choosing a name assignment algorithm for relational attributes. We chose a combination of the principles discussed below.

Retention of EER Names. We assume that much thought was given to the selection of names for the various components of the EER structure. Therefore, we wish the relational attribute names to be as close as possible to these names. Accordingly, we decided to adopt the principle of deriving the relational attribute names from the names of EER attributes, object-sets, and roles. For the sake of simplicity, we chose to concatenate these names (using the customary "." notation, such as COURSE.NUMBER) when necessary. Furthermore, we chose not to make up new names without consulting the user. We also chose to avoid any systematic abbreviation of names (such as AT.GR for ATTEND.GRADE) because this can obscure the meaning.

Clarity. It is important to assign names that carry the semantic clarity intended in the EER design. Thus, as a general criterion, we prefer to assign a name to a foreign attribute that includes the name of the object-set to which the corresponding EER attribute belongs. For example, as can be seen in figure 1(ii), there is a foreign attribute (A_{3_2}) in the relation corresponding to relationship-set ATTEND, that corresponds to EER attribute NUMBER of entity-set COURSE. This foreign attribute can be assigned several reasonable names, such as NUMBER, COURSE.NUMBER, ATTEND.NUMBER, or ATTEND.COURSE.NUMBER. According to the clarity principle mentioned above, we prefer COURSE.NUMBER because in this way the original association in the EER schema is reflected by the relational attribute name. Using the name of the EER attribute alone (e.g. NUMBER) may obscure this association.

Brevity. Long names are difficult to remember and can be confusing. In the example above, the attribute name ATTEND.COURSE.NUMBER is unnecessarily long. Furthermore, if relationship-set ATTEND is further aggregated, then concatenating the names of all the involved object-sets leads to longer and longer names. For example, if relationship-set ATTEND was

(i)



(ii)

Object Set	**ER Attribute**	**Relation Scheme**	**Relation Name**	**Assign**$_L$	**Assign**$_G$
PERSON	NAME	$R_1(A_{1_1})$	PERSON	NAME	PERSON.NAME
COURSE	NUMBER	$R_2(A_{2_1})$	COURSE	NUMBER	COURSE.NUMBER
ATTEND		$R_3(A_{3_1},$	ATTEND	PERSON.NAME	PERSON.NAME
		$A_{3_2},$		COURSE.NUMBER	COURSE.NUMBER
	GRADE	$A_{3_3})$		GRADE	ATTEND.GRADE
TEACH		$R_4(A_{4_1},$	TEACH	PERSON.NAME	TEACHER.NAME
		$A_{4_2})$		COURSE.NUMBER	COURSE.NUMBER

Figure 1. Name Assignments for a Relational Schema Representing an ER Schema.

connected to an entity-set ROOM through a relationship-set ASSIGN, then the foreign attribute corresponding to EER attribute NUMBER in the relation corresponding to ASSIGN will be given the name ASSIGN.ATTEND.COURSE.NUMBER. In general, it is easy to understand names that have two components, one corresponding to the object-set and one to the EER attribute, such as COURSE.NUMBER. Thus, we prefer names of length two, which is a compromise between clarity and brevity. We show in this paper that in most cases it is possible to limit the number of name components to two. However, in some rare structures, more than two components are necessary.

Non–Proliferation Of Attribute Names. As mentioned in the introduction, we examine in this paper the problem of attribute name assignment for two different relational environments: one in which users know both attributes and relation names, and another in which users know only attribute names. In the later case it is desirable to minimize the overall number of relational attributes. Since attributes are uniquely identified by their global names, the name assignment algorithm should minimize the number of global names. As noted in [1], the proliferation of attribute names leads to confusing and unmanageable attribute names. Consequently, the overall number of global names for relational attributes should be kept at a minimum.

The criteria above should be followed without compromising the correctness of the mapping or normalization, and without restricting the functional capability of EER modeling.

2.2. Informal Description of Two Name Assignment Algorithms.

One of the purposes of this paper is to demonstrate that the mapping of EER schemas into relational schemas can be associated with various name assignment algorithms. Our approach is to generate first a *canonical* relational schema in which the attributes are represented by (internal) symbolic names. Subsequently, a name assignment algorithm may be applied in order to replace these symbolic names by (semantically) meaningful names. Provided that this algorithm complies with the UR assumptions that underly relational normalization, the relational schema can then be normalized. Conversely, normalization can be applied directly to the canonical schema and then a name assignment algorithm can be applied to the normalized schema. Naturally, the name assignment algorithm in this case must still comply with the UR assumptions that underly relational normalization.

Following the criteria for name assignment discussed above, we propose two name assignment algorithms. The difference between these algorithms can be characterized in terms of *local* vs. *global* attribute names. The local name assignment algorithm, called $\mathbf{Assign}_L$, is intended for relational interfaces whose users know both attributes and relation names; therefore it is sufficient for the attribute names to be locally (i.e. relative to their relations) unique. The global name assignment algorithm, called $\mathbf{Assign}_G$, is intended for UR-oriented interfaces, whose users know only attribute names; therefore the attribute names must be globally unique.

In order to illustrate the difference between these two algorithms, we refer again to the simple example of figure 1(i). The canonical schema for the ER schema of figure 1(i) includes four relation-schemes, R_1, R_2, R_3, and R_4, shown in figure 1(ii). Relation-schemes R_1 and R_2 have one relational (non-foreign) attribute each, corresponding to the identifiers of the corresponding entity-sets. Relation-schemes R_3 and R_4 have two relational (foreign) attributes each, corresponding to the identifiers of the entity-sets involved, and R_3 has an additional (non-foreign) attribute. For both algorithms, the relation-schemes can be assigned the names of the corresponding object-sets. Since $\mathbf{Assign}_L$ is used under the assumption that relation names are visible to the user, it is sufficient for $\mathbf{Assign}_L$ to simply assign to the non-foreign attributes the (local) names of the corresponding EER attributes, as shown in figure 1(ii). In contrast, $\mathbf{Assign}_G$ assigns to non-foreign attributes, names of length 2, by prefixing the names of the corresponding EER attributes with the name of their object-sets, as also shown in figure 1(ii). Recall that in this second case the names of the relations are concealed from the user, and hence the need for global attribute names. Note also that in this simple example we did not have to prefix the ER attribute names in order to achieve global uniqueness of names, but we chose to do that because of the clarity criterion discussed above.

The problem is more complex when assigning names to foreign attributes, especially when the EER structure is cyclic. Under $\mathbf{Assign}_L$ it is possible to assign the same (local) names to attributes A_{3_1} and A_{4_1}, and to attributes A_{3_2} and A_{4_2}, respectively. This does not present a problem because the relation names are visible to the user and a reference to an attribute will include the relation name. However, if the same name assignment was chosen for interfaces without visible relations, there would be no way of distinguishing the different roles of these foreign attributes. These ambiguity of names is captured by the assumptions underlying both normalization and UR interfaces. $\mathbf{Assign}_G$ solves this ambiguity by assigning to some of the foreign attributes different names, or in other words by *renaming* them. The decision of what foreign attributes to rename is based on a graph algorithm that determines which edges in the graph representing the EER structure are involved in such name ambiguities. Some of the edges are selected and *marked* for renaming. For the ER structure represented in figure 1(i), for example, one of the four edges connecting the object-sets may be marked. Suppose that the edge connecting TEACH and PERSON is marked; then, using the role name associated with the marked edge, attribute A_{4_1} will be assigned the new name TEACHER.NAME as shown in figure 1(ii). Note that in principle, additional attributes can be renamed, but according to the non-proliferation of attribute names criterion mentioned above, only the minimum number of foreign attributes should be renamed. The choice of what edge to mark is semantically motivated. There are no algorithmically compelling reasons to prefer one edge over another besides the minimality goal mentioned above. In the example above, if the edge between PERSON and ATTEND is marked, and we choose to use ATTEND for renaming the foreign attributes involved, then it may be unclear whether the attribute name ATTEND.NAME refers to the attending person or the attended course. A good policy is to refer this choice to the database designer, and ask for a *role* name for the marked edge. $\textbf{Assign}_G$ gives preference to the edges with roles over those without roles.

The discussion above is only a simple example intended to illustrate some of the aspects of name assignment. Similar considerations have to be given for inherited foreign attributes that result from more complex structures. The formal treatment of name assignment in this paper ensures the correctness of the result regardless of the complexity of the EER schema.

A final remark on how $\textbf{Assign}_G$ can support the methodologies underlying UR interfaces. In general, these methodologies are based on relation correlations implied by attribute names (i.e. two relations are correlated if they have attributes with identical names), and renaming attributes limits this capability. However, $\textbf{Assign}_G$ does not unnecessarily reduces this capability since it minimizes the attribute renamings. Moreover, $\textbf{Assign}_G$ can be used in order to keep track of the attribute renamings. This information on attribute renamings can be used in order to overcome the limitation mentioned above. An extended UR methodology that uses such information has been proposed in [5].

3. CANONICAL MAPPING OF EER SCHEMAS

As mentioned above, the mapping of Extended Entity-Relationship (EER) schemas into relational schemas has three aspects: (i) the generation of relational schemas that represent the structural semantics of EER schemas; (ii) the normalization of relational schemas representing EER schemas; and (iii) the assignment of names to the relational attributes generated in (i). In [7] we have proposed a mapping, called **Crep** (**C***anonical* **rep***resentation*), which is provably correct and which is independent of a particular attribute name assignment. In this section we review the results of [7]. We first review briefly the version of the EER model used in this paper.

Next, we review **Crep** . Finally, we discuss the constraints that the relational attribute names must satisfy in order to comply with the assumptions underlying normalization and Universal-Relation interfaces.

3.1 The Extended Entity-Relationship Model.

The concepts of the basic *Entity-Relationship (ER)* model, (*entity, relationship, entity—set, relationship—set, value—set, attribute, entity—identifier, weak* entity-set, relationship *cardinality, role*) have been defined originally in [2] and have been repeatedly reviewed since then (e.g. see [9]). We refer commonly to entities and relationships as *objects*. Unlike the basic ER model of [2] the *extended ER* (EER) model that we use in this paper has two additional abstraction capabilities, generalization and full aggregation. *Generalization* is an abstraction mechanism that views a set of entity-sets as a single *generic* entity-set. The inverse of generalization is called *specialization*. An entity-set which is not specified as the specialization of any other entity-set is called *generalization—source*. In the basic ER model the *aggregation* construct takes three forms: (i) the aggregation of a collection of attributes into an entity-set; (ii) the aggregation of a collection of attributes and the entity-identifiers of several existing entity-sets into a weak entity-set; and (iii) the aggregation of two or more entity-sets into a relationship-set. The EER model provides the full capability of aggregation by allowing in addition relationship-sets to associate any object-set, rather than only entity-sets. For the sake of brevity, we omit the definitions of these concepts. For detailed definitions and explanations see [8] and [9].

EER-schemas are expressible in a diagrammatic form called *EER diagram* (EERD). Entity-sets, relationship-sets, and attributes, are represented graphically by rectangles, diamonds, and ellipses, respectively. Every vertex is labeled by the name of the represented object-set or attribute. Edges that represent ID-dependencies and generalizations are labeled with *ID* and *ISA* labels, respectively. Roles can be represented by edge labels in the corresponding EER diagram. Unlike in [2], we define the EER diagram as a *directed graph*. Note that in the

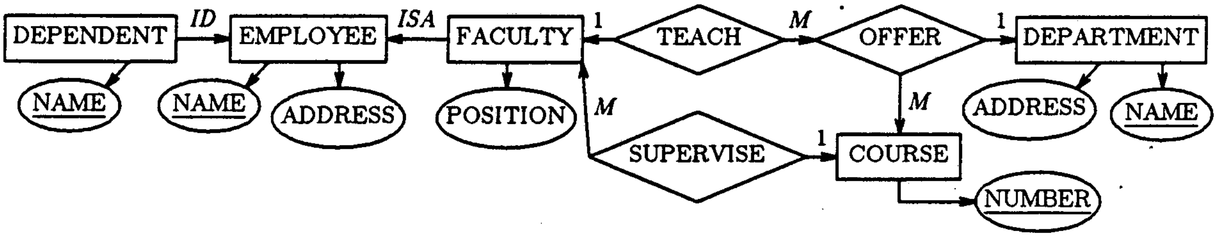


Figure 2. Extended Entity-Relationship Diagram Example (*identifiers are underlined*).

EER model the edge directionality is essential for the unambiguous representation of generalizations and aggregations. A self-explanatory example of an EER diagram is shown in figure 2.

In order to be semantically well-formed, EER structures must satisfy certain restrictions concerning the combination of different EER constructs. A detailed discussion can be found in [8]. The first restriction refers to disallowing *directed* cycles in EER diagrams. Second, specialization entity-sets are restricted to have unique generalization-sources. The third restriction concerns the interaction of generalization and aggregation, namely that an entity-set cannot be specified by using both generalization and aggregation. The fourth restriction concerns the *names* used for the specification of EER structures. For a given EER structure, object-sets must have unique *global* names, all the attributes of an object-set (including the inherited attributes, for specialization entity-sets) must have unique *local* names among the attributes associated with that object-set, and the roles must have unique names among the multiple roles of some object-set in other object-sets.

3.2 Canonical Relational Representations for EER Schemas.

We use letters from the beginning of the alphabet to denote attributes and letters from the end of the alphabet to denote sets of attributes. A sequence of attributes (e.g. ABC) denotes the set containing these attributes and a sequence of sets (e.g. XY) denotes the union of these sets.

EER value-sets are represented straightforwardly by relational domains. **Crep** represents independent entity-sets, aggregations, and generalizations as specified below; an example, for the EER schema of figure 2, is shown in figure 3.

(i) An independent (i.e. neither weak nor specialization) entity-set, E_i, is represented by a relation-scheme, $R_i(X_i)$, such that X_i is in a one-to-one correspondence with the EER attributes of E_i. R_i is associated with functional dependency $R_i : Z_i \rightarrow (X_i - Z_i)$, where Z_i is the subset of X_i that corresponds to the identifier of E_i.

(ii) Let object-set O_i be the aggregation of object-sets O_{i_j}, $1 \leq j \leq m$, and let each object-set O_{i_j} be represented by relation-scheme $R_{i_j}(Y_{i_j})$, $1 \leq j \leq m$, respectively. Then O_i is represented by relation-scheme $R_i(X_i)$ together with inclusion dependencies $R_i[X_{i_j}] \subseteq R_{i_j}[Y_{i_j}]$, $1 \leq j \leq m$, where X_i is the union of two disjoint sets of attributes, X'_i and X''_i : (1) X'_i is in a one-to-one correspondence with the attributes of O_i;

(2) $X''_i = \bigcup_{j=1}^{m} X_{i_j}$, is a set of foreign attributes, where attribute sets X_{i_j}, $1 \leq j \leq m$, are pair-wise disjoint, and X_{i_j} is in a one-to-one correspondence with Y_{i_j}, $1 \leq j \leq m$. R_i is associated

with functional dependency $R_i : Z_i X''_i \rightarrow (X_i - Z_i X''_i)$, where Z_i is the subset of X_i that corresponds to the identifier of O_i. If O_i is a relationship-set, then for every object-set that is involved in O_i with cardinality *one*, O_{i_j}, R_i is associated with the additional functional dependency $R_i : (X''_i - X_{i_j}) \rightarrow X_{i_j}$, where X_{i_j} is defined as above.

For example, the weak entity-set DEPENDENT of figure 2, is represented by relation-scheme $R_5\,(X_5)$, where $X'_5 = A_{5_1}$ and $X''_5 = X_{5_1} = A_{5_2}A_{5_3}$. The relationship-set OFFER is represented by relation-scheme $R_6\,(X_6)$, where X'_6 is empty and X''_6 consists of two disjoint subsets, $X_{6_2} = A_{6_1}A_{6_2}$ and $X_{6_3} = A_{6_3}$.

(iii) Let entity-set E_i be the specialization of entity-sets E_{i_j}, $1 \leq j \leq m$, and let E_s be the generalization-source of E_i. The set of *inherited* attributes of E_i consists of the attributes associated with all the generic entity-sets of E_i. Let E_s be represented by relation-scheme $R_s(Y_s)$ and each entity-set E_{i_j} be represented by relation-scheme $R_{i_j}(Y_{i_j})$, $1 \leq j \leq m$, respectively. Then E_i is represented by relation-scheme $R_i(X_i)$ together with inclusion dependencies $R_i[\,X_{i_j}\,] \subseteq R_{i_j}[Y_{i_j}]$, $1 \leq j \leq m$, where X_i is the union of two disjoint sets of

Relation : Object-Set	**Attribute : ER Attribute**	**Foreign Attribute : Attribute**
$R_1\,(X_1)$: EMPLOYEE	A_{1_1} : NAME A_{1_2} : ADDRESS	
$R_2\,(X_2)$: DEPARTMENT	A_{2_1} : NAME A_{2_2} : ADDRESS	
$R_3\,(X_3)$: COURSE	A_{3_1} : NUMBER	
$R_4\,(X_4)$: FACULTY	A_{4_1} : POSITION	$A_{4_2} : A_{1_1}$ $A_{4_3} : A_{1_2}$
$R_5\,(X_5)$: DEPENDENT	A_{5_1} : NAME	$A_{5_2} : A_{1_1}$ $A_{5_3} : A_{1_2}$
$R_6\,(X_6)$: OFFER		$A_{6_1} : A_{2_1}$ $A_{6_2} : A_{2_2}$ $A_{6_3} : A_{3_1}$
$R_7\,(X_7)$: TEACH		$A_{7_1} : A_{4_1}$ $A_{7_2} : A_{4_2}$ $A_{7_3} : A_{4_3}$ $A_{7_4} : A_{6_1}$ $A_{7_5} : A_{6_2}$ $A_{7_6} : A_{6_3}$
$R_8\,(X_8)$: SUPERVISE		$A_{8_1} : A_{4_1}$ $A_{8_2} : A_{4_2}$ $A_{8_3} : A_{4_3}$ $A_{8_4} : A_{3_1}$

Functional Dependencies	**Inclusion Dependencies**	
$R_1 : A_{1_1} \rightarrow A_{1_2}$		
$R_2 : A_{2_1} \rightarrow A_{2_2}$		
$R_3 : A_{3_1} \rightarrow \emptyset$		
$R_4 : A_{4_2}A_{4_3} \rightarrow A_{4_1}$	$R_4[A_{4_2}A_{4_3}] \subseteq R_1[A_{1_1}A_{1_2}]$	
$R_5 : A_{5_1}A_{5_2}A_{5_3} \rightarrow \emptyset$	$R_5[A_{5_2}A_{5_3}] \subseteq R_1[A_{1_1}A_{1_2}]$	
$R_6 : A_{6_3} \rightarrow A_{6_1}A_{6_2}$	$R_6[A_{6_1}A_{6_2}] \subseteq R_2[A_{2_1}A_{2_1}]$	$R_6[A_{6_3}] \subseteq R_3[A_{3_1}]$
$R_7 : A_{7_4}A_{7_5}A_{7_6} \rightarrow A_{7_1}A_{7_2}A_{7_3}$	$R_7[A_{7_1}A_{7_2}A_{7_3}] \subseteq R_4[A_{4_1}A_{4_2}A_{4_3}]$	$R_7[A_{7_4}A_{7_5}A_{7_6}] \subseteq R_6[A_{6_1}A_{6_2}A_{6_3}]$
$R_8 : A_{8_1}A_{8_2}A_{8_3} \rightarrow A_{8_4}$	$R_8[A_{8_1}A_{8_2}A_{8_3}] \subseteq R_4[A_{4_1}A_{4_2}A_{4_3}]$	$R_8[A_{8_4}] \subseteq R_3[A_{3_1}]$

Figure 3. Canonical Relational Representation for the EER Schema of Figure 2.

attributes, X'_i and X''_i: (1) X'_i is in a one-to-one correspondence with the set consisting of the attributes of E_i and the inherited attributes of E_i; (2) X''_i is in a one-to-one correspondence with $Y''_s \subseteq Y_s$, where Y''_s is defined as in (ii.2) above; (3) each set of foreign attributes, X_{i_j}, includes X''_i and is in a one-to-one correspondence with Y_{i_j}, $1 \leq j \leq m$, such that the corresponding attributes of X_{i_j} and Y_{i_j} result from the mapping of either the same EER attribute or of the same attribute of Y''_s. R_i is associated with functional dependency $R_i : X_{i_s} \rightarrow (X_i - X_{i_s})$, where $X_{i_s} \subseteq X_i$ corresponds to Y_s.

For example, the specialization entity-set of the EER schema of figure 2 is represented as shown in figure 3, where $X'_4 = A_{4_1}A_{4_2}A_{4_3}$, $X_{4_1} = A_{4_2}A_{4_3}$, and X''_4 is empty.

To summarize, **Crep** generates relational schemas of the form $(R, I \cup F)$, where (R, I) represents the EER schema, and F represents entity identifiers and relationship cardinalities. In [7] we have proved that the relational schemas generated by **Crep** represent *correctly* the corresponding EER schemas. Note that all the attributes of the object-sets that are part of some aggregation or generalization have correspondents in the relation-scheme representing the aggregate or specialization object-set. This is caused by the lack of any relational key information at this stage; keys are computed in a latter stage and redundant attributes can be removed by a normalization mapping (see [7] for details and the specification of such a mapping). Normalization requires a special framework discussed below.

3.3 Assumptions Underlying Normalization and Universal-Relation Interfaces.

In this section we examine the assumptions, called *Universal—Relation* (UR) assumptions, underlying normalization and UR interfaces. Satisfying the UR assumptions leads to three conditions that the names assigned to the relational attributes generated by **Crep** must satisfy. We refer in this section only to the *global* names of relational attributes, that is, names that are sufficient for the identification of the attributes within the entire relational schema. Clearly, two attributes are identical iff they are assigned the same global name. We briefly review below the UR assumptions and examine their impact on attribute name assignments. The UR assumptions are surveyed in [6].

A relational attribute, A, generated by **Crep** represents the EER attribute (a) to which it corresponds directly (see (i), (ii.1), and (iii.1) of **Crep**), or (b) which is represented by the relational attribute to which A corresponds as a foreign attribute (see (ii.2) and (iii.3) of **Crep**). The *Universal-Relation Scheme Assumption* requires each attribute to represent a property of the same class of objects in every relation-scheme in which it appears. Accordingly, in a relational schema generated by **Crep** the following condition must hold:

(U1) attributes representing distinct EER attributes must be assigned distinct global names.

In the relational schema of figure 3, for example, the attributes that represent EER attribute NAME of entity-set DEPARTMENT (e.g. A_{2_1}, A_{6_1}) must be assigned different names than the attributes that represent EER attribute NAME of entity-set EMPLOYEE (e.g. A_{1_1}, A_{4_2}, A_{7_2}).

The *Unique Role Assumption* (URA) requires every attribute set W consisting of more than one attribute, to represent *at most one* basic *association* among the attributes of W. The first aspect of URA refers to W as part of the attribute-set of some relation-scheme: if W appears in more than one relation-scheme then W must represent the same class of objects in all the relation-schemes in which it appears. Let an EER schema associated with EER diagram G_{ER} be mapped by **Crep** into relational schema $(R, I \cup F)$. It can be verified that the inclusion dependency digraph associated with I is isomorphic to the subgraph of G_{ER} induced by the vertices representing object-sets [8]. In order to comply with URA the relational attributes generated by **Crep** must be assigned names that satisfy the following conditions:

(U2) (i) The attribute digraph associated with R must be a subgraph of the inclusion dependency digraph associated with I; and (ii) for any two relation-schemes of R, $R_i(X_i)$ and $R_j(X_j)$, if $X_i \cap X_j$ consists of more than one attribute, then there exists a relation-scheme $R_k(X_k)$ such that $X_i \cap X_j = X_k$.

Note that conditions (U2.i) and (U2.ii) above correspond to the *containment condition* and *association integrity* of [6], respectively.

In general, the *basic association* represented by some attribute set W refers to the projection on W, of the join of a set of relations. The corresponding join expression is based on a *join-path* which consists of a sequence of relation-schemes. As noted in [6], if multiple join-paths can be associated with a given set of attributes, then URA implies the additional *One-Flavor Assumption* (OFA). OFA requires all the join-paths that can be associated with some attribute set to represent the same *flavor* of relationship [6]. In order to comply with OFA the relational attributes generated by **Crep** must be assigned names that satisfy the following condition:

(U3) The attribute digraph associated with R is allowed to contain (undirected) cycles only of the following form: all the vertices on such a cycle correspond to relation-schemes that represent entity-sets belonging to the same generalization hierarchy.

For example, if in the relational schema of figure 3 attributes A_{8_1}, A_{8_2}, A_{8_3}, and A_{8_4} are assigned the same names as attributes A_{7_1}, A_{7_2}, A_{7_3} and A_{7_6}, respectively, then condition (U2) is not satisfied because, following this assignment, attribute set X_8 is included in X_7 although the corresponding relationship-sets are independent. More details on the three conditions above and their derivation from the UR assumptions can be found in [7] and [8].

4. A LOCAL NAME ASSIGNMENT ALGORITHM

We have discussed in section 3.2 how to represent correctly an EER schema by a relational schema. The use of symbolic names for the representation of relational attributes allowed us to keep the mapping of EER schemas into relational schemas independent of a specific attribute name assignment. Since **Crep** generates a set of relation-schemes that is in a one-to-one correspondence with the set of mapped object-sets, every relation-scheme generated by **Crep** can be simply assigned the name of its object-set correspondent. In the present and next sections we propose two name assignment algorithms, **Assign**$_L$ and **Assign**$_G$, for the attributes of relational schemas representing EER schemas. Both algorithms satisfy conditions (U1), (U2), and (U3) discussed in section 3.3, in order to ensure the compatibility of relational schemas with relational normalization. These algorithms are intended for different relational environments: while **Assign**$_L$ assumes that users know both attributes and relations, **Assign**$_G$ is intended for users which are expected to know only attribute names.

Let $R_i(X_i)$ be the relation-scheme corresponding to object-set O_i. The attributes of every relation-scheme $R_i(X_i)$ generated by **Crep** are partitioned into two disjoint sets of attributes, X'_i and X''_i, where X'_i is in a one-to-one correspondence with the local or inherited EER attributes of O_i, and X''_i is the set of foreign attributes in X_i that do not correspond to inherited EER attributes of O_i. The correspondence of X'_i with the (inherited) EER attributes of O_i permits the assignment of the names of the EER attributes to the corresponding attributes of X'_i. Thus, for example, in the relational schema of figure 3, attribute A_{3_1} (which corresponds to the EER attribute NUMBER of COURSE) can be assigned the local name NUMBER, while attribute A_{4_2} (which corresponds to the inherited EER attribute NAME of EMPLOYEE) can be assigned the local name NAME (see figure 5).

For foreign attributes that do not correspond directly to EER inherited attributes the assignment of local names cannot follow the simple strategy above because of two potential sources of conflicts: (i) conflicts between the names of different foreign attributes, or between the names of foreign attributes and non-foreign attributes, where the relational attributes correspond to EER attributes with identical names; and (ii) between the names of different foreign attributes that correspond to the same EER attribute. For example, if attributes A_{5_1} and A_{5_2} of the relational schema of figure 3, which correspond to EER attributes NAME of DEPENDENT and NAME of EMPLOYEE, respectively, are assigned the names of their EER attribute correspondents, then a name conflict of the first kind mentioned above will arise. Such a name conflict can be avoided by assigning as local names for foreign attributes the names of their EER correspondents, prefixed by the name of the associated object-sets. Thus, for example, attributes A_{5_1} and A_{5_2} mentioned above can be assigned NAME and EMPLOYEE.NAME, respectively. This assignment strategy, however, does not resolve the conflicts between the names of foreign attributes corresponding to the same EER attribute. Consider the EER diagram of figure 4(i). Let $R_9(X_9)$ be the relation-scheme corresponding to object-set O_9 ; then X_9 includes two foreign attributes corresponding to EER attribute D of object-set O_8: one as a result of the direct involvement of O_8 in O_9 , and the second one as a result of the indirect involvement of O_8 in O_9 , via O_7 and O_6 . Graphically, this structure is characterized by two paths from O_9 to O_8 in the EER diagram of figure 4(i). Note, however, that these multiple paths differ in non-ISA edges. Multiple paths of ISA-edges from an entity-set, E_i , to another entity-set, E_j , represent different ways of inheritance of the EER attributes of E_j by E_i . Clearly, every inherited EER attribute of E_i corresponds to a single foreign attribute in the relation-scheme corresponding to E_i . In the EER diagram of figure 4(i), for instance, there are two distinct paths from O_4 to O_2 , but the relation-scheme corresponding to O_4 , $R_4(X_4)$, is associated with a single foreign attribute that corresponds to EER attribute B of O_2.

Proposition 4.1. Let G_{ER} be an EER diagram and let $(R, I \cup F)$ be the corresponding relational schema generated by **Crep**. Let $R_i(X_i)$ be a relation-scheme of R corresponding to object-set O_i . Then X_i includes two distinct foreign attributes, A_{i_j} and A_{i_k}, corresponding to the same EER attribute of some object-set O_j iff there exist two distinct paths from O_i to O_j in G_{ER} that differ in at least one non-ISA edge.

Proof Sketch . The proof is by induction on the number of steps of **Crep**. □

The multiple paths mentioned in proposition 4.1 form undirected cycles in the underlying undirected graph of the EER diagram. Since such cycles characterize name conflicts between the local names of foreign attributes, we call them *name-conflict (nc) cycles.* Thus, the two paths from O_9 to O_8 mentioned above form an nc-cycle.

We need a strategy for assigning local names to foreign attributes so that conflicts such as those mentioned above would be avoided. The solution we propose is to resolve such conflicts by prefixing the local names of foreign attributes with additional object-set or role names. We specify below a procedure that determines when such additional concatenations are required. Specifically, the procedure finds the object-sets whose EER attributes have multiple foreign attribute correspondents in the same relation-scheme. The roles of the edges incident to such object-sets will be used in generating local names for foreign attributes. If such roles have not been specified, then either the database designer can be asked to provide them or, by default, object-set names can be used instead of the missing role names.

The main steps of the procedure are exemplified in figure 4. Starting with an EER diagram, G_{ER} (figure 4(i)), (1) first we construct the reduced EER diagram, $\overline{G}_{ER}$, by removing the attribute vertices and their incident edges, and by unifying the vertices representing entity-sets that belong to the same generalization hierarchy (figure 4(ii)); next, (2) we determine all the vertices that have indegree 0 (O_9 and O_1 in figure 4(ii)); for each such vertex, O_i , (3) we determine the subgraph of $\overline{G}_{ER}$ that is induced by O_i together with all the edges reachable from O_i (figures 4(iii) and 4(v)); (4) for every subgraph determined in (3) we find a *directed spanning tree* (figures 4(iv) and 4(v)); finally, (5) the edges that belong to a subgraph found in step (3), but do not belong to the corresponding spanning tree found in step (4), are *marked* and removed from $\overline{G}_{ER}$. We give below the precise definition of the procedure outlined above. The roles of the edges marked by this procedure are used by the name assignment algorithm **Assign**$_L$.

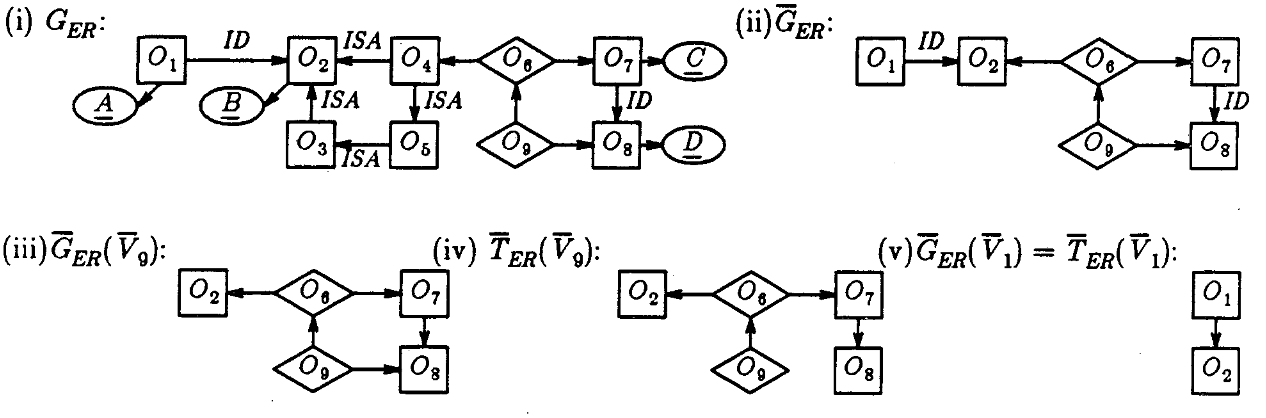


Figure 4. Marking EER Diagram Edges for Local Name Assignment.

Definition 4.1 – $\textbf{Mark}_L$.

Input: EER diagram $G_{ER} = (V, H)$ representing an EER structure.

Output: Marked edges of G_{ER} .

(1) Construct the *reduced* EER diagram $\overline{G}_{ER} = (\overline{V}, \overline{H})$ from G_{ER} as follows:

$\overline{V}$: remove from V vertices that represent attributes and specialization entity-sets;

$\overline{H}$: (a) remove from H edges that are incident to attribute vertices and ISA-edges, and

(b) replace edges of H , $O_i \rightarrow E_j$ where E_j represents a specialization entity-set, by $O_i \rightarrow E_k$ where E_k represents the generalization-source of E_j .

(2) Let O_i be a vertex of $\overline{G}_{ER}$ such that there is no $O_j \in \overline{V}$ such that $O_j \rightarrow O_i \in \overline{H}$;

O_i is called a *root–vertex* in $\overline{G}_{ER}$.

While there exist root-vertices in $\overline{G}_{ER}$ *Do*

(3) Choose a root-vertex O_i ; determine $\overline{G}_{ER}(\overline{V}_i)$, the subgraph of $\overline{G}_{ER}$ induced by $\overline{V}_i$, where $\overline{V}_i$ is the subset of vertices that are reachable from O_i in $\overline{G}_{ER}$.

(4) For $\overline{G}_{ER}(\overline{V}_i)$ determined in step (3), *find* a *directed spanning tree*, $\overline{T}_{ER}(\overline{V}_i)$.

Mark the edges of G_{ER} that belong to $\overline{G}_{ER}(\overline{V}_i)$ and do not belong to $\overline{T}_{ER}(\overline{V}_i)$.

(5) Update $\overline{G}_{ER}$ by removing from $\overline{H}$ the marked edges determined in (4).

EndDo □

Relation Scheme	**Non–Foreign Attributes**	**Foreign Attributes**
EMPLOYEE	NAME, ADDRESS	
DEPARTMENT	NAME, ADDRESS	
COURSE	NUMBER	
FACULTY	POSITION, NAME, ADDRESS	
DEPENDENT	NAME	E.NAME, E.ADDRESS
OFFER		D.NAME, D.ADDRESS, C.NUMBER
TEACH		F.NAME, F.ADDRESS, F.POSITION D.NAME, D.ADDRESS, C.NUMBER
SUPERVISE		F.NAME, F.ADDRESS, F.POSITION, C.NUMBER

Abbreviations: C–COURSE, D–DEPARTMENT, E–EMPLOYEE, F–FACULTY

Figure 5. Local Attribute Names Under $\textbf{Assign}_L$ for the Relational Schema of Figure 3.

The complete specification of **Assign**$_L$ is given below.

Definition 4.2 — Assign$_L$.

Let $(R, I \cup F)$ be the relational schema generated by **Crep** for an EER schema. An attribute A_{i_m} associated with relation-scheme $R_i(X_i)$ of R , where R_i correspond to object-set O_i , is assigned a *local name* as follows:

```
If $A_{i_m}$ corresponds to an (inherited) EER attribute of $O_i$
(1) Then $A_{i_m}$ is assigned the local name of that EER attribute;
(2) Else let $A_{i_m}$ correspond to relational attribute $A_{j_n}$ of relation-scheme $R_j(X_j)$,
                        where $R_j$ corresponds to object-set $O_j$ ;
        If $O_i \rightarrow O_j$ is not marked by Mark$_L$
          Then If $A_{j_n}$ corresponds to an (inherited) EER attribute of $O_j$
(3)                Then $A_{i_m}$ is assigned the local name of $A_{j_n}$ in $X_j$
                        prefixed by the name of $O_j$ ;
(4)                Else $A_{i_m}$ is assigned the local name of $A_{j_n}$ in $X_j$ EndIf
        EndIf
        If $O_i \rightarrow O_j$ is marked by Mark$_L$
(5)       Then $A_{i_m}$ is assigned the local name of $A_{j_n}$ in $X_j$ prefixed by the role of $O_j$
                        in $O_i$ , or (when the role is unspecified) by the name of $O_j$ .
        EndIf
EndIf □
```

The global names of relational attributes under **Assign**$_L$ consist of their local names prefixed by the name of the corresponding relation-scheme. An example of how **Assign**$_L$ can be applied on a relational schema generated by **Crep** is given in figure 5. The correctness of **Assign**$_L$ is stated below.

Proposition 4.2. Let $(R, I \cup F)$ be the relational schema generated by **Crep** and let the relational attribute names be assigned by **Assign**$_L$. Then the relational attribute names are (i) consistent with the specification of **Crep** and (ii) satisfy conditions (U1), (U2), and (U3).

Proof Sketch. (i) The proof is by induction on the number of steps of **Crep**. (ii) Conditions (U1), (U2), and (U3) refer to the global names of relational attributes. Since under **Assign**$_L$ attributes have distinct global names, these conditions are trivially satisfied. □

5. A GLOBAL NAME ASSIGNMENT ALGORITHM

The global attribute name assignment algorithm, $\mathbf{Assign}_G$, is intended for relational interfaces whose users are expected to know only attribute names. Let $(R, I \cup F)$ be generated by **Crep** and let $R_i(X_i)$ be a relation-scheme of R ; the foreign attributes of X_i are involved in left-hand sides of inclusion dependencies. When the name of a foreign attribute is different from the name of its corresponding (via the inclusion dependency) attribute, then it is said to be *renamed*. Following the principle discussed in section 2.1 of retaining the EER names, renaming can be achieved by embedding the names of several EER elements, such as EER attributes, object-sets, roles, into a relational attribute name. As noted in [1], attribute renaming has several negative aspects: (i) it leads to the proliferation of relational attributes (every renamed attribute is an additional attribute); (ii) when it is based on concatenation it leads to large attribute names (e.g. attribute A_{7_4} of the relational schema of figure 3 can be assigned the name TEACH.OFFER.DEPARTMENT.NAME); and (iii) as a result of name embedding, it leads to unclear and, from a user's point of view, unmanageable, attribute names. These disadvantages are particularly flagrant when all foreign attributes are renamed. For example, under $\mathbf{Assign}_L$, we allow all foreign attributes to be renamed because the relation names are visible, but such an assignment is inappropriate for an interface in which users deal with attribute names only. Consequently, the renaming should be limited only to the necessary cases so that the number of renamed attributes would be kept at a minimum.

The best known relational interfaces that allow users to manipulate only attribute names are the *Universal–Relation* (UR) interfaces [6]. If the relational schema generated by **Crep** is intended to be used as a UR interface, then the UR assumptions underlying such an interface should be satisfied. As discussed in section 3.3, satisfying the UR assumptions leads to conditions (U1), (U2) and (U3) that the global names assigned to the relational attributes generated by **Crep** must satisfy. As noted in section 2.2, the methodologies underlying UR interfaces are based on relation correlations implied by attribute names (i.e. two relations are correlated if they have attributes with identical names), so that attribute renaming limits the capability of such methodologies to support UR interfaces. This restriction has been overcome by the methodology of [5] which takes into account the information on attribute renaming kept in the form of a directed R (*role*) graph, whose set of vertices consists of relational attributes and each edge, $A_i \rightarrow A_j$ represents the fact that attribute A_j resulted by renaming attribute A_i. We show below that $\mathbf{Assign}_G$ can be easily adapted in order to construct such an R-graph.

Let G_{ER} be an EER diagram and $(R, I \cup F)$ the relational schema generated by **Crep** and corresponding to G_{ER}. As mentioned in section 3.3, the inclusion dependency digraph G_I is isomorphic to the subgraph of G_{ER} induced by the object-set vertices, and the attribute digraph associated with R, G_A, is a subgraph of G_I. Clearly, the edges of G_I which do not belong to G_A are those corresponding to the renamed foreign attributes, that is, if $R_i \rightarrow R_j$ is an edge of G_I but not an edge of G_A, then some foreign attributes of R_i referencing R_j are renamed. Undirected cycles in the attribute digraph associated with a relational schema generated by **Crep** must be only of the form allowed by condition (U3). Consequently, cycles that have a different form must be broken by renaming attribute names. Some of the cycles in the attribute digraph are caused by the assignment of the same global name to distinct foreign attributes of the same relation-scheme. Such cycles are isomorphic to the EER diagram nc-cycles discussed in section 4, and therefore are also called *nc—cycles*. For example, for the attribute digraph which is identical to the inclusion dependency digraph of figure 6(ii), the cycle involving vertices R_6, R_7, R_8, and R_9, is an nc-cycle.

We specify below a procedure that determines a strategy for renaming foreign attributes so that (i) the corresponding attribute digraph will be free of undesired cycles, and (ii) the overall number of attributes (i.e. global names) will be minimal. Note that the foreign attributes referencing some relation-scheme must be either renamed together or not renamed at all in order to satisfy condition (U2.ii). We want to rename attributes using role names, whenever possible. For an object-set O_j that has no role for its involvement in object-set O_i, if O_j is not involved in any other object-set, then the name of O_j is considered its *default role* in O_i.

The main steps of the procedure are exemplified in figure 6. Starting with an EER diagram, G_{ER} (figure 6(i)), (1) first we construct the corresponding inclusion dependency graph, G_I (figure 6(ii)); then (2) we unify in G_I the vertices corresponding to entity-sets that belong to the same generalization hierarchy, thus obtaining a reduced digraph $\overline{G}_I$ (figure 6(iii)); (3) next we associate with every edge of $\overline{G}_I$ a *weight* that represents the number of additional relational attributes which would result by renaming the foreign attributes involved in the left-hand side of the inclusion dependency corresponding to that edge (see the edge labels in figure 6(iii)); additionally, edges that correspond to EER edges that are associated with (default) roles have a star (*) label; (4) we find the edges of $\overline{G}_I$ that must be removed from G_A in order to break the undesirable cycles mentioned above; in order to minimize the number of renamed attributes, we must maximize the number of foreign attributes that are not renamed, therefore we must find the subgraph of $\overline{G}_I$ corresponding to the maximum spanning tree of the underlying graph of $\overline{G}_I$; if there are multiple choices in generating the maximum spanning tree, then the star-labeled edges are preferred (figure 6(iv)). The foreign attributes that are involved in the left-hand sides of the inclusion dependencies corresponding to the edges that do not belong to the spanning tree found in (4) are candidates for renaming.

Definition 5.1 – Mark$_G$.

Input: EER diagram G_{ER} representing an EER schema.

Output: Marked edges of G_I .

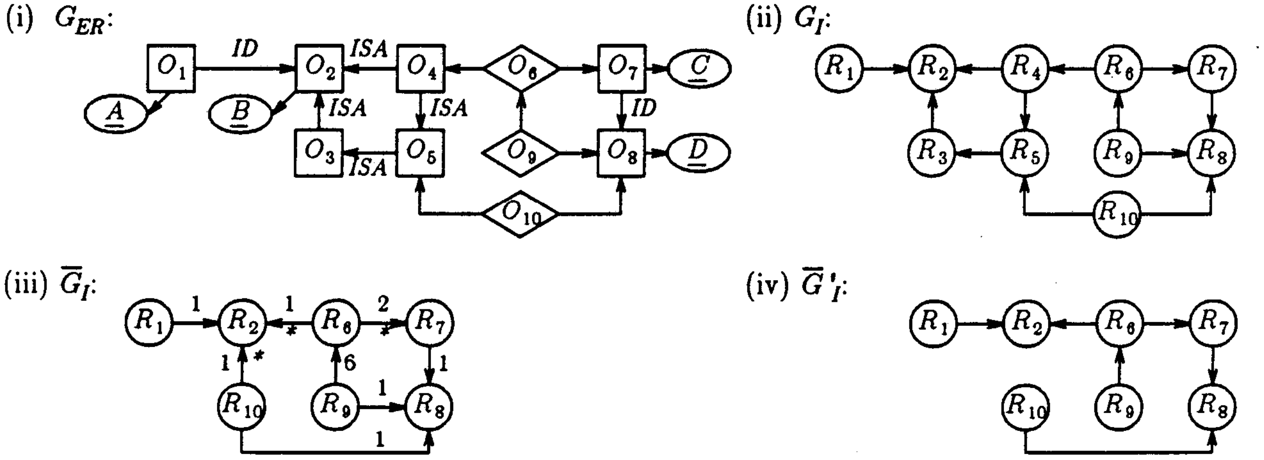
(1) Construct the inclusion dependency digraph $G_I = (V, H)$ isomorphic to G_{ER} .

(2) Construct the digraph $\overline{G}_I = (\overline{V}, \overline{H})$ as follows:

$\overline{V}$: remove from V vertices that correspond to specialization entity-sets;

$\overline{H}$: (a) remove from H edges that correspond to ISA-edges and (b) replace edges of H, $R_i \rightarrow R_j$ where R_j corresponds to a specialization entity-set, E_j , by $R_i \rightarrow R_k$ where R_k corresponds to the generalization-source of E_j .

(3) Every edge of $\overline{H}$, $R_i \rightarrow R_j$, is associated with a *weight* , ω_{ij} , representing the number of foreign attributes associated with R_i that correspond to attributes associated with relation-scheme R_j : $\omega_{ij} = n_j + f_j$, where n_j and f_j denote the number of non-foreign and


(i) G_{ER}:

(ii) G_I:

(iii) $\overline{G}_I$:

(iv) $\overline{G}'_I$:

Figure 6. Marking Inclusion Dependency Digraph Edges for Global Name Assignment.

foreign attributes of R_j, respectively, and $f_j = \sum_{\{R_j \to R_k \in \overline{H}\}} f_k$.

(4) Find a connected subgraph of $\overline{G}_I$, $\overline{G}'_I = (\overline{V}', \overline{H}')$, whose underlying graph is the *maximum* spanning tree of the underlying graph of $\overline{G}_I$, that is, such that $\overline{V}' = \overline{V}$ and for which the sum of edge weights is maximum; in every step of choosing an edge for the maximum spanning tree, star-labeled edges are preferred over unlabeled edges.

(5) **Mark** the edges of G_I that belong to $(\overline{H} - \overline{H}')$. □

The specification of $\mathbf{Assign}_G$ is given below.

Definition 5.2 – $\mathbf{Assign}_G$.

Let $(R, I \cup F)$ be the relational schema generated by **Crep** for an EER schema. An attribute A_{i_m} associated with relation-scheme $R_i(X_i)$ of R, where R_i corresponds to object-set O_i, is assigned a *global name* as follows:

```
If $A_{i_m}$ corresponds to an EER attribute of $O_i$
(1) Then $A_{i_m}$ is assigned the name of that EER attribute prefixed by the name of $O_i$;
(2) Else let $A_{i_m}$ correspond to relational attribute $A_{j_n}$ of relation-scheme $R_j(X_j)$,
                                            where $R_j$ corresponds to object-set $O_j$;
        If $R_i \to R_j$ is not marked by $\mathbf{Mark}_G$
(3)         Then $A_{i_m}$ is assigned the global name of $A_{j_n}$    EndIf
        If $R_i \to R_j$ is marked by $\mathbf{Mark}_G$
            Then If $R_i \to R_j$ belongs to an nc-cycle
(4)                 Then $A_{i_m}$ is assigned the global name of $A_{j_n}$ prefixed by the role of $O_j$
                              in $O_i$, or (when the role is unspecified) by the name of $O_j$.
(5)                 Else $A_{i_m}$ is assigned the global name of $A_{j_n}$ in $X_j$ in which the prefix
                              is replaced by the role of $O_j$ in $O_i$, or
                              (when the role is unspecified) is prefixed by the name of $O_i$.
                 EndIf
        EndIf
EndIf □
```

An example of how $\mathbf{Assign}_G$ can be applied on a relational schema generated by **Crep** is given in figure 7. Consider the R-graph mentioned above, employed by the UR methodology of

[5]. This graph can be easily constructed by adding an edge to it whenever an attribute is renamed in steps (4) or (5) of **Assign**$_G$. The correctness of **Assign**$_G$ is stated below.

Proposition 5.1. Let $(R, I \cup F)$ be the relational schema generated by **Crep** and let the relational attribute names be assigned by **Assign**$_G$. Then the relational attribute names (i) are consistent with the specification of **Crep**, (ii) satisfy conditions (U1), (U2), and (U3), and (iii) are minimal in number.

Proof Sketch. (i) The proof is by induction on the number of steps of **Crep**.

(ii) Conditions (U1) and (U2.i) are satisfied by assigning the relational attributes the names of the corresponding EER attributes, prefixed by the names of the associated object-sets. For (U2.ii) and (U3) the proof follows the specification of **Assign**$_G$.

(iii) Suppose that an attribute that is chosen for renaming in step 4 or 5 of **Assign**$_G$, is not renamed. Then it can be verified that either condition (U2.ii) or (U3) is not satisfied. □

Relation Scheme	**Non–Foreign Attributes**	**Foreign Attributes**
EMPLOYEE	E.NAME, E.ADDRESS	
DEPARTMENT	D.NAME, D.ADDRESS	
COURSE	C.NUMBER	
FACULTY	F.POSITION, E.NAME, E.ADDRESS	
DEPENDENT	W.NAME	E.NAME, E.ADDRESS
OFFER		D.NAME, D.ADDRESS, C.NUMBER
TEACH		E.NAME, E.ADDRESS, F.POSITION D.NAME, D.ADDRESS, C.NUMBER
SUPERVISE		F.NAME, F.ADDRESS, F.POSITION, C.NUMBER

Abbreviations: C–COURSE, D–DEPARTMENT, E–EMPLOYEE, F–FACULTY, W–DEPENDENT

Figure 7. Global Attribute Names Under **Assign**$_G$ for the Relational Schema of Figure 3.

6. SUMMARY

We have examined in this paper the criteria for assigning names to attributes of relational schemas representing Extended Entity-Relationship (EER) structures. Following these criteria we have developed two name assignment algorithms, $\mathbf{Assign}_L$ and $\mathbf{Assign}_G$. These algorithms are meant for different relational interfaces: (i) interfaces that require the users to know both relations and attributes ($\mathbf{Assign}_L$), and (ii) interfaces that spare the users the details of how attributes are grouped into relations and allow them to refer only to attributes ($\mathbf{Assign}_G$).

The name assignment algorithms presented in this paper can follow the canonical mapping from EER schemas into relational schemas, or the normalization mapping from relational schemas into BCNF schemas, proposed in [7]. This flexibility is achieved by requiring the name assignment algorithms to satisfy the assumptions underlying normalization. As a byproduct of this requirement, the canonical mapping coupled with $\mathbf{Assign}_G$ generates Universal-Relation schemas. As shown in [7], the use of name assignments that do not comply with the normalization assumptions, is the main cause for lack of precision in most mappings of ER and EER schemas into relational schemas.

In conclusion, the canonical and normalization mappings proposed in [7] can be combined with the name assignment algorithms presented in this paper in order to automatically produce from EER specifications correct and normalized relational schemas. While the mappings reduce the redundancy through normalization and generate the key and referential integrity constraints that must be maintained by the database management system, the attribute name assignment algorithm is tailored to the user's need by producing names that are semantically close to the original EER names.

REFERENCES

[1] P. Atzeni and D.S. Parker, "Assumptions in Relational Database Theory", *ACM Symposium on Principles of Database Systems*, 1982, pp. 1-9.

[2] P.P. Chen, "The Entity-Relationship Model- Towards a Unified View of Data", *ACM Trans. on Database Systems* **1**,1 (March 1976), pp. 9-36.

[3] S. Even, *Graph Algorithms*, Computer Science Press, 1979.

[4] S. Jajodia, P.A. Ng, and F.N. Springsteel, "Entity-Relationship Diagrams which are in BCNF", *Int. Journal of Computer and Information Sciences* **12**,4 (1983), pp. 269-283.

[5] D. Maier, D. Rozenshtein, and J. Stein, "Representing Roles in Universal Scheme Interfaces", *IEEE Trans. on Software Engineering*, vol SE-11,7, July 1985, pp. 644-652.

[6] D. Maier, D. Rozenshtein, and D.S. Warren, "Window Functions", *Advances in Computing Research*, vol.3, JAI Press, 1986, pp. 213-246.

[7] V.M. Markowitz and A. Shoshani, "On the Correctness of Representing Extended Entity-Relationship Structures in the Relational Model", Proc. of *1989 SIGMOD Conference*, SIGMOD Record **18**, 2, June 1989, pp. 430-439.

[8] V.M. Markowitz and A. Shoshani, "On the Correctness of Representing Extended Entity-Relationship Structures in the Relational Model", Lawrence Berkeley Laboratory Technical Report LBL-26389, December 1988.

[9] T.J. Teorey, D. Yang, and J.P. Fry, "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model", *Computing Surveys* **18**,2 (June 1986), pp. 197-222.

[10] J.D. Ullman, *Principles of Database Systems*, Computer Science Press, 1982.

APPENDIX : GRAPH AND RELATIONAL CONCEPTS

A.1 Graph Concepts.

We denote by $G = (V, H)$ a directed graph (*digraph*) with set of vertices V and set of edges H, and by h a directed edge $v_i \rightarrow v_j$, from vertex v_i to vertex v_j; h is said to be *incident* from v_i to v_j. The *underlying* undirected graph of a digraph results by ignoring the edge directions in the digraph. An undirected *path* from (*start*) vertex v_{i_0} to (*end*) vertex v_{i_m} is a sequence of alternating vertices and edges, $v_{i_0} h_{j_1} v_{i_1} \ldots h_{j_m} v_{i_m}$, such that h_{j_k} is incident from (to) $v_{i_{k-1}}$ to (from) v_{i_k}, $1 \leq k \leq m$. The *indegree* of a vertex v_i is the number of edges incident *to* v_i. A *cycle* is a path whose start and end vertices are the same. A path is called *simple* if a vertex appears on it at most once. A path (cycle) is said to be directed if all the edges on the path have the same direction and the first edge is incident *from* the start vertex. If there exists a directed path from vertex v_i to vertex v_j then v_j is said to be *reachable* from v_i.

A digraph $G' = (V', H')$ is a *subgraph* of $G = (V, H)$ if $V' \subseteq V$ and $H' \subseteq H$. The subgraph *induced* by a subset of V, V', is denoted $G(V')$ and is defined as follows: $G(V') = (V', H')$, where $H' = \{v_i \rightarrow v_j \mid v_i \in V', v_j \in V' \text{ and } v_i \rightarrow v_j \in H\}$.

An undirected graph is called a *tree* iff it has no cycles and any edge added to it forms a cycle. A digraph is called a *directed tree* iff its underlying undirected graph has no cycles and it has one vertex with indegree 0, while all the other vertices have indegree 1. A *(directed) spanning tree* of a (directed) graph is a (directed) tree containing all the vertices of the graph. If the edges of the graph are associated with *lengths* (*weights*) then the *maximum* (*minimum*) spanning tree is the spanning tree with the maximum (minimum) sum of edge lengths.

A.2 Relational Concepts.

A *relational schema* is a pair (R, Δ) where R is a set of relation-schemes and Δ is a set of dependencies over R. We consider relational schemas which are associated with set of dependencies $\Delta = F \cup I$, where F and I denote sets of functional and inclusion dependencies, respectively. A *relation—scheme* is a named set of attributes, $R_i(X_i)$, where R_i is the relation-scheme name and X_i denotes the associated set of attributes. A set of relation-schemes, R, can be associated with the following *attribute digraph*: $G_A = (V, H)$, where $V = R$, and $R_i \rightarrow R_j \in H$ iff $R_i(X_i) \in R$, $R_j(X_j) \in R$, $X_j \subseteq X_i$, and $\nexists R_k(X_k) \in R$ such that $X_j \subseteq X_k \subseteq X_i$.

Every attribute is assigned a *domain*, and every relation-scheme, $R_i(X_i)$, is assigned a *relation*, r_i. We denote by t a tuple, by $t[W]$ the sub-tuple of t corresponding to attribute set W, and by $r_i[W]$ the *projection* of r_i on $W \subseteq X_i$, where relation r_i is associated with $R_i\,(X_i)$.

Let $R_i(X_i)$ be a relation-scheme associated with relation r_i. A *functional dependency* over R_i is a statement of the form R_i: $Y \rightarrow Z$ where Y and Z are subsets of X_i ; R_i: $Y \rightarrow Z$ is *satisfied* by r_i iff for any two tuples of r_i, t and t', $t[Y] = t'[Y]$ implies $t[Z] = t'[Z]$. Let $R_i(X_i)$ and $R_j\,(X_j)$ be two relation-schemes associated with relations r_i and r_j, respectively. An *inclusion dependency* is a statement of the form $R_i[Y] \subseteq R_j[Z]$, where Y and Z are subsets of X_i and X_j, respectively, and the corresponding attributes of Y and Z are associated with the same domain. $R_i[Y] \subseteq R_j[Z]$ is *satisfied* by r_i and r_j iff $r_i[Y] \subseteq r_j[Z]$. The attributes involved in the left-hand side of an inclusion dependency are called *foreign* attributes. The set of inclusion dependencies I over the relation-schemes of R can be represented graphically by the following *inclusion dependency digraph*: $G_I = (V, H)$, where $V = R$, and $R_i \rightarrow R_j \in H$ iff $R_i[Y] \subseteq R_j[Z] \in I$.

A *key* associated with R_i is a subset of X_i, K_i, such that $R_i : K_i \rightarrow X_i$ is satisfied by any r_i associated with R_i and there does not exist any proper subset of K_i which has this property. If $R_i[Y] \subseteq R_j[Z]$ is an inclusion dependency and Z is the *primary* key of R_j then Y is called a *foreign key* of R_i *referencing* R_j. For relation-schemes associated with functional dependencies the highest *normal form* is *Boyce-Codd Normal Form (BCNF)* which requires all functional dependencies to be *key dependencies*.

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
1 CYCLOTRON ROAD
BERKELEY, CALIFORNIA 94720