A new approach to information structuring is presented.

Basic to the structure is the notion of an Entity—an object, concept, or event—and associations among Names for Entities.

Discussed on the basis of these concepts is an Entity Set Model for information structuring.

Data structures and accessing in data-base systems II Information organization

The ultimate purpose of a data-base system is not to read punched cards or magnetic spots on disks. These functions are only incidental aspects of the information representation being used. The real purpose is to store and present valuable structured information in a timely and efficient manner.

Although practical steps have been taken toward achieving data structure independence, the processing of structured information has been constrained by a given technological representation. The appearance of high-performance, random-access technology, however, provides the capability for overriding the constraints of existing representations and for dealing with structured information more naturally.

To capitalize on the random-access capability, we must first understand the properties of information and information processing in their own terms, and not in terms derived from existing representations. Given a more meaningful and stable terminology, we should be better able to discuss and understand the use of particular concepts of existing systems and provide simplified compatible coverage for them in our future systems.

Organization of information

It is not easy to discuss the properties of information, and only a few papers in the computer literature (including a pioneering one by Mealy[1], and later ones by Engles,[2] Meltzer,[3] and Davies[4])

have even attempted to address this problem. Most papers discuss the properties of particular representations for information.

In discussing information structuring we would prefer to propose and define our terms with mathematical accuracy. We find, however, that the state of the art of information structuring contains a complex of ill-defined observations that are far from being reduced to the precision we desire. Thus we see two choices. (1) Discuss those few aspects that may be conveniently described in terms of an existing precise mathematical framework, and relate that framework to our conceptual model. Otherwise (2), seek to construct a complete, faithful description of the intrinsic characteristics of structured information based on concepts that are more appropriate, but less precisely defined, and then indicate investigative directions for seeking better definitions.

Since a global discussion of information is essential to an overall understanding of work in the information systems area, we have chosen the second course. Thus, much of what we present in this section is not presently provable, but hopefully it makes useful common sense now and will yield to better definition in the long term.

structured information

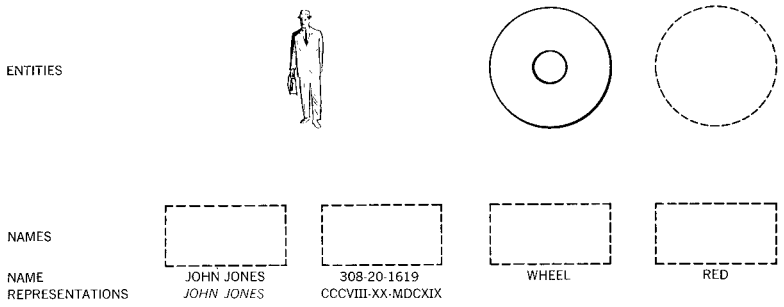
The information content of a particular data-base system is of sufficient value that users are willing to work hard to structure it and to get at least an implicit, general agreement on the meaning of the structure. This structuring has many of the formal aspects of taxonomy, logic, and set theory, which helps in making discussion more specific. There are, however, aspects of information structuring such as identifiers that are not included in existing formal structures.

As we have mentioned, information consists of facts about things. These facts and things exist independently of any representation, but it is essentially impossible to deal with them conceptually except in terms of some representation. As soon as we draw a picture of something or give it a name, we are dealing with the thing in terms of a representation. In a data-base system, we deal with information almost completely in terms of name representations of information.

entities

The top level of Figure 1 shows things of the real world, which we term Entities. It may seem strange to think of the concept color Red as an Entity, but the concept color Red may be as important to the owner of a file as the object Wheel. He might

Figure 1 Relationships among Entities, Names, and Name Representations



wish to name the color and store some structured information about it. An *Entity* is, therefore, defined as anything that has reality and distinctness of being in fact or in thought, e.g., objects, associations, concepts, and events.

When discussing Entities, we seldom use the objects themselves. Rather we use names and named associations between names to stand for Entities. In this paper, we use the term *Name* in its most general sense to mean a symbol or combination of symbols by which a person, place, thing, body, class, or any object of thought is known. Example Names are 6, Red, and Jones.

We further define *Name Representations* because there is often a choice of equivalent coding schemes available for expressing Names. In our thoughts, we use Names in some unknown form to stand for Entities, but in writing these Names or storing them in the computer, we must use Name Representations. It is usually unnecessary to distinguish between Names and Name Representations. Rather, we use the term Name for both. For completeness, however, a *Name Representation* is the Name as expressed in a selected coding scheme. In Figure 1, the man has two Names, and each Name has two printed Name Representations.

entity names

Unique names are needed for identifying Entities whose uniqueness is important to us. For example, it is unnecessary to uniquely identify each wheel in a bin of wheels because, in the properties that are important to us, they are all the same. Differences among people (and in some cases among parts), however, are very often important because those differences define significant unique traits. When such differences are important to the user of a data-base system, then each person Entity in a particu-

lar context (employee within department) must have at least one unique name so that he may be identified unambiguously.

The human process of associating unique names with unique Entities in conversation normally appears somewhat different from the process used by people in interaction with machines. In a dialogue, we need only a few unique names, which we construct—without lifting a finger—from the names for the set of properties and associations that an entity has had over its lifetime (say, the John Jones who married Jane's friend, Karen). Our interactions with computers in terms of names, however, require a pressing of keys at some stage. The cost is high in both time to enter and space to store the large number of property or association names that might be needed to find a unique name by dialogue. This situation motivates us to specify rules for constructing a unique name for an entity out of a finite set of terms.

As shown in Figure 1, individual Entities may have more than one unique name. One reason for this is that an Entity may be important in a number of contexts—family, country, company—and in a particular context it should be as easy as possible to uniquely name the Entity and to remember its name. A small context (family) requires only a selection from a small set of distinctive given names. A large context (country) requires selection from a large set of Social Security numbers. The larger the context, the less distinctive and more difficult the names are to remember. We tend to use the simplest context possible when referring to Entities.

name organization

In thinking about the real world, we seem first to group Entities with similar properties into sets. In this paper, these sets are called *Entity Sets*, and are given Entity Set Names. An *Entity Set Name* is a unique name for an Entity Set. People, part types, and colors could be Entity Set Names. Also in information systems, to provide a context for assigning unique names to individual Entities, we seem to define subsets of Entity Sets based on properties held in common (for example, "is a member of the Jones Family"). In this paper, the unique names for Entities are called *Entity Names*. The sets of unique names are called *Entity Name Sets*, and Entity Name Sets are given *Entity Name Set Names*. An Entity may have an Entity Name in more than one Entity Name Set. For example, a person usually has names in the sets Social Security number and employee number. Entity Name Set Names are made unique across a particular data-base system by creating a hierarchic structure of qualified names such as company name, division name, department name and so forth. Each of such qualified names is unique within the context of the higher level name. If departments in different divisions have the same name, there is no ambiguity because those departments have been qualified by their division names. We realize that these and

Table 1 Entity Set model terminology

	Concept in the real world	*Name for the concept*	*Description of the concept*
Element	Entity	Entity Name	Entity Description
Set	Entity Set	(Entity) Name Set	(Entity) Description Set
Name of set	Entity Set Name	(Entity) Name Set Name	(Entity) Description Set Name

other terms used in this paper may become shortened in practice. To ease the learning procedure for the reader, we use a systematic composition for the terminology in this paper. In certain cases, we can shorten the terminology by dropping the prefix Entity. This is shown in Table 1.

entity names

Within each Entity Name Set, we give each included Entity a unique Entity Name and construct Entity Names in two ways. To guarantee uniqueness in a particular context, we may for that context create a special set of names such as the three-letter air terminal Entity Name Set or the nine-digit Social Security number.

Alternatively, we may use a combination of one or more names of other Entities associated with the Entity to be named. This is possible only where we believe that the combination is unique within the Entity Name Set context. For example, it is highly probable that the combination, FIRST NAME/JOHN, LAST NAME/JONES, BIRTHDATE/JUNE 7, 1938, and BIRTHPLACE/OGDEN, UTAH, is an adequate Entity Name for a person in the United States. Another combination is the Entity Name Set DATE, which constructs unique names for its Entities from a combination of names from the Entity Name Sets MONTH, DAY, and YEAR. It is often thought that such combinations are created simply for programming convenience. We can see, however, that each name within the DATE set is a name for a completely new and different entity. That it is not simply a name for a straightforward combination of the entities within the sets MONTH, DAY, YEAR can be seen by noting that while February and 31 are valid names for Entities in the sets MONTH and DAY, there is no Entity Name Set DATE corresponding to the name MONTH = FEBRUARY, DAY = 31.

Similarly, an Entity may be a unique association between two other Entities, in which case, a unique name for the association may be constructed out of a combination of the names for the two associated Entities. In the association named SUPERIOR PART with COMPONENT PART, one unique name is a combination of SUPERIOR PART NUMBER/17 with COMPONENT PART NUMBER/32. Here again, there is no Entity corresponding to the

combination SUPERIOR PART NUMBER/17, COMPONENT PART NUMBER/17. Therefore, this Entity Name is invalid. Note that no order is necessary among the component names. We need only indicate from which Entity Name Set the component names are drawn.

These concepts are intended to provide a good characterization for all the names of real-world entities that might be stored in a data-base system. With reasonably high confidence, we believe that Entity Names and Entity Name Set Names rather than fields, records, and files are more useful and basic building blocks of structured information.

Descriptions of entities

In some cases, the Name Set Name/Entity Name pair (such as the set of auto body color names used by many departments of motor vehicles) is all the description required of an Entity by the data-base system. In other cases, the system may require a more detailed description of the Entity. People seem to create a more complete structural description by associating a name of the Entity being described with a name of another entity used in a descriptive role. For example, the Entity named PART NAME/GEAR may be described by the Entity named COLOR NAME/BLUE in the descriptive role, COLOR OF PART. In this paper, such an association is given a *Role Name*.

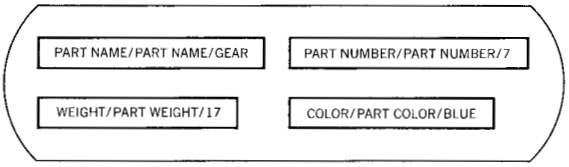
Commonly called a fact, this association is descriptive of both Entities. Thus, "James is the father of John," but also "John is the son of James." This observation is somewhat at variance with Mealy's paper and other sources that postulate a rigid distinction between attributes and identifiers in records where an Attribute/Attribute Value combination (COLOR NAME/COLOR) is primarily a description of the Identifier (PART NAME/GEAR). There is some utility to this notion of rigid distinction, and we will use it later in constructing an Entity Description. Note, however, that there is some measure of reciprocal description.

Decisions as to the form and relative stature in the model of descriptive structures based on the association of Entities are presently a matter for critical judgment. There are at least three possible choices for the selection of the most basic form of description.

two-place association

The first and simplest form involves a two-place association between entities. Each place consists of a unique Entity Name. That name could be composed of a combination of Entity Names. For example, Quantity of Subordinate Part for a given Superior Part may be in one place of an association. The other

Figure 2 An Entity description



place may be an Entity Name composed by combining Entity Names for the Superior Part and the Subordinate Part. Such a form resembles an *n*-place association, and has some underlying structure that can be used by the system (for example, in listing the component parts and quantities of a particular superior part). The dominant essence, however, is that of a two-place association. This type of form can probably be used to express any information that might be stored in a data-base system.

It is simply a matter of judgment whether this is the most desirable form. Its main disadvantage with respect to the next two forms is that it requires a unique name for each type of association, and each added name makes the system more complex and prone to user error. Queries are more difficult to construct because they have to specify a name for each referenced association.

entity set model

A second possible choice, the Entity Set Model, uses Mealy's distinction with regard to asymmetry in the association of Entity Names. In this case, an Entity is described in terms of its associations with other Entities. Here we define the terms Entity Description, Entity Description Set, Role Name, and Identifier for the Entity Set Model.

An *Entity Description* is composed of a set of name triplets that form a description of an identified Entity. Each triplet consists of a Name Set Name, a Role Name, and an Entity Name drawn from the named Name Set as shown in Figure 2.

The Entity being described is uniquely identified by one or more of its Entity Names, which appear as specified subsets of the triplets in the Entity Description. These subsets are called *Identifiers*. PART NAME/PART NAME/GEAR is one Identifier in the example in Figure 2, and PART NUMBER/PART NUMBER/7 is a second independent Identifier.

Each *Role Name* is unique across the Entity Description and indicates the role that its associated Entity plays in describing the identified Entity. To allow reduction in the number of different names used, each Name Set Name may optionally be used as a Role Name in one of the one or more triplets in which

it appears as a Name Set Name, as shown in the preceding Entity Description triplet. As is usual with sets, no order is implied among the name triplets. ("Attribute name," in earlier terminology, is the implicit result of this type of reduction, and only imperfectly performs the two functions required: Name Set Name and Role Name.

The requirement for unique Role Names derives from a desire for simplicity and homogeneity of structure for the Entity Description, which leads to the property that each Entity Description can be represented as a simple vector. We neither need nor wish to complicate the Entity Description form with parameters for specifying multivalued attributes or hierarchic records.

The *Description Set* (Entity Description Set) is a uniquely named set of Entity Descriptions for Entities drawn from a single Entity Set. Since the identified Entities in an Entity Set have similar properties, each Entity Description in a Description Set has the same set of Role Names. No order is implied among the Entity Descriptions of a Description Set.

At this point, it is useful to digress for a moment and discuss terminology. To avoid inconsistencies of attribute-oriented terminology, we have decided to move to an Entity Name and Entity Name Set terminology. It may be difficult to think of Colors—Red, Blue, Green—as Entities, but the Entity named Color/Red can be described by its spectral distribution, even though it is normally used to describe other Entities. We also feel that it is important to distinguish among Entities that exist in the real world, their associated Entity Names, and their associated Entity Descriptions. This distinction is not very clear in earlier documentation, and has sometimes led to such ambiguities as the use of Entity Set as the name for a set of Entity Descriptions rather than a set of Entities. The *Entity Set Model*, which is summarized in Table 1, is the entry mechanism between the world of tangible information external to the data-base system and the Data Independent Accessing Model (DIAM).

In existing systems proposals, the Entity Description concept appears quite frequently, although it is very often embedded in a complex aggregate of inseparable logical and physical considerations. Close parallels exist between the Entity Description and the following concepts:

- *Logical* level of the DBTG Report[5]
- *Segment* of the Information Management System[6]
- *Segment* of the Generalized Information System[7]
- *Level* of a COBOL Record[8]
- *Logical Record* of the Integrated Data Store[9]
- *Relation* in several systems

The main difference is that the Entity Set Model for organizing Names has made a clean, separate logical construct of the Entity Description.

hierarchic model

A third possible choice of the most basic form of description is to use a *hierarchic data structure* such as that described by Mealy. With this choice there is much less need for Description Set Names to provide a context for accessing language search specifications. Queries simply name hierarchic record types. The Time Shared Data Management System (TDMS) is an example of this form of data description. For a particular view of the data, the hierarchic record is essential to the end user or application programmer and, therefore, must be provided, but it does have difficulties relative to the Entity Set Model as a system interface. The basic units (hierarchic records) are neither simple nor homogeneous in structure and, therefore, require a relatively complex description. Users often find that incompatible hierarchies best fit their need for information processing, and there is no objective means for selecting the hierarchy that is to be the system view. As an evolutionary step, we would like to provide a more basic interface that allows multiple logical hierarchies to be defined on it.

DIAM selection

Based on the preceding discussion of possible data descriptive structures for the Data Independent Accessing Model (DIAM), the Entity Set Model best fills our requirements. Thus owners of information to be stored in the system describe their information in terms of the Entity Set Model Name Organization. DIAM then catalogs the information in these terms.

An interface of lesser stature must be provided to realize the convenience of logical hierarchic structures for specific users and to achieve a convenient migration path for users' programs addressed to existing data bases. In DIAM, this basic system interface would be provided by facilities that particular transactions would invoke for naming, defining, and accessing logical hierarchic structures. To use this optional interface, for example, hierarchic records might be composed of one Entity Description from the Description Set named Project and all the matching Entity Descriptions from the set Employee. The matching Employee descriptions contain a triplet having Name Set Name = Project Number, Role Name = Project Worked In and an Entity Name = Name of Project in the matching Project Entity.

Catalog for the Entity Set Model

Data-base systems describe and process information about sets of Entities with similar properties. Some of this information is

the same for each Entity Description in a Description Set (Name Set Names, Identifiers in terms of Role Names, and Role Names themselves), whereas other aspects are generally different for each instance of an Entity Description (for example, the Entity Names associated with the Role Names). To conceptualize instances of a particular Description Set (Entities of the same type) and make thinking more efficient, information that is common to all instances of a particular type is collected and placed in a catalog. The complete information about a particular instance is thus a combination of the information common to all instances of its type and the information that is specific to it. The process of looking for collections of instance information common to all instances of a collection and placing it into a type description is a powerful method of organizing, simplifying, and condensing the information about a collection of instances. In some situations, this process has been called *factoring*.

In information systems, this classification process has been used instinctively to create things like data description tables. We have used it explicitly throughout the DIAM to aid in organizing and simplifying the special-instance terminology in data-base systems. The reclassification and simplification of existing terminology is, we believe, one of the major contributions of DIAM.

entity description entries

There are two requirements for type description of an Entity Description in the Entity Set Model. A Description Set Name that is unique across the system is required plus a list of Identifiers in the Entity Description in terms of Role Names. Also required is a list of Role Names with their associated Name Set Names. The Role Names need only be unique under the Description Set Name. A Description Set catalog is illustrated in Table 2. Here the slash symbol separates a Description Set Name from a Role Name. The combination of the two names is required to provide unique names across the system. Role Names required to form an identifier are separated by commas and enclosed in parentheses.

name set entries

The preliminary requirements for entries for Name Sets are as follows:

- Name Set Name unique to the system (e.g., PART)
- List of Role Names that the Name Set assumes in Description Sets of the system wherein system uniqueness is obtained by using the form Description Set Name/Role Name (e.g., PART/PART. SHIPMENT/PART SUPPLIED)
- List of the other Name Set Names that apply to the same set of Entities (e.g., PART NUMBER)
- List that correlates all Entity Names for a particular Entity (e.g., Social Security number = 308-20-1619 and employee number = 862 are names for the same Entity)

Table 2 A Description Set catalog

Name	*Type of name*	*Identifiers*	*Name Set Name*
PART	DSN	(PART)	
PART/PART COLOR	RN		COLOR
PART/PART	RN		PART
PART/PART WEIGHT	RN		WEIGHT
SHIPMENT	DSN	(PART SUPPLIED, PROJECT SUPPLIED, SUPPLIER) (SHIPMENT NUMBER)	
SHIPMENT/PART SUPPLIED	RN		PART
SHIPMENT/PROJECT SUPPLIED	RN		PROJECT
SHIPMENT/SHIPMENT NUMBER	RN		NUMBER
SHIPMENT/SHIPMENT WEIGHT	RN		WEIGHT
SHIPMENT/SUPPLIER	TN		CORPORATION

where DNS = Description Set Name
RN = Role Name

- List of the component Name Set Names, if the Entity Names in this Entity Name Set are combinations of Entity Names drawn from other Name Sets

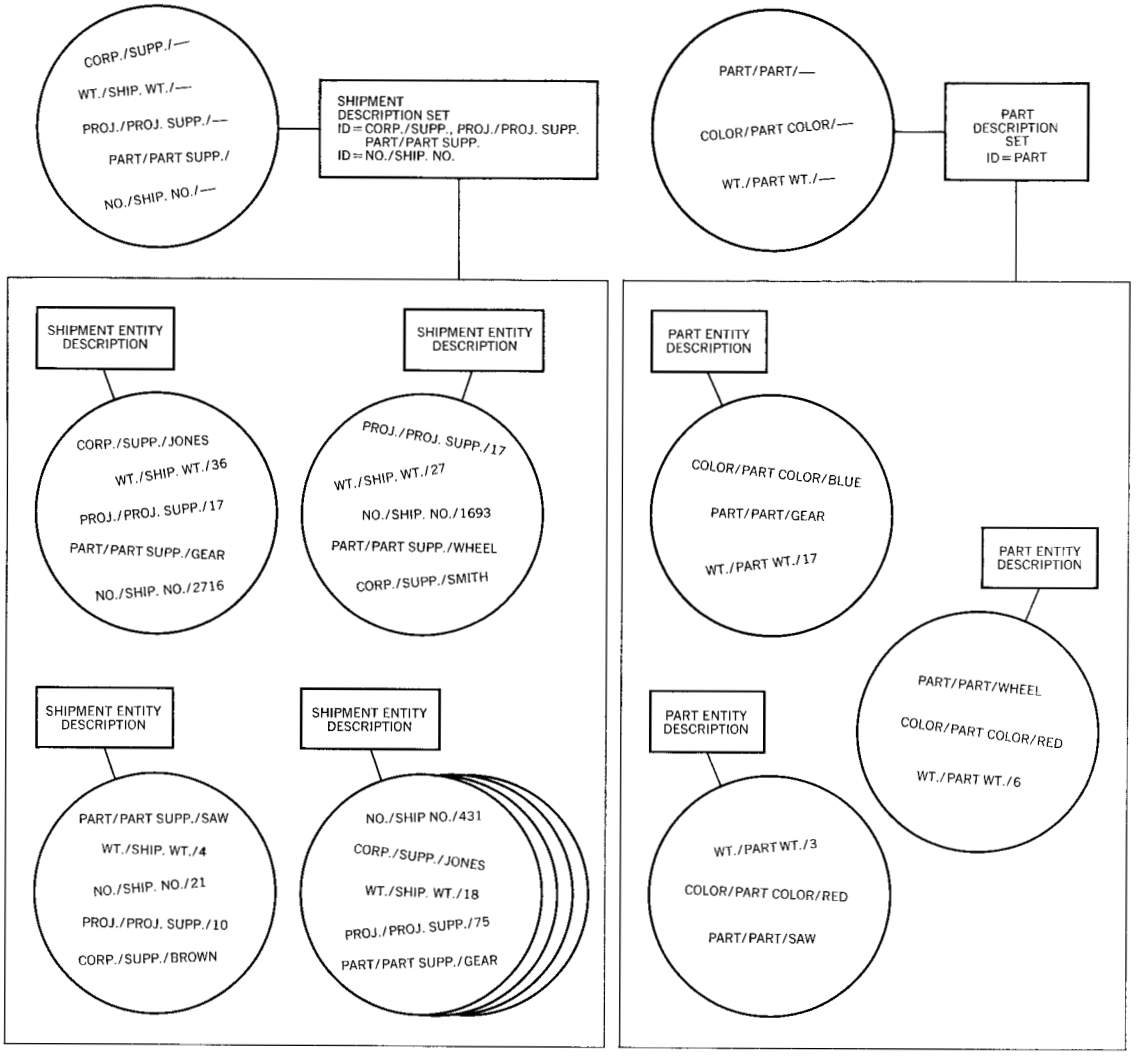
Within these requirements there is the basis of a *data directory*, that is, "field names," "where-used references," and the "composition of fields" that are composed of groups of other fields. The Name Set entries provide a useful place to store validity-checking information. For security against the disclosure of particular facts and to avoid update integrity problems, appropriate information can be associated with the Role Names under the Description Set entries.

generality of the entity set model

Figure 3 is a small Entity Set for the type description in Table 2. In looking at Figure 3 (and recalling one's experience with hierarchies of records, trees, and graphs) there appear to be three types of meaningful associations in data-base systems. It is a thesis of this paper that the Entity Set Model, including the following three associations, provides all the structure required to accommodate the kinds of facts that are stored in data-base systems. This is a useful thesis even if only approximately valid because it provides a simple basis for describing the facts stored in data-base systems. Exceptions can probably be accommodated with a minimum of disruption. The following are the three types of associations.

Associations among Name Set Name/Role Name/Entity Name triplets within an Entity Description permit the obtaining of answers to questions about the immediate characteristics of the identified Entity.

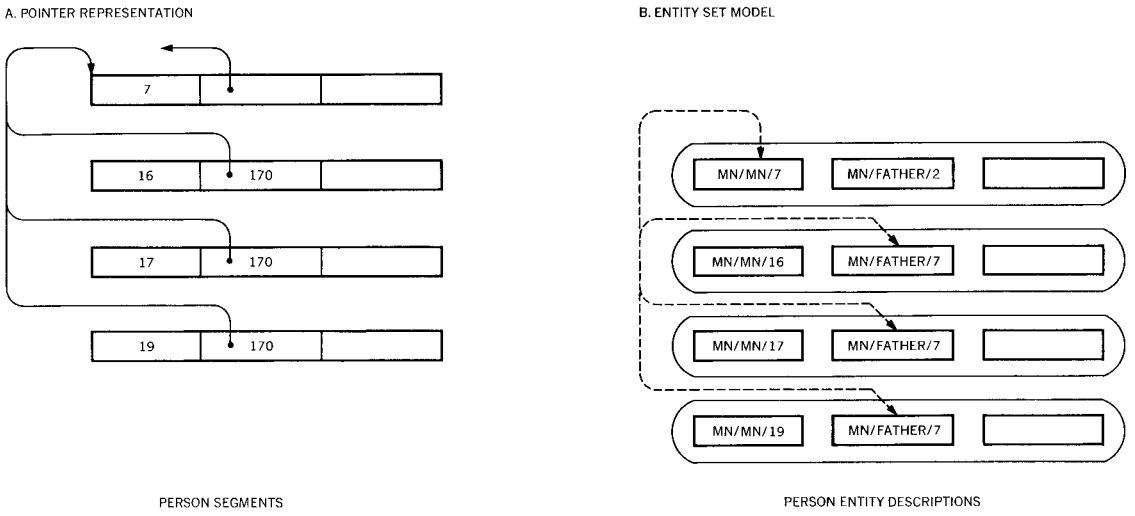
Figure 3 Entity Sets described by Table 2



Associations among Entity Descriptions in the same Description Set make it possible to obtain answers to questions about the entities that have specific properties.

The third is a somewhat more elusive association, the association among Entity Descriptions of the same or different types because the Descriptions have within them Entity Names for the same Entity. This association allows us to combine information about a department with information about its employees, and to ask questions about the department that involve its extended characteristics, i.e., characteristics of a department's component Entities or about the characteristics of its associations with other departments. Thus, for example, the DEPART-

Figure 4 Comparison of pointer representation and Entity Set Model



MENT Entity Description with the Identifier DEPT/DEPARTMENT/K06 is meaningfully associated with the Entity Descriptions from the EMPLOYEE Description Set containing DEPT/DEPARTMENT OF EMPLOYEE/K06.

One of the main elements of concern with regard to this generality thesis arises from the impression that pointers are required to fully represent information. It is true that pointers are used in certain representations to connect segments to represent a logical relationship between the segments and to provide a fast access path between the segments. It is unnecessary to include pointers in the Entity Set Model because the relationships that they represent can be made more naturally, generally, and appropriately by the match of Entity Names for the same Entity in two different Emity Descriptions—the third association, just discussed.

Figure 4 compares relationship pointers with an equivalent Entity Set representation. The pointer representation on the left is unidirectional and implies that the pointers have to be changed when the location of the segment identified by Man Number (MN) = 7 is changed. The Entity Set Model does not require a location concept and the relationship may be searched either from the parent to the child or from the child to the parent. It is clear that there is no explicit link that defines the relationship, but the implicit association can still be made by set operations. In this sense, no real-world information is lost by using the Entity Set Model. Another way of stating the same system property is that accessing language statements need not specify order or pointers to be followed. They need only specify Description Set Name/Role Names (which are unique in the system) and Entity

Names to obtain any substantive information stored in the system.

Figure 2 can be used as a starting basis for a physical analogy that is useful in relating the Entity Set Model to the model that is used for specifying access paths, which is discussed in Part III of this paper.

The analogy for the Entity Set Model is to assume that, for each Entity Description, its Name Set Name/Role Name/Entity Name triplets are each written on a separate card and that the cards corresponding to each particular Entity Description are placed in one separate envelope. To create a physical analogy for a particular Description Set, one must place all the Entity Description envelopes for that set into a larger envelope.

Note that all the tangible information contained in the data-base system can be extracted from this physical analogy by an exhaustive set of searches of the various envelopes, considering the three specified associations. Such a search procedure is feasible, but for most transactions it is inefficient. In fact, we have specified a conceptual structure of names and named associations that is neutral to efficiency considerations in the sense that no ordered subset access paths are yet defined on it to increase search efficiency.

From the physical analogy, associations of names are relatively stable. That is, to add a new property to the Entity Description, simply write the appropriate Name Set Name/Role Name/Entity Name cards for each of the Entities. Then place the cards in the appropriate Entity envelopes. Associations among existing names used by old programs remain unaffected. Discussed in Part III are general specifications of representations to provide efficient support for the conceptual structure.

Adapting the model to the application

The Entity Set Model contains only the essential information needed by the system accessor. With appropriate rules for the composition of Entity Descriptions to be discussed next, it may be possible to create a conceptual structure for a specific application that has only one place to store a fact. Having defined the form for the information structure, it is appropriate to seek a set of rules for assigning facts to Description Sets.

Inherent in any data-base system are the following requirements:

- Simplified maintenance of the integrity and consistency of information

- General guidance to location of a given fact in the system
- Simplified ways of preserving the validity of long-term programs for querying and maintaining the data base in an environment of changing representations (data independence)

These system requirements usually translate into two user requirements at the system interface. The system has but one place to store each fact, and also has a reasonably stable name structure to be used by programs that access the data base. This is necessary so that programs do not have to be rewritten whenever there are minor changes in structure.

role name identifier allocation rules

It may not be possible to fully achieve these requirements in the near-term future. It should be clear, however, that their eventual achievement is closely intertwined with means for specifying the fact content of and improving the stability of Description Sets used at each installation. Needed instead of our present process of distributing facts redundantly throughout the data base on the basis of requirements of special applications is a set of logical rules or guidelines to assist in specifying sensible Description Sets and their associated Role Name content. Role Name Identifier Allocation (RNIA) is the term for rules that govern this process. An indication of the strategic direction to be taken is given by presenting a few useful rules concerned with RNIA.

Component-whole rules. If a Role Name/Entity Name description of an Entity is not assigned to the description of the Entity, but instead, to the Entity Descriptions of components of the Entity, then this assignment is equivalent to storing a fact in many places. When the fact changes, all correlated values (Entity Names) in the component Entity Descriptions must be changed. Thus, if a project is always a component of a department and located on the same site as the department, then LOCATION OF is a proper role name for the Entity Description of the Department but not for the project Entity Description.

Further, a Role Name that is descriptive of a component should not be assigned to an Entity that subsumes the component. If projects can ever be located at different sites than their Departments, then LOCATION OF is a valid Role Name for the Project Entity Description even if all projects are presently located at the same site as their Department. Current location is thus an accidental rather than fundamental property of projects.

Many-to-one association. If one set of Entities is associated with another (or the same) set of Entities in a many-to-one relationship (such as Employees in a Department), then—to maintain a homogeneous structure—the department association (say) must be recorded in the Description Set of the many employee Entities.

Many-to-many association. Again for homogeneity, in a many-to-many association (Superior Parts to Component Parts), we must create a new Description Set where the many-to-many association is the identified Entity. The new Entity can also have properties of its own, such as quantity of a Component Part per Superior Part.

One-to-one association. Here it is important to determine the type of one-to-one relationship involved. One type involves two different Entities (Manager-Department). In another case, one Entity is a unique, permanent component of the second with no similar components involved. In the third type, we are dealing with two different Identifiers of the same Entity (employee number-Social Security number).

As a guide in applying these rules, the history of a contemplated association is generally helpful. In the first case, associations of specific Entities change over time. (Department managers, for instance, change with time.) Therefore, two different Description Sets (one for department and one for manager) are called for. The criteria for Description Set allocation of associations of these individuals with other individuals are, however, less quantitative. Essentially, the relative stability of each association is important. Normally, an employee is more stably associated with a department than with a manager, so minimum maintenance is involved when the department number, rather than the manager number, is placed in the employee Entity Description.

The second type of one-to-one association is probably an unusual occurrence. This association does bring up problems of classification that require further study. One question is whether a component is really unique in properties that are important to the system. (Does a person have more than one head?) If a component is unique, then the structure of the Role Name can be more complex or a new Entity Description can be created especially for a given Entity. If a component is not unique (the arm of a person), we create a new Description Set for it and for related similar Entities.

The third type of one-to-one association leads to a single Entity Description Set with multiple Identifiers.

Subsets of Entities with additional properties. An even more interesting association is involved when a person is also a manager. Such a person has certain descriptive Role Names that are associated with him only because of his Role as manager. Should these kinds of descriptive Role Names be placed in a new Entity Description for manager or should they simply be an extension of an existing Entity Description for person? In DIAM, either choice can be made with little effect on efficiency of stor-

age representation or accessing speed. The question is which one of the two alternatives is easiest for the user to understand and use. This question remains to be decided.

Thus we conclude our discussion of some aspects of the problem of improving the user interface through meaningful Role Name Identifier allocation rules. Note that all the rules like those for selecting IMS logical structures are based on logical relationship considerations, and not one is based on access efficiency considerations. Any Role Name Identifier Allocation based on access efficiency considerations instead of logical relationships often changes in response to changes in the transaction stream and this kind of change violates the goal of data structure independence. The rules presented make the user interface significantly more stable, consistent, and understandable. Even considering this progress, RNIA is an area that merits further investigation.

Meaningful operations

Existing and proposed data-base systems provide the user with unconstrained operations that allow him to operate on the stored information representations (bit patterns) as though they were devoid of meaning outside the system. There are a few exceptions, such as the tendency of compilers to distinguish between integer and floating point representations. Nevertheless, these unconstrained operations allow the user to create pseudo-facts that have no basis in the real world. (For example, it is possible today for a user to store a Quantity of Part in the location for Age of Person.) The aim of improvement here should be to provide better characterizations of the meaning of the bit patterns, and to define operations that are constrained by these characterizations. For example, we should not be able to add quantities of apples to quantities of oranges without specifying that the resultant quantity should have a new name. This kind of problem at the simplest level might be handled by forbiding the addition of quantities of elements from two different Entity Name Sets. More complex rules may actually be needed, but this indicates a proposed direction.

A second problem deals with Entity Names for an item that are derived by function from Entity Names for its Components, such as "Number of Component Part Types" for a given "Superior Part." Here it is clear that the user should only be able to change the Number of Component Part Types indirectly by inserting or deleting a Component Part Type. He should not be able to modify the number directly.

A more complex problem pertains to the special meanings of values for Identifiers and for Role Names. We should, for exam-

ple, not be allowed to combine the following two Entity Descriptions:

	Identifier		Identifier		
Role Name:	Supplier	Part Supplied	Project	Part Supplied	Quantity
		with			
Entity Name:	JONES	GEAR	34	GEAR	17

to form the Entity Description:

Identifier			
Supplier	Part Supplied	Project	Quantity
JONES	GEAR	34	17

where both Supplier-Part Supplied and Project-Part Supplied are many-to-many associations and Identifiers for their entities.

In this case—without reference to the real world to determine whether such an Entity exists and what its characteristics are—we have created a Name and Description (Quantity) for it. The system no longer is a faithful representation of the real world. This situation is emphasized when we look at the meaning of the value of Quantity in the two Entity Descriptions. In the initial Entity Description, Quantity stands for number of parts supplied to the project by all suppliers. The Entity Description resulting from the combination incorrectly implies that Quantity stands for the total number of Parts Supplied to the Project by the specific supplier, Jones. Anyone reading such a system output comes naturally to that erroneous conclusion. In essence, unconstrained mathematical operations do—but they should not—have the capability of changing the meaning of Role Name/Entity Name Pairs. In the example, the meaning of the value for Quantity of Parts Supplied to the Project should not be changed into Quantity of Part Supplied to the Project by supplier by the simple unconstrained operation of concatenation of identifier fields. The value for the second Quantity must be supplied from outside the system.

It appears that with adequate RNIA rules in the Entity Set Model, it may be possible to prohibit combination operations that construct or modify identifiers. In essence, there should be one and only one Entity Description for each Entity, and that description should contain all directly related Role Name/Entity Name pairs. Useful operations defined for the Entity Description would be the following: selection of a subset of Role Name/Entity Name pairs, and the combination of Entities to form hierarchic records where the relationship between Role Names and Identifiers is preserved.

There are also certain operations used in fact retrieval systems that can create new, meaningful associations from old, stored

associations, but these depend on functions supplied by the user and are applicable only to specific stored functions. For example, the two-place association "grandfather of" can be calculated from the stored two-place association "father of." Such operations, however, seem to be relatively infrequent in practice.

Summary

In this section on information structure, we have defined an Entity Set Model and related its basic concepts to notions of classification, naming, and description of Entities. We have also indicated how the Entity Set Model can provide a basis for a database system with certain very desirable characteristics. In Part III, we move from the discussion of information structures and associations among names, to a discussion of models for representing the names in computer storage.

CITED REFERENCES

1. G. H. Mealy, "Another look at data," *AFIPS Conference Proceedings, Fall Joint Computer Conference* **31**, 525–534 (1967).
2. R. W. Engles, *A Tutorial on Data Base Organization*, Report TR 00.2004, International Business Machines Corporation, System Development Division, Poughkeepsie, New York (1970).
3. H. S. Meltzer, *Data Base Concepts and Architecture for Data Base Systems*, IBM Report to SHARE Information Systems Research Project (August 20, 1969).
4. C. T. Davies, *A Logical Concept for the Control and Management of Data*, Report AR-0803-00, International Business Machines Corporation, System Development Division, Poughkeepsie, New York (1967).
5. CODASYL Data Base Task Group, *Report to the CODASYL Programming Language Committee*, Report CR 11, 5(70)19,080; ACM, New York, New York (October 1969).
6. *Information Management System IMS/360, Application Description Manual (Version 2)*, Form GH20-0765-1, International Business Machines Corporation, Data Processing Division, White Plains, New York 10604 (1971).
7. *Generalized Information System GIS/360, Application Description Manual (Version 2)*, Form GH20-0892-0, International Business Machines Corporation, Data Processing Division, White Plains, New York 10604 (1970).
8. *Common Business Oriented Language (COBOL) General Information*, IBM Systems Reference Library, File No. GENL-24, Form F28-8053-2, International Business Machines Corporation, Data Processing Division, White Plains, New York (1960–61).
9. *GE-600 Line Integrated Data Store*, Publication CPB-1565A, General Electric Information Systems Department, Phoenix, Arizona (September 1969).
10. E. F. Codd, "A relational model for large shared data data banks," *Communications of the ACM* **13**, No. 6, 377–387 (June 1970).
11. R. E. Bleier, "Treating hierarchical data structures in the SDC Time Shared Data Management System TDMS," *Proceedings of the ACM 22nd National Conference*, 41–49, MDI Publications, Wayne, Pennsylvania (1967).