DATA BASE STANDARDIZATION

A STATUS REPORT

Thomas B. Steel, Jr.

Equitable Life Assurance Society

New York, N.Y., USA

This paper is a report on the current (1975 September) status of the Study Group on Data Base Management Systems in the United States, together with some remarks on the ISO activity in the area. While the official purpose of this Study Group is an investigation of standardization potential in the area of data base management systems, an important by-product of the work of the Group has been the development of a set of requirements for effective data base management systems. As no existing or proposed implementation of a data base management system satisfies these requirements, it is appropriate to expose these ideas as widely as possible for evaluation.

Among the responsibilities of the Standards Planning and Requirements Committee (SPARC) of the American National Standards Committee on Computers and Information Processing (ANSI/X3) is the generation of recommendations for action by the parent Committee on appropriate areas for the initiation of standards development. For some time it has been evident that data base management systems are in the process of becoming central elements of information processing systems, and that there is less than full agreement on appropriate design. In addition to the existence of a number of implementations of such systems (CODASYL 1969), there are several documents generated out of the collective wisdom of some segment of

the information processing community which are either proposals for specific systems (CODASYL 1971) or statements of requirements (GUIDE-SHARE 1970), (CMSAG 1971). As is well known there is a debate in the community on whether existing and proposed implementations meet the indicated requirements or whether the requirements as drawn are all really necessary. Further, there are serious questions about the economics of meeting all the stated requirements.

In addition to the above considerations there is argument on the appropriate data model to use: relational, hierarchical, network. This particular debate has been referred to as the "theological" discussion of the data base management system theorists. There has been criticism of the use of this word; I can only respond to that criticism by quoting Hilaire Belloc: "All political questions are ultimately theological". Indeed, such it seems to be, from which it follows that the correct answer to the question of what data model to use is necessarily "all of the above". One of the outcomes of the work reported in this paper is a mechanism that permits this answer in a meaningful sense.

In the autumn of 1972, responding to the clearly perceived need to rationalize the growing confusion, SPARC, then under the Chairmanship of the author, took formal action to initiate investigation of the subject of data base management systems in the context of potential standardization. Consistent with its normal practice when confronted with a complex subject, SPARC established an ad hoc Study Group on Data Base Management Systems, initially under the Chairmanship of D. M. Smith of the EXXON Corporation and now under the Chairmanship of the author. This Study Group was convened with a charge to investigate the subject of data base

management systems with the objective of determining which, if any, aspects of such systems are at present suitable candidates for the development of American National Standards. The "if any" qualification is important because a negative response is just as meaningful as a positive response in a standards context. The "at present" qualification is equally significant, indicating the continuing need for review as the requirements, technologies and economics change over time.

The eventual result of the deliberations of this Study Group will be a series of reports in a specified format (SPARC 1974), identifying potentially standardizable elements of data base management systems and recommending whether or not there is a need, technological feasibility and economic justifications for the initiation of a standards development project in the area. The first interface to be examined is 7 with respect to COBOL. The present target date for completion of this work is the beginning of 1976. As an Interim Report the Study Group has prepared a document (SPARC 1975) which has had wide circulation and is soon to be generally published.

It is appropriate at this juncture to provide a list of the members of the Study Group and their affiliations to indicate the breadth of representation. It is worth noting the extent to which the user community is participating in this effort, a rare event in data processing standardization on any continent.

Bachman, C.W.	Honeywell Information Systems
Cohn, L.	IBM Corporation
Florance, W.E.	Eastman Kodak Company
Kirshenbaum, F.	Equitable Life

Kunecke, H.	Boeing Computer Services
Lavin, M.	Sperry Univac
Mairet, C.E.	Deere and Company
Sibley, E.H.	University of Maryland
Steel, T.B., Jr.	Equitable Life
Turner, J.A.	Columbia University
Yormark, B.	The RAND Corporation

The initial tasks of the Study Group were the difficult ones of understanding and coming to respect the varying views of the different individuals--all theologies were (and still are) represented--and developing a vocabulary that was consistent and mutually comprehensible. It is not clear whether this last task has yet been fully accomplished, although considerable closure has been attained.

In the course of the early discussions it emerged that what any standardization should treat is interfaces. There is no merit and potential disaster in developing standards that specify how components are to work. What is potentially proper for standards specification is how the components are meshed together; in other words, the interfaces. With this notion in mind a generalized model of a data base management system has been developed that highlights the interfaces and the kind of information and data passing across them. Figure 1 is a simplified diagrammatic view of this model.

It should be noted that, except for the man-system interfaces, the technological nature of the interface is not determined; it could be hardware, software, firmware or some mixture. Indeed, some of the

interfaces could be man-man, although pursuit of that notion is not germane to what follows. The important point is that the implementation of the system is not prescribed, only the requirements that must be satisfied. As was noted above, this is a simplified diagram, but in order to maintain consistency with the detailed picture, the numerical identifications of the exhibited interfaces have not been changed so there are some numbers missing.

The hexagonal boxes depict people in specific roles. The rectangular boxes represent processing functions, the arrow terminated lines represent flow of data, control information, programs and descriptions, and the dashed boxes represent program preparation and execution subsystems (including compilation and interpretation functions). Finally, the solid bars represent essential interfaces, the ultimate subject matter of the Study Group's deliberations. These interfaces are numbered rather than conventionally named for simplicity of discussion and to avoid confusion.

Among the processes and interfaces omitted on this cut down version of the diagram are the various ways that system programmers and machine operators can invade the system to make ad hoc repairs, certain bypasses of the system mechanism that are asserted to promote efficiency but of debatable desirability in view of their impact on data independence, integrity and security, and the entire structure of physical mapping of data onto specific storage devices. All of the latter structure is to be found to the left of interface 21, much of it will be dictated by the laws of physics and, as such, is of little concern to the current investigation. The principal elements of the Study Group's view of a data base management system are displayed and, in particular, the three schema approach,

reflecting the new element introduced by this work, is illustrated.

The lower right hand side of the diagram, the hexagon labelled "application programmer", the dashed rectangle labelled "application program subsystem" and the two interfaces labelled "7" and "12" comprise the entire non-data base activity of preparing and executing an application program. This structure may be viewed as replicated into a variety of subsystems, all interfacing with the data base management system through interface 12, differing in the nature of the language used by the programmer to communicate across the man-machine interface. This language may be a conventional procedure language such as COBOL, ALGOL or PL/I, recognizable special languages like report generators, inquiry languages or update specifiers, or some potentially new type of procedure or problem language. The critical thing to note here is that all data description passes into the application program subsystem across interface 12 from the data base system itself. This, of course, is nothing new.

The lower left hand side of the diagram, the hexagon labelled "system programmer", the dashed rectangle labelled "system program subsystem" and the two interfaces labelled "16" and "18" comprise the entire normal interface available to the system programmer when it is necessary to bypass the ordinary mode of access to the system. Routine system maintenance and modification will occur through this subsystem. There are some exceptions, as noted above, but they do not concern the thrust of this paper. It should also be noted that there is clearly available the installation option of permitting application programmers to operate across this interface, potentially dangerous as that may be. Again, there is nothing new in this construction.

It is the upper portion of the diagram that is of concern in this paper. Current data base systems envision a two level structure; the data as seen by the machine and the data as seen by the programmer. A plethora of confusing terminology has been employed to distinguish between these views. The Study Group has chosen to employ the terms "internal" and "external" to make this distinction. In addition, the Study Group has taken note of the reality of a third level, which we chose to call the "conceptual", that has always been present but never before called out explicitly. It represents the enterprise's view of the structure it is attempting to model in the data base. This view is that which is informally invoked when there is a dispute between the user and the programmer over exactly what was meant by program specifications. The Study Group contends that in the data base world it must be made explicit and, in fact, made known to the data base management system. The proposed mechanism for doing this is the conceptual schema. The other two views of data, internal and external, must necessarily be consistent with the view expressed by the conceptual schema.

This required consistency can be maintained and verified in a reasonably fail safe manner only if the conceptual schema is machine processable. The bulk of the remainder of this paper will discuss the nature of the conceptual schema and how it may be made explicit to the system. However, it is worth examining what its presence means to the dynamics of the data base management system operation in terms of the diagrammatic representation of Figure 1.

Ignoring the system programmers, who are extraneous to normal operation, there are four human roles identified: the enterprise administrator, the data base administrator, the application administrator(s), and the application programmer(s). Notice that

these are roles as opposed to individuals. The same individual may function in different roles and one role may involve several individuals simultaneously. It is critical, however, that there is only one enterprise administrator and one data base administrator (viewed as roles) while there may be several application administrators and several application programmers. This leads to the notion that there can be several external schemas, each representing a different view of the data, provided each is consistent with and derivable from the single conceptual schema. By extension there can be several application programmers, not necessarily working on the same program, that use the same external schema.

Each "administrator" is responsible for providing to the system a particular view of the necessary data and the relevant relationships among that data. The central view, as noted above, is that of the enterprise administrator who provides the conceptual schema. It must be emphasized, and apparently with repetition as this point seems to be the most frequently missed by those not on the Study Group who have examined its work, that the conceptual schema is a real, tangible item, made most explicit in machine readable form, couched in some well defined and potentially standardizable syntax. Much of the remainder of this paper is concerned with conceptual schemas and the author's view of the possibilities for the semantics of such schemas. In order to provide a context, however, a preliminary examination of the dynamics of the process envisioned is appropriate.

The enterprise administrator defines the conceptual schema and, to the extent possible and practicable, validates it. Some, but in general not all, of this schema can be checked for consistency by

mechanical means. As the conceptual schema is a formal model of the interesting (for the data base management system) aspects of the enterprise, if the situation is at all complex then the problem of logical incompletability will be encountered (Godel 1931). The conceptual schema will contain, among other things, definitions of all the entities to be comprehended--up to the isomorphism determined by identity of those properties defined in the schema as relevant. Relatonships amongst these entities will also be explicated, as will the constraints on allowable values of "data". By defining those persons with some access to the data base management system as entities of interest, it is possible to directly model the rules of access and, thus, provide security control at the level of the conceptual schema. This is a key point. It is well known that there are substantial problems with security control and the importance if a centralized point having a view of the entire system must not be overlooked.

The data base administrator (a definition of this role somewhat at variance with the conventional conception of the task) is responsible for defining the internal schema. This schema contains an abstract description of the storage strategy currently employed by the data base management system. Whether the data is actually stored flat, hierarchical, networked, inverted or otherwise, including any meaningful combination, is contained in the internal schema. The "internal syntax" of the data values will also be found in the internal schema; such items as the radix for numeric values, coding schemes used, units of measure, and the like. Access paths and the relational connectivity between data representations will be defined. All of this must be consistent with and derivable from the conceptual schema, which, therefore, must be available for display to the data base administrator,. The internal schema processor (see

Figure 1) provides a mechanical check on this consistency. Within the limits imposed by this requirement of consistency with the conceptual schema, the data base administrator is free to alter the internal schema in any way appropriate to optimization of the data base management system operation. Indeed, by use of suitable interpreters it will be possible to reorganize the internal structure of the data base dynamically while normal operations continue. In view of the massive size of some data bases currently comtemplated, this is an essential requirement, and it would seem that only the guarantee of separation of the users' view and the system's view of data provided by interposition of the conceptual schema permits this.

The third "administrator" role, the application administrators, provide the external schemas (analogues of the DBTG "sub-schemas") which define the application programmers' views of the data. These external schemas are a multiplicity in concept and will, in general, only encompass the portion of the data base relevant to a particular application. It is envisioned that each general application area will have its own application administrator who provides the appropriate schemas for that area. These are the only data descriptions (schemas) seen by an application program and provide the only avenue of data name resolution. It would carry this essay too far afield to discuss the complexities of name resolution and symbol binding; suffice it to say that all external name resolution, whether performed at compile time, program invocation time, or module execution time are done across interfaces 7, 12 and 31 through the intermediation of the appropriate external schema across interface 5.

Exactly the same remarks about the consistency of the various

external schemas with respect to the conceptual schema as was noted about the internal schema are to be understood, with the qualification that one external schema may be a true subset of another and, under the hypothesis that consistency in this sense is transitive, the external schema processor may only validate one external schema against a more comprehensive one known to be consistent with the conceptual schema.

After the appropriate schemas are defined, the system dynamics becomes quite straightforward and little different from current systems. The application programmer (report specifier, inquiry specifier, etc.) does his job in the usual way, using the provided external schema, both explicitly and implicitly, as his set of data declarations, providing procedural input across interface 7 and invoking compilation, generation or other relevant processes through the application program subsystem. Upon entry to execution mode, requests for data are passed across interface 12 to the conceptual/external transformer which computes the mapping between the external data description and the conceptual data description. This description passes across interface 31 to the conceptual/internal transformer which in its turn computes the mapping between the conceptual data description and the internal data description. In general, the internal and conceptual schemas will be static, so, depending upon the mapping complexity and the nature of the implementation, it may well be possible to collapse the two transformers (into and out of the conceptual data description) by computing the composite mapping function. This should not obscure the face that in order to maintain true data independence it must always remain possible to force this process to occur in two steps.

Finally, the data request as transformed is passed across interface 30 to the internal/storage transformer. The internal schema will recognize storage as something like a linear, multiorigined address space, and it will be necessary to remap this abstract model of storage onto hardware constructs such as tracks, cylinders and the like. This "dirty" description then is passed across interface 21 into the bowels of the machine (and may go through other transformations therein) until actual data is obtained and the process reversed. This brief description has been couched in terms of obtaining data but, of course, storage of data proceeds in the same way, mutatis mutandis.

Question of locks, avoidance of "deadly embrace", security, integrity and other data base management system problems all have their place in this scheme of things, but it is beyond the scope of this paper to consider them. By and large they present no distinct aspects in this three level view from those found in conventional approaches, except that in some instances--security, for example--the solutions may be both easier and more assured.

Before turning to a discussion of the conceptual schema it is appropriate to insert a brief excursus on the status of data base management system standardization in ISO. At the Eight Plenary Meeting of ISO/TC97, held 1974 May 14-17 in Geneva, Resolution 11, passed with 14 affirmative and two negative (Canada, France) votes, assigned responsibility for data base management to Subcomittee 5 (Programming Languages) and instructed SC5 to establish a study group on the subject (ISO 1974).

Such a Study Group was established by SC5 and several countries submitted position papers. The USA position paper was the SPARC Interim Report. An 1975 June 24-26, the Study Group met in

Washington, DC with delegations from France, Germany, Sweden and the USA. Written input was also available from Switzerland and the United Kingdom. The following six points are the conclusions of that meeting:

1. The Study Group concludes in response to the Netherlands Proposal on Data Base Management (ISO/TC 97/598), that any standardization action in the area of data base management systems based on existing proposals is premature in the absence of criteria against which to measure such proposals.

2. The Interim Report of the ANSI/X3/SPARC Study Group on Data Base Management Systems (ISO/TC 97/SC 5 (USA-75) N359) is accepted by the ISO/TC 97/SC 5 Study Group on Data Base Management Systems as an initial basis for discussion on a gross architecture of data base management systems.

3. The Study Group acknowledges the need to identify all types of data base management systems users and to specify their requirements.

4. The Study Group proposes to review and augment the terminology used in N359 and the concepts therein. As the initial effort, the Study Group will establish priorities in terms of the interfaces identified in N359 for further investigation. These priorities will be chosen to optimize the benefits derived from standardization.

5. As a parallel activity to those identified above, the current CODASYL data base specifications will be evaluated. The Study Group notes at this time that preliminary studies by various national and internationl bodies have indicated that the CODASYL specifications are not suitable for standardization as they

stand.

6. The Study Group will recommend development work for those interfaces appropriate for standardization for which no adequate candidate exists.

The next meeting of this Study Group will be in Paris, 1976 January 12-15.

The underlying notion behind the conceptual schema as envisioned by the Study Group is the "entity-property-value" trinity made explicit in GUIDE-SHARE requirements study (GUIDE-SHARE 1970). There is general agreement among the members of the Study Group on the overall nature and objectives of the conceptual schema, but in my judgment there is less real agreement on its exact place in the scheme of things than might seem the case from the Study Group reports. To a considerable extent this lack of agreement does not hamper progress, and may well not matter in the long run provided the distinct views are carefully articulated. What follows is the author's view of the conceptual schema notion and some indications on how it can be formalized.

Figure 2 is a schematic illustration of how one can proceed from "reality" to the data models actually used by application programs. It is derived from a metaphysics that may not be wholly congenial to everyone but should at the very least be familiar to those acquainted with the principles of scientific explanation (Braithwaite 1953). It is assumed that a "real world" exists in some meaningful sense. Subordinate to this "true" reality can be found the "perceived" reality obtained through our sensory inputs as transformed by our brains. This immediate, primitive image of reality is, or at least can be, transformed into a rational mental

model of reality by a process known as scientific abstraction.

This process can be roughly described as: (1) observation (noting one's perceptions); (2) experimentation (stimulation of the perceived reality to generate new perceptions); (3) generalization (intuiting that similar stimulation will generate similar perceptions); (4) theorizing (identifying fundamental generalizations); (5) deducing (inferring that new and different stimulations will produce new, albeit expected, perceptions); and, finally, (6) verification (initiating these new stimuli and observing the results). Repeated iteration of this sequence leads to a gradually more refined mental model of the real world.

In order to communicate this model to someone--or something--else, it is necessary to use a language. As is well known, natural languages are unsatisfactory media for precise communication of the content of scientific models. At present the best available vehicle for such precise communication is that of formal languages (Tarski 1930). While there are complications in the reduction of scientific descriptions of reality to existing formalisms, most of these problems are to be found on the outer limits of the models. Generally one does not really wish to describe a total model of all reality--the "best" model whose boundary is fuzzy and moves with the growth and modification of scientific knowledge. What is desired is to describe some limited model of a portion of reality, extracted from the "best" model by a process we can call "engineering abstraction". While it may be the case that the universe is "best" described by the interactions of 3.10^{80} quarks, the typical engineer is more apt to build his bridge by combining girders, cross braces and rivets. The molecular biologist may view the human being as a complex structure of water, protein molecules, DNA and other,

assorted chemicals, but to the insurance agent a human being is not much more than an age, sex and checkbook. For any application one abstracts those aspects of "reality" considered relevant and ignores the rest. Thus, formal descriptions need only deal with the appropriate level of abstraction.

This resultant formalism--the "symbolic" model--is derived from the limited, "engineering" model of the interesting subset of reality as embodied in the mind of the perceiver by a process we will call "symbolic abstraction", and is the linguistic expression in some conventional, predetermined syntax of a set of forms to which suitable semantic content is given by the adoption of rules of designation and rules of truth (Carnap 1942). It expresses the totality of what is known and interesting about the enterprise being modeled. It is the conceptual schemma. The processes of mapping from this formal model to the data models we call "internal schema" and "external schemas" may be complex and difficult in practice, but they are straightforward in principle, providing only that the conceptual schema has sufficient detail to permit all necessary expression.

In the author's view the proper choice of formalism--indeed, the only acceptable choice--is that of modern symbolic logic; the first order predicate calculus with identity (Hilbert & Ackermann 1938), together with a suitable axiomatic set theory (Bernays & Fraenkel 1958), augmented by appropriate modal logics (von Wright 1951), and, finally, supplemented by "individuals" (Quine 1961) and the associated non-logical predicates and the axioms for their behavior. The reasoning behind this position is quite simple. Use of the conventional formalisms of symbolic logic and set theory permit the invocation of all the analysis that has been devoted to this topic

by three generations of logicians. Both the pitfalls and possibilities are well understood and the limitations clearly defined. Further, it is in some sense the most general scheme available. If one accepts Church's Thesis (Kleene 1952), as do most contemporary logicians, it is the most general scheme that can be contemplated for use with digital machinery. From this it is possible to deduce that anything expressible to a machine with precision at all is necessarily expressible in this fashion.

As an aside let me emphasize a point which should be obvious but is, perhaps, worth making explicit for clarity. Whenever in this paper I use the word "set" I intend it in the strictly logical sense as a synonym for "collection" or the German "Menge" or the French "ensemble", not in any way as that linguistic atrocity perpetrated by the DBTG Report wherein the nineteenth, fifth and twentieth letters of the Roman alphabet are used in that order as the name of a peculiar object. This may seem harsh, but the point at issue represents a prize example of the manner in which the information processing sciences generate confusion for themselves and others by casual misuse of words. Indeed, it reminds me of Orwell's Newspeak.

In a paper of this character it is not possible to probe the possibilities of the sketch above in any depth. However, certain examples may clarify the power of the approach. It is unequivocally precise in any modern version of set theory as to what is meant by a "relation". A relation is a set of ordered pairs (the ordered pair $\langle x,y \rangle$ being definable as $\{\{x\},\{x,y\}\}$) and one can say that x bears the relationship R to y provided that $\langle x,y \rangle \in R$ ("$\in$" being the predicate of set membership). Thus, the confusion between a "relation" and a "relationship", which is another example of terminological idiocy, is made quite precise.

Relations of interest can be given names and defined either by enumeration of their members or by any property that must be possessed by a pair to enjoy membership, in exactly the same fashion that any other set is completely defined by its members.

The equally troublesome concept of "order" can be explicitly defined. A partial ordering is any relation having the properties of reflexivity, anti-symmetry and transitivity. A linear ordering is a partial ordering where any two elements in its field are comparable and a well-ordering is a nowhere dense linear ordering.

Structures of arbitrary complexity can be constructed. The concept of a general array (Steel 1964) developed out of some early data structure studies, and it can be shown that any nondense complex is expressible as a general array so defined. As digital computers cannot deal with dense structures except in finite approximation, this would seem to be sufficient.

The modal predicate of deontic logic, "O" (for "obliged to"), and its derived predicates "O-" ("obliged to not" ≡ "forbidden to"), and "-O-" ("not forbidden to" ≡ "permitted to") provide the required paradigm for expressing either legal constraints in the model or defining the rules of access.

These examples could be multiplied a considerable length, but should be sufficient to illustrate the point. From a theoretical point of view there is no more suitable vehicle for expressing a conceptual schema. This is, of course, not the whole story.

First, theoretical possibility and practical possibility are not identical. There is the danger that the necessary expressions get too large and cumbersome for effective use. In an age where we deal with million instruction operating systems, this is not a fully

persuasive argument in any event. It is, however, moot. The number and character of the necessary expressions do not get excessive; unlike, say, the contrast between conventional procedure languages and Turing machines. On the contrary, nearly a century of search for compact notation has resulted in definitional sequences that provide more compact expression than one typically finds in programming language data descriptions (or sub-schemas) which perform less of the task. Some of this is due, of course, to the use of large character sets, but in any case economy of notation is not a problem.

A second potential difficulty is the actual use of the tools to construct the desired models, which is a task that is necessarily an art rather than a science. Clearly, if the process of constructing a model could be itself formalized one would already have the model in the input. To this point I can only say that I have personally been partially successful in constructing models of relatively complex insurance procedures, and in a matter of a few days, inventing notation as I went along. This effort was only partially successful in the sense that, while I was able to generate static models with no difficulty, the problem with time and the dynamic behavior of the model caused difficulties of two types. First, thre was the philosophical problem of the potential as opposed to the actual. How does one treat the property "age at death" prior to the actual death of the individual? Formally, of course, this is trivial, but obtaining some assurance that the formalism does not hide an ambiguity or paradox is far from trivial.

The second problem with time has to do with the inelegance of making the variable denoting time distinguished and, therefore, a special case. While there is nothing inherently wrong with mathematical

inelegance per se, several thousand years of logical and mathematical history suggest intuitively that something is wrong. Some recent work (Thomasen 1974) on the reduction of tense logic to modal logic hints at a solution to this problem.

I have gone far enough with this work to become convinced that the approach is sound and no fundamental invention is required; only some hard work to refine the ideas. There remains, however, one further potential criticism of this approach with which it is necessary to deal. It is a criticism to which I would prefer to comment "a pox on those who raise it" and then ignore the matter. As a practical consideration, however, it will not go away. It is much the same argument that has been raised in the past against every programming language except COBOL; i.e., the language is too much like algebra, only the mathematicians can use it. The argument is irrefutable for if people believe they cannot understand something, they won't! However, there is one difference between this situation and the programming language situation. The only one who must construct models is the enterprise administrator and only the data base administrator and the applications administrators need to read such models. These individuals are presumably senior and well compensated. They can be required to have a little education. Furthermore, while I have no proof, it is my belief that once the barrier of belief in its esoteric character is overcome, it is no harder to teach reasonably intelligent people the relevant logic than it is to teach them COBOL and the DDL.

To summarize this personal view of the nature of a conceptual schema, any alternative is either equivalent and therefore equally complex while being less understood for lack of familiarity, or it is not equivalent and therefore can only model a subset of that

reality otherwise amenable to modelling. The only real issue is whether some less powerful but more acceptable formalism exists that is adequate for modelling anticipated enterprises for a reasonable future. In my view neither data structure diagrams (Bachman 1969) nor normalized relations (Codd 1970) nor the CODASYL DDL (CODASYL 1971) being discussed at this Working Conference are candidates for such an alternative. As overlaid structures for internal and external schemas they may be quite suitable; the criteria for acceptability being different.

In conclusion, let me reiterate that the latter portion of this paper is my personal view of the appropriate structure for a conceptual schema and does not necessarily represent the view of other members of the ANSI/SPARC Study Group on Data Base Management Systems. On the other hand, the general principle of the three level approach and the essential requirement for the conceptual schema <u>is</u> fundamental to the deliberations of the Study Group. It is reasonable to claim that this position will be maintained in the Final Report of the Study Group and will continue to characterize the official position taken by ANSI on behalf of the USA in any deliberations on data base management systems in the ISO.

REFERENCES

Bachman, C. W.: "Data Structure Diagrams", Data Base, 1:2 (1969).

Bernays, P. and Fraenkel, A. A.: "Axiomatic Set Theory", North-Holland (Amsterdam 1958).

Braithwaite, R. B.: "Scientific Explanation", Cambridge University Press (London 1953).

Carnap, R.: "Introduction to Semantics", Harvard University Press (Cambridge, MA 1942).

CMSAG Joint Utilities Project: "Date Management Systems Requirements", CMSAG (Orlando, FL 1971).

CODASYL: "A Survey of Generalized Data Base Management Systems", available from NTIS (Washington, DC 1969).

CODASYL": "Data Base Task Group Report", ACM (New York 1971).

Codd, E. F.: "A Relational Model of Data for Large Shared Data Banks", CACM, 13:6 (1970), pp. 377-387.

Gödel, K.: "Uber formal unentscheidbare Sätze der Principia Mathematica und verwandter Systems I", Monatshefte, 38 (1931), pp. 173-198.

GUIDE/SHARE: "Data Base Management System Requirements", SHARE Inc. (New York, N. Y. 1970).

Hilbert, D. and Ackermann, W.: "Grundzuge der Theoretischen Logik", Julius Springer (Berlin, 1938).

ISO: ISO/TC97 (Geneva-3) 669.

Kleene, S. C.: "Introduction to Metamathematics", van Nostrand (Princeton, N. J. 1952).

Quine, W. V. O.: "Mathematical Logic", rev.ed., Harvard University Press (Cambridge, MA 1961).

SPARC: "Outline for Preparation of Proposals for Standardization", document SPARC/90, CBEMA (Washington, DC 1974).

SPARC: "Interim Report: Study Committee on Data Base Management Systems:, SIGMOD NEWSLETTER (forthcoming).

Steel, T. B., Jr.: "Beginnings of a Theory of Information Handling", CACM, 7:2 (1964), pp. 97-103.

Tarski, A.: "Fundmentale Begriffe der Methodogie der deduktiven Wissenschaften I", Monatshefte für Mathematik und Physik, 37 (1930), pp. 361-404.

Thomason, S. K.: "Reduction of tense logic to modal logic, I", J. Symbolic Logic, 39:3 (1974), pp. 549-551.

Von Wright, G. H.: "An Essay in Modal Logic", North-Holland (Amsterdam 1951).

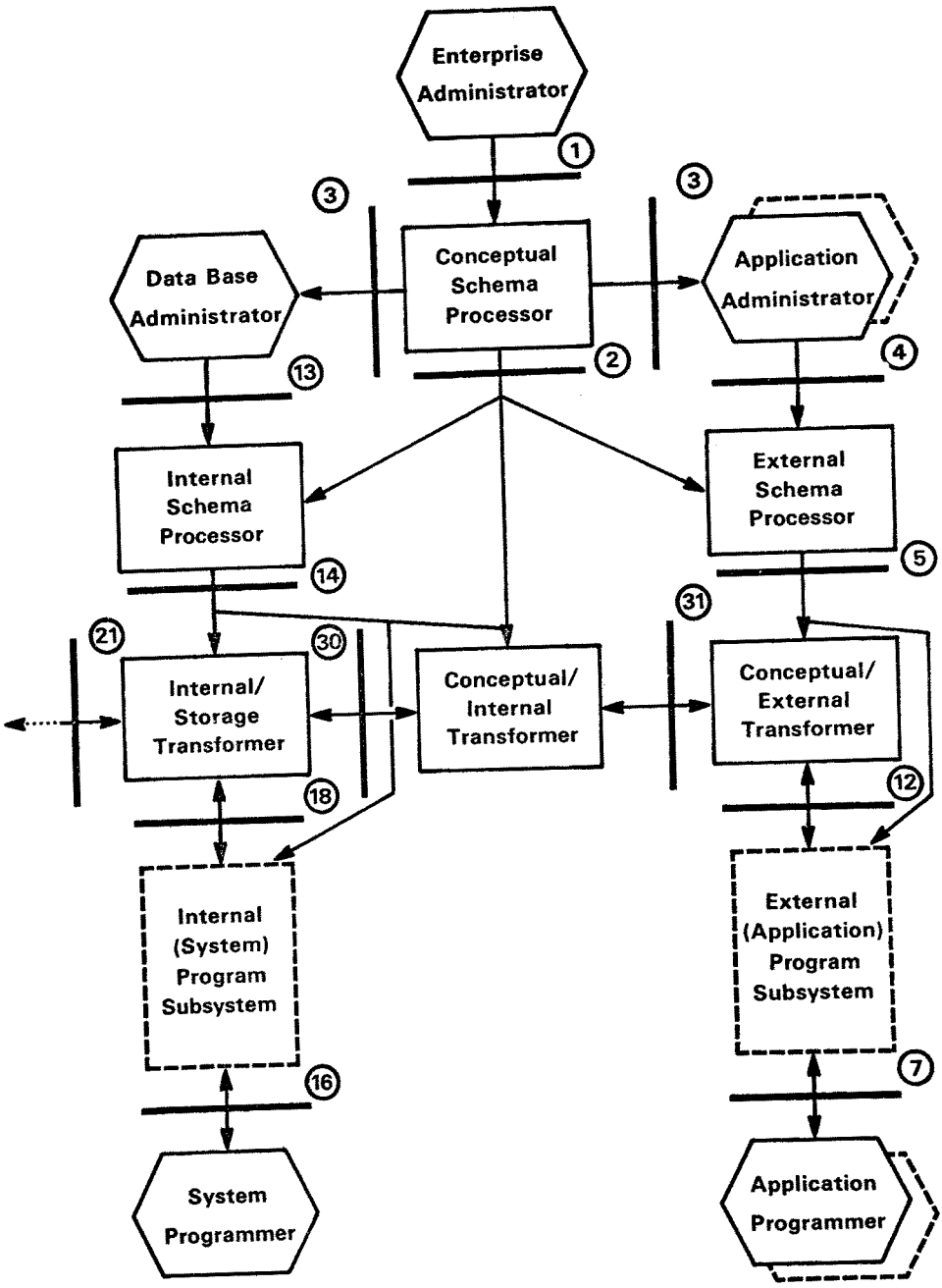
Enterprise Administrator
1
3
3
Data Base Administrator
Conceptual Schema Processor
Application Administrator
13
2
4
Internal Schema Processor
External Schema Processor
14
5
21
30
31
Internal/ Storage Transformer
Conceptual/ Internal Transformer
Conceptual/ External Transformer
18
12
Internal (System) Program Subsystem
External (Application) Program Subsystem
16
7
System Programmer
Application Programmer

Figure
1

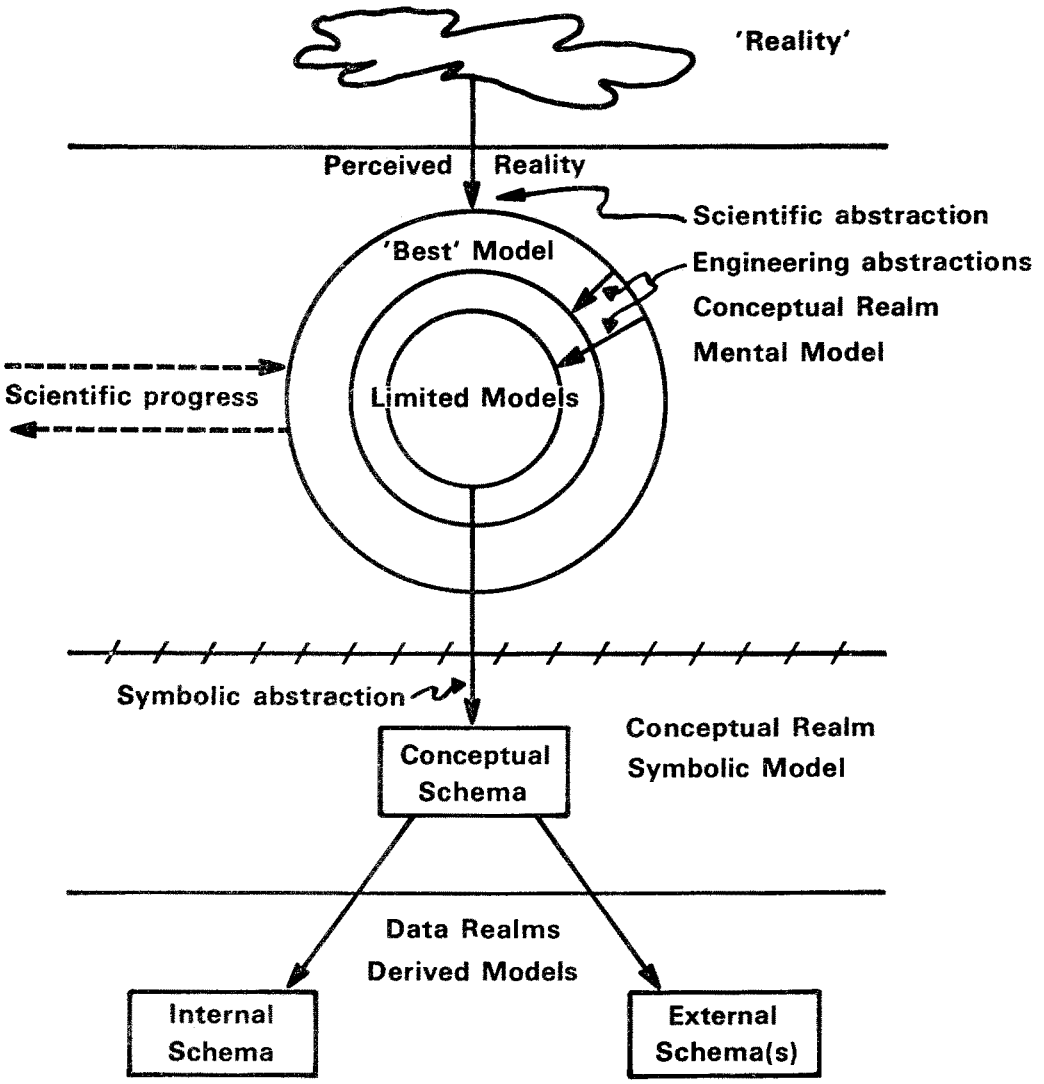
'Reality'
Perceived Reality
Scientific abstraction
'Best' Model
Engineering abstractions
Conceptual Realm
Mental Model
Scientific progress
Limited Models
Symbolic abstraction
Conceptual Realm
Symbolic Model
Conceptual Schema
Data Realms
Derived Models
Internal Schema
External Schema(s)

Figure
2