BASEMENT

HD28
.M414
839-76

MASS. INST. TECH.
OCT 26 1976
LIBRARIES

WORKING PAPER
ALFRED P. SLOAN SCHOOL OF MANAGEMENT

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

BY PETER PIN-SHAN CHEN

OCT 25 1976

WP 839-76 MARCH, 1976

MASSACHUSETTS
INSTITUTE OF TECHNOLOGY
50 MEMORIAL DRIVE
CAMBRIDGE, MASSACHUSETTS 02139

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

BY PETER PIN-SHAN CHEN

OCT 25 1976

WP 839-76 MARCH, 1976

* A revised version of this paper will appear in the ACM Transactions on Database Systems.

HD28
.M414
no. 837-76

M.I.T. LIBRARIES
OCT 26 1976
RECEIVED

ABSTRACT:

A data model, called the entity-relationship model, is proposed. This model incorporates some of the important semantic information in the real world. A special diagramatic technique is introduced as a tool for data base design. An example of data base design and description using the model and the diagramatic technique is given. Some implications on 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:

data base 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.

C72Sv89

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 [24]. 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 [24].

This paper presents the entity-relationship model, which has most of the advantages of the three models stated in the previous paragraph. 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. Part 1 introduces the entity-relationship model using a framework of multi-level views of data. Part 2 describes

the semantic information in the model and its implications for data description and data manipulation. A special diagramatic technique, the entity-relationship diagram, is introduced as a tool for data base design. Part 3 analyzes the network model, the relatonal 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. Multi-level 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 [18,24], 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 -- tha 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*.

---

* 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 [7]. He should define what are entities and what are relationships so that the distinction is suitable for his environment.

The data base 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 data base 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 data base.

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 aforementioned 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 [20] 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 role

of entities in the relationship are explicitly stated as follows:

$(r_1/e_1, r_2/e_2, \ldots, r_n/e_n)$, 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:

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

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 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 of a particular employee 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 information about relationships. We shall see that this separation is useful in identifying functional dependencies among data.

Figure 4 illustrates the information about entities in an entity set. This information is shown in table form. Each row of values is related to the same entity, and each column is related to a value set which is, in turn, 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.

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 can not find such one-to-one mapping on available data, or 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 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 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, 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 relationships 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,...) 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.

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 diagramatic 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 diagramatic 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.

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 entitiy 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,...) employees and each employee works for only one department. The relationship set PROJECT-WORKER is a 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* do not include this type of information; the network model can not 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 illustrated 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).

---

* This mapping information is included in DIAM II [25].

3.2. An example of a data base design and description

There are four steps in designing a data base using the entity-relationship model: (1) identify the entity sets and the relationship sets of interest; (2) identify semantic information in the relatonship 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 the last section as an example. The results of the first two steps of data base design are expressed in an entity-relationship diagram as shown in Figure 11. The third step is to define value sets and attributes (see Fig. 2&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 shall illustrate a schema (data definition) for a small part of the data base 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 DEPENDENT. 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 the last section.

(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 data base. 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,

$$TAX(e) \leq SALARY(e), \; e \in EMPLOYEE$$

or

$$BUDGET(e_i) = \sum BUDGET(e_j), \text{ where } e_i \in COMPANY, \; e_j \in DEPARTMENT$$

and $[e_i, e_j] \in COMPANY\text{-}DEPARTMENT$.

3.4 Semantics amd 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 (or 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:

$$\{AGE(e) \mid e \in EMPLOYEE,\ WEIGHT(e) > 170,$$
$$[e, e_j] \in PROJECT\text{-}WORKER,\ e_j \in PROJECT,$$
$$PROJECT\text{-}NO\ (e_j) = 254\}\ ;$$

or,

$$\{AGE(EMPLOYEE) \mid WEIGHT(EMPLOYEE) > 170,$$
$$[EMPLOYEE, PROJECT] \in PROJECT\text{-}WORKER,$$
$$PROJECT\text{-}NO(EMPLOYEE) = 254\}\ .$$

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

$\{(NAME(e_i),NAME(e_j))|e_i \varepsilon EMPLOYEE, e_j \varepsilon SHIP, AGE(e_i)=AGE(e_j)\}$.

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 data base. 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 the following tables, we discuss the semantics and rules* for insertion, deletion, and updating in both level 1 and level 2. Level 1 is used to clarify the semantics.

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 PK's check: whether the entity PK's 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 relationship tuple check: whether the values are acceptable

* 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 the users.

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 consequence • if it is part of an entity PK, •• change the entity PK's in all related relationship relations •• change PK's of other entities which use this value as part of their PK's (for example, DEPENDENTS' PK's 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)

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

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 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 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 caustiously (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 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, non-key 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 non-key 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 non-key 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 vs. 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 PK's of entities in relationships and the value set names have been ommitted.

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 data base design.* 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. 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,...) 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 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 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.

---

* 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.

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 same as (2) (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 Figure 19(a)) [20]. The corresponding entity-relationship diagram is shown in Figure 19(b).

It is clear now that the arrows in a data structure diagram do not always represent relationships of entities. Even in the case that an arrow represents a 1:n relationship, the arrow only represents an uni-directional relationship [10] (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 entities 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) in the network model we must create a relationship record DEPARTMENT-EMPLOYEE. 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 data bases 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 in the 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: change the diamonds to boxes, and 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 "EMPLOYEE-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 treated as values. Note that 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,* while n-ary relationships may be used in the entity-relationship model.

---

* In DIAM II [25], n-ary relationships may be treated as special cases of identifiers.

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.

ACKNOWLEDGEMENT

The author wishes to express his thanks to George Mealy, Stuart Madnick, Murray Edelberg, Susan Brewer, Stephen Todd, and the referees for their valuable suggestions (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, Klimbie, J.W. and Koffeman, K.L. (Eds.), North-Holland Publishing Co., Amsterdam, 1974, pp. 1-60.

2. Bachman, C.W. Software for random access processing. Datamation (April, 1965), pp. 36-41.

3. Bachman, C.W. Data structure diagrams. Data Base 1,2 (Summer 1969).

4. Bachman, C.W. Trends in database management- 1975. AFIPS Conf. Proc. Vol. 44 (June, 1975), 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, 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), pp. 377-387.

9. Codd, E.F. Normalized data base structure: a brief tutorial. Proc. ACM-SIGFIDET 1971, Workshop, San Diego, Calif., November, 1971, pp. 1-18.

10. Codd, E.F. A data base sublanguage founded on the reltional calculus. Proc. ACM-SIGFIDET 1971, Workshop, San Diego, Calif., November, 1971, pp. 35-68.

11. Codd, E.F. Recent investigations in relational data base systems. Proc. IFIP Congress 74, North-Holland Publishing Co., Amsterdam, pp. 1017-1021.

12. Deheneffe, C., Hennebert, H., and Paulus, W. Relational model for data base. Proc. IFIP Congress 74, North-Holland Publishing Co., Amsterdam, pp. 1022-1025.

13. Dodd, G.G. APL -- a language for associate data handling in PL/I. Fall Joint Computer Conference, 1966.

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., September, 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 74, North-Holland Publishing 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., September, 1975, pp. 25-47.

17. Lindgreen, P. Basic operations on information as a basis for data base design. Proc IFIP Congress 74, Stockholm, Sweden, August, 1974, pp. 993-997.

18. Mealy, G.H. Another look at data base. AFIPS Conference Proc., Vol. 31 (1967), pp. 525-534.

19. Nijssen, G.M. Data structuring in the DDL and the relational model. In Data Base Management, Klimbie, J.W. and Koffeman, K.L. (Eds.), North-Holland Publishing Co., Amsterdam, 1974, pp. 363-379.

20. Olle, T.W. Current and future trends in data base management systems. Proc. IFIP Congress 74, North-Holland Publishing 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., September, 1975, pp. 144-172.

22. Rustin, R. (Ed.) Proc. ACM-SOGMOD 1974 - debate on data models, Ann Arbor, Michigan, May, 1974.

23. Schmid, H.A., and Swenson, J.R. On the semantics of the relational model. Proc. ACM-SIGMOD 1975, Conferance, San Jose, Calif., May, 1975, pp. 211-233.

24. Senko, M.E., Altman, E.B., Astrahan, M.M., and Fehder, P.L. Data structures and accessing in data-base systems. IBM Systems Journal, No. 1(1973), pp. 30-93.

25. Senko, M.E. Data description language in the concept of multilevel structured description: DIAM II with FORAL. In Data Base Description, Dougue, B.C.M., and Nijssen, G.M. (Eds.), North-Holland Publishing Co., Amsterdam, pp. 239-258.

26. Sibley, E.H. On the equivalence of data base systems. Proc. ACM-SIGMOD 1974 debate on data models, Ann Arbor, Mich., May, 1974, pp. 43-76.

27 Steel, T.B. Data base standardization - a status report. Proc. ACM-SIGMOD

1975 Conf., San Jose, Calif., May, 1975, pp. 65-78.

28. Stonebreaker, M. Implementation of integrity constraints and views by query modification. Proc. ACM-SIGMOD 1975 Conf., San Jose, Calif., May, 1975, pp. 65-78.

29. Sundgren, B. Conceptual foundation of the infological approach to data bases. In Data Base Management, Klimbie, J.W., and Koffeman, K.L. (Eds.), North-Holland Publishing Co., Amsterdam, 1974, pp. 61-95.

30. Taylor, R.W. Observations on the attributes of database sets. In Data Base Description, Douge, B.C.M., and Nijssen, G.M. (Eds.), North-Holland Publishing Co., Amsterdam, pp. 73-84.

31. Tsichritzis, D. A network framework for relation implementation. In Data Base Description, Dougue, B.C.M., and Nijssen, G.M. (Eds.), North-Holland Publishing Co., Amsterdam, pp. 269-282.

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

Figure 2. Attributes defined on the entity set PERSON.

Figure 3. Attributes defined on the relationship set PROJECT-WORKER.

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

Figure 5. Information about relationships in a relationship set (table form).

Figure 6. Representing entities by values (employee numbers).

Figure 7. Regular entity relation EMPLOYEE.

Figure 8. Regular relationship relation PROJECT-WORKER.

Figure 9. A weak entity relation DEPENDENT.

FIGURE 10. A simple entity-relationship diagram.

Figure 11. An entity-relationship diagram for analysis of information in a manufacturing firm.

Figure 12. Relation EMPLOYEE.

Figure 13. Relation EMPLOYEE-PROJECT.

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

Figure 15. Relation SHIP.

Figure 16. Relationship DEPARTMENT-EMPLOYEE. (a) data structure diagram. (b) entity-relationship diagram.

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

Figure 18. Data-structure-set defined on the same record type.

Figure 19. Relationship MARRIAGE. (a) data structure diagram. (b) entity-relationship diagram.

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

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

Figure 22. The entity set view.

Figure 23. An "entity description" in the entity set model.

Fig. No. 1

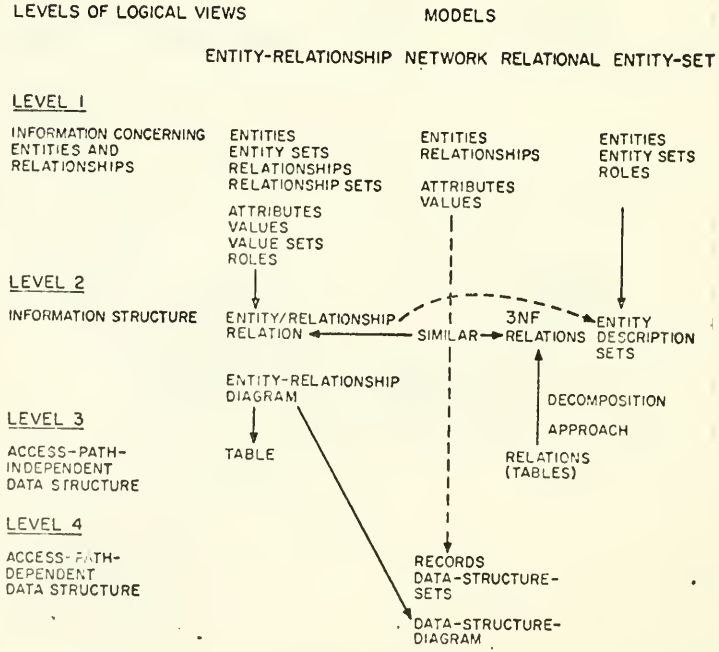

Fig. No. 2

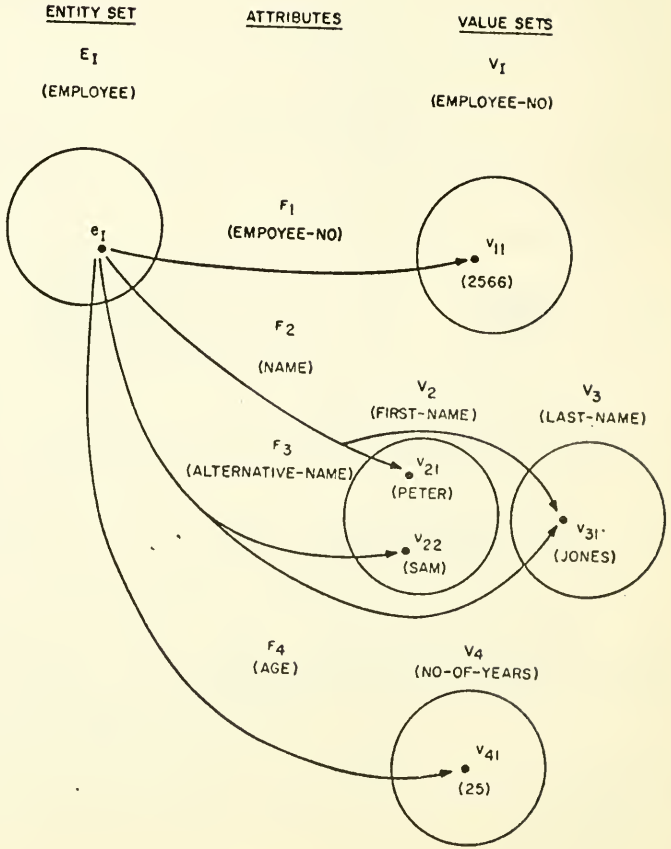

Fig. No. 3

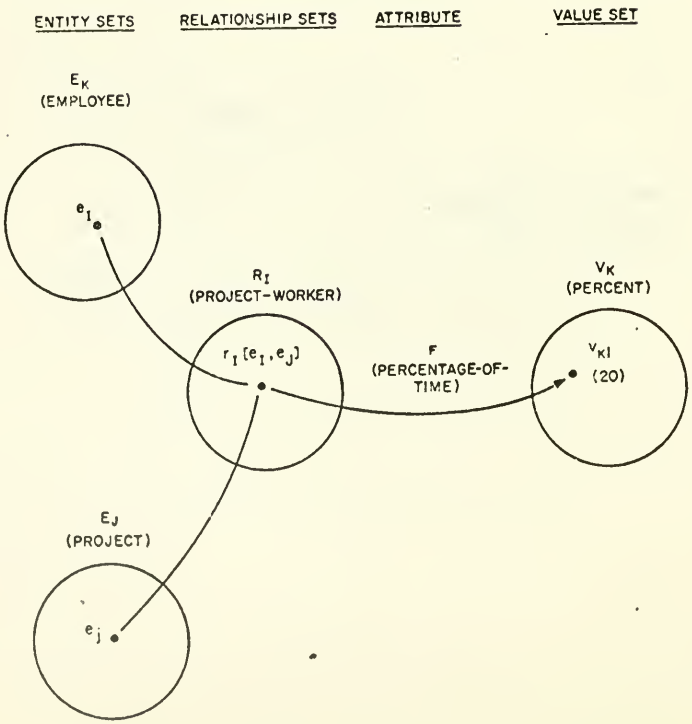

Fig. No. 4

ATTRIBUTE / ENTITY SET / AND VALUE SET	F_1 (EMPLOYEE-NO)	F_2 (NAME)		F_3 (ALTERNATIVE-NAME)		F_4 (AGE)
E_1 (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)
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Fig. No. 5

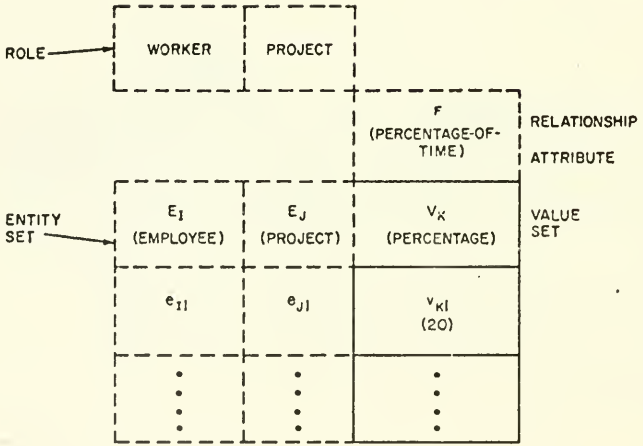

Fig. No. 6

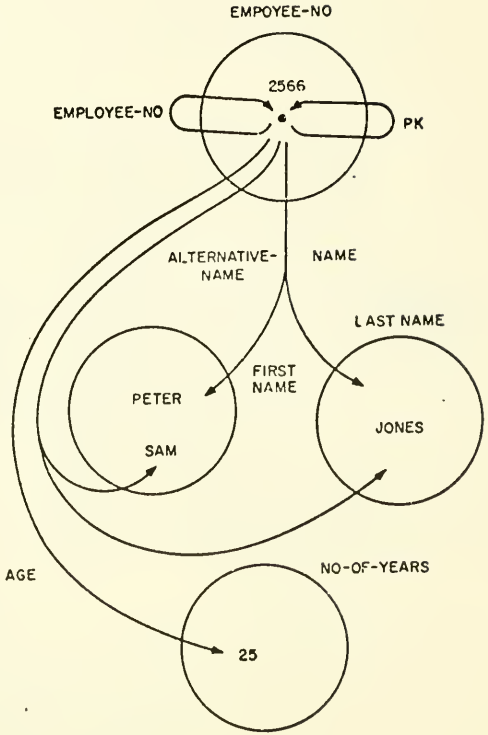

Fig. No. 7

PRIMARY KEY

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
	⋮	⋮	⋮	⋮	⋮	⋮

RELATIONAL VIEW

Fig. No. 8

PRIMARY KEY

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

RELATIONSHIP ATTRIBUTE

RELATIONAL VIEW

Fig. No. 9

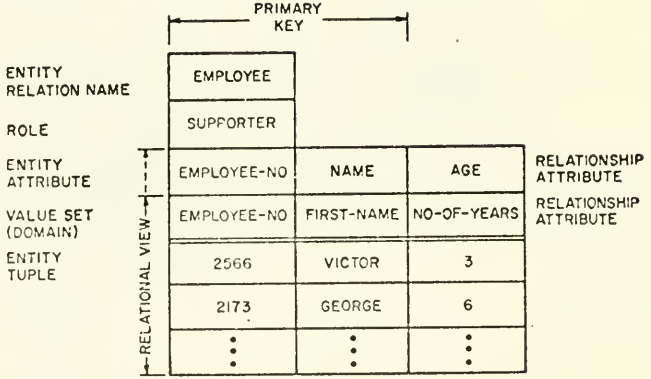

Fig. No. 10

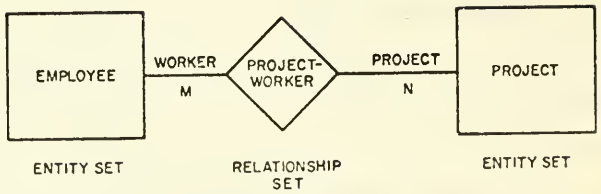

Fig. No. 11

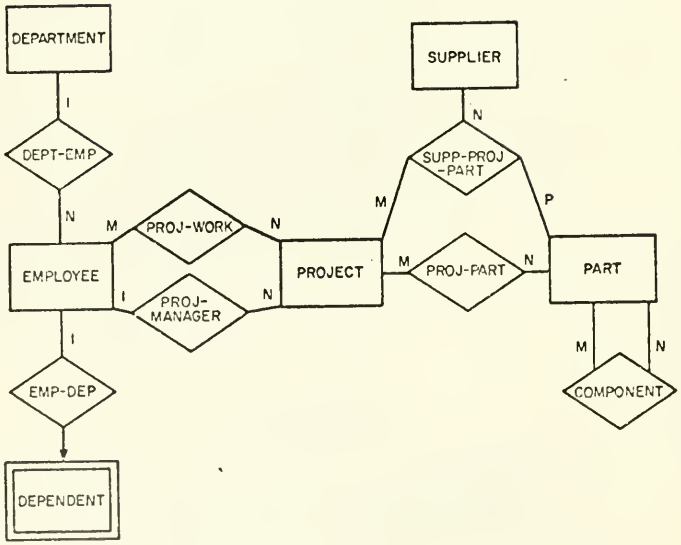

Fig. No. 12

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. No. 13

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

Fig. No. 15

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

Fig. No. 14

		LEGAL	LEGAL	ALTERNATIVE	ALTERNATIVE	
PROJECT-NO	EMPLOYEE-NO	FIRST-NAME	LAST-NAME	FIRST-NAME	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. No. 16

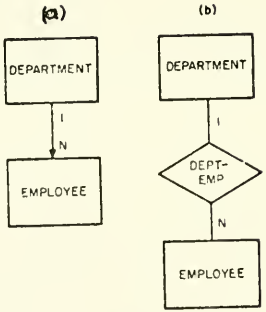

Fig. No. 17

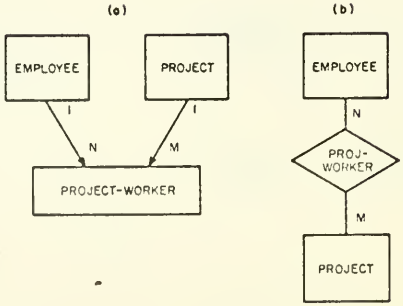

Fig. No. 18

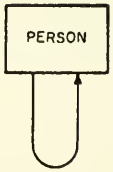

Fig. No. 19

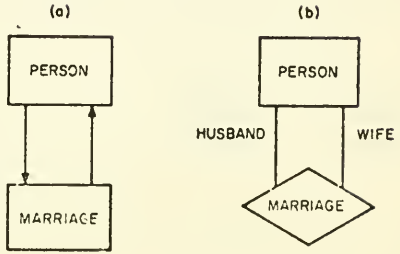

Fig. No. 20

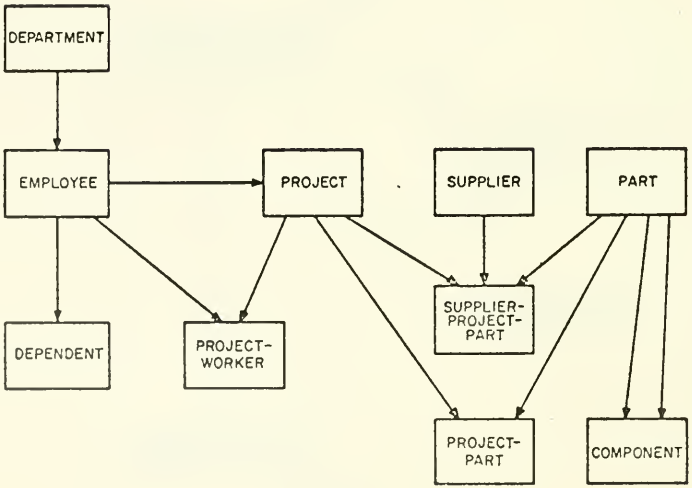

Fig. No. 21

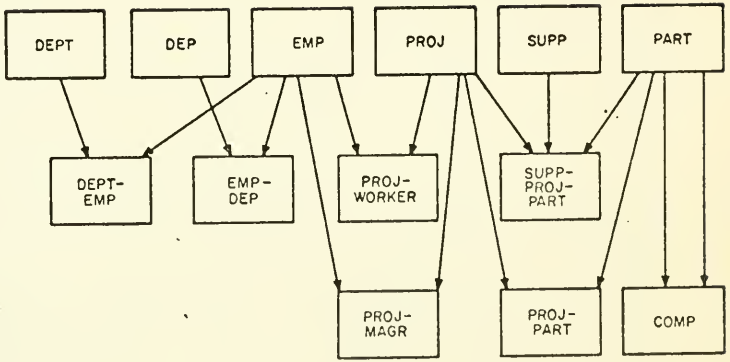

Fig. No. 22

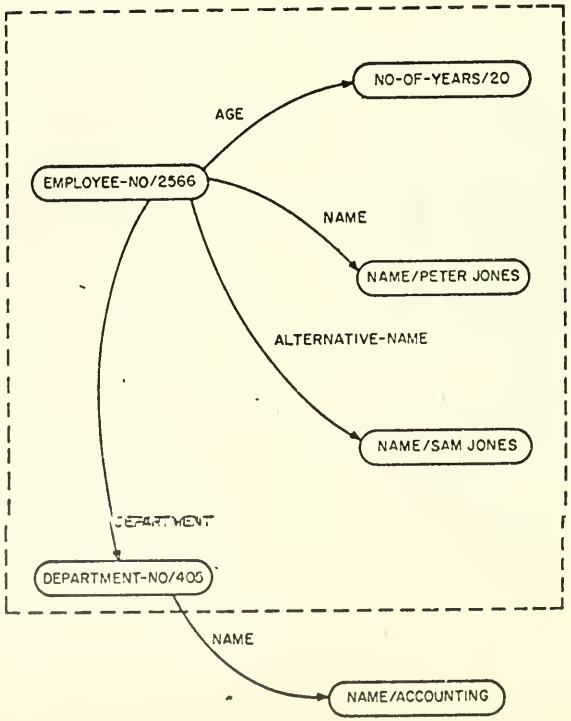

Fig. No. 23

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