THE ENTITY-RELATIONSHIP MODEL:
TOWARD A UNIFIED VIEW OF DATA

By

Peter Pin-Shan Chen

CISR No. 30

WP 913-77

March 1977

THE ENTITY-RELATIONSHIP MODEL:
TOWARD A UNIFIED VIEW OF DATA

By

Peter Pin-Shan Chen

CISR No. 30

WP 913-77

March 1977

The Entity-Relationship Model—Toward a Unified View of Data

PETER PIN-SHAN CHEN

Massachusetts Institute of Technology

A data model, called the entity-relationship model, is proposed. This model incorporates some of the important semantic information about the real world. A special diagrammatic technique is introduced as a tool for database design. An example of database design and description using the model and the diagrammatic technique is given. Some implications for data integrity, information retrieval, and data manipulation are discussed.

The entity-relationship model can be used as a basis for unification of different views of data: the network model, the relational model, and the entity set model. Semantic ambiguities in these models are analyzed. Possible ways to derive their views of data from the entity-relationship model are presented.

Key Words and Phrases: database design, logical view of data, semantics of data, data models, entity-relationship model, relational model, Data Base Task Group, network model, entity set model, data definition and manipulation, data integrity and consistency
CR Categories: 3.50, 3.70, 4.33, 4.34

1. INTRODUCTION

The logical view of data has been an important issue in recent years. Three major data models have been proposed: the network model [2, 3, 7], the relational model [8], and the entity set model [25]. These models have their own strengths and weaknesses. The network model provides a more natural view of data by separating entities and relationships (to a certain extent), but its capability to achieve data independence has been challenged [8]. The relational model is based on relational theory and can achieve a high degree of data independence, but it may lose some important semantic information about the real world [12, 15, 23]. The entity set model, which is based on set theory, also achieves a high degree of data independence, but its viewing of values such as "3" or "red" may not be natural to some people [25].

This paper presents the entity-relationship model, which has most of the advantages of the above three models. The entity-relationship model adopts the more natural view that the real world consists of entities and relationships. It

incorporates some of the important semantic information about the real world (other work in database semantics can be found in [1, 12, 15, 21, 23, and 29]). The model can achieve a high degree of data independence and is based on set theory and relation theory.

The entity-relationship model can be used as a basis for a unified view of data. Most work in the past has emphasized the difference between the network model and the relational model [22]. Recently, several attempts have been made to reduce the differences of the three data models [4, 19, 26, 30, 31]. This paper uses the entity-relationship model as a framework from which the three existing data models may be derived. The reader may view the entity-relationship model as a generalization or extension of existing models.

This paper is organized into three parts (Sections 2–4). Section 2 introduces the entity-relationship model using a framework of multilevel views of data. Section 3 describes the semantic information in the model and its implications for data description and data manipulation. A special diagrammatic technique, the entity-relationship diagram, is introduced as a tool for database design. Section 4 analyzes the network model, the relational model, and the entity set model, and describes how they may be derived from the entity-relationship model.

2. THE ENTITY-RELATIONSHIP MODEL

2.1 Multilevel Views of Data

In the study of a data model, we should identify the levels of logical views of data with which the model is concerned. Extending the framework developed in [18, 25], we can identify four levels of views of data (Figure 1):

(1) Information concerning entities and relationships which exist in our minds.

(2) Information structure—organization of information in which entities and relationships are represented by data.

(3) Access-path-independent data structure—the data structures which are not involved with search schemes, indexing schemes, etc.

(4) Access-path-dependent data structure.

In the following sections, we shall develop the entity-relationship model step by step for the first two levels. As we shall see later in the paper, the network model, as currently implemented, is mainly concerned with level 4; the relational model is mainly concerned with levels 3 and 2; the entity set model is mainly concerned with levels 1 and 2.

2.2 Information Concerning Entities and Relationships (Level 1)

At this level we consider entities and relationships. An *entity* is a "thing" which can be distinctly identified. A specific person, company, or event is an example of an entity. A *relationship* is an association among entities. For instance, "father-son" is a relationship between two "person" entities.[1]

---

[1] It is possible that some people may view something (e.g. marriage) as an entity while other people may view it as a relationship. We think that this is a decision which has to be made by the enterprise administrator [27]. He should define what are entities and what are relationships so that the distinction is suitable for his environment.

LEVELS OF LOGICAL VIEWS   MODELS

ENTITY-RELATIONSHIP   NETWORK   RELATIONAL   ENTITY-SET

LEVEL 1

INFORMATION CONCERNING   ENTITIES   ENTITIES   ENTITIES
ENTITIES AND   ENTITY SETS   RELATIONSHIPS   ENTITY SETS
RELATIONSHIPS   RELATIONSHIPS   ROLES
                            RELATIONSHIP SETS
                                       ATTRIBUTES
                      ATTRIBUTES   VALUES
                      VALUES
                      VALUE SETS
                      ROLES

LEVEL 2

INFORMATION STRUCTURE   ENTITY/RELATIONSHIP   3NF   ENTITY
                             RELATION ←   SIMILAR → RELATIONS   DESCRIPTION
                                                SETS

                     ENTITY-RELATIONSHIP
                     DIAGRAM   DECOMPOSITION

LEVEL 3   APPROACH

ACCESS-PATH-   TABLE   RELATIONS
INDEPENDENT   (TABLES)
DATA STRUCTURE

LEVEL 4

ACCESS-PATH-   RECORDS
DEPENDENT   DATA-STRUCTURE-
DATA STRUCTURE   SETS

                               DATA-STRUCTURE-
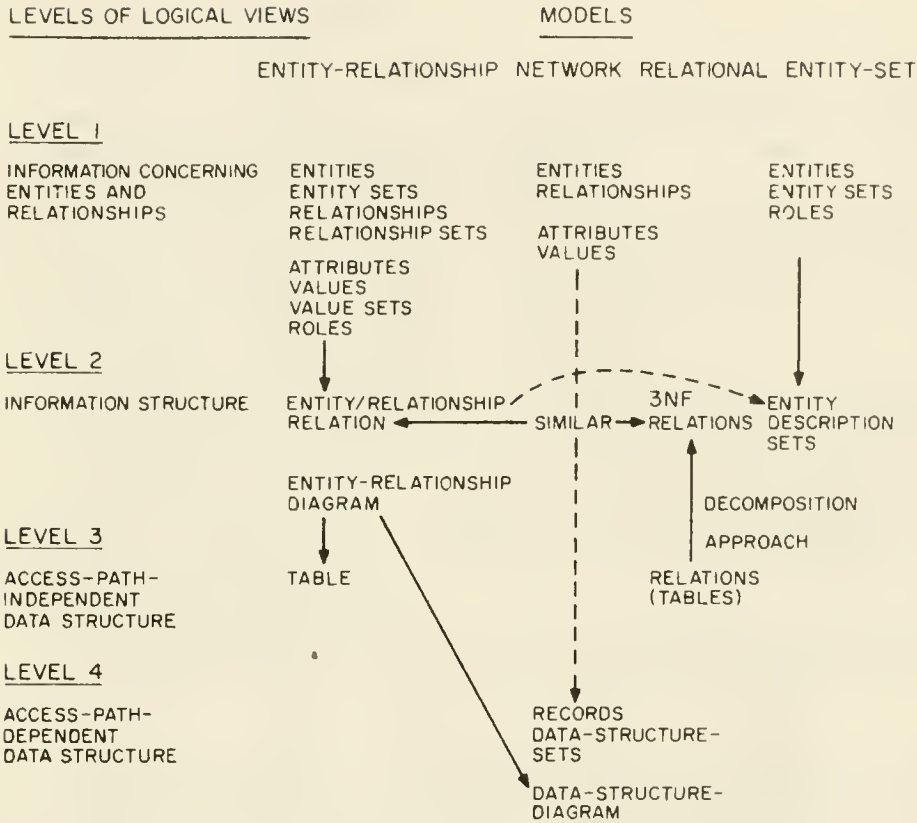   DIAGRAM

Fig. 1. Analysis of data models using multiple levels of logical views

The database of an enterprise contains relevant information concerning entities and relationships in which the enterprise is interested. A complete description of an entity or relationship may not be recorded in the database of an enterprise. It is impossible (and, perhaps, unnecessary) to record every potentially available piece of information about entities and relationships. From now on, we shall consider only the entities and relationships (and the information concerning them) which are to enter into the design of a database.

2.2.1 Entity and Entity Set. Let e denote an entity which exists in our minds. Entities are classified into different *entity sets* such as EMPLOYEE, PROJECT, and DEPARTMENT. There is a predicate associated with each entity set to test whether an entity belongs to it. For example, if we know an entity is in the entity set EMPLOYEE, then we know that it has the properties common to the other entities in the entity set EMPLOYEE. Among these properties is the afore-mentioned test predicate. Let E_i denote entity sets. Note that entity sets may not be mutually disjoint. For example, an entity which belongs to the entity set MALE-PERSON also belongs to the entity set PERSON. In this case, MALE-PERSON is a subset of PERSON.

2.2.2 Relationship, Role, and Relationship Set. Consider associations among entities. A *relationship set*, R_i, is a mathematical relation [5] among n entities,

each taken from an entity set:

$$\{[e_1, e_2, \ldots, e_n] \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\},$$

and each tuple of entities, $[e_1, e_2, \ldots, e_n]$, is a *relationship*. Note that the E_i in the above definition may not be distinct. For example, a "marriage" is a relationship between two entities in the entity set PERSON.

The *role* of an entity in a relationship is the function that it performs in the relationship. "Husband" and "wife" are roles. The ordering of entities in the definition of relationship (note that square brackets were used) can be dropped if roles of entities in the relationship are explicitly stated as follows: $(r_1 \cdot e_1, r_2 \cdot e_2, \ldots, r_n \cdot e_n)$, where r_i is the role of e_i in the relationship.

2.2.3 *Attribute, Value, and Value Set.* The information about an entity or a relationship is obtained by observation or measurement, and is expressed by a set of attribute-value pairs. "3", "red", "Peter", and "Johnson" are values. Values are classified into different *value sets*, such as FEET, COLOR, FIRST-NAME, and LAST-NAME. There is a predicate associated with each value set to test whether a value belongs to it. A value in a value set may be equivalent to another value in a different value set. For example, "12" in value set INCH is equivalent to "1" in value set FEET.

An *attribute* can be formally defined as a function which maps from an entity set or a relationship set into a value set or a Cartesian product of value sets:

$$f: E_i \text{ or } R_i \rightarrow V_i \text{ or } V_{i_1} \times V_{i_2} \times \cdots \times V_{i_n}$$

Figure 2 illustrates some attributes defined on entity set PERSON. The attribute AGE maps into value set NO-OF-YEARS. An attribute can map into a Cartesian product of value sets. For example, the attribute NAME maps into value sets FIRST-NAME, and LAST-NAME. Note that more than one attribute may map from the same entity set into the same value set (or same group of value sets). For example, NAME and ALTERNATIVE-NAME map from the entity set EMPLOYEE into value sets FIRST-NAME and LAST-NAME. Therefore, attribute and value set are different concepts although they may have the same name in some cases (for example, EMPLOYEE-NO maps from EMPLOYEE to value set EMPLOYEE-NO). This distinction is not clear in the network model and in many existing data management systems. Also note that an attribute is defined as a function. Therefore, it maps a given entity to a single value (or a single tuple of values in the case of a Cartesian product of value sets).

Note that relationships also have attributes. Consider the relationship set PROJECT-WORKER (Figure 3). The attribute PERCENTAGE-OF-TIME, which is the portion of time a particular employee is committed to a particular project, is an attribute defined on the relationship set PROJECT-WORKER. It is neither an attribute of EMPLOYEE nor an attribute of PROJECT, since its meaning depends on both the employee and project involved. The concept of attribute of relationship is important in understanding the semantics of data and in determining the functional dependencies among data.

2.2.4 *Conceptual Information Structure.* We are now concerned with how to organize the information associated with entities and relationships. The method proposed in this paper is to separate the information about entities from the infor-

ENTITY SET   ATTRIBUTES   VALUE SETS

E_I V_I

(EMPLOYEE)   (EMPLOYEE-NO)
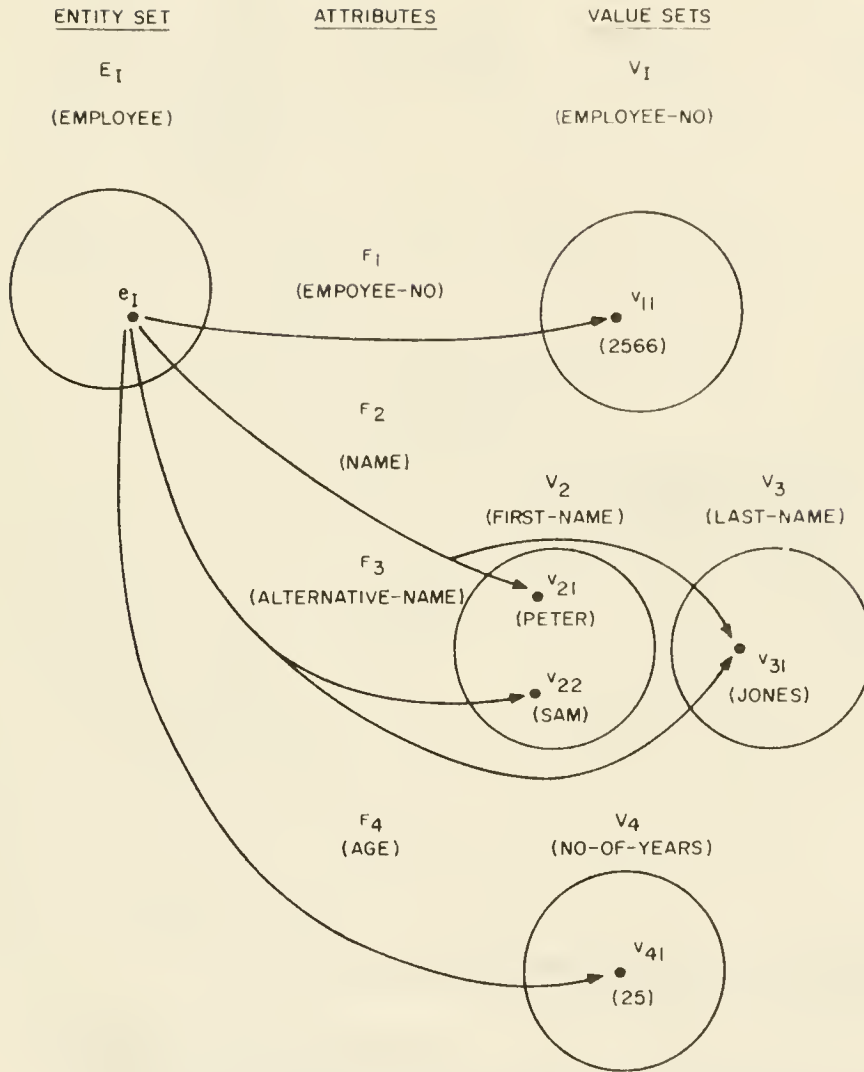

Fig. 2. Attributes defined on the entity set PERSON

mation about relationships. We shall see that this separation is useful in identifying functional dependencies among data.

Figure 4 illustrates in table form the information about entities in an entity set. Each row of values is related to the same entity, and each column is related to a value set which, in turn, is related to an attribute. The ordering of rows and columns is insignificant.

Figure 5 illustrates information about relationships in a relationship set. Note that each row of values is related to a relationship which is indicated by a group of entities, each having a specific role and belonging to a specific entity set.

Note that Figures 4 and 2 (and also Figures 5 and 3) are different forms of the same information. The table form is used for easily relating to the relational model.
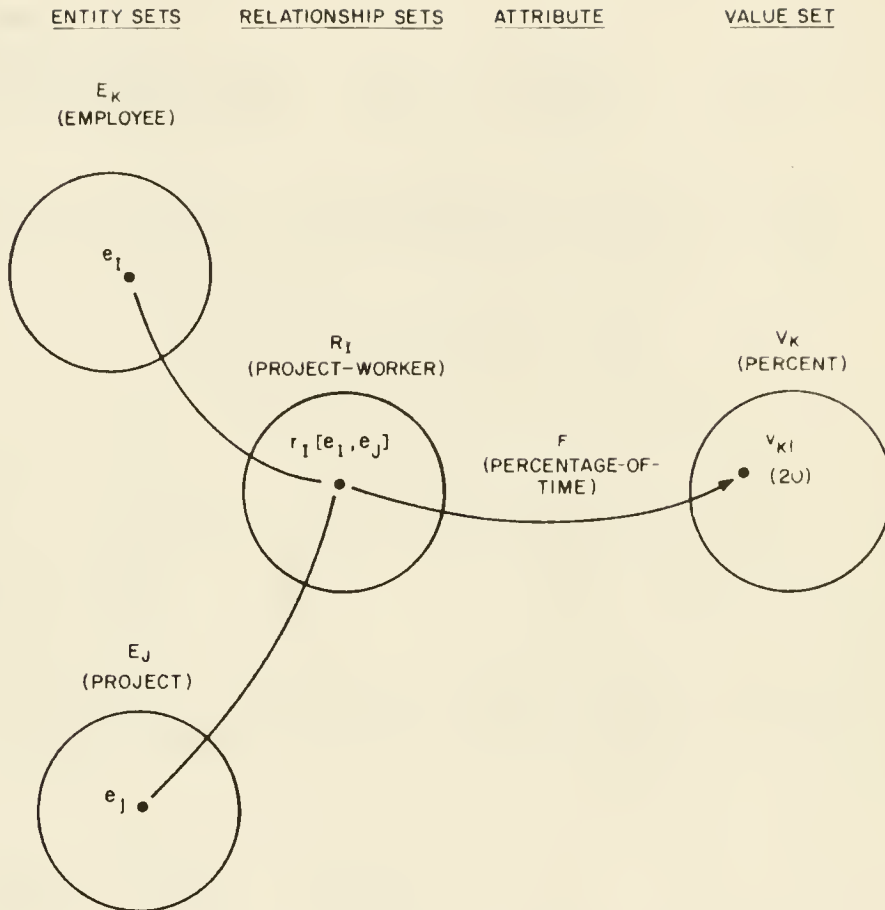
ENTITY SETS   RELATIONSHIP SETS   ATTRIBUTE   VALUE SET

Fig. 3. Attributes defined on the relationship set PROJECT–WORKER

2.3 Information Structure (Level 2)

The entities, relationships, and values at level 1 (see Figures 2–5) are conceptual objects in our minds (i.e. we were in the conceptual realm [18, 27]). At level 2, we consider representations of conceptual objects. We assume that there exist direct representations of values. In the following, we shall describe how to represent entities and relationships.

2.3.1 Primary Key. In Figure 2 the values of attribute EMPLOYEE–NO can be used to identify entities in entity set EMPLOYEE if each employee has a different employee number. It is possible that more than one attribute is needed to identify the entities in an entity set. It is also possible that several groups of attributes may be used to identify entities. Basically, an *entity key* is a group of attributes such that the mapping from the entity set to the corresponding group of value sets is one-to-one. If we cannot find such one-to-one mapping on available data, or if simplicity in identifying entities is desired, we may define an artificial attribute and a value set so that such mapping is possible. In the case where

ATTRIBUTE ENTITY SET AND VALUE SET →	F_1 (EMPLOYEE-NO)	F_2 (NAME)		F_3 (ALTERNATIVE-NAME)		F_4 (AGE)
E_I (EMPLOYEE)	V_1 (EMPLOYEE-NO)	V_2 (FIRST-NAME)	V_3 (LAST-NAME)	V_2 (FIRST-NAME)	V_3 (LAST-NAME)	V_4 (NO-OF-YEARS)
e_1	v_{11} (2566)	v_{21} (PETER)	v_{31} (JONES)	v_{22} (SAM)	v_{31} (JONES)	v_{41} (25)
e_2	v_{12} (3378)	v_{23} (MARY)	v_{32} (CHEN)	v_{24} (BARB)	v_{33} (CHEN)	v_{42} (23)
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$

Fig. 4. Information about entities in an entity set (table form)

ROLE →	WORKER	PROJECT		
			F (PERCENTAGE-OF-TIME)	RELATIONSHIP ATTRIBUTE
ENTITY SET →	E_I (EMPLOYEE)	E_J (PROJECT)	V_K (PERCENTAGE)	VALUE SET
	e_{I1}	e_{J1}	v_{K1} (20)	
	$\vdots$	$\vdots$	$\vdots$	

Fig. 5. Information about relationships in a relationship set (table form)
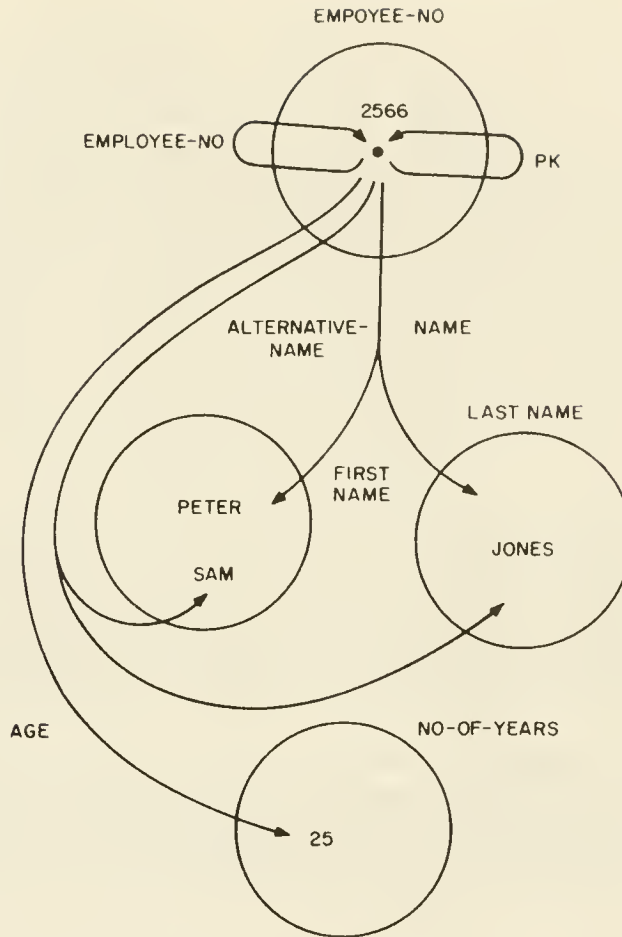
Fig. 6. Representing entities by values (employee numbers)

several keys exist, we usually choose a semantically meaningful key as the *entity primary key* (PK).

Figure 6 is obtained by merging the entity set EMPLOYEE with value set EMPLOYEE–NO in Figure 2. We should notice some semantic implications of Figure 6. Each value in the value set EMPLOYEE–NO represents an entity (employee). Attributes map from the value set EMPLOYEE–NO to other value sets. Also note that the attribute EMPLOYEE–NO maps from the value set EMPLOYEE–NO to itself.

2.3.2 Entity/Relationship Relations. Information about entities in an entity set can now be organized in a form shown in Figure 7. Note that Figure 7 is similar to Figure 4 except that entities are represented by the values of their primary keys. The whole table in Figure 7 is an *entity relation*, and each row is an *entity tuple*.

Since a relationship is identified by the involved entities, the *primary key of a relationship* can be represented by the primary keys of the involved entities. In

ATTRIBUTE	EMPLOYEE-NO	NAME		ALTERNATIVE-NAME		AGE
VALUE SET (DOMAIN)	EMPLOYEE-NO	FIRST-NAME	LAST-NAME	FIRST-NAME	LAST-NAME	NO-OF-YEARS
ENTITY (TUPLE)	2566	PETER	JONES	SAM	JONES	25
	3378	MARY	CHEN	BARB	CHEN	23
	⋮	⋮	⋮	⋮	⋮	⋮

Fig. 7 Regular entity relation EMPLOYEE

Figure 8, the involved entities are represented by their primary keys EMPLOYEE-NO and PROJECT-NO. The role names provide the semantic meaning for the values in the corresponding columns. Note that EMPLOYEE-NO is the primary key for the involved entities in the relationship and is not an attribute of the relationship. PERCENTAGE-OF-TIME is an attribute of the relationship. The table in Figure 8 is a *relationship relation*, and each row of values is a *relationship tuple*.

In certain cases, the entities in an entity set cannot be uniquely identified by the values of their own attributes; thus we must use a relationship(s) to identify them. For example, consider dependents of employees: dependents are identified by their names and by the values of the primary key of the employees supporting them (i.e. by their relationships with the employees). Note that in Figure 9,

ENTITY RELATION NAME	EMPLOYEE	PROJECT		
ROLE	WORKER	PROJECT		
ENTITY ATTRIBUTE	EMPLOYEE-NO	PROJECT-NO	PERCENTAGE-OF-TIME	RELATIONSHIP ATTRIBUTE
VALUE SET (DOMAIN)	EMPLOYEE-NO	PROJECT-NO	PERCENTAGE	
RELATIONSHIP TUPLE	2566	31	20	
	2173	25	100	
	⋮	⋮	⋮	

Fig. 8. Regular relationship relation PROJECT-WORKER

Fig. 9. A weak entity relation DEPENDENT

EMPLOYEE–NO is not an attribute of an entity in the set DEPENDENT but is the primary key of the employees who support dependents. Each row of values in Figure 9 is an entity tuple with EMPLOYEE NO and NAME as its primary key. The whole table is an entity relation.

Theoretically, any kind of relationship may be used to identify entities. For simplicity, we shall restrict ourselves to the use of only one kind of relationship: the binary relationships with $1:n$ mapping in which the existence of the n entities on one side of the relationship depends on the existence of one entity on the other side of the relationship. For example, one employee may have n ($= 0, 1, 2, \ldots$) dependents, and the existence of the dependents depends on the existence of the corresponding employee.

This method of identification of entities by relationships with other entities can be applied recursively until the entities which can be identified by their own attribute values are reached. For example, the primary key of a department in a company may consist of the department number and the primary key of the division, which in turn consists of the division number and the name of the company.

Therefore, we have two forms of entity relations. If relationships are used for identifying the entities, we shall call it a *weak entity relation* (Figure 9). If relationships are not used for identifying the entities, we shall call it a *regular entity relation* (Figure 7). Similarly, we also have two forms of relationship relations. If all entities in the relationship are identified by their own attribute values, we shall call it a *regular relationship relation* (Figure 8). If some entities in the relationship are identified by other relationships, we shall call it a *weak relationship relation*. For example, any relationships between DEPENDENT entities and other entities will result in weak relationship relations, since a DEPENDENT entity is identified by its name and its relationship with an EMPLOYEE entity. The distinction between regular (entity/relationship) relations and weak (entity/relationship) relations will be useful in maintaining data integrity.
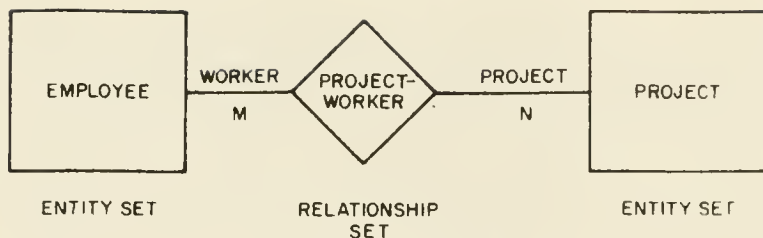
Fig. 10. A simple entity-relationship diagram

3. ENTITY-RELATIONSHIP DIAGRAM AND INCLUSION OF SEMANTICS IN DATA DESCRIPTION AND MANIPULATION

3.1 System Analysis Using the Entity-Relationship Diagram

In this section we introduce a diagrammatic technique for exhibiting entities and relationships: the entity-relationship diagram.

Figure 10 illustrates the relationship set PROJECT-WORKER and the entity sets EMPLOYEE and PROJECT using this diagrammatic technique. Each entity set is represented by a rectangular box, and each relationship set is represented by a diamond-shaped box. The fact that the relationship set PROJECT-WORKER is defined on the entity sets EMPLOYEE and PROJECT is represented by the lines connecting the rectangular boxes. The roles of the entities in the relationship are stated.
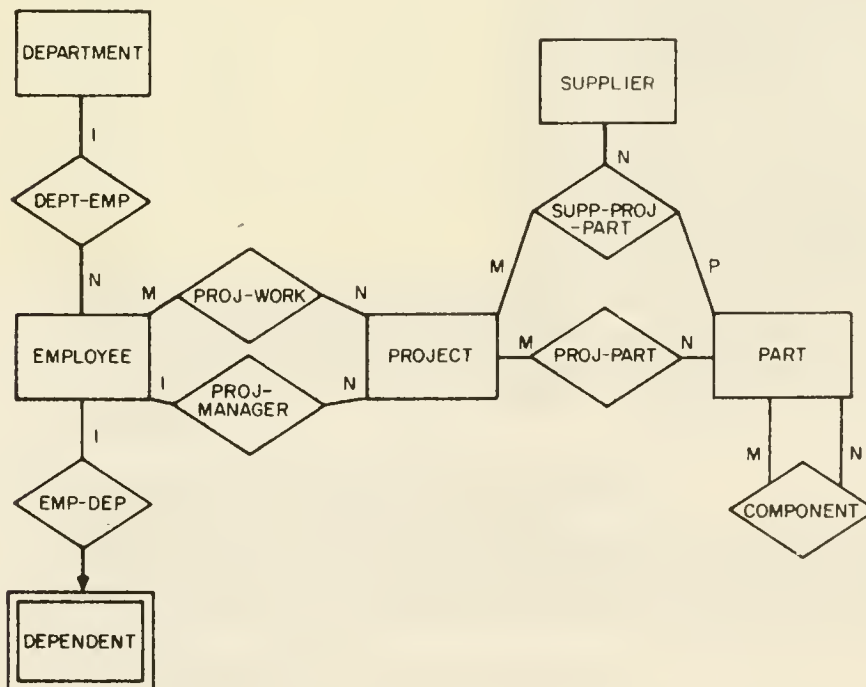
Fig. 11. An entity-relationship diagram for analysis of information in a manufacturing firm

P 17
Weak & angular

Figure 11 illustrates a more complete diagram of some entity sets and relationship sets which might be of interest to a manufacturing company. DEPARTMENT, EMPLOYEE, DEPENDENT, PROJECT, SUPPLIER, and PART are entity sets. DEPARTMENT-EMPLOYEE, EMPLOYEE-DEPENDENT, PROJECT-WORKER, PROJECT-MANAGER, SUPPLIER-PROJECT-PART, PROJECT-PART, and COMPONENT are relationship sets. The COMPONENT relationship describes what subparts (and quantities) are needed in making superparts. The meaning of the other relationship sets need not be explained.

Several important characteristics about relationships in general can be found in Figure 11:

(1) A relationship set may be defined on more than two entity sets. For example, the SUPPLIER-PROJECT-PART relationship set is defined on three entity sets: SUPPLIER, PROJECT, and PART.

(2) A relationship set may be defined on only one entity set. For example, the relationship set COMPONENT is defined on one entity set, PART.

(3) There may be more than one relationship set defined on given entity sets. For example, the relationship sets PROJECT-WORKER and PROJECT-MANAGER are defined on the entity sets PROJECT and EMPLOYEE.

(4) The diagram can distinguish between $1:n$, $m:n$, and $1:1$ mappings. The relationship set DEPARTMENT-EMPLOYEE is a $1:n$ mapping, that is, one department may have n ($n = 0, 1, 2, \ldots$) employees and each employee works for only one department. The relationship set PROJECT-WORKER is an $m:n$ mapping, that is, each project may have zero, one, or more employees assigned to it and each employee may be assigned to zero, one, or more projects. It is also possible to express $1:1$ mappings such as the relationship set MARRIAGE. Information about the number of entities in each entity set which is allowed in a relationship set is indicated by specifying "1", "m", "n" in the diagram. The relational model and the entity set model[2] do not include this type of information; the network model cannot express a $1:1$ mapping easily.

(5) The diagram can express the *existence dependency* of one entity type on another. For example, the arrow in the relationship set EMPLOYEE-DEPENDENT indicates that existence of an entity in the entity set DEPENDENT depends on the corresponding entity in the entity set EMPLOYEE. That is, if an employee leaves the company, his dependents may no longer be of interest.

Note that the entity set DEPENDENT is shown as a special rectangular box. This indicates that at level 2 the information about entities in this set is organized as a weak entity relation (using the primary key of EMPLOYEE as a part of its primary key).

3.2 An Example of a Database Design and Description

There are four steps in designing a database using the entity-relationship model: (1) identify the entity sets and the relationship sets of interest; (2) identify semantic information in the relationship sets such as whether a certain relationship

---

set is an 1:n mapping; (3) define the value sets and attributes; (4) organize data into entity/relationship relations and decide primary keys.

Let us use the manufacturing company discussed in Section 3.1 as an example. The results of the first two steps of database design are expressed in an entity-relationship diagram as shown in Figure 11. The third step is to define value sets and attributes (see Figures 2 and 3). The fourth step is to decide the primary keys for the entities and the relationships and to organize data as entity/relationship relations. Note that each entity/relationship set in Figure 11 has a corresponding entity/relationship relation. We shall use the names of the entity sets (at level 1) as the names of the corresponding entity/relationship relations (at level 2) as long as no confusion will result.

At the end of the section, we illustrate a schema (data definition) for a small part of the database in the above manufacturing company example (the syntax of the data definition is not important). Note that value sets are defined with specifications of representations and allowable values. For example, values in EMPLOYEE-NO are represented as 4-digit integers and range from 0 to 2000. We then declare three entity relations: EMPLOYEE, PROJECT, and DE-PENDENT. The attributes and value sets defined on the entity sets as well as the primary keys are stated. DEPENDENT is a weak entity relation since it uses EMPLOYEE.PK as part of its primary key. We also declare two relationship relations: PROJECT-WORKER and EMPLOYEE-DEPENDENT. The roles and involved entities in the relationships are specified. We use EMPLOYEE.PK to indicate the name of the entity relation (EMPLOYEE) and whatever attribute-value-set pairs are used as the primary keys in that entity relation. The maximum number of entities from an entity set in a relation is stated. For example, PROJECT-WORKER is an $m:n$ mapping. We may specify the values of m and n. We may also specify the minimum number of entities in addition to the maximum number. EMPLOYEE-DEPENDENT is a weak relationship relation since one of the related entity relations, DEPENDENT, is a weak entity relation. Note that the existence dependence of the dependents on the supporter is also stated.

DECLARE	VALUE-SETS	REPRESENTATION	ALLOWABLE-VALUES
	EMPLOYEE-NO	INTEGER (4)	(0,2000)
	FIRST-NAME	CHARACTER (8)	ALL
	LAST-NAME	CHARACTER (10)	ALL
	NO-OF-YEARS	INTEGER (3)	(0,100)
	PROJECT-NO	INTEGER (3)	(1,500)
	PERCENTAGE	FIXED (5.2)	(0,100.00)

DECLARE REGULAR ENTITY RELATION EMPLOYEE
ATTRIBUTE/VALUE-SET:
  EMPLOYEE-NO/EMPLOYEE-NO
  NAME/(FIRST-NAME, LAST-NAME)
  ALTERNATIVE-NAME/(FIRST-NAME,LAST-NAME)
  AGE/NO-OF-YEARS
PRIMARY KEY:
  EMPLOYEE-NO

DECLARE   REGULAR ENTITY RELATION PROJECT
ATTRIBUTE/VALUE-SET:
    PROJECT-NO/PROJECT-NO
PRIMARY KEY:
    PROJECT-NO

DECLARE   REGULAR RELATIONSHIP RELATION PROJECT-WORKER
ROLE/ENTITY-RELATION.PK/MAX-NO-OF-ENTITIES
    WORKER/EMPLOYEE.PK/m
    PROJECT/PROJECT.PK/n   (m:n mapping)
ATTRIBUTE/VALUE-SET:
    PERCENTAGE-OF-TIME/PERCENTAGE

DECLARE   WEAK RELATIONSHIP RELATION EMPLOYEE-DEPENDENT
ROLE/ENTITY-RELATION.PK/MAX-NO-OF-ENTITIES
    SUPPORTER/EMPLOYEE.PK/1
    DEPENDENT/DEPENDENT.PK/n
EXISTENCE OF DEPENDENT DEPENDS ON
EXISTENCE OF SUPPORTER

DECLARE   WEAK ENTITY RELATION DEPENDENT
ATTRIBUTE/VALUE-SET:
    NAME/FIRST-NAME
    AGE/NO-OF-YEARS
PRIMARY KEY:
    NAME
    EMPLOYEE.PK THROUGH EMPLOYEE-DEPENDENT

3.3 Implications on Data Integrity

Some work has been done on data integrity for other models [8, 14, 16, 28]. With explicit concepts of entity and relationship, the entity-relationship model will be useful in understanding and specifying constraints for maintaining data integrity. For example, there are three major kinds of constraints on values:

(1) Constraints on *allowable values* for a value set. This point was discussed in defining the schema in Section 3.2.

(2) Constraints on *permitted* values for a certain attribute. In some cases, not all allowable values in a value set are permitted for some attributes. For example, we may have a restriction of ages of employees to between 20 and 65. That is,

$$AGE(e) \in (20,65), \text{ where } e \in EMPLOYEE.$$

Note that we use the level 1 notations to clarify the semantics. Since each entity/relationship set has a corresponding entity/relationship relation, the above expression can be easily translated into level 2 notations.

(3) Constraints on *existing values* in the database. There are two types of constraints:

(i) Constraints between sets of existing values. For example,

$$\{NAME(e) \mid e \in MALE\text{-}PERSON\} \subseteq \{NAME(e) \mid e \in PERSON\}.$$

(ii) Constraints between particular values. For example,

$$\text{TAX}(e) \leq \text{SALARY}(e),\ e \in \text{EMPLOYEE}$$

or

$$\text{BUDGET}(e_i) = \sum \text{BUDGET}(e_j),\ \text{where } e_i \in \text{COMPANY}$$
$$e_j \in \text{DEPARTMENT}$$
$$\text{and } [e_i, e_j] \in \text{COMPANY-DEPARTMENT}.$$

3.4 Semantics and Set Operations of Information Retrieval Requests

The semantics of information retrieval requests become very clear if the requests are based on the entity-relationship model of data. For clarity, we first discuss the situation at level 1. Conceptually, the information elements are organized as in Figures 4 and 5 (on Figures 2 and 3). Many information retrieval requests can be considered as a combination of the following basic types of operations:

(1) Selection of a subset of values from a value set.

(2) Selection of a subset of entities from an entity set (i.e. selection of certain rows in Figure 4). Entities are selected by stating the values of certain attributes (i.e. subsets of value sets) and/or their relationships with other entities.

(3) Selection of a subset of relationships from a relationship set (i.e. selection of certain rows in Figure 5). Relationships are selected by stating the values of certain attribute(s) and/or by identifying certain entities in the relationship.

(4) Selection of a subset of attributes (i.e. selection of columns in Figures 4 and 5).

An information retrieval request like "What are the ages of the employees whose weights are greater than 170 and who are assigned to the project with PROJECT-NO 254?" can be expressed as:

```
{AGE(e) | e ∈ EMPLOYEE, WEIGHT(e) > 170,
[e, e_i] ∈ PROJECT-WORKER, e_i ∈ PROJECT,
PROJECT-NO(e_i) = 254};
```

or

```
{AGE(EMPLOYEE) | WEIGHT(EMPLOYEE) > 170,
[EMPLOYEE,PROJECT] ∈ PROJECT-WORKER,
PROJECT-NO(EMPLOYEE) = 254}.
```

To retrieve information as organized in Figure 6 at level 2, "entities" and "relationships" in (2) and (3) should be replaced by "entity PK" and "relationship PK." The above information retrieval request can be expressed as:

```
{AGE(EMPLOYEE.PK) | WEIGHT(EMPLOYEE.PK) > 170
(WORKER/EMPLOYEE.PK,PROJECT/PROJECT.PK) ∈ {PROJECT-WORKER.PK},
PROJECT-NO (PROJECT.PK) = 254}.
```

To retrieve information as organized in entity/relationship relations (Figures 7, 8, and 9), we can express it in a SEQUEL-like language [6]:

```
SELECT      AGE
FROM        EMPLOYEE
WHERE       WEIGHT > 170
```

Table I. Insertion

level 1	level 2
operation: insert an entity to an entity set	*operation:* create an entity tuple with a certain entity-PK *check:* whether PK already exists or is acceptable
operation: insert a relationship in a relationship set *check:* whether the entities exist	*operation:* create a relationship tuple with given entity PKs *check:* whether the entity PKs exist
operation: insert properties of an entity or a relationship *check:* whether the value is acceptable	*operation:* insert values in an entity tuple or a relation- ship tuple *check:* whether the values are acceptable

```
AND        EMPLOYEE.PK =
           SELECT    WORKER/EMPLOYEE.PK
           FROM      PROJECT-WORKER
           WHERE     PROJECT-NO = 254.
```

It is possible to retrieve information about entities in two different entity sets without specifying a relationship between them. For example, an information retrieval request like "List the names of employees and ships which have the same

Table II. Updating

level 1	level 2
operation: • change the value of an entity attribute	*operation:* • update a value *consequence:* • if it is not part of an entity PK, no conse- quence • if it is part of an entity PK, •• change the entity PKs in all related relationship relations •• change PKs of other entities which use this value as part of their PKs (for example, DEPENDENTS' PKs use EMPLOYEE'S PK)
operation: • change the value of a relationship attribute	*operation:* • update a value (note that a relationship attribute will not be a relationship PK)

Table III. Deletion

level 1	level 2
operation: • delete an entity *consequences:* • delete any entity whose existence depends on this entity • delete relationships involving this entity • delete all related properties	*operation:* • delete an entity tuple *consequences* (applied recursively): • delete any entity tuple whose existence depends on this entity tuple • delete relationship tuples associated with this entity
operation: • delete a relationship *consequences:* • delete all related properties	*operation:* • delete a relationship tuple

age" can be expressed in the level 1 notation as:

$$\{(NAME(e_i),NAME(e_j)) \mid e_i \in EMPLOYEE, e_j \in SHIP, AGE(e_i) = AGE(e_j)\}.$$

We do not further discuss the language syntax here. What we wish to stress is that information requests may be expressed using set notions and set operations [17], and the request semantics are very clear in adopting this point of view.

3.5 Semantics and Rules for Insertion, Deletion, and Updating

It is always a difficult problem to maintain data consistency following insertion, deletion, and updating of data in the database. One of the major reasons is that the semantics and consequences of insertion, deletion, and updating operations usually are not clearly defined; thus it is difficult to find a set of rules which can enforce data consistency. We shall see that this data consistency problem becomes simpler using the entity-relationship model.

In Tables I–III, we discuss the semantics and rules[3] for insertion, deletion, and updating at both level 1 and level 2. Level 1 is used to clarify the semantics.

4. ANALYSIS OF OTHER DATA MODELS AND THEIR DERIVATION FROM THE ENTITY-RELATIONSHIP MODEL

4.1 The Relational Model

4.1.1 The Relational View of Data and Ambiguity in Semantics. In the relational model, *relation*, R, is a mathematical relation defined on sets $X_1, X_2, \ldots, X_n$:

$$R = \{(x_1, x_2, \ldots, x_n) \mid x_1 \in X_1, x_2 \in X_2, \ldots, x_n \in X_n\}.$$

The sets $X_1, X_2, \ldots, X_n$ are called *domains*, and $(x_1, x_2, \ldots, x_n)$ is called a *tuple*. Figure 12 illustrates a relation called EMPLOYEE. The domains in the relation

---

[3] Our main purpose is to illustrate the semantics of data manipulation operations. Therefore, these rules may not be complete. Note that the consequence of operations stated in the tables can be performed by the system instead of by the users.

ROLE		LEGAL	LEGAL	ALTERNATIVE	ALTERNATIVE	
DOMAIN	EMPLOYEE-NO	FIRST-NAME	LAST-NAME	FIRST-NAME	LAST-NAME	NO-OF-YEARS
TUPLE	2566	PETER	JONES	SAM	JONES	25
	3378	MARY	CHEN	BARB	CHEN	23

Fig. 12. Relation EMPLOYEE

are EMPLOYEE-NO, FIRST–NAME, LAST–NAME, FIRST-NAME, LAST NAME, NO OF–YEAR. The ordering of rows and columns in the relation has no significance. To avoid ambiguity of columns with the same domain in a relation, domain names are qualified by *roles* (to distinguish the role of the domain in the relation). For example, in relation EMPLOYEE, domains FIRST–NAME and LAST–NAME may be qualified by roles LEGAL or ALTERNATIVE. An *attribute name* in the relational model is a domain name concatenated with a role name [10]. Comparing Figure 12 with Figure 7, we can see that "domains" are basically equivalent to value sets. Although "role" or "attribute" in the relational model seems to serve the same purpose as "attribute" in the entity-relationship model, the semantics of these terms are different. The "role" or "attribute" in the relational model is mainly used to distinguish domains with the same name in the same relation, while "attribute" in the entity-relationship model is a function which maps from an entity (or relationship) set into value set(s).

Using relational operators in the relational model may cause semantic ambiguities. For example, the join of the relation EMPLOYEE with the relation EMPLOYEE–PROJECT (Figure 13) on domain EMPLOYEE–NO produces the

PROJECT-NO	EMPLOYEE-NO
7	2566
3	2566
7	3378

Fig. 13. Relation EMPLOYEE–PROJECT

PROJECT-NO	EMPLOYEE-NO	LEGAL FIRST-NAME	LEGAL LAST-NAME	ALTERNATIVE FIRST-NAME	ALTERNATIVE LAST-NAME	NO-OF-YEARS
7	2566	PETER	JONES	SAM	JONES	25
3	2566	PETER	JONES	SAM	JONES	25
7	3378	MARY	CHEN	BARB	CHEN	23

Fig. 14. Relation EMPLOYEE-PROJECT' as a "join" of relations EMPLOYEE and EMPLOYEE-PROJECT

relation EMPLOYEE-PROJECT' (Figure 14). But what is the meaning of a join between the relation EMPLOYEE with the relation SHIP on the domain NO-OF-YEARS (Figure 15)? The problem is that the same domain name may have different semantics in different relations (note that a role is intended to distinguish domains in a given relation, not in all relations). If the domain NO-OF-YEAR of the relation EMPLOYEE is not allowed to be compared with the domain NO-OF-YEAR of the relation SHIP, different domain names have to be declared. But if such a comparison is acceptable, can the database system warn the user?

In the entity-relationship model, the semantics of data are much more apparent. For example, one column in the example stated above contains the values of AGE of EMPLOYEE and the other column contains the values of AGE of SHIP. If this semantic information is exposed to the user, he may operate more cautiously (refer to the sample information retrieval requests stated in Section 3.4). Since the database system contains the semantic information, it should be able to warn the user of the potential problems for a proposed "join-like" operation.

4.1.2 Semantics of Functional Dependencies Among Data. In the relational model, "attribute" B of a relation is *functionally dependent* on "attribute" A of the same relation if each value of A has no more than one value of B associated with it in the relation. Semantics of functional dependencies among data become clear

SHIP-NO	NAME	NO-OF-YEARS
037	MISSOURI	25
056	VIRGINIA	10

Fig. 15. Relation SHIP

in the entity-relationship model. Basically, there are two major types of functional dependencies:

(1) Functional dependencies related to description of entities or relationships. Since an attribute is defined as a function, it maps an entity in an entity set to a single value in a value set (see Figure 2). At level 2, the values of the primary key are used to represent entities. Therefore, nonkey value sets (domains) are functionally dependent on primary-key value sets (for example, in Figures 6 and 7, NO-OF-YEARS is functionally dependent on EMPLOYEE-NO). Since a relation may have several keys, the nonkey value sets will functionally depend on any key value set. The key value sets will be mutually functionally dependent on each other. Similarly, in a relationship relation the nonkey value sets will be functionally dependent on the prime-key value sets (for example, in Figure 8, PERCENTAGE is functionally dependent on EMPLOYEE-NO and PROJECT-NO).

(2) Functional dependencies related to entities in a relationship. Note that in Figure 11 we identify the types of mappings ($1:n$, $m:n$, etc.) for relationship sets. For example, PROJECT-MANAGER is a $1:n$ mapping. Let us assume that PROJECT-NO is the primary key in the entity relation PROJECT. In the relationship relation PROJECT-MANAGER, the value set EMPLOYEE-NO will be functionally dependent on the value set PROJECT-NO (i.e. each project has only one manager).

The distinction between level 1 (Figure 2) and level 2 (Figures 6 and 7) and the separation of entity relation (Figure 7) from relationship relation (Figure 8) clarifies the semantics of functional dependencies among data.

4.1.3 3NF Relations Versus Entity-Relationship Relations. From the definition of "relation," any grouping of domains can be considered to be a relation. To avoid undesirable properties in maintaining relations, a normalization process is proposed to transform arbitrary relations into the first normal form, then into the second normal form, and finally into the third normal form (3NF) [9, 11]. We shall show that the entity and relationship relations in the entity-relationship model are similar to 3NF relations but with clearer semantics and without using the transformation operation.

Let us use a simplified version of an example of normalization described in [9]. The following three relations are in first normal form (that is, there is no domain whose elements are themselves relations):

EMPLOYEE (EMPLOYEE-NO)
PART (PART-NO, PART-DESCRIPTION, QUANTITY-ON-HAND)
PART-PROJECT (PART-NO, PROJECT-NO, PROJECT-DESCRIPTION,
           PROJECT-MANAGER-NO, QUANTITY-COMMITTED).

Note that the domain PROJECT-MANAGER-NO actually contains the EMPLOYEE-NO of the project manager. In the relations above, primary keys are underlined.

Certain rules are applied to transform the relations above into third normal form:

EMPLOYEE' (EMPLOYEE-NO)
PART' (PART-NO, PART-DESCRIPTION, QUANTITY-ON-HAND)

PROJECT' (PROJECT-NO, PROJECT-DESCRIPTION, PROJECT-MANAGER-NO)
PART-PROJECT' (PART-NO, PROJECT-NO, QUANTITY-COMMITTED).

Using the entity-relationship diagram in Figure 11, the following entity and relationship relations can be easily derived:

entity relations	PART''(PART-NO, PART-DESCRIPTION, QUANTITY-ON-HAND) PROJECT''(PROJECT-NO, PROJECT-DESCRIPTION) EMPLOYEE''(EMPLOYEE-NO)
relationship relations	PART-PROJECT''(PART/PART-NO, PROJECT/PROJECT-NO, QUANTITY-COMMITTED) PROJECT-MANAGER''(PROJECT/PROJECT-NO, MANAGER/EMPLOYEE-NO).

The role names of the entities in relationships (such as MANAGER) are indicated. The entity relation names associated with the PKs of entities in relationships and the value set names have been omitted.

Note that in the example above, entity/relationship relations are similar to the 3NF relations. In the 3NF approach, PROJECT-MANAGER-NO is included in the relation PROJECT' since PROJECT-MANAGER-NO is assumed to be functionally dependent on PROJECT-NO. In the entity-relationship model, PROJECT-MANAGER NO (i.e. EMPLOYEE-NO of a project manager) is included in a relationship relation PROJECT-MANAGER since EMPLOYEE-NO is considered as an entity PK in this case.

Also note that in the 3NF approach, changes in functional dependencies of data may cause some relations not to be in 3NF. For example, if we make a new assumption that one project may have more than one manager, the relation PROJECT' is no longer a 3NF relation and has to be split into two relations as PROJECT'' and PROJECT-MANAGER''. Using the entity-relationship model, no such change is necessary. Therefore, we may say that by using the entity-relationship model we can arrange data in a form similar to 3NF relations but with clear semantic meaning.

It is interesting to note that the decomposition (or transformation) approach described above for normalization of relations may be viewed as a bottom-up approach in database design.[4] It starts with arbitrary relations (level 3 in Figure 1) and then uses some semantic information (functional dependencies of data) to transform them into 3NF relations (level 2 in Figure 1). The entity-relationship model adopts a top-down approach, utilizing the semantic information to organize data in entity/relationship relations.

4.2 The Network Model

4.2.1 Semantics of the Data-Structure Diagram. One of the best ways to explain the network model is by use of the *data-structure diagram* [3]. Figure 16(a) illustrates a data-structure diagram. Each rectangular box represents a record type.

---

[4] Although the decomposition approach was emphasized in the relational model literature, it is a procedure to obtain 3NF and may not be an intrinsic property of 3NF.
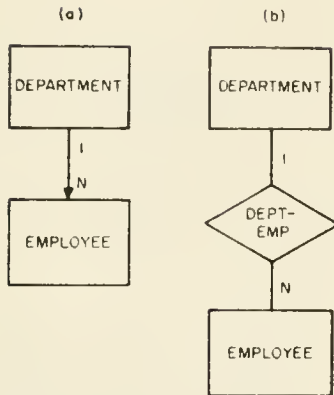
Fig. 16. Relationship DEPART-
MENT-EMPLOYEE
(a) data structure diagram
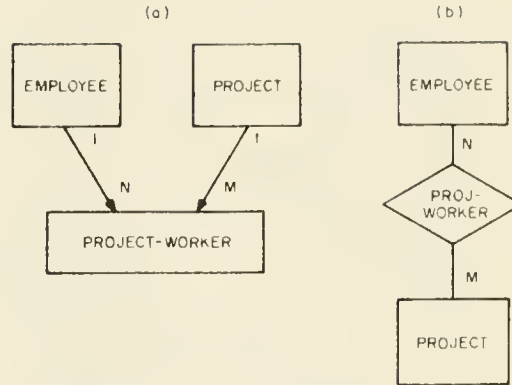(b) entity-relationship diagram

Fig. 17. Relationship PROJECT-WORKER
(a) data structure diagram
(b) entity-relationship diagram

The arrow represents a data-structure-set in which the DEPARTMENT record is the *owner-record*, and one owner-record may own n ($n = 0, 1, 2, \ldots$) *member-records*. Figure 16(b) illustrates the corresponding entity-relationship diagram. One might conclude that the arrow in the data-structure diagram represents a relationship between entities in two entity sets. This is not always true. Figures 17(a) and 17(b) are the data-structure diagram and the entity-relationship diagram expressing the relationship PROJECT-WORKER between two entity types EMPLOYEE and PROJECT. We can see in Figure 17(a) that the relationship PROJECT-WORKER becomes another record type and that the arrows no longer represent relationships between entities. What are the real meanings of the arrows in data-structure diagrams? The answer is that an arrow represents an $1:n$ relationship between two *record* (not entity) types and also implies the existence of an access path from the owner record to the member records. The data-structure diagram is a representation of the organization of records (level 4 in Figure 1) and is not an exact representation of entities and relationships.

4.2.2 Deriving the Data-Structure Diagram. Under what conditions does an arrow in a data-structure diagram correspond to a relationship of entities? A close comparison of the data-structure diagrams with the corresponding entity-relationship diagrams reveals the following rules:

1. For $1:n$ binary relationships an arrow is used to represent the relationship (see Figure 16(a)).

2. For $m:n$ binary relationships a "relationship record" type is created to represent the relationship and arrows are drawn from the "entity record" type to the "relationship record" type (see Figure 17(a)).

3. For k-ary ($k \geq 3$) relationships, the same rule as (2) applies (i.e. creating a "relationship record" type).

Since DBTG [7] does not allow a data-structure-set to be defined on a single record type (i.e. Figure 18 is not allowed although it has been implemented in [13]), a "relationship record" is needed to implement such relationships (see
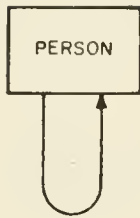
(a)   (b)

Fig. 18. Data-structure-set de-
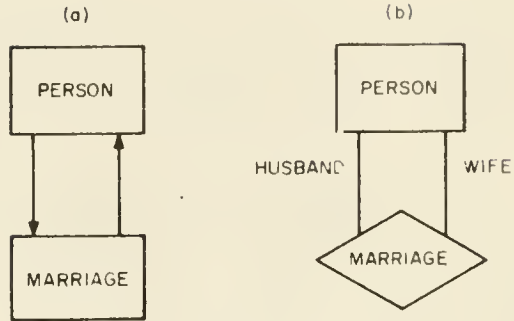fined on the same record type

Fig. 19. Relationship MARRIAGE (a) data-struc-
ture diagram (b) entity-relationship diagram

Figure 19(a)) [20]. The corresponding entity-relationship diagram is shown in
Figure 19(b).

It is clear now that arrows in a data-structure diagram do not always represent
relationships of entities. Even in the case that an arrow represents a $1:n$ relation-
ship, the arrow only represents a unidirectional relationship [20] (although it is
possible to find the owner-record from a member-record). In the entity-relationship
model, both directions of the relationship are represented (the roles of both en-
tities are specified). Besides the semantic ambiguity in its arrows, the network
model is awkward in handling changes in semantics. For example, if the relationship
between DEPARTMENT and EMPLOYEE changes from a $1:n$ mapping to an
$m:n$ mapping (i.e. one employee may belong to several departments), we must
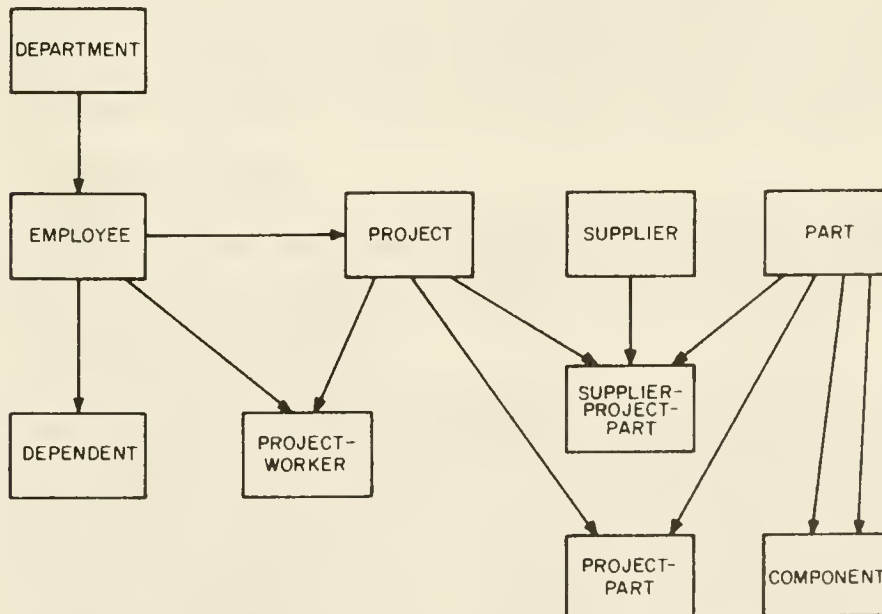create a relationship record DEPARTMENT-EMPLOYEE in the network model.

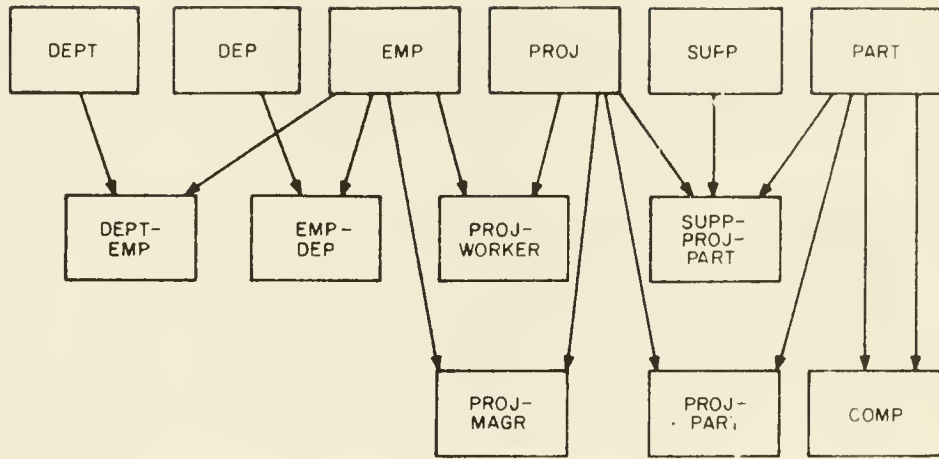Fig. 20. The data structure diagram derived from the entity-relationship diagram in Fig. 11

Fig. 21. The "disciplined" data structure diagram derived from the entity-relationship diagram in Fig. 11

In the entity-relationship model, all kinds of mappings in relationships are handled uniformly.

The entity-relationship model can be used as a tool in the structured design of databases using the network model. The user first draws an entity-relationship diagram (Figure 11). He may simply translate it into a data-structure diagram (Figure 20), using the rules specified above. He may also follow a discipline that every entity or relationship must be mapped onto a record (that is, "relationship records" are created for all types of relationships no matter that they are $1:n$ or $m:n$ mappings). Thus, in Figure 11, all one needs to do is to change the diamonds to boxes and to add arrowheads on the appropriate lines. Using this approach three more boxes—DEPARTMENT-EMPLOYEE, EMPLOYEE-DEPENDENT, and PROJECT MANAGER—will be added to Figure 20 (see Figure 21). The validity constraints discussed in Sections 3.3–3.5 will also be useful.

4.3 The Entity Set Model

4.3.1 The Entity Set View. The basic element of the entity set model is the entity. Entities have names (*entity names*) such as "Peter Jones", "blue", or "22". Entity names having some properties in common are collected into an *entity-name-set*, which is referenced by the *entity-name-set-name* such as "NAME", "COLOR", and "QUANTITY".

An entity is represented by the entity-name-set-name/entity-name pair such as NAME/Peter Jones, EMPLOYEE-NO/2566, and NO-OF-YEARS/20. An entity is described by its association with other entities. Figure 22 illustrates the entity set view of data. The "DEPARTMENT" of entity EMPLOYEE-NO/2566 is the entity DEPARTMENT-NO/405. In other words, "DEPARTMENT" is the role that the entity DEPARTMENT-NO/405 plays to describe the entity EMPLOYEE-NO/2566. Similarly, the "NAME", "ALTERNATIVE-NAME", or "AGE" of EMPLOYEE-NO/2566 is "NAME/Peter Jones", "NAME/Sam Jones", or "NO-OF-YEARS/20", respectively. The description of the entity EMPLOYEE-

NO/2566 is a collection of the related entities and their roles (the entities and roles circled by the dotted line). An example of the *entity description* of "EM-PLOYEE–NO 2566" (in its full-blown, unfactored form) is illustrated by the set of role-name entity-name-set-name/entity-name triplets shown in Figure 23. Conceptually, the entity set model differs from the entity-relationship model in the following ways:

(1) In the entity set model, everything is treated as an entity. For example, "COLOR/BLACK" and "NO-OF-YEARS/45" are entities. In the entity-relationship model, "blue" and "36" are usually treated as values. Note treating values as entities may cause semantic problems. For example, in Figure 22, what is the difference between "EMPLOYEE–NO/2566", "NAME Peter Jones", and "NAME/Sam Jones"? Do they represent different entities?

(2) Only binary relationships are used in the entity set model,[5] while *n*-ary relationships may be used in the entity-relationship model.
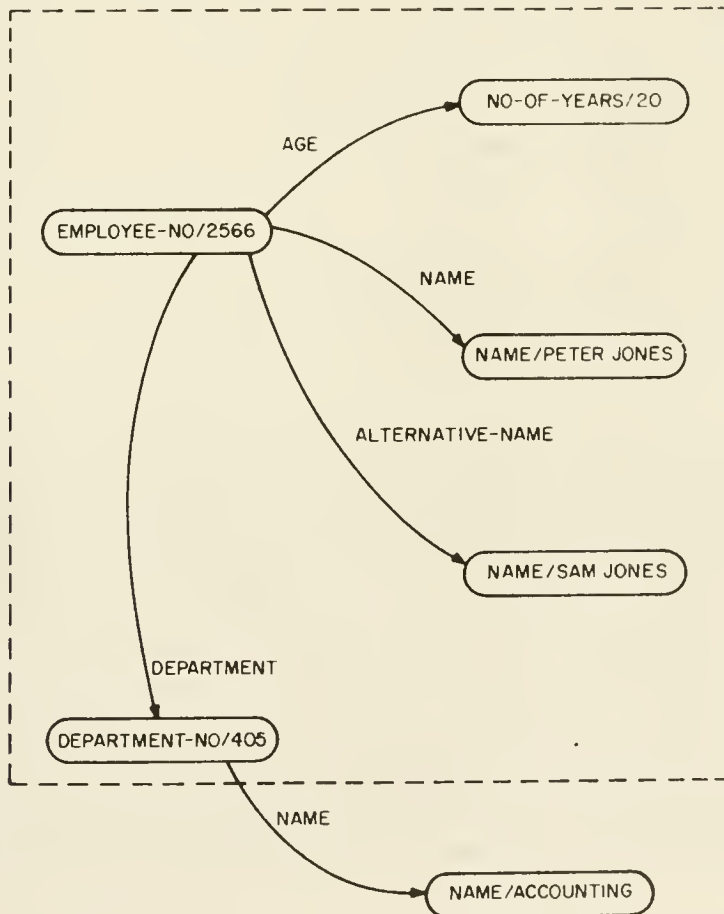

Fig. 22. The entity-set view

---

[5] In DIAM II [24], *n*-ary relationships may be treated as special cases of identifiers.

THE ENTITY-RELATIONSHIP MODEL TERMINOLOGY	ATTRIBUTE OR ROLE	VALUE SET	VALUE
THE ENTITY SET MODEL TERMINOLOGY	"ROLE-NAME"	"ENTITY-NAME-SET-NAME"	"ENTITY-NAME"
	IDENTIFIER	EMPLOYEE-NO	2566
	NAME	NAME	PETER JONES
	NAME	NAME	SAM JONES
	AGE	NO-OF-YEARS	25
	DEPARTMENT	DEPARTMENT-NO	405

Fig. 23. An "entity description" in the entity-set model

4.3.2 Deriving the Entity Set View. One of the main difficulties in understanding the entity set model is due to its world view (i.e. identifying values with entities). The entity-relationship model proposed in this paper is useful in understanding and deriving the entity set view of data. Consider Figures 2 and 6. In Figure 2, entities are represented by e_i's (which exist in our minds or are pointed at with fingers). In Figure 6, entities are represented by values. The entity set model works both at level 1 and level 2, but we shall explain its view at level 2 (Figure 6). The entity set model treats all value sets such as NO-OF-YEARS as "entity-name-sets" and all values as "entity-names." The attributes become role names in the entity set model. For binary relationships, the translation is simple: the role of an entity in a relationship (for example, the role of "DEPARTMENT" in the relationship DEPARTMENT-EMPLOYEE) becomes the role name of the entity in describing the other entity in the relationship (see Figure 22). For n-ary ($n > 2$) relationships, we must create artificial entities for relationships in order to handle them in a binary relationship world.

gestions (Figure 21 was suggested by one of the referees). This paper was motivated by a series of discussions with Charles Bachman. The author is also indebted to E.F. Codd and M.E. Senko for their valuable comments and discussions in revising this paper.

REFERENCES

1. ABRIAL, J.R. Data semantics. In *Data Base Management*, J.W. Klimbie and K.L. Koffeman, Eds., North-Holland Pub. Co., Amsterdam, 1974, pp. 1–60.
2. BACHMAN, C.W. Software for random access processing. *Datamation 11* (April 1965), 36–41.
3. BACHMAN, C.W. Data structure diagrams. *Data Base 1*, 2 (Summer 1969), 4–10.
4. BACHMAN, C.W. Trends in database management—1975. Proc., AFIPS 1975 NCC, Vol. 44, AFIPS Press, Montvale, N.J., pp. 569–576.
5. BIRKHOFF, G., AND BARTEE, T.C. *Modern Applied Algebra*. McGraw-Hill, New York, 1970.
6. CHAMBERLIN, D.D., AND RAYMOND, F.B. SEQUEL: A structured English query language. Proc. ACM-SIGMOD 1974, Workshop, Ann Arbor, Michigan, May, 1974.
7. CODASYL. Data base task group report. ACM, New York, 1971.
8. CODD, E.F. A relational model of data for large shared data banks. *Comm. ACM 13*, 6 (June 1970), 377–387.
9. CODD, E.F. Normalized data base structure: A brief tutorial. Proc. ACM-SIGFIDET 1971, Workshop, San Diego, Calif., Nov. 1971, pp. 1–18.
10. CODD, E.F. A data base sublanguage founded on the relational calculus. Proc. ACM-SIG-FIDET 1971, Workshop, San Diego, Calif., Nov. 1971, pp. 35–68.
11. CODD, E.F. Recent investigations in relational data base systems. Proc. IFIP Congress 1974, North-Holland Pub. Co., Amsterdam, pp. 1017–1021.
12. DEHENEFFE, C., HENNEBERT, H., AND PAULUS, W. Relational model for data base. Proc. IFIP Congress 1974, North-Holland Pub. Co., Amsterdam, pp. 1022–1025.
13. DODD, G.G. APL—a language for associate data handling in PL/1. Proc. AFIPS 1966 FJCC, Vol. 29, Spartan Books, New York, pp. 677–684.
14. ESWARAN, K.P., AND CHAMBERLIN, D.D. Functional specifications of a subsystem for data base integrity. Proc. Very Large Data Base Conf., Framingham, Mass., Sept. 1975, pp. 48–68.
15. HAINAUT, J.L., AND LECHARLIER, B. An extensible semantic model of data base and its data language. Proc. IFIP Congress 1974, North-Holland Pub. Co., Amsterdam, pp. 1026–1030.
16. HAMMER, M.M., AND McLEOD, D.J. Semantic integrity in a relation data base system. Proc. Very Large Data Base Conf., Framingham, Mass., Sept. 1975, pp. 25–47.
17. LINDGREEN, P. Basic operations on information as a basis for data base design. Proc. IFIP Congress 1974, North-Holland Pub. Co., Amsterdam, pp. 993–997.
18. MEALY, G.H. Another look at data base. Proc. AFIPS 1967 FJCC, Vol. 31, AFIPS Press, Montvale, N.J., pp. 525–534.
19. NIJSSEN, G.M. Data structuring in the DDL and the relational model. In *Data Base Management*, J.W. Klimbie and K.L. Koffeman, Eds., North-Holland Pub. Co., Amsterdam, 1974, pp. 363–379.
20. OLLE, T.W. Current and future trends in data base management systems. Proc. IFIP Congress 1974, North-Holland Pub. Co., Amsterdam, pp. 998–1006.
21. ROUSSOPOULOS, N., AND MYLOPOULOS, J. Using semantic networks for data base management. Proc. Very Large Data Base Conf., Framingham, Mass., Sept. 1975, pp. 144–172.
22. RUSTIN, R. (Ed.). Proc. ACM-SOGMOD 1974—debate on data models. Ann Arbor, Mich., May 1974.
23. SCHMID, H.A., AND SWENSON, J.R. On the semantics of the relational model. Proc. ACM-SIGMOD 1975, Conference, San Jose, Calif., May 1975, pp. 211–233.
24. SENKO, M.E. Data description language in the concept of multilevel structured description: DIAM II with FORAL. In *Data Base Description*, B.C.M. Dougue, and G.M. Nijssen, Eds., North-Holland Pub. Co., Amsterdam, pp. 239–258.