A Normal Form For Entity-Relationship Diagrams

Tok-Wang Ling

National University of Singapore

Abstract

In this paper, a normal form for entity-relationship diagrams and the objectives for such a normal form are presented. A normal form entity-relationship diagram may consist of composite attributes, multivalued attributes, and special types of relationship sets such as existence dependent, identifier dependent, ISA, UNION, INTERSECT, and DECOMPOSE relationships. A comprehensive algorithm is given to translate a normal form entity-relationship diagram to a set of relations. We show that all relations generated are either in third normal form or fifth normal form. General guidelines and detailed steps for converting an entity-relationship diagram to a normal form entity-relationship diagram are presented. These steps can also be used as guidelines for designing databases using entity-relationship approach.

1. Introduction

The entity-relationship(ER) approach for database schema design was proposed in [Chen 76]. It uses the concepts of entity type and relationship set and incorporates some of the important semantic information about the real world. The structure of a database organized according to the entity-relationship approach can be represented by a diagrammatic technique called an entity-relationship (ER) diagram [Chen 76]. Two of the main reasons for using ER approach for database designs are: first, the ER approach allows us to capture and preserve some of the important semantics of the real world; second, it can achieve a high degree of data independence and therefore it becomes an attractive candidate for logical database design. Since ER approach is used for logical database designs, a method is needed to translate ER diagrams to other data models, such as relational data models, network data models, etc. Methods for translating an ER diagram to a relational database have been studied by many researchers [Chung 81, Dumpala 81, Hwang 81, Jajodia 83, Melkanoff 80, Morgenstern 81, Ng 80, Wong 80]. Most of these methods are simply a form of guidelines, rules of thumb, and/or only consider simple ER diagrams which do not include some of the ER concepts such as composite attributes, weak entity types, recursive relationship sets, and weak relationship sets, namely, ISA, UNION, INTERSECT, and DECOMPOSE relationships. The relations generated by these methods do not guarantee the results to be in good normal form. Unlike the relational model which is based on the mathematical theory of relations where methods (normalization theory) are used to determine whether a set of relations is a good design for a given database, it is very difficult to determine whether an ER diagram is the best representation for a given database. This is a major problem in the field of ER approach research. In this paper, a normal form for ER diagrams is defined based on the concepts in ER approach and the theory of dependencies in relational model [Codd 72, Bernstein 76, Fagin 77, Ling 81, Nicolas 78]. Unlike the definition of a normal form ER diagram given in [Chung 81], we allow composite attributes, multivalued attributes, and special types of relationship sets, namely, existence dependent, identifier dependent, ISA, UNION, INTERSECT, and DECOMPOSE relationships, to be included in a normal form ER diagram. We also ensure that all **basic** relationship sets and dependencies are represented in the diagram, that no redundant relationship sets exist in the diagram, and adopt the universal relation assumption [Atzeni 82] in the definition. A comprehensive algorithm which translates a normal form ER diagram to a set of relations is given in section 5. The concepts of a role name and ISA relationship in relational model [Ullman 80] are used in the translating process. We show that all relations generated are either in 3NF or 5NF. General guidelines and detailed steps for converting an ER diagram to a normal form ER diagrams are given in section 6.

2. The Relational Model

A relational database, consisting of several interrelated relations, was first introduced by Codd [Codd 70]. A relation is defined as follows: Given sets of atomic (non-decomposable) elements D1,D2,...,Dn (not necessarily distinct), R is a **first normal form ralation** (or simply **relation**) on these n sets if it is a set of ordered n-tuples <d1,d2,...,dn> such that di belongs to Di for i=1,2,...,n. Thus R is a subset of D1 x D2 x .... x Dn where x denotes the Cartesian product. D1,D2,...,Dn are called the domains of R. Rather than referencing each use of a domain by a position number, each is assigned a unique role name, called an **attribute** of R. For any tuple in T, the value for the attribute named B is referred

to as a B-value; for a set of attributes X={B1,B2,...,Bp}, the tuple's value for the attributes in X is referred to as an X-value; the values of the other attributes in the tuple are said to be associated with that X-value.

A set of attributes Y of R is said to be **functionally dependent** on a set of attributes X of R if each X-value in R has associated with it exactly one Y-value in R (at any time). This is denoted by X-->Y and is called a **functional dependency (FD)** of R; X and Y are termed the **left** and **right sides** of the dependency, respectively. A functional dependency X-->Y of R is said to be a **full dependency** of R (or Y is fully dependent on X) if there exists no proper subset X' of X such that Y is functionally dependent on X'. A set of attributes K of a relation R is said to be a **candidate key** (or simply a **key**) of R if all attributes of R are functional dependent on K and there exists no proper subset K' of K such that all attributes of R are functional dependent on K'. An attribute of R is called a **prime attribute** if it is contained in some key of R. All other attributes of R are called **non-prime attributes** of R.

Codd recognised that certain relations may contain some redundancy and the redundancy may cause some updating anomalies. One process that attempts to remove undesirable updating anomalies and redundancy from a relation is called **normalization**, which was originally defined in two stages [Codd 72]. A relation R is in **second normal form (2NF)** if every non-prime attribute of R is fully dependent on each candidate key of R. Let A and B be two non-identical sets of attributes of a relation R and d be an attribute of R which does not belong to the union of A and B, such that A-->B, B-->d, B-/->A (i.e. A is not functionally dependent on B), then we say that d is **transitively dependent** on A under R. A relation R is in **Codd third normal form (3NF)** if it is in second normal form and every non-prime attribute of R is not transitively dependent on each key of R [Codd 72]. A relation R is in **Boyce-Codd Normal Form (BCNF)** iff whenever there exists an non-trivial functional dependency in R, say A-->B (i.e. B is not a subset of A), then all attributes of R are also functionally dependent on A.

Let R(A,B,C) be a relation defined on three pairwise disjoint sets of attributes A, B, and C. A **multivalued dependency (MVD)** A-->>B or A-->>B|C is said to hold for the relation R if, whenever tuples <a,b,c> and <a,b',c'> are both in R, then tuples <a,b,c'> and <a,b'c> are also in R. The statement "A-->>B" is read as "attribute set B is multi-dependent on attribute set A", or, equivalently, "attribute set A multi-determines attribute set B". A multivalued dependency A-->>B|C is **non-trivial** if neither B nor C is an empty set of attributes. A non-trivial MVD A-->>B is said be a **strong MVD** if B does not functionally dependent on A. A relation R is said to be in **fourth normal form (4NF)** if, whenever there exists a non-trivial MVD in R, say A-->>B, then all attributes of R are also functionally dependent on A [Fagin 77]. Note that although there may be certain multivalued dependencies that we expect to hold in a projection relation of a relation R, we do not expect these dependencies to hold in R itself. Such a dependency is said to be an **embedded multivalued dependency (EMVD).**

Let R(A,B,C) be a relation defined on three pairwise disjoint sets of attributes A, B, and C. We say that R satisfies the **join dependency (JD)** "*(AB,BC,AC,)" if, whenever tuples <a,b,c'>, <a',b,c>, <a,b',c> appears in R, then <a,b,c> also appears in R. The concept of an embedded join dependency (EJD) can be defined similarly as the definition of EMVD. In general, relation R satisfies the JD *(X,Y,...,Z) iff it is the join of its projections on X,Y,...,Z, where X,Y,...,Z are subsets of the set of attributes of R. A relation R is in **fifth normal form (5NF)** iff every join dependency in R is implied by the candidate keys of R. [Nicolas 78, Date 81]. It has been shown that any 5NF relation is also in 4NF; any 4NF relation is also in BCNF; and any BCNF relation is also in 3NF.

3. The Entity-Relationship Approach

The entity-relationship approach for database schema design was proposed in [Chen 76]. Its uses the concepts of entity type and relationship set and incorporates some of the important semantic information about the real world. Information in the real world is recognized by either entities or relationships among entities. An entity is an object which exists in our minds and can be distinctly identified. For example, a particular house, a person, and a car, etc, are all entities. Entities can be classified into different entity types; each **entity type** (or **entity set**) contains a set of entities, each satisfying a set of predefined common properties. For example, we may group all the employees in a company as an entity type EMPLOYEE; We may group all the wards in a hospital as an entity type WARD.

Let E={E1,E2,...,En} be a set of entity types. A **relationship set** R over E is defined as a subset of the Cartesian product of the entity types which satisfies a set of predefined common properties, i.e.

$$R \subseteq \{<e1,e2,...en> \mid ei \in Ei,\ 1 <= i <= n\}$$

Each element of R is called a **relationship.** For example, let the two entity types DEPT and EMP be the set of all departments and the set of all employees in a company, respectively. We define a relationship set EMP-DEPT over these two entity types, i.e. EMP-DEPT is a subset of EMP x DEPT, and each relationship <e,d> in EMP-DEPT means the employee e belongs to the department d.

An entity type E (or a relationship set R) has attributes representing structural properties of E (or R). An **attribute** A is a mapping from E (or R) into a Cartesian product of n value sets, V1xV2x...xVn. If n>=2 then we call A a **composite attribute**, e.g. DATE is a composite attribute with the associated values sets {DAY,MONTH, YEAR}.

The mappings can be one of the four types: one-to-one, many-to-one, one-to-many, and many-to-many. If an attribute A is a one-to-many or many-to-many mapping from E (or R) into the associated value-type, then A is called a **multivalued attribute.** A minimal set of attributes K of an entity type E which defines a one-to-one mapping from E into the Cartesian product of the associated value sets of K is called a **key** of E. Note that there may exist more than one key for an entity type. If an entity type has more than one key, then we designate one of the keys as the **primary key** or **identifier** of the entity type. Let K be a set of identifiers of some entity types participating in a relationship set R. K is called a **key** of the relationship set R if there is a one-to-one mapping from R into the Cartesian Product of the associated value sets of K and no proper subset of K has such property. Note that a relationship set may have more than one key. One of the keys of a relationship set is designated as the **primary key** or **identifier** of the relationship set.

A relationship set R can also be described by the set of entity types involved and a set of functional dependencies defined among the identifiers of the entity types involved in R. In the entity-relationship approach, recursive relationship sets are allowed. For example, we can defined a relationship set MARRIED which is a subset of PERSON x PERSON, where PERSON is an entity type. We can also have more than one relationship sets between the same entity types. For example we can define two relationship sets namely ATTENDING-PHYSICIAN and CONSULTING-PHYSICIAN between the two entity types DOCTOR and PATIENT. Given two relationship sets R1 and R2 defined on the two sets of entity types S1={A1,...,Ai,B1,...,Bj} and S2={B1,...,Bj,C1,...Ck} where B1,...,Bj are entity types participating in both relationship sets. The **join** of R1 and R2, denoted by R1 * R2, is defined as follows:

R1 * R2 ={<a1,...,ai,b1,...,bj,c1,...,ck> | <a1,...,ai,b1,...,bj>∈S1 and <b1,...,bj,c1,...,ck>∈S2}

Note that there are entity types in which entities cannot be identified by the values of its own attributes, but has to be identified by its relationship with other entities. Such an entity type is called a **weak entity type** and the relationship set which is used to identify the entity is said to be an **identifier dependent relationship set.** If the existence of an entity in one entity type depends upon the existence of a specific entity in another entity type, such a relationship set and entity type are called **existence dependent relationship set** and **weak entity type.** Clearly an identifier dependent relationship set is also an existence dependent relationship set. An entity type which is not a weak entity type is called a **regular entity type.** A relationship set which involves weak entity type(s) is called a **weak relationship set.** A relationship set which does not involve weak entity type(s) is called a **regular relationship set.** If an entity in one entity type E1 is also in another entity type E2, we call the relationship between these two entity types E1 and E2 an **ISA relationship set.** Clearly the identifiers of the two entity types of an ISA relationship set must be defined on the same value set. If an entity type is equal to the union (or intersection) of some other entity types, such a relationship is called a UNION (or INTERSECT) relationship. If an entity type can be partitioned or decomposed into several other entity types, such a relationship is called a DECOMPOSE relationship. Note that all the special relationship sets such as existence dependent, identifier dependent, ISA, UNION, INTERECT, DECOMPOSE relationships are weak relationship sets and with no attributes associated.

The structure of a database organized according to the entity-relationship approach can be represented by a diagrammatic technique called an entity-relationsip diagram (ERD) [Chen 76]. A regular entity type is represented by a rectangle, labelled by its name. A weak entity type is represented by a double-rectangle, labelled by its name. A relationship set is represented by a diamond, labelled by its name. For existence dependent and identifier dependent relationship sets, the diamond boxes are also labelled with the symbols E and ID respectively. For ISA, UNION, INTERSECT, and DECOMPOSE relationship sets, the corresponding diamond boxes are labelled by ISA, UNION, INTERSECT, and DECOMPOSE respectively.

Arcs are used to connect a relationship set and the entity types which participate in the relationship set. For existence dependent and identifier dependent relationship sets, arrows are used to join the diamond boxes to the weak entity types instead of arcs. Arrows are also used in the ISA, UNION, INTERSECT, and DECOMPOSE relationship sets. An arc which joins a relationship set and an entity type may be labelled by a role name if required. If a relationship set involves only two entity types, and the relationship is a one-to-one, one-to-many, many-to-one, or many-to-many mapping, then the two arcs which join the two entity types and the relationship set are labelled by '1' and '1', '1' and 'n', 'n' and '1', or 'n' and 'm' respectively. Sometimes, we do not label the arcs of a many-to-many relationship set in order to simplify the ER diagram. It is difficult to represent all the functional dependencies among all the entity types in a relationship set which involves more than two entity types.

Figures 3.1 to 3.4 show 4 different relationship sets which involves 3 entity types, namely A, B, and C. The relationship set in Figure 3.1 shows no functional dependency defined amongst the three entity type. Figure 3.2 depicts a functional dependency AB-->C, where A, B, C are used as the identifiers of the entity types A, B, and C respectively. Figure 3.3 represents the functional dependency A-->BC and Figure 3.4 represents the functional dependencies A-->BC, B-->AC, and C-->AB. Note that it is very difficult to represent a relationship set which involves three entity types A, B, and C with

functional dependencies AB-->C and AC-->B. We can describe such a relationship set by Figure 3.2 with a constraint (functional dependency) AC-->B.

Since we do not need to use the concept of value sets of attributes in our discussion, we will not represent value sets in ER diagrams. Attributes of entity types and relationship sets are represented by circles, labelled by attribute names. Arrows are used to connect entity types or relationship sets and their many-to-one attributes. Double sided arrows, <->, are used to connect entity types or relationship sets and their one-to-one attributes. Double arrows, <->> and -->>, are used to connect entity types or relationship set and their one-to-many and many-to-many multivalued attributes respectively. A key of an entity type which consist of more than one attribute are indicated by joining the attributes of the key with a line. Figure 3.5 depicts an entity type E with attributes A, B, C, D, E, F. A and BC are keys of E; D, E, F are many-to-one, many-to-many and one-to-many attributes respectively.

Figure 3.6 shows a ER diagram of a EMP-PROJ database. Work is a relationship set defining a many-to-many mapping between the two entity types EMPLOYEE and PROJECT. EMPLOYEE has three single valued attributes namely, EMP#, NAME and SALARY, and one multivalued attribute QUALIFICATION. EMP# is its identifier. PROJECT has 3 single valued attributes namely, PROJ#, P-Name, and BUDGET. PROJ# is the identifier. The relationship set WORK has a single valued attribute PROGRESS.

4. A Normal Form For ER Diagrams

In this section, we define what is meant by a *normal form ER diagram (NF-ER diagram)*. The objectives for defining such a normal form for ER diagrams are :

(1) to capture and preserve all the semantics of the real world of a database which can be expressed in term of functional, multivalued, and join dependencies, by representing them explicitly in the ER diagram.

(2) to ensure that all the relationships represented in the ER diagram are non-redundant, i.e. none of the relationships can be derived from other relationships.

(3) to ensure that all the relations translated from the ER diagram are in good normal form, either in 3NF or 5NF.

A ER normal form is defined in [Chung 81]. It is a very restrictive definition and has several drawbacks. First, it does not allow the existence of multivalued attributes; all multivalued attributes have to be converted to relationship sets. This defeats the objective stated in (1) above, namely to preserve the semantics of the real world in a ER diagram. Second, it does not allow the existence of non-trivial functional dependencies whose left sides do not include one identifier of some entity type or relationship set. This implies that no one-to-one or one-to-many attributes are allowed for relationship sets. This also implies that all relations translated are in BCNF. This contradicts the fact that there are 3NF relations which cannot be decomposed into BCNF if we require a database to cover the given set of functional dependencies. Third, the concept of composite attribute is not included in the paper [Chung 81]. In this section, we enhance the definition of a ER-NF [Chung 81] by allowing the existence of multivalued attributes, composite attributes, and one-to-one and one-to-many attributes for relationship sets in order to remove the above mentioned drawbacks. First, we define what are meant by entity type normal form and relationship set normal form. Then we give a precise definition for an ER normal form diagram.

Definition 4.1 Let E be an entity type and K be its identifier. The **set of basic dependencies of E**, BD(E), is defined as follows :

(1) For each many-to-one attribute A of E, K-->A is a FD in BD(E).
(2) For each one-to-many multivalued attribute A of E, A-->K is a FD in BD(E).
(3) For each one-to-many and many-to-many multivalued attribute A of E, K-->>A is a MVD in BD(E).
(4) For each key K1 of E which is not the identifier of E, K-->K1 and K1-->K are FDs in BD(E).
(5) No other FDs or MVDs are in BD(E).

Informally, the set of basic dependencies of an entity type E are the functional dependencies and multivalued dependencies of E which are explicitly shown in the ER diagram.

Definition 4.2 An entity type E of a ER diagram is said to be in **entity normal form (E-NF)** if all functional dependencies and multivalued dependencies which only involve attributes of E , can be derived (or implied) from the set of basic dependencies of E, BD(E),by using the Armstrong's axioms for functional dependencies and inference rules for multivalued dependencies [Beeri 77].

In Figure 4.1, the set of basic dependencies of the entity type EMPLOYEE consists of the following dependencies :

E# -->SSN, NAME, SEX
SSN --> E#
E# -->>SKILL
E# -->>DEGREE

where E# is the identifier of EMPLOYEE and SSN is a key of EMPLOYEE.

EMPLOYEE is in E-NF because there is no other dependencies which only involve attributes of E and cannot be derived from the set of basic dependencies of EMPLOYEE.

In figure 4.2 the entity type SUPPLIER has a

many-to-one composite attribute ADDRESS which consists of three attributes, namely, CITY, STREET, and ZIP. The entity type SUPPLIER is not in E-NF because the two functional dependencies : CITY,STREET-->ZIP and ZIP-->CITY are not in the set of basic dependencies of the entity type. A method to convert such a non E-NF entity type to a E-NF will be presented in section 5.

Lemma 4.1 Let E be an entity type. If E is in E-NF, then the following statements hold.

(1) Each single valued attribute A of E is fully dependent on each key of E which does not contain A, and on each one-to-many attribute of E.
(2) All components of any composite single valued attribute A of E are fully dependent on each key of E which does not contain B.
(3) There is no non-trivial functional dependencies defined among components of any composite attribute of E.
(4) For each one-to-many attribute A of E and for each many-to-many attribute B of E, A-->>B is a strong MVD.
(5) For each key K of E and for each multivalued attribute A of E, K-->>A is a strong MVD.
(6) No multivalued attribute of E is multi-dependent on a part of a key of E.
(7) No component of a composite multivalued attribute of E is multi-dependent on the identifier of E.

Lemma 4.2 An entity type E of an ER diagram is in E-NF if and only if it satisfies the following conditions:

(1) Any non-trivial canonical form full dependency (i.e. the right side of the FFD is a single attribute) A-->B which only involves attributes and components of composite attributes of E implies
  (a) A is a key of E or A is a one-to-many attribute of E, and
  (b) B is a single valued attribute or B is a component of a composite single valued attribute of E.

(2) Any strong MVD A-->>B which only involves attributes and components of composite attributes of E and in which B is not multi-dependent on any proper subset of A and no proper subset of B is multi-dependent on A, implies
  (a) A is a key of E or A is a one-to-many attribute of E , and
  (b) B is a multivalued attribute of E.

Lemmas 4.1 and 4.2 can be proved directly using the definition of an E-NF entity type, Armstrong's axioms for functional dependencies, and inference rules for multivalued dependencies [Beeri 77,Ullman 80].

Definition 4.3 Let R be a relationship set with identifier K and F be the associated set of functional dependencies which only involve the identifiers of the set of entity types participating in R. The **set of basic dependencies of R**, denoted by BD(R),is defined as follows:
(1) For each one-to-one attribute A of R, K-->A and A-->K are FDs in BD(R).
(2) For each many-to-one attribute A of R, K-->A is a FD in BD(R).
(3) For each one-to-many multivalued attribute A of R, A-->K is a FD in BD(R).
(4) For each one-to-many and many-to-many multivalued attribute A of R, K-->>A is a MVD in BD(R).
(5) Let A-->B be a full dependency in F such that A is a set of identifiers of entity types participating in R , and B is an identifier of some entity type participating in R. If A is a key of R or B is part of a key of R then A-->B is a FD in BD(R).
(6) No other FDs or MVDs are in BD(R).

Definition 4.4 A relationship set R of a ER diagram is said to be in **relationship normal form (R-NF)** if all functional dependencies and multivalued dependencies which only involve attributes of R and identifiers of entity types participating in R are implied by the set of dependencies of R, i.e. BD(R).

Informally speaking, the set of basic dependencies of a relationship set R includes those functional dependencies and multivalued dependencies which involves attributes of R and are explicitly shown in the ER diagram. Item 5 of the definition of the basic set of dependencies of a relationship set is to ensure that all relations which correspond to a R-NF relationship set are at least in 3NF.

Figure 4.3 shows a relationship set R defined on three entity types A, B, and C with attributes E, F, and G. If the set of functional dependencies, which only involves identifiers of the entity types of R, only consists of the functional dependency A-->BC, where A, B, and C are used as the identifiers of the entity types A,B and C respectively, then the set of basic dependencies of R contains the following dependencies:

A-->EG, G-->A, A-->>F, A-->BC.

Note that R is R-NF if there is no FDs or MVDs which involves the attributes E, F, G and cannot be derived from DB(R). Note that if R is associated with another FD B-->C, then R is not in R-NF since B-->C cannot be derived from DB(R).

Lemma 4.3 Let R be a regular relationship set and BD(R) be the set of basic dependencies of R. If R is in R-NF, then the following statements hold:
(1) All many-to-one attributes of R are fully dependent on each key, each one-to-one attribute and each one-to-many attribute of R.
(2) All components of any composite single valued attribute A of R are fully dependent on each key of R, each one-to-many attributes and each one-to-one attribute of R which is not equal to A.
(3) There is no non-trivial functional dependencies defined among components of any composite attribute of R.
(4) For each one-to-many (or one-to-one) attribute A of R and for each many-to-many attribute B of R, A-->>B is a strong MVD.

(5) For each key K of R and for each multivalued attribute A of R, K-->>A is a strong MVD.
(6) No multivalued attribute of R is multi-dependent on a part of a key of E.
(7) No component of a composite multivalued attribute of R is multi-dependent on the identifier of E.

This lemma is similar to lemma 4.1.

Lemma 4.4 Let R be a regular relationship set and BD(R) be the set of basic dependencies of R. R is in R-NF if and only if it satisfies the following conditions :

(1) Any non-trivial full dependency A-->B which only involves attributes, components of composite attributes, and identifiers of entity types participating in R, and in which B is an attribute or a component of a composite attribute of R, implies
(a) A is a key of R, or A is a one-to-many or one-to-one attribute of R, and
(b) B is a single valued attribute or a component of a composite single valued attribute of R.

(2) Any non-trivial full dependency A-->B which only involves attributes of identifiers of entity types participating in R and in which B is a single attribute, implies either A is a key of R or B is part of a key of R.

(3) Any strong MVD A-->>B which only involves attributes, components of composite attributes, and identifiers of entity types participating in R and in which B is not multi-dependent on any proper subset of A and no proper subset of B is multi-dependent on A, implies
(a) A is a key of R, or A is a one-to-one or one-to-many attribute of R, and
(b) B is a multivalued attribute of R.

Lemma 4.4 is similar to lemma 4.2 except the extra condition (2) in lemma 4.4.

Definition 4.5 Let D be an entity relationship diagram. The **set of basic dependencies of D**, denoted by BD(D), is defined as the union of the sets of dependencies of all entity types of D and the sets of basic dependencies of all relationship sets of D.

Definition 4.6 An entity relationship diagram D is in **normal form** (ER-NF) if it satisfies the following conditions :

(1) All attribute names are distinct and of different semantics.
(2) Every entity type in the ER diagram is in E-NF.
(3) Every relationship set in the ER diagram is in R-NF.
(4) All relationships and dependencies are implied by the set of basic dependencies of D.
(5) Every relationship set R with no associated attribute defined on it, satisfies the following conditions :
(a) R is not equal to the join of any two other relationship sets, and
(b) R is not equal to the join of any three other relationship sets

Informally speaking, the first condition of the definition for a normal form ER diagram is required in order to conform to the universal relation assumption. The second condition of the definition ensures that all relations generated for all entity types are in 5NF. The fourth condition of the definition ensures that the ER diagram has captured all the relationships and dependencies of the given database. The second and fifth conditions ensure that all relations generated for all regular relationship sets are either in 3NF or 5NF and that there is no relation in BCNF but not in 4NF or 5NF. In section 5, we will show that all relations generated for a normal form ER diagram are in good normal form.

5. Translation of a Normal Form ER diagram to a relational database

In this section, an algorithm is given to translate a normal form ER diagram to a set of relations. Such a normal form ER diagram may consist of composite attributes, recursive relationship sets, weak entity types, and special types of relationship sets, such as existence dependent, identifier dependent, ISA, UNION, INTERSECT, and DECOMPOSE relationship sets. All the relations produced by the algorithm are either in 3NF or 5NF. Since we adopt the concept of covering the given set of functional dependencies in the normalization process, some of the relations produced are in 3NF and cannot be decomposed into BCNF. Since we also adopt the concept of the universal relation assumption, role names are required for some entity types participating in some relationship sets and the concept of ISA in relational model is used to link the role names and the identifiers of the corresponding entity types. In fact, this algorithm can also be used to translate any ER diagram, but the relations produced may not be in good normal form. It is more comprehensive and precise than any other translation methods given in [Chung81, Dumpala 81, Hwang 81, Jajodia 83, Melkanoff 80, Morgenstern 81, Ng 80, Wong 80].

Algorithm 1: Translate a normal form ER diagram to a set of relations.

Step 1: {Assign role names to certain arcs in order to conform to the universal relation assumption }

For each cycle in the ER diagram, we assign each arc which connects an entity type and a relationship set in the cycle, a unique role name if there is no role name assigned.

{Here, a cycle in an ER diagram is defined as a cycle in the corresponding graph of the ER diagram in which all entity types and regular relationship sets (i.e. except those special relationships: existence dependent, identifier dependent, ISA, UNION, INTERSECT, and DECOMPOSE relationship sets) are nodes in the graph and arcs which connect

entity types and relationship sets are edges in the graph.}

Step 2: {Assign identifiers for entity types involved in special relationship such as ISA, UNION, INTERSECT, DECOMPOSE}

(1) If entity types A and B are involved in a ISA relationship such that A ISA B, and K is the identifier of B, then we assign a unique identifier name say K1, for A and record the constraint: K1 ISA K.

(2) If entity types A and B are involved in a UNION (or DECOMPOSE) relationship such that A is a union of (or A can be decomposed to) B and some other entity types, and K is the identifier of A, then we assign a unique identitifer name, say K1, for B and record the constraint: K1 ISA K.

(3) If entity types A and B are involved in an INTERSECT relationship such that A is the intersection of B and some other entity types, and K is the identifier of B, then we assign a unique identifier name, say K1 for A and record the constraint: K1 ISA K.

Step 3: {Generate relations for each entity type}
For each entity type E (either weak or regular entity type) in the ER diagram, we construct the following relations for E.

(1) All the keys of E and all the many-to-one single valued attributes form a relation. The keys and primary key of this relation are the keys and identifier of E respectively. We call this relation a type E-1 relation.

(2) Each many-to-many multivalved attribute and the identifier of E form form a relation. This relation is an all key relation, i.e. all the attributes of the relation form the key of the relation. We call this relation a type E-2 relation.

(3) Each one-to-many attribute and the identifier of E form a relation and the key of this relation is the one-to-many attribute. We call this relation a type E-3 relation.

Note that we replace all composite attributes of E by their components in all the relations generated.

Step 4: {Translate each regular relationship set to relations}

For each regular set R, we construct the following relations for R:

(1) All the identifiers of the entity types participating in R and all the many-to-one and one-to-one attributes form a relation. The keys and and the one-to-one attributes of R are keys of this generated relation. The primary key of this relation is the identifier of R. If A-->B is a non-trivial canonical form full dependency in the set of dependencies of R and A is not a key of R, then we record A-->B as a constraint of this relation. We call this relation a type R-1 relation.

(2) Each many-to-many attribute of R and the identifier of R form an all key relation. We call this relation a type R-2 relation.

(3) Each one-to-many attribute A of R and the identifier of R form a relation and A is the key of the relation. We call this relation a type R-3 relation.

Note that if an arc which joins an entity type (with identifier K) and R is labelled by a role name, say N in step 1, we replace the occurrences of K in relations generated and the functional dependencies constraints generated (if any by (1) of this step) by the role name N, and record the constraint: N ISA K. We also replace all composite attributes in the relations by their components. Note that there is no relation generated for any special relationships. Instead, constraints are generated as described in step 2 of the algorithm.

Example 5.1 Figure 5.1 is a normal-form ER diagram which has 5 entity types, 3 regular relationship sets, and one ISA relatioship. The identifier C# of entity type C is assigned in step 2 of algorithm 1. The roles name AX, AY, BX, BY are assigned by step 1. Assume that the relationship set R3 has 2 functional depdendencies, namely E#,D# --> A# and A# --> D#. Clearly the keys of R3 are {E#,D#} and {A#,E#} and we designate {E#,D#} as the identifier of R3. The set of relations generated by algorithm 1 consists of the followings

(1) relations of entity type A:
AE1(A# , A1 , A2)
AE3(A3 , A#)

(2) relations of entity type B:
BE2(B# , B1)
BE2'(B# , B2)

(3) relation of entity type C:
CE1(C# , C1)
constraint: C# ISA B#

(4) relation of entity type D:
DE1(D# , D1 , D2)

Note that D1 and D2 are components of a composite attribute.

(5) relation of entity type E:
EE1(E# , E1 , E2 , E3)
where E# is the primary key.

(6) relation for relationship set R1:
R1R1(AX, BX)
constraints: AX ISA A# , BX ISA B#

(7) relations for relationship set R2:
R2R1(AY, BY, S2)
R2R2(AY, BY, S3)
constraints: AX ISA A# , AY ISA A#, BX ISA B#, BY ISA B#

(8) relations for relationship set R3
R3R1(A#, D#, E#, S1)

Keys are {D#,E#},{A#,E#} and primary key is {D#,E#}
Constraint: A# --> D#

Note that relation R3R1 is in 3NF but not BCNF. There is no relation corresponding to relationship ISA; it is translated to the constraint: C# ISA B#.

It is obvious that the relations and constraints generated by algorithm 1 cover the given set of functional dependencies and relationship sets of the given normal form ER diagram. In the following two theorems, we show that all these relations are either in 3NF or 5NF.

Theorem 1: The relations generated for each of the entity types of a normal form ER diagram by step 3 of algorithm 1 are in 5NF.

Proof: Let E be an entity type of the given normal form ER diagram. From algorithm 1, there are three types of relations generated for E.

(1) Let S1 be a type E-1 relation of E, i.e. S1 consists of all the keys and many-to-one single valued attribute of E. By lemma 4.1 each single valued attribute is fully dependent on each key of S1, therefore S1 is in 2NF. Let A-->B be a non-trivial canonical form full dependency in S1. By lemma 4.2, A is a key of E. Hence S1 is in BCNF. Also, since there is no strong multivalued or join dependency involved in S1, S1 is also in 4NF and 5NF.

(2) Let S2 be a type E-2 relation of E, i.e. S2 consists of one many-to-many multivalued attribute and the identifier of E. Since E is in E-NF, by lemma 4.2, there is no non-trivial function dependency or non-trivial multivalued dependency in S2. Hence S2 in 4NF. Clearly there is no join dependency in S2, therefore S2 is also in 5NF.
(3) Let S3 be a type E-3 relation of E, i.e. S3 consists of an one-to-many attribute A and the identifier of E. Clearly A is a key of S3. Since E is in E-NF, by lemma 4.2 there is no non-trivial full dependency with the left side not equal to A in S3. Hence S3 is in BCNF. Clearly there is no MVD or JD in S3, therefore S3 is also in 4NF and 5NF.

From the above discussion, we have proved that all relations generated by step 3 of Algorithm 1 are in 5NF.

Theorem 2: The relations generated for each of the relationship sets of a normal form ER diagram by step 4 of algorithm 1 are either in 3NF or 5NF.

Proof: As we have discussed before, we do not need to generate relations for the special relationships such as existence dependency, identifier dependency, ISA, UNION, INTERSECT, and DECOMPOSE relationships. Let R be a relationship set other than the special relationships and BD(R) be the set of basic dependencies of R. We have the following three cases:

Case 1: R has no associated attribute and there is no non-trivial functional dependency defined among the identifiers of the entity types participating in R.

Clearly algorithm 1 only generates a type R-1 relation, say S, for R which consists of all the identifiers of the entity types participating in R. Clearly S is an all key relation. By the definition of a normal form ER diagram, R is not equal to the join of any two or three other relationship sets and all relationships are implied by the set of basic dependencies of D. Therefore there is no non-trivial multivalued dependency or join dependency in S. Hence S is in 5NF.

Case 2: R has no associated attribute and there are non-trivial functional dependencies defined among the identifiers of the entity types participating in R, ie. BD(R) is not empty.

Clearly algorithm 1 only generates a type R-1 relation, say S, for R which consists of all the identifiers of the entity types participating in R. Since R is in R-NF, by lemma 4.3, all the non-prime attributes are fully dependent on each key of R, and therefore S is in 3NF. Note that if there is no non-trivial full dependency, say A-->B in BD(R) such that A is not a key of R then there is no non-trivial full dependency A-->B in S such that A is not a key S. Therefore S is in BCNF. By item 5 of the definition of a normal form ER diagram, there is no non-trivial strong multivalued dependencies or join dependency in S, therefore S is 5NF.

Case 3: R has associated attributes. In this case, we have to consider the three types of relations possibly generated for R.

(1) Let S1 be a type R-1 relation of R, i.e. S1 consists of the identifiers (or role names) of the entity types participating in R and all the many-to-one and one-to-one single valued attributes of R. If R has no single valued attribute associated with it, then S1 is either the same as the relation in case 1 or case 2. Here we assume that S has some single valued attributes associated with it. Clearly the keys of S1 are the keys of the relationship set R and all the one-to-one attributes of R. Since R is in R-NF, by lemma 4.3, all the many-to-one attributes of R are fully dependent on each key of R and each one-to-one attribute of R. This means that all non-prime attributes of S1 are fully dependent on each key of S1 and therefore S1 is in 2NF. By lemma 4.4, there is no non-trivial full dependency A-->B in S1 such that B is a non-prime attribute and A is not a key of S1. Therefore there is no transitive dependency in S1. Hence S1 is in the 3NF. Using the similar argument stated in case 2, if there is no non-trivial full dependency A-->B in BD(R) such that A is not a key of R then S1 is in 5NF; otherwise S1 is not in BCNF but in 3NF.

(2) Let S2 be a type R-2 relation of R, i.e. S2 consists of a many-to-many attribute and the identifier of R. Since R is in R-NF, by lemma 4.4, there is no non-trivial functional dependency

or strong multivalued dependency in S2. Hence S2 is in 4NF. Clearly there is no join dependency in S2. Therefore S2 is also in 5NF.

(3) Let S3 be a type R-3 relation of R, i.e. S3 consists of an one-to-many attribute A and identifier of R. Clearly A is a key of S3. Since R is in R-NF, by lemma 4.4, there is no non-trivial full dependency with the left side not equal to A in S3. Hence S3 is in BCNF. Clearly there is no MVD or JD in S3, therefore S3 is also in 4NF and 5NF.

From the above discussion, we have shown that all relations generated by step 4 of algorithm 1 are either in 3NF or 5NF.

Theorem 3: All the relations generated for a normal form ER diagram by algorithm 1 are either in 3NF or 5NF.

Proof: The proof follows directly from theorems 1 and 2.

6. Converting an ER diagram to a normal form ER diagram

In this section, guidelines and steps for converting an ER diagram to a normal form ER diagram are present. We will not discuss each step in detail. A detailed algorithm for the converting process and the proof for the correctness of the algorithm will be presented in another paper by the author. Basically, the converting process is based on the four lemmas described in section 4 and the definition of a normal form ER diagram. The basic steps for the converting process are as follows:

Step 1: Ensure that all attribute names are distinct and of different semantics.

Step 2: Convert any non E-NF entity type to E-NF. We remove all undesirable functional dependencies and/or multivalued dependencies by introducing new entity types and relationship sets.

Step 3: Convert any non R-NF relationship set to R-NF. We remove all undesirable functional dependencies, multivalued dependencies, and/or join dependencies either by introducing new entity types and relationship sets or by splitting the relationship set into smaller ones.

Step 4: Remove those relationship sets which have no associated attributes and is equal to the join of two or three other relationship sets.

Below we discuss each step in more detail.

6.1 STEP 1

Step 1 is necessary in order to ensure that the translated relations conform to the universal relation assumption. To ensure that all attribute names are unique is trivial. Now if there are two attributes, say A and B, are of the same semantic meaning (ie. refer to the same thing). There are 4 possible cases:

(1) A and B are attributes of two entity types,
(2) A and B are attributes of two relationship sets,
(3) A is an attribute of an entity type, and B is an attribute of a relationship type,
(4) A is an attribute of a relationship set and B is an attribute of an entity type.

Let us consider the first case and assume that A and B are attributes of entity type E1 and E2 respectively. Note A and B can be a one-to-many, many-to-many, or many-to-one attribute, a key or part of a key, or even the identifier or part of the identifier of the entity types E1 and E2 respectively. We can't have a single converting rule for all the cases, although the main idea is the same: replace one attribute by another one and create a new entity type for them if necessary. Figures 6.1 and 6.2 show two of the cases. Only relevant information are shown in the figures.

6.2 Convert a non E-NF entity type to E-NF

The results in lemmas 4.1 and 4.2 are used to test whether a given entity type E is in E-NF. This implies we need to test the following conditions:

(1) Each single valued attribute A is fully dependent on each key of E which does not contain A.
(2) All components of any composite Single valued attribute A are fully dependent on each key of E which does not contain A.
(3) There is no non-trivial functional dependencies defined among components of any composite attribute of E.
(4) No multivalued attribute of E is multi-dependent on a part of a key of E.
(5) No component of a composite many-to-many attribute of E is multi-dependent on the identifier of E.
(6) No component of a composite many-to-one attribute determines the identifier of E.
(7) Condition 1 of lemma 4.2.

If an entity type does not satisfy any of the above conditions, we have to remove some of the attributes involved and create some new entity types and relationship sets. Figures 6.3 to 6.5 show three of the many possible cases.

6.3 Convert a non R-NF relationship set to R-NF

The results in lemmas 4.3 and 4.4 are used to test whether a regular relationship set R is in R-NF, i.e. we need to test the following conditions:

(1) Each single valued attribute of R is fully dependent on **each** key of R
(2) All components of any composite single valued attribute are fully dependent on each key of R
(3) There is no non-trivial functional dependencies defined among components of any composite attribute of R
(4) No multivalued attribute of R is multi-dependent on a part of a key of R
(5) No component of a composite multivalued attribute of R is multi-dependent on the identifier of R
(6) No component of a composite many-to-one or

one-to-one attribute determines the identifier of E
(7) Condition 1 of lemma 4.4
(8) Condition 2 of lemma 4.4

Note that the first seven conditions are similar to the seven conditions in section 6.2. If a relationship set does not satisfy any of the above conditions, we have to remove some of the attributes involved and create some new entity types and relationship sets, or split the relationship set into two or more smaller relationship sets. Figures 6.6 to 6.8 show three of the many possile cases.

6.4 Remove redundant relationships

If a relationship set R which has no associated attribute and is equal to the join of two or three other relationship sets, then clearly the relation generated for R will not be in 4NF or 5NF. We have to remove this type of relationship sets from ER diagrams. In fact, given an ER diagram and a set of dependencies associated with the ER diagram, we can only achieve the following:

> Given a relationship set R with no associated attribute, we can detect the existence of two or three relationship sets such that the set of entity types participating in R is equal to the union of the entity types participating in the other two or three relationship sets.

To detect whether the join of two or three relationship sets is equal to R or not, we require more information about the semantic meaning of the relationship sets which can be provided by the database designer or database owner.

7. Conclusion

In this paper, we first defined the concept of a basic set of dependencies of an entity type and a relationship set, a normal form entity type, and a normal form relationship set. We then defined what is meant by a normal form ER diagram. A normal form ER diagram may consist of composite attributes, multivalued attributes, weak entity types, and special relationships such as existence dependent, identifier dependent, ISA, UNION, INTERSECT, DECOMPOSE relationships. The definition for a normal form ER diagram gives the necessary and sufficient condition for ensuring all relations of the corresponding ER diagram are either in 3NF or 5NF. An algorithm was given to translate a normal form ER diagram to a set of relations which conforms to the universal relation assumption, and a set of constraints which consists of some functional dependencies and a set of ISA relationships. We have proved that all relations are in 3NF or 5NF. We also gave a method to convert an ER diagram to a normal form ER diagram. Further research to simplify and reduce the number of cases to be considered in order to convert a ER diagram to normal form is required.

References

[Atzeni 82] P Atzeni and D S Parker Jr, Assumptions in Relational Database Theory, ACM Symposium on Principles of Database Systems, Los Angeles 1982.

[Beeri 77] C Beeri, R Fagin, and J H Howard, A Complete Axiomatization for functional and multivalued dependencies in database relations, 3rd ACM SIGMOD Int. Conf. Management of Data, 1977.

[Bernstein 76] P A Bernstein, Synthesizing third normal form relations from functional dependencies, ACM Transaction on Database Systems 1 4, 1976.

[Chen 76] P P Chen, The entity-relationship model: Toward a unified view of data, ACM Transaction on Database Systems 1 1, 1976.

[Codd 70] E F Codd, A Relational Model of Data for Large Shared Data Banks, CACM 13, 6, June 1970, 377-387

[Codd 72] E F Codd, Further Normalization of the Data Base Relational Model, Data Base Systems, edit by Randell Rustin, Prentice Hall 1972.

[Chung 81] Ilchoo Chung, Fumio Nakamura, P P Chen, A Decomposition of Relations Using the Entity Relationship Approach, Entity-Relationship Approach to Information Modeling and Analysis, P P Chen (ed.), ER Institute, 1981.

[Date 81] C J Date, An Introduction to Database Systems, 3rd edition, Addison-Wesley, 1981.

[Dumpala 81] Surya R Dumpala and Sudhir K Arora, Schema Translation using the Entity-Relationship Approach, Entity-Relationship Approach to Information Modeling and Analysis, P P Chen (ed.), ER Institute, 1981.

[Fagin 77] R Fagin, Multivalued Dependencies and a new Normal Form for Relational Databases, ACM Transaction on Database Systems 2 3, Sept 1977.

[Hwang 81] Hai-Yann Hwang and Umeshwar Dayal, Using The Entity- Relationship Model for Implementing Multi-Model Database Systems, Entity-Relationship Approach to Information Modeling and Analysis, P P Chen (ed.), ER Institute, 1981.

[Jajodia 83] Sushil Jajodia and Peter A Ng, On Representation of Relational Structures By Entity-Relationship Diagrams, Entity-Relational Approach to Software Engineering, C G Davis, S Jajodia, P A Ng and R T Yeh (eds) North Holland, 1983.

[Ling 81] T W Ling, F W Tompa and T Kameda, An improved third normal for relational databases, ACM Transaction on Database Systems 6 2, June 1981.

[Melkanoff 80] Michel A Melkanoff and Carlo Zaniolo, Decompositon of Relations and Synthesis of Entity-Relationship Diagrams, Entity-Relationship Approach to Sytems Analysis and Design, P P Chen (ed) North-Holland Pub Co., 1980

[Morgenstern 81] Matthew Morgenstern, A Unifying Approach for Conceptual Schema to Support Multiple Data Models, Entity-Relationship Approach to Information Modelling and Analysis, P P Chen (ed), ER Institue, 1981

[Ng 80] Peter A Ng and Jean F Paul, A Formal Definition of Entity-Relationship Models, Entity-Relationship Approach to Systems Analysis and Design, P P Chen (ed), North-Holland Pub. Co., 1980.

[Nicolas 78] J M Nicolas, Mutual dependencies and some results on undecomposable relations, Proc 4th International conference on Very Large Data Bases, 1978.

[Ullman 80] J D Ullman, Principles of Database Systems, Computer Science Press, 1980.

[Wong 80] Eugene Wong and Randy H Katz, Logical Design and Schema Conversion for Relational and DBTG Databases, Entity-Relationship Approach to Systems Analysis and Design, P P Chen (ed), North-Holland Pub. Co., 1980.

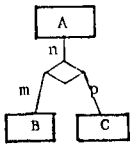

Figure 3.1

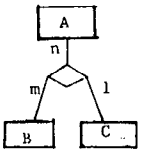

Figure 3.2

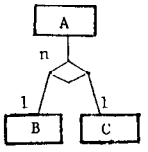

Figure 3.3

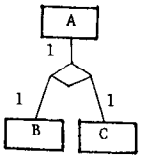

Figure 3.4

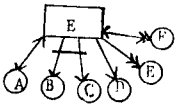

Figure 3.5 An entity type with its attributes

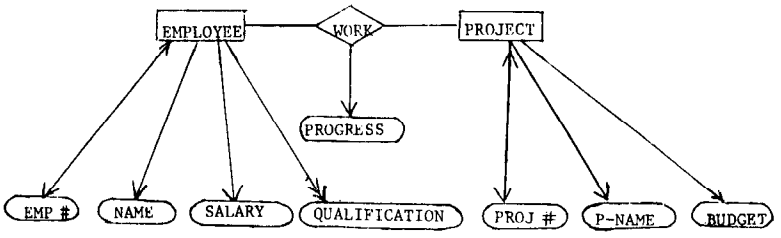

Figure 3.6

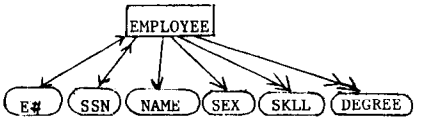

Figure 4.1 An entity type in E-NF

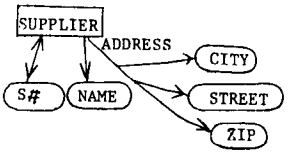

Figure 4.2 An entity type not in E-NF

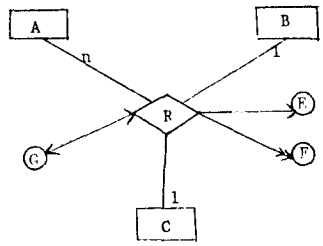

Figure 4.3

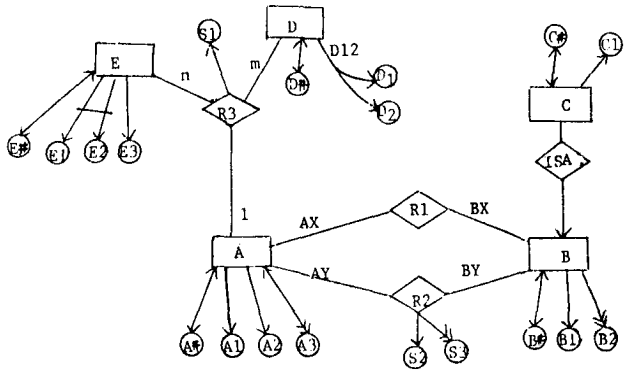

Figure 5.1

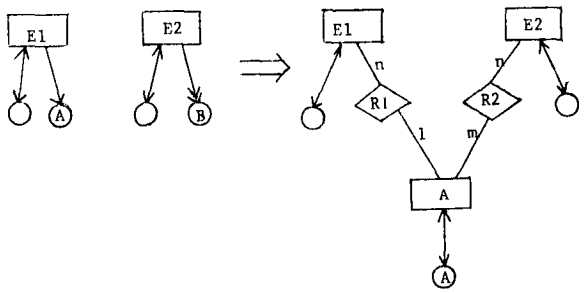

Figure 6.1 A and B are of the same semantic

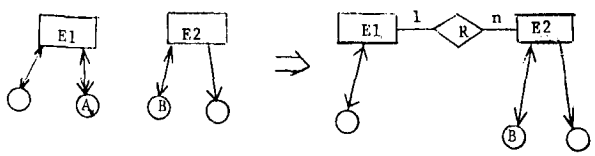

Figure 6.2 A and B are of the same semantic

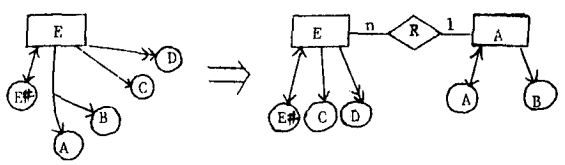

Figure 6.3 A ⟶ B in a composite attribute of E

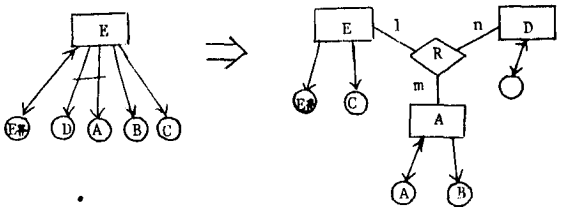

Figure 6.4 A ⟶ B in E and A is part of a key of E. Constraint generated is E# ⟶ AD.

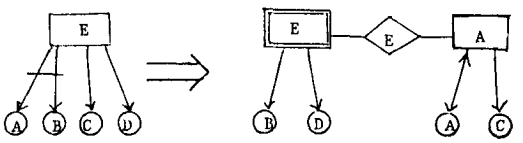

Figure 6.5 A ⟶ C and A is part of the identifier of E. E becomes a weak entity type and the identifier is still AB.

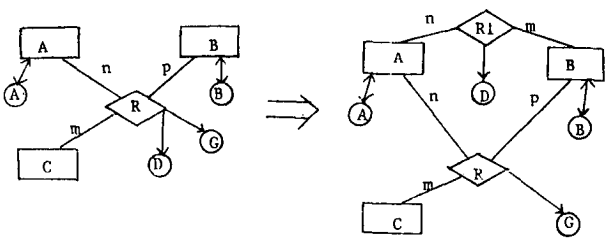

Figure 6.6 AB ⟶ D

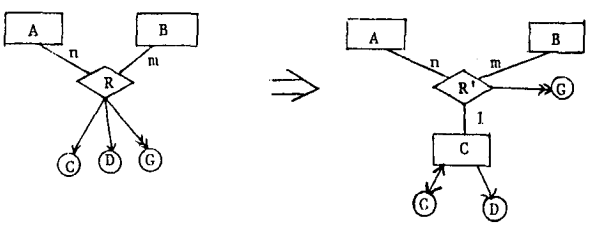

Figure 6.7 C ⟶ D in R

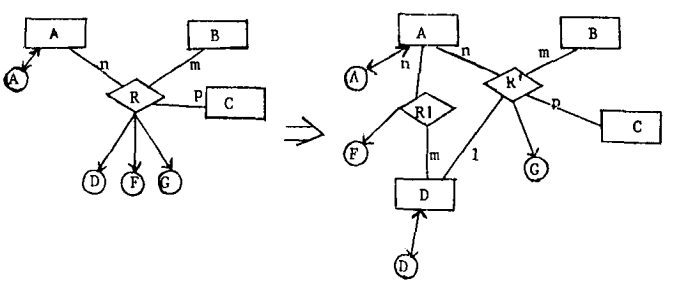

Figure 6.8 AD ⟶ F in R