The entity-relationship model— A basis for the enterprise view of data

by PETER PIN-SHAN CHEN
Massachusetts Institute of Technology
Cambridge, Massachusetts

ABSTRACT

The concept of the enterprise view of data is very useful in the database design process and in the construction of conceptual schema. This paper discusses the use of the entity-relationship approach in describing and maintaining the enterprise view of data. Fundamental operations for changing the enterprise schema are presented. Finally, an example is given to show the differences between the entity-relationship approach and the data-structure approach in modeling the enterprise view of data.

INTRODUCTION

The subject of the logical view of data has attracted considerable attention in the past ten years. However, most researchers have focused on the user view of data. The need for studying the enterprise view of data was not recognized until recently. Different users of a database may have different views of the database, but the enterprise should have a unique and consistent view of the database. This is particularly important in designing a logically meaningful and consistent database. The concept of the enterprise view of data is very useful in the database design process and in the design of conceptual schema.

Enterprise view and database design

Database design is a process to organize data into a form which matches the underlying data model of the database management system. There are three major types of database management systems: network, hierarchical, and relational. In the network database management systems, which include Honeywell's IDS and UNIVAC' DMS-1100, data will be organized into different types of records and can be represented by a data-structure diagram[1] (see Figure 1). In the hierarchical database management systems, which include IBM's IMS, data will be organized into a form similar to but more restricted than the data-structure diagram. In the relational database management systems,[2] data will be organized into a set of tables (or "relations"). In general, to design a database is to decide how to organize data into specific forms (record types, tables) and how to access them. Up to now, there are very few tools available to aid the database design process. Usually, the database designer relies on his own intuition and experience. Thus, the resulting database may not satisfy company's objectives and may cause problems in company's operations.

Another related problem in database design is that the output of the database design process—the user schema (a description of the user view of data)—is not a "pure" representation of the real world. One of the reasons is that the database designer is restricted by the limited capabilities of the database management system. For example, the many-to-many relationships between entities are difficult to represent directly in some database systems. Another reason is that the user schema may contain some features related to the storage representation of the database. For instance, it may describe which record types can be directly accessed and how to access other record types. In addition, the user schema is usually designed to be efficient for a certain type of data processing operations. For example, the data about employees may be grouped into two record types, employee-master and employee-detail, to improve the retrieval performance. Therefore, the user schema is usually not a direct representation of the real world. This makes the user schema difficult to understand and difficult to change.

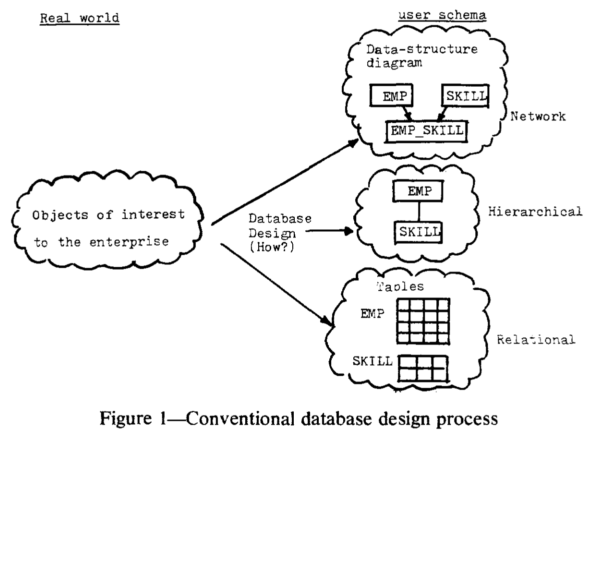


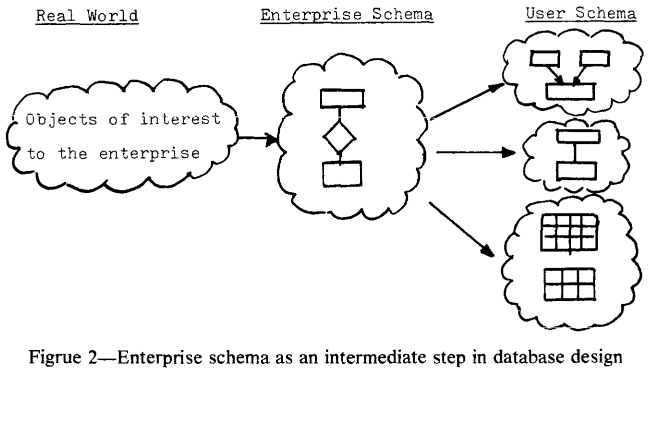
Figure 1—Conventional database design process

A possible solution to the above problems is to introduce an intermediate stage in the database design process: defining the enterprise schema, which is a "pure" representation of the real world and is independent of storage and efficiency considerations. The enterprise schema will then be translated into different types of schemata for different database management systems (see Figure 2). It can also be translated into several schemata for the same database management system to optimize different types of data processing operations. There are several advantages of this approach:

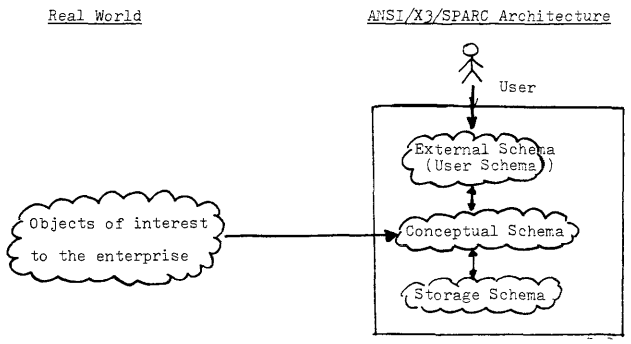
(1) The enterprise schema is easier to understand than a user schema since the former does not have the restrictions of the underlying database management system;
(2) The enterprise schema is more stable than the user schema, since some types of changes in the user schema may not require any change in the enterprise schema. If the enterprise schema needs to be changed to reflect the changes in the enterprise environment, the changes can be performed easily since efficiency and storage issues are not considered.

Enterprise view and conceptual schema

What is the difference between the enterprise schema and the conceptual schema proposed by the ANSI/X3/SPARC group?[3] Basically, they are very similar since both are descriptions of the enterprise view of data. In the SPARC's approach, the conceptual schema serves as the interface between the external schema (user view of data) and the storage schema (physical view of data) (see Figure 3). The requirement of serving as an interface between two other schemata may introduce some undesirable features into the conceptual schema. If this restriction on the conceptual schema is ignored, there is almost no difference between the conceptual schema and the enterprise schema. Therefore, the techniques discussed in this paper are also suitable for describing and maintaining the conceptual schema in the SPARC's architecture.



Figrue 2—Enterprise schema as an intermediate step in database design


Real World
ANSI/X3/SPARC Architecture
User
External Schema (User Schema)
Objects of interest to the enterprise
Conceptual Schema
Storage Schema
Figure 3—Enterprise view and conceptual schema

Approach used in the paper

In order to describe the enterprise view of data, a mental framework to model the real world is needed. Different people may be used to different mental frameworks. The mental framework used in this paper is the Entity-Relationship (E-R) model.[4,5] The E-R model and similar approaches[6-9] have been found useful in modeling the real world. A diagrammatic technique called the Entity-Relationship (E-R) diagram will be used in this paper to represent the enterprise view of data.

This paper is divided into three parts. The first part discusses how to use the E-R model and diagrammatic technique to describe the enterprise view of data. This is an extension of the work reported in Reference 5. The second part describes fundamental operations for changing the enterprise view of data. This is an area where very little work has been done. The operations proposed in this paper will be useful in maintaining the enterprise schema. The third part uses the E-R approach to analyze an example given by Bachman[10] concerning changes in the conceptual schema.

MODELING THE REAL WORLD USING THE ENTITY-RELATIONSHIP MODEL AND DIAGRAMMATIC TECHNIQUE

In this section, we shall use examples to show how to use the Entity-Relationship (E-R) model and diagrammatic technique to describe the enterprise view of data. A more formal definition of the model can be found in Reference 5.

It is assumed that the responsibility of defining and maintaining the enterprise schema belongs to a person called the *enterprise administrator*. The following is the suggested procedure for the enterprise administrator to define the enterprise schema:

(1) *identify entity sets of interest to the enterprise*
An *entity* is a "thing" which can be distinctly identified. According to the needs of the enterprise, entities can be classified into different *entity types* such as EMPLOYEE, STOCK_HOLDER. An *entity set* is a group of entities of the same type. In the E-R

EMPLOYEE
STOCK_HOLDER

Figure 4—Entity sets

diagram, an entity set is represented by a rectangular-shaped box (see Figure 4). The terms, "set" and "type," can be interchanged in the E-R diagram. The reader may use either one to interpret the E-R diagram.

There are many "things" in the real world. In addition, different enterprises may view the same thing differently. It is the responsibility of the enterprise administrator to select the entity types which are most suitable for his company.

(2) *identify the relationship sets of interest to the enterprise*

Entities are related to each other. Different *types* of relationships may exist between different types of entities. A *relationship set* is a set of relationships of the same type. For example, PROJ_EMP, which describes the assignment of employees to projects, is a relationship set defined on two entity sets, EMP and PROJ. A relationship set can also be defined on more than two entity sets. For example, PROJ_SUPP_PART is a relationship set defined on three entity sets PROJ, SUPP, and PART. In the entity-relationship diagram, a relationship set is represented by a diamond-shaped box with lines connecting to the related entity sets (see Figure 5). The "m" and "n" associated with the PROJ_EMP relationship in the E-R diagram indicate that the relationship is an m:n mapping. That is, each employee may be associated with several projects, and each project may have several employees. In certain companies, each employee belongs to at most one project, and the PROJ_EMP relationship is a 1:n mapping.

There are many types of relationships between entities. The responsibility of the enterprise administrator is to select the relationship sets (or types) which are of interest to the enterprise. He also has to specify the type of mappings (1:1, 1:n, m:1, or m:n) of the relationships.

(3) *identify relevant properties of entities and relationships (i.e., define value sets and attributes)*

Entities and relationships have properties, which can be expressed in terms of *Attribute-value* pairs. "Blue," and "4" are examples of *values*. Values can be classified into different *types* such as COLOR or QUANTITY. A *value set* is a group of values of the same type. An *attribute* is a mapping from an entity set (or a relationship set) to a value set (or a group of value sets). For example, "address" is an attribute which maps entities in the entity set EMP to values in the value set NAMES_OF_LOC. Note that we relax the constraint imposed in Reference 5 that the mapping from the entity set to the value set has to be a function (i.e., m:1 mapping). In other words, we now allow that an attribute (such as address) can have several values (such as locations) for the same entity (employee). This relaxation in the definition of attribute will make the changes in the enterprise view simpler. This point will become clear in the next section.

In the E-R diagram, a value set is represented by a circle, and an attribute is represented by an arrow directed from the entity set (or the relationship set) to the desired value set(s) (see Figure 6). After selecting entity sets and relationship sets, the enterprise administrator identifies the attributes and value sets which are relevant to the company's operations.

The three steps stated above cover a major part of the enterprise schema. For simplicity, we shall not discuss in this paper other issues related to the enterprise schema such as integrity constraints.

To design a database, the enterprise administrator first draws an E-R diagram such as the one shown in Figure 7. He then drew the attributes and value sets for each entity set and relationship set. The E-R diagram is then translated into a data-structure diagram or a set of tables ("relations") (see Figure 2). The rules and procedures used in the translation process were discussed in Reference 5. Here, we shall investigate how to change the enterprise schema (the E-R diagram) itself.

MODIFICATION OF THE ENTERPRISE VIEW

Although the enterprise schema is more stable than a user schema, it still needs to be changed from time to time

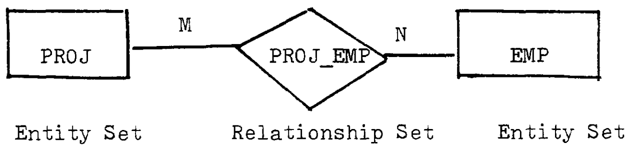


Figure 5—Relationship set

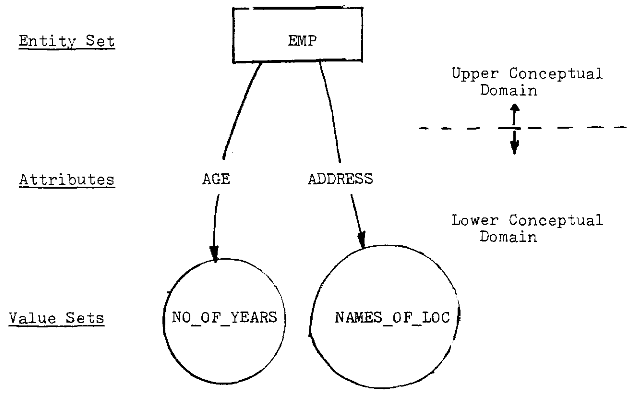


Figure 6—Attributes and value sets

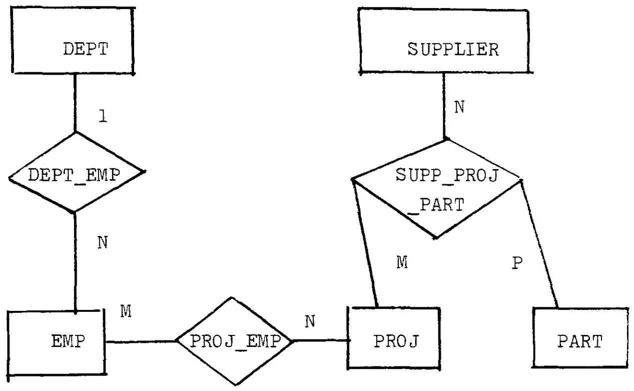


Figure 7—An entity-relationship diagram (with entity sets and relationship sets only)

to reflect the changes in the enterprise environment. Excepting a paper by Bachman,[10] very little work has been done in this area. In this paper, we use the E-R model as a basis for analyzing different types of changes in the enterprise view of data. We not only propose a set of operations but also analyze the consequences of these operations.

There are five basic types of operations: *add, delete, split, merge,* and *shift*. The first four operations are applicable to entity sets, relationship sets, attributes, and value sets. The *shift* operation is used when the enterprise administrator would like to view a value set in the old enterprise schema as an entity set in the new schema or vice versa. It is useful to think that the E-R diagram consists of two conceptual domains: (1) the upper conceptual domain which consists of entity sets and relationship sets; (2) the lower conceptual domain which consists of attributes and value sets. We shall discuss the first four operations in both the upper and lower conceptual domains. Finally, we shall discuss the shifting an entity set from the upper conceptual domain to the lower conceptual domain and the shifting a value set in the opposite direction.

Operations in the upper conceptual domain

The following are the basic operations applicable to entity sets and relationship sets:

(1) *Split an entity into several subsets*
For instance, the entity set EMP in Figure 8a can be split into two entity sets: MALE_EMP AND FEMALE_EMP in Figure 8b. The consequence of this operation is that the relationship sets associated with the entity set may also have to be split. For example, PROJ_EMP is split into PROJ_M_EMP and PROJ_F_EMP (see Figure 8b).

(2) *Merge several entity sets into one entity set*
This is the opposite operation of (1). The consequence is that the related relationship sets may have to be merged.

(3) *Split a relationship set into several subsets*
An example of this operation is: the relationship set PROJ_EMP in Figure 8a can be split into two relationship sets, PROJ_MANAGER and PROJ_WORKER, in Figure 8c. Note that the type of mapping in the new relationships may be different from that in the original relationship. For instance, the mapping in PROJ_MANAGER is 1:n while the mapping in PROJ_EMP is m:n.

(4) *Merge several relationship sets into one set*
This is the opposite operation of (3). Note that these relationship sets have to be defined on the same group of entity sets.

(5) *Add a new entity set*
For example, a new entity set called SUPPLIER may be added to the E-R diagram in Figure 8a. The result is shown in Figure 9a. Note that it is possible to have stand-alone entity sets in the enterprise schema, although in many cases relationships between the new entity set and the existing entity sets are established immediately (see the next operation).

(6) *Add a new relationship set*
We may add a new relationship set for the new entity set such as the relationship set PROJ_SUPP in Figure 9b. We may also add a new relationship set for existing entity sets such as the relationship set PROJ_MANAGER in Figure 9b.

(7) *Delete an entity set*
For instance, after deleting the entity set EMP in Figure 9b, we have Figure 9c. The consequences are: (i) the relationship sets related to the entity set are

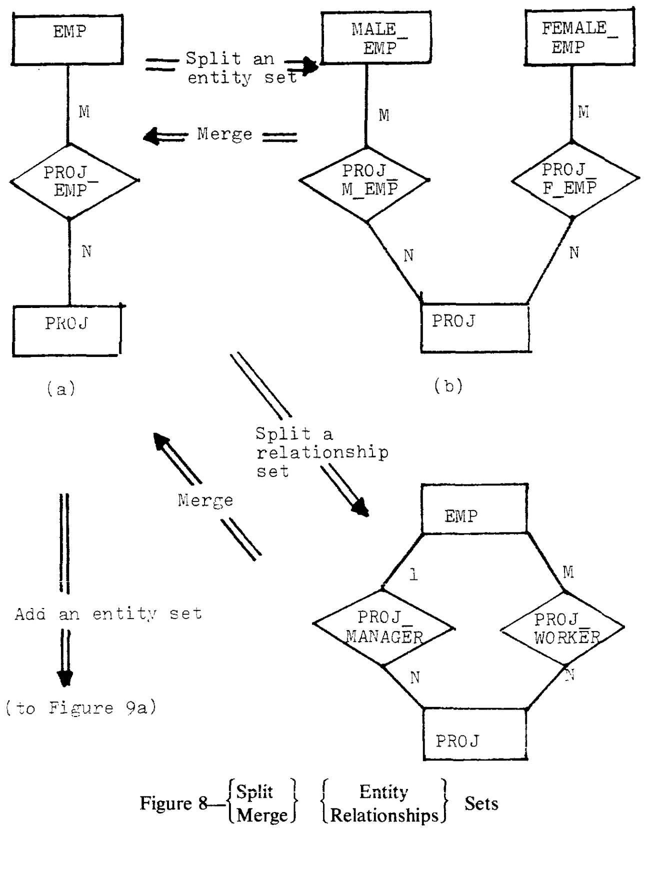


Figure 8—$\left\{\begin{matrix}\text{Split}\\ \text{Merge}\end{matrix}\right\}$ $\left\{\begin{matrix}\text{Entity}\\ \text{Relationships}\end{matrix}\right\}$ Sets

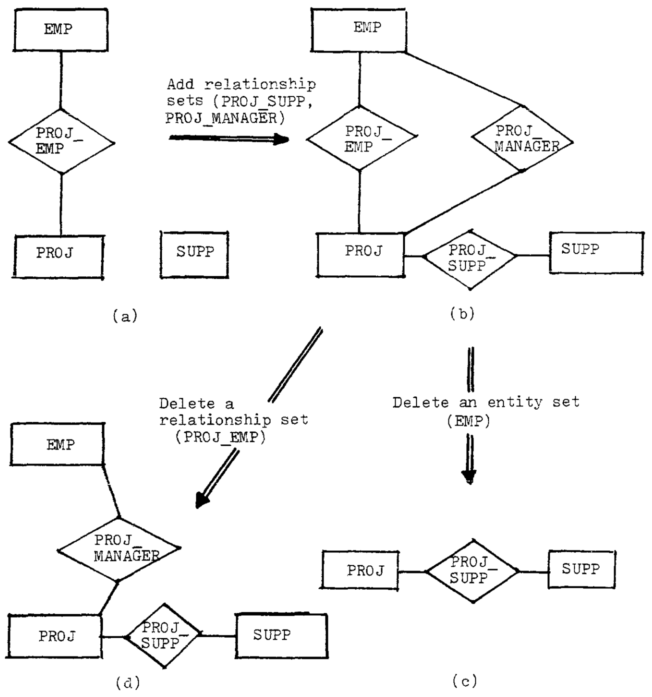

EMP
PROJ_EMP
PROJ
SUPP
Add relationship sets (PROJ_SUPP, PROJ_MANAGER)
PROJ_MANAGER
PROJ_SUPP
(a)
(b)
Delete a relationship set (PROJ_EMP)
Delete an entity set (EMP)
(d)
(c)

Figure 9—{Add / Delete} {Entity / Relationship} Sets

also deleted; (ii) attributes related to the deleted entity set and related relationship sets are also deleted.

(8) *Delete a relationship set*
An example is: delete the relationship set PROJ_EMP in Figure 9b, and the result is shown in Figure 9d. The consequence of this operation is that the attributes of the relationships are deleted (not shown in Figure 9d).

Operations in the lower conceptual domain

Assume that the entities in the entity set EMP have two attributes, LEGAL_NAME and PHONE, which map the entities to the value sets NAME and PHONE_# (see Figure 10a). We shall use these attributes and value sets as the basis for the discussion of the following operations:

(1) *Add a value set*
For example, a new value set called DOLLARS may be added to Figure 10a. The result is shown in Figure 10b. Usually, this operation is followed by an "add attribute" operation.
(2) *Delete a value set*
After deleting the value set PHONE_# in Figure 10a, we get Figure 10c. The consequence is that all attributes associated with this value set will be deleted.
(3) *Split a value set into several subsets*
The value set NAMES in Figure 10a may be split into two value sets FIRST_NAMES and LAST_NAMES in Figure 10d. The consequence is that attributes related to the value set may have to be adjusted. Although the attribute LEGAL_NAME is not split in Figure 10d, it is possible to split it into two attributes: LEGAL_FIRST_NAME and LEGAL_LAST_NAME. It is the responsibility of the enterprise administrator to make this decision.
(4) *Merge several value sets into a value set*
This is the opposite operation of (3).
(5) *Add an attribute*
For instance, Figure 11b is obtained by adding the attribute OTHER_NAME to Figure 11a.
(6) *Delete an attribute*
Deleting the attribute LEGAL_NAME from Figure 11a, we have Figure 11c. The value set associated with the attribute will be deleted by another operation ("delete value set") if desired. In some cases, the value set may be still associated with other attributes (see Figure 11c).
(7) *Split an attribute into several attributes*
For example, Figure 11d is obtained by splitting the attribute PHONE in Figure 11a into two attributes, OFFICE_PHONE and HOME_PHONE.
(8) *Merge several attributes into one attribute*
This is the opposite operation of (7). The attributes have to be defined on the same entity set (or relationship set).

Operations between two conceptual domains

Assume that there are two entity sets (EMP and PROJ), one relationship set (PROJ_EMP), four value sets (NAMES_OF_PLACES, SOC_SEC_#, PHONE_#, and

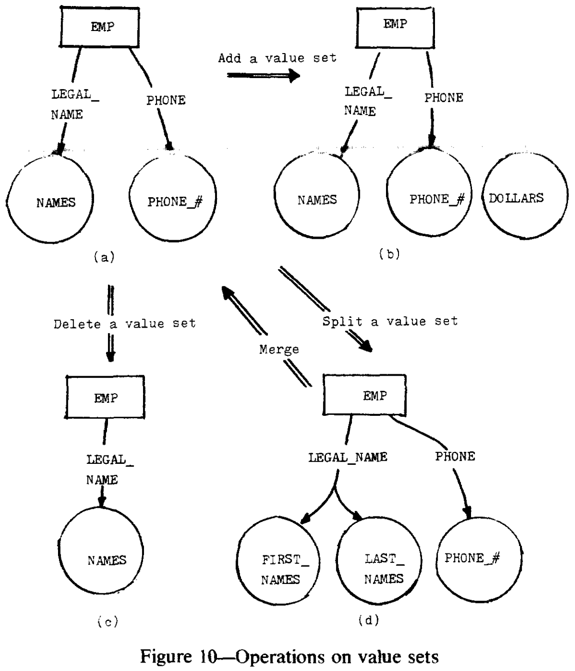


Figure 10—Operations on value sets

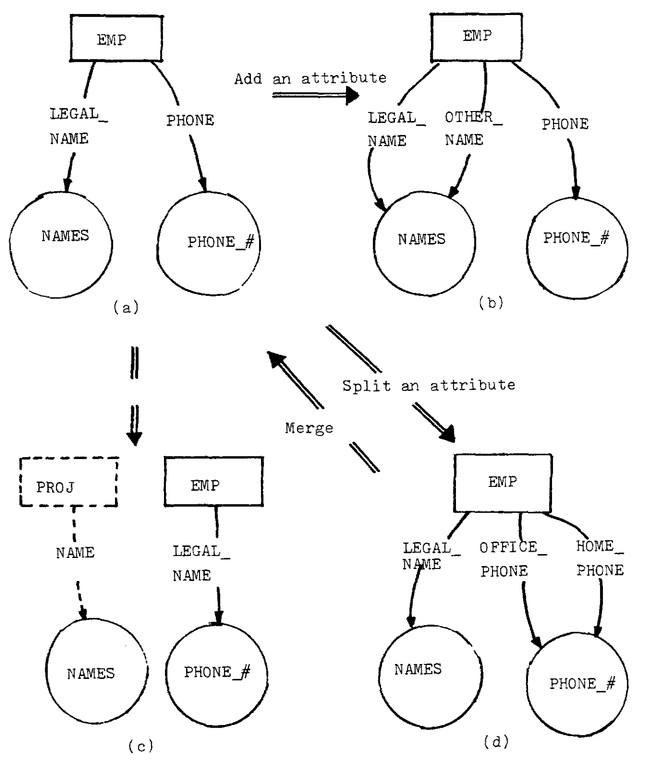


Figure 11—Operations on attributes

ANALYSIS OF AN EXAMPLE

In a recent paper, Bachman[10] uses data-structure diagrams to illustrate the changes in a conceptual schema. In this section, we shall first state his example and then use E-R diagrams to interpret his example.

Description of the example using data-structure diagrams

The following is a simplified version of Bachman's example:

(a) In the beginning, the enterprise administrator declared a conceptual schema as shown in Figure 13a. The reader is assumed to have some knowledge of the data-structure diagram.[1] Simply speaking, a rectangular-shaped box represents a record type, and an arrow represents a data-structure-set (i.e., 1:n relationship between record types). In Figure 13a, there are two types of conceptual records, COMPANY and PERSON, and a data-structure-set "a" representing the fact that each person is associated with exactly one company and that each company has a set of personnel.

(b) Later, the enterprise administrator recognized that the personnel of the company were persons in their own right. This fact may be discovered at the merger

PROJ_NAMES), and four attributes (ADDRESS, SOC_SEC_NO, PHONE, and NAME) as shown in Figure 12a. We shall use them as the basis for the discussion on the following operations:

(1) *Shift a value set from the lower conceptual domain to the upper conceptual domain*
When the enterprise environment changes, it may become natural to view PLACE as an entity set instead of a value set. Thus, in Figure 12b "ADDRESS" becomes a relationship set, and "PLACE" has an attribute "NAME" which points to the value set NAMES_OF_PLACES. Since PLACE is an entity set, we may establish new relationships of it with other entity sets such as PROJ or add more attributes and value sets to describe properties of "places."

(2) *Shift an entity set from the upper conceptual domain to the lower conceptual domain*
When the enterprise environment changes again, it may become natural to view PROJ as a value set instead of an entity set. In Figure 12c, PROJ is deleted from the upper conceptual domain, and the relationship set PROJ_EMP becomes the attribute INVOLVED_PROJ. The entity set PROJ in Figure 12b may have been associated with several value sets, but only the value set PROJ_NAMES which is used to identify the entities PROJ remains in the lower conceptual domain.

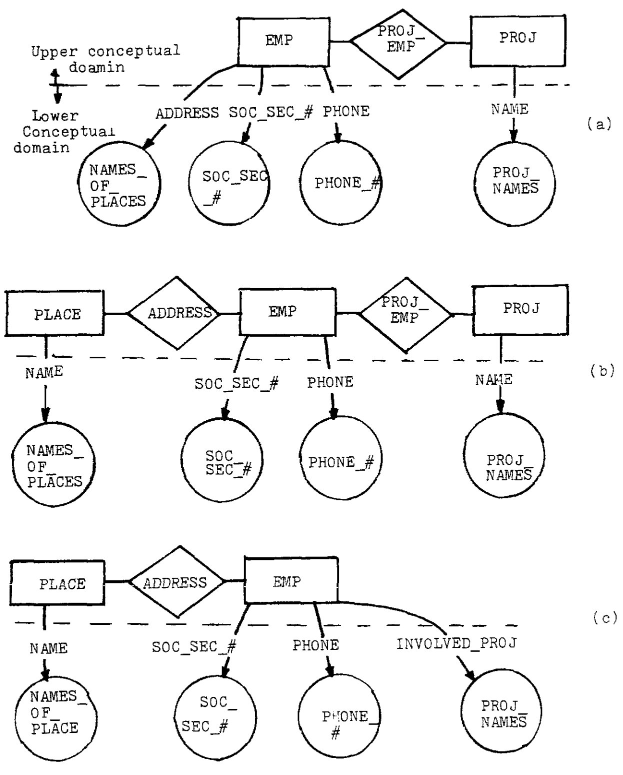


Figure 12—Shifting a set from the upper conceptual domain to the lower conceptual domain and vice versa

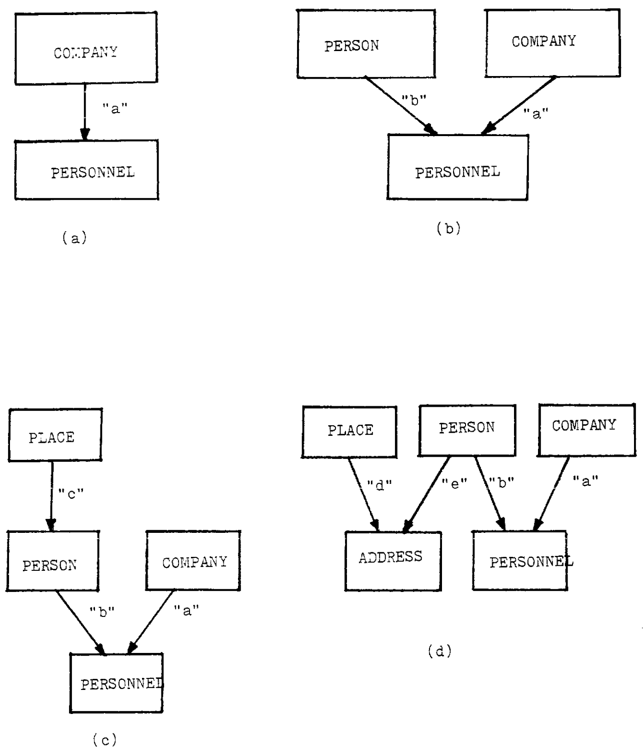


Figure 13—Expressing changes in the enterprise view using data-structure diagrams

of several companies that some of the persons held two jobs and were personnel to two of the merged companies. Figure 13b illustrates the data-structure diagram for the new conceptual schema. Basically, the old personnel type record has been split into two record types, PERSONNEL and PERSON. The "PERSON" has attributes NAME and ADDRESS (not shown in the figure).

(c) After a while, the enterprise administrator decided to factor the address of residence out of the person record. Figure 13c illustrates the addition of the "PLACE" conceptual record type and the data-structure-set type "c." It was also assumed that each person has a unique address (place).

(d) It is now recognized that people move from place to place and that it is desirable to know current address as well as past addresses. Another reason may be: it is discovered that a person may have more than one address. In either case, a new conceptual record type ADDRESS is added to the conceptual schema (see Figure 13d).

Analysis using entity-relationship diagrams

In the following, we shall use E-R diagrams to explain the above example:

(a) The E-R diagram in Figure 14a is corresponding to the data-structure diagram in Figure 13a. There are two types of entities, PERSON and COMPANY, in the enterprise view. Since the mapping between COMPANY and PERSON is 1:n, the relationship set PERSONNEL is represented by a data-structure-set "a" in Figure 13a.

(b) Figure 14b is the corresponding E-R diagram for Figure 13b. Since the relationship set PERSONNEL is an m:n mapping, it is represented by a relationship record type PERSONNEL and two data-structure-sets "a" and "b" in Figure 13b. Note that Figures 14a and 14b have the same entity sets and relationship set in the upper conceptual domain, and the difference is the type of mapping between the entity sets.

(c) Now the enterprise administrator prefers to view "PLACE" as an entity set rather than a value set. Thus, we have Figure 14c. The attribute ADDRESS in Figure 14b becomes a relationship set in Figure 14c. Since the mapping between PLACE and PERSON is 1:n, the relationship set ADDRESS is represented by the data-structure-set "c" in Figure 13c.

(d) The enterprise administrator discovers that the mapping between PLACE and PERSON is an m:n mapping instead of a 1:n mapping. The new enterprise view is represented by Figure 14d. Since the mapping is m:n, the relationship set ADDRESS is represented by the record type ADDRESS and two data-structure-sets "d" and "e." Note that Figures 14c and 14d

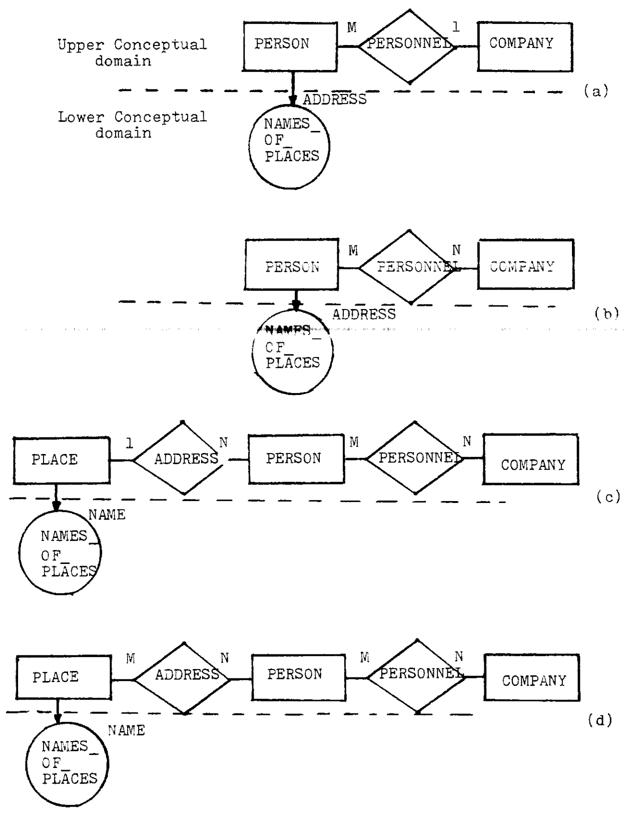


Figure 14—Analysis of Figure 13 using E-R diagrams

are almost the same except that the type of mapping between PLACE and PERSON is different.

In general, the E-R diagram is easier to use to analyze the changes in the enterprise view than the data-structure diagram. Bachman also raised the issue of the ambiguity in Figure 13d: If one wants to modify a person's address, does he have to create a new "address" record or to change the name of the place where the person is living? This question can be easily answered using the E-R approach. Consider Figure 14d. Since the PLACE is an entity set, to change a person's address is to change the relationship between the person and "his place." We should not change the name of the place where the person is living since "NAME" and "NAMES_OF_PLACES" are used to describe a property of the PLACE entities (see Figure 14d).

SUMMARY

The enterprise schema is useful as an intermediate step in database design. In this paper, we have shown how to use the entity-relationship model and diagrammatic technique to describe the enterprise schema. Since the enterprise environment changes from time to time, the enterprise schema will have to change to reflect these changes. Five basic types of operations (add, delete, split, merge, and shift) which are useful in modifying the enterprise schema have been presented, and the consequences of these operations have been discussed. Finally, we have used an example to analyze the differences between the entity-relationship approach and the network approach in modeling the enterprise view of data.

REFERENCES

1. Bachman, C. W., "Data Structure Diagrams," *Data Base 1*, 2, Summer 1969, pp. 4–10.
2. Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," *Comm. ACM 13*, 6, June 1970, pp. 377–387.
3. ANSI, *Interim Report of ANSI/X3/SPARC Group on Database Management Systems*, ANSI, February 1975.
4. Chen, P. P., "The Entity-Relationship Model," (abstract), *Proc. 1st Very Large Database Conf.*, Framingham, Mass., Sept. 1975, ACM.
5. Chen, P. P., "The Entity-Relationship Model: Toward a Unified View of Data," *ACM Tran. on Database Systems 1*, 1, March 1976, pp. 9–36.
6. Moulin, P., J. Randon, M. Teboul, et al., "Conceptual Model as a Database Design Tool," *Proc. IFIP TC-2 Working Conf.*, Jan. 1976, Black Forest, Germany, pp. 459–479.
7. Hall, P., Todd S. Owlett, "Relations and Entities," *Proc. IFIP TC-2 Working Conf.*, Jan. 1976, Black Forest, Germany, pp. 430–458.
8. Deheneffe C. and H. Hennebert, "NUL: a Navigational User's Language for a Network Structured Data Base," *Proc. ACM 1976 SIGMOD Conf.*, Washington, D.C., June 1976, pp. 135–142.
9. Tozer, E. E., "Database Systems Analysis and Design," Technical report, Software Sciences Limited, England, April 1976.
10. Bachman, C. W., "Trends in Database Management—1975," *Proc. AFIPS 1975 NCC*, Vol. 44, AFIPS Press, Montvale, N. J., pp. 569–576.