CHALMERS TEKNISKA HÖGSKOLA

GÖTEBORGS UNIVERSITET

CHALMERS UNIVERSITY OF TECHNOLOGY
GÖTEBORG
SWEDEN

UNIVERSITY OF GOTHENBURG

A Study on Conceptual Data Modeling

Eva Lindencrona-Ohlin

Göteborg 1979

A Study on Conceptual Data Modeling

Eva Lindencrona-Ohlin

Department of Computer Sciences
Chalmers University of Technology
and
University of Gothenburg

Akademisk avhandling för doktorsexamen vid
Chalmers Tekniska Högskola,
Institutionen för Informationsbehandling-ADB
412 96 GÖTEBORG

A Dissertation for Doctor's Degree at
Chalmers University of Technology,
Department of Computer Sciences
S-412 96 GOTHENBURG

ISBN 91–7222–249–2
ISSN 0348–1050
Sundt Offset Stockholm

A STUDY ON CONCEPTUAL DATA MODELING

Akademisk avhandling
som för avläggande av filosofie doktorsexamen
offentligen försvaras
måndagen den 21 maj 1979, kl. 10.00
i sal SB, Hörsalsvägen 1, (Skeppsbyggnad),
Chalmers Tekniska Högskola

av

Eva Lindencrona-Ohlin
fil.kand.

Institutionen för Informationsbehandling-ADB
412 91 GÖTEBORG

Göteborg 1979

151 sidor, ISBN-91-7222-249-2

ABSTRACT

The conceptual level of data base systems was introduced by the ANSI/SPARC Interim Report 1975. During the late 1970's a number of data models and design methods intended for the conceptual level have been suggested.
There does not exist any common agreement on the scope and contents of the conceptual level design area. In this study, an attempt is made to informally define the contents of conceptual level design in terms of problem areas.

Further, an attempt is made to identify and discuss semantical aspects of conceptual level data base design. For this purpose a number of design methods are analyzed. This analysis aims at identification of relevant aspects and problems within the area. Important aspects identified and discussed concern ; inference, redundancy and a temporal dimension. Among these aspects, inference is most thoroughly discussed. Different types of inference are identified. Redundancy is considered as closely related to inference. Requirements of "non-redundancy" in conceptual level models is questioned.

The results of this study hopefully contributes to further developement of conceptual level models and methods, and to further approaches to classification of such methods.

ABSTRACT

The conceptual level of data base systems was introduced by the ANSI/SPARC Interim Report 1975. During the late 1970's a number of data models and design methods intended for the conceptual level have been suggested.
There does not exist any common agreement on the scope and contents of the conceptual level design area. In this study, an attempt is made to informally define the contents of conceptual level design in terms of problem areas.

Further, an attempt is made to identify and discuss semantical aspects of conceptual level data base design. For this purpose a number of design methods are analyzed. This analysis aims at identification of relevant aspects and problems within the area. Important aspects identified and discussed concern ; inference, redundancy and a temporal dimension. Among these aspects, inference is most thoroughly discussed. Different types of inference are identified. Redundancy is considered as closely related to inference. Requirements of "non-redundancy" in conceptual level models is questioned.

The results of this study hopefully contributes to further developement of conceptual level models and methods, and to further approaches to classification of such methods.

ACKNOWLEDGEMENT

During several years I have had the priviledge to work within the CADIS (Computer Aided Design of Information Systems) Research Group in the Department of Information Processing at the Royal Institute of Technology and the University of Stockholm. The research group has had the priviledge to be led by Professor Janis Bubenko j:r.
I am deeply grateful to Janis Bubenko for his stimulating guidance during earlier research work as well as for his encouragement and interest in this study.

I am also grateful to my collegues within the CADIS Research Group, and other collegues at the Department of Information Processing. Special thanks go to Professor Börje Langefors for his valuable criticism and suggestions.

I also want to thank Björn Nilsson at the National Central Bureau of Statistics (SCB) for many stimulating discussions.

The research work has partly been financed by the Swedish Board for Technical Development (STU).

For typing and retyping with never ending patience and for skillful management of an editing program I am truely grateful to Gunnel Gustafsson.

Stockholm, April 1979

Eva Lindencrona-Ohlin

CONTENTS page

1. INTRODUCTION

1.1 Problem specification

In recent years, several data models and some design methods for the conceptual level of data base systems have been presented.

The "conceptual level" of a data base system is an undefined but often used expression. In 1975 ANSI/X3/SPARC Study Group proposed an architecture for data base management systems. The most important aspect of this proposal was the introduction of the so called "conceptual level".
"Current data base management systems envision a two level organization: the data as seen by the system 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 neutral terms "internal" and "external" to make this distinction. In addition, the Study Group has taken note of the reality of a third level, which is called the "conceptual". It represents the enterprise's description of the information as modelled in the data base. This description is that which is informally involved when there is a dispute between the user and the programmer over exactly what is meant by program specifications. The Study Group contends that in the data base world this description must be made explicite and in fact, made known to the data base management system." [ANS-75].

Since 1975, several data models have been suggested as formalisms for the conceptual level data description of database management systems. There has been a debate of what is the "best" data model for the conceptual level. Today, this debate seems to have been replaced by a common understanding of the need of different data models in different situations, for different applications and for different organizations. This, however, indicates the need of a classification system for data models. Such a classification system requires relevant and characteristic properties of data models, in terms of which the different data models can be distinguished and described.

Some research on comparison of data models has been reported. However, there are several problems involved in a comparison of conceptual level data models and design methods.

First, there does not exist any commonly accepted definition or description on what is included in the "conceptual level" data base design.

Different data models and design methods originate from different areas of computer science and information processing as for example from "information systems analysis and design", from "artificial intelligence, specifically natural language processing" and from "file organizations and file design". Depending on the area from which the data models and designs methods originate, different types of problems are focused.

No commonly accepted or commonly used terminology exists. Different authors use the same terms for different concepts or different terms for seemingly the same concepts.

Very little of practical experience of the usage of different conceptual models and design methods has been reported. The practical value and relevance of different properties of data models and design methods is hard to estimate.

To evaluate data models and/or design methods for the conceptual level of data base systems is unrealistic. A classification system would be desirable. One important step towards such a classification system is identification of relevant properties of conceptual level data models and design methods. In order to identify such relevant properties, practical experience and theoretical analysis of different models and methods is required.

The work presented here is an attempt to informally analyze and discuss some semantical properties and problems within conceptual level data base design.

1.2 Relevance of the problem

The conceptual level of data base systems is an area of practical and theoretical importance. Although, to our knowledge, there does not exist any commercially available data base management systems of ANSI/SPARC type, the conceptual level design is essential in design of any data base system and in design of any information system.

The conceptual level design is concerned with specification, determination, formal description and analysis of information to be contained in a (data base) system. When data is shared between different persons and used for different purposes it is important that the meaning of the data can be precisely described. It is important to organize the data in an economical way. At the conceptual level, economy in data organization concerns the efforts that persons involved in the information system have to spend on finding and understanding which information that is (to be) contained in the system, and with the accuracy, concistency and semantical integrity of the information (to be) contained in the system.

There is today, among practitioners, a strong demand for methods and tools for data base design. Conceptual level data base design methods however, also have relevance for design of information systems to be implemented in other ways that by data base management systems. Although, in this work, conceptual level data base design methods are not studied from a practical point of view, the problem of characterizing and describing properties of different design methods has relevance to practical work within data base and information systems design.

Interest and research within the area of data models and design methods for the conceptual level of data base systems is rapidly increasing. The number of papers presented at conferences and the number of articles and research reports published indicates this interest for the area. New data models are proposed for the conceptual level of data base system and methods for designing application data base systems are being presented. There is from a theoretical point of view a demand for research within the area of data models and design methods for the conceptual level.

As an example, in the ISO/TC97 working paper [ISO-78] on concepts for the conceptual schema it is stated:

"It is important to remember that this conceptual level description will eventually be mechanized. It will then be expected to provide significant capabilities which are not available with current data base management systems. A principal requirement will be interactive support for data base access and data base update. In addition to this there will be a need for greatly improved data integrity, and improved support for data base design and data base restructuring. We believe that the situation is hopeful and that certain actions can legitimately be undertakes within the context of standardisation. Firstly, there is considerable confusion over terminology and attempts to clarify this must be welcomed Further work within this area is appropriate.....
Secondly, there is a pressing need for a notation by which particular conceptual schemas may be documented. Such a notation would be of immediate use in the data base design process and in the implementation of data dictionaries. In time it will become an extremely important form of communication between all users of an information system....
Thirdly, it would be extremely useful to have a number of well documented case studies of conceptual schemas.
These could act not only as "bench-marks" for discussions about the conceptual schema, but also as reference documents for those seeking to design their own conceptual schema". [ISO-78].
Research within this area, so far, has not been primarily concerned with models and methods for the conceptual level, but has been concerned with data models in general and information systems design methods in general.

One part of an ANSI/SPARC type data base management system which is not clearly defined is the mapping functions. Mapping functions exist between the external and the conceptual models as well as between the conceptual and internal models. Detailed specification of an ANSI/SPARC type data base management system will require specification of mapping functions.
Research concerning these mappings will involve the conceptual level data definition.
In this case, the conceptual level data definition and the mapping functions are means for acchieving data independence in a data base management system.

Conceptual level data base design has relevance for the information systems design area. The conceptual level data base design is assumed to be independent of any specific data base management system and independent of any specific way of implementing the information system being designed. Thus, conceptual level data base design can be regarded as a part of information systems design. Conceptual level data base design is concerned with

precise descriptions of the information content of a data base system. It includes analysis of information and specification of constraints which are necessary for assuring consistency and semantic integrity in the information contained in the system.
It seems to us that methods for information systems design have been more concerned either with identification of systems requirements in general and/or with file and process design, than with the part of information systems design that concerns formal description and analysis of semantical aspects of the data to be contained in the system. It is also interesting to notice how, from a data base system point of view, the scope of the conceptual level data base design is slowly being widened into what traditionally has not been regarded as data base design, but rather as information systems design. An example of this widened view can be found in [ANS-77] in its description of the task of the enterprise administrator.

In summary, research within the area of conceptual level data base design methods and conceptual level data models has relevance for theoretical work concerning data base management systems of ANSI/SPARC type. It also has relevance for theoretical and practical work within data base design and information systems design. An increasing number of organizations and enterprises are developing and using integrated information systems - eventually implemented with data base systems - an do require methods and models for description and organization of information contained in their systems.

1.3 Related work

In recent years, there has been an increasing demand for comparative studies of data models and design methods.
So far, relatively few such studies have been presented.
Comparative studies related to this one can be characterized as comparisons of either:

- data models
- general information systems design methods
- conceptual models/information models and design approaches.

Comparative studies of data models have been presented by Kerchberg et. al. [KER-76] and by Lochovsky [LOC-77] and of others.
The most well known and exhaustive comparison and classification of data models is "A taxonomy of data models" by Kerchberg and Tsichritzis [KER-76]. This

comparison and classification concerns 23 data models and is divided into comparison of the various models'

- structural properties
- conceptual properties
- sematic properties.

The structural comparison concerns the mathematical structure of the data models. The data models are classified as being graph theoretic or set theoretic oriented. Within the graph theoretic models the nodes and edges of the data models are compared. Nodes are classified as being either structured (for example records, tuples) or unstructured (point sets). Edges are regarded as binary relations and their eventual functionality is used in the classification.

Set theoretic models are classified according to tuple size and according to set nesting. By a nested structure is meant that elements of a tuple can themselves be sets. The set theoreitic models are classified as having a nested or a flat structure.

Similarity and difference of often used concepts as entities, properties, relationships etc. etc. are discussed under conceptual properties. The various models are compared according to which of these concepts that are used, how they are denoted etc. Some of the models for example, do not identify the entity concept. For those models that do identify entities the possibility to relate entities and the possibility to apply properties to entities are discussed.

The semantic comparison classifies the data models as supporting surface semantics, mixed semantics or deep semantics. The authors summarize: "We note that surface semantics models capture the way we speak and communicate verbally with one another. The mixed semantic level represents functional relationships as properties of an entity. Lastly, at the deep semantic level, entities relinquish their importance to data items which are treated as entities. Associations among entities (in graph models) are labeled with long verb phrases which describe the relationship" [KER-76].

In Kerchbergs comparison and classification of data models no distinction is made between data models intended for the different levels of data base systems, i.e. for conceptual-, external- or internal-levels.

An experimental approach to comparison of data models is reported by Lochovsky [LOC-77]. In Lochovsky's experiment, hierarchical, network and relational data models are compared from a "user performance" aspect. Lochovsky's point of view is that, as data base management system are meant to be used by people, the facilities that they provide for user interaction - data models and data manipulation languages - should help to maximize user performance. Aspects of data model/data language affecting user performance are complexity of data model/data language, level and power of the data language, representation capabilities of the data model and conciseness and readability of the data language.
The experiments mainly concern the first of three aspects of data models/data languages and are designed for measuring user effectiveness of a specific class of users, namely of application programmers. The effectiveness of application programmers is related to program developments. The factors considered and measured of program development are:

" 1. portion of correctly coded application programs for a given set of queries and a given application,
2. time required to code application programs, and
3. time required to implement the applicational programs".

Experiments were carried out in which three different DBMS representing a relational, a network and a hierarchical data model were used in program development for three different applications. The result of the experiments are presented in tables showing results of measuring the factors considered for program development for each of the three data models.

Most tutorial textbooks on data base systems include some sort of comparison between relational, network and hierarchical data models. For exampel, Date in [DAT-77] discusses aspects of the three data models as:
- language/operations defined for the data model, especially whether or not the language supports "record-at-a-time" and/or "set-at-a-time" functions
- simplicity, for example, the number of basic constructs in the models.
- theoretical base of the data models.

Comparative studies of design methods seem to be rare. Two recently presented comparative studies do however concern design methods. The one presented by Thaggert and Tharp [THA-77] concern methods intended for the early steps within information systems design. The one

presented by the VIA-project [VIA-77] concerns methods for information systems design in general.

Thaggert's and Tharp's comparative study is presented in the article "A survey of Information Requirements Analysis Techniques".

First, in this article, a framework representing the problem area of information requirement analysis is presented. This framework is built up from 1) The development process, 2) Information, i.e. the problem of identifying information, 3) Decision making and 4) Organization, i.e. the problem of acchieving objectives from the organization. The categories 2-4 are each further divided into three subcategories of problems. In all ten aspects of information requirements analysis are identified.

In relation to these ten aspects, 22 methods for information requirements analysis are classified and compared. For each of the ten aspects, and for each of the 22 methods, the extent to which the aspect or problem is treated in the method is estimated. A three point scale is used in this estimation: 1 = aspect not considered, 2 = recognition given to aspect, 3 = significant treatment of aspect. The result of applying this scale to each of the ten aspects is called the profile of a method.

Important questions raised and discussed by the authors are "First, does the framework represent an adequate view of the problem area?... Second, are the profiles representative of the methods, given this particular framework as the basis for evaluation? [THA-77].

The VIA-project at University of Gothenburg [VIA-77] is concerned with classification and evaluation of methods for information systems design. In their approach a number of methods are classified and evaluated relative eleven main categories of criteria. The first category concerns the theoretical base for the methods. The second concern where, within an over all process of information system design, the methods are primarily applicable. Third, the so called "level" of the methods is discussed. By "level" is here meant distinctions based on defined usage of concepts as method, technique, model etc. The fourth category concerns the degree of formalism in the methods. Fifth, operationality in terms of the methods applicability in a practical situation is discussed. Sixth, acceptance of the methods is studied.

Acceptance is related to category of users, by whom and for what purpose the method has been developed, is introduced and used within an organization. Seventh, flexibility of the methods are discussed. The eighth category concerns the complexity and the possibility of structuring the object (organization) and the information system being designed by the method. Ninth, aspects as time required for users to learn and understand the method and whether or not the method stimulates user engagement is discussed. Tenth the possibility of using the result of applying the method in different implementation techniques (for example, with data base management systems) is studied. Eleventh and last, documentation aspects of the methods are discussed.

Comparative studies of specific interest to this study are those especially concerned with the conceptual or information level modeling.
Such studies have been presented by Senko [SEN-77] and by Bubenko [BUB-76C],[BUB-77].

In a paper named "Conceptual schemas, abstract structures, enterprise descriptions" [SEN-77] Michael Senko compares some conceptual data models according to the number of name categories used by the models.
The base fo the comparison is the "Entity Set Model" proposed by Senko [SEN-72], [SEN-75], [SEN-76]. The Entity Set Models assumes that the real world phenomena are categorized into one of the four basic categories "associations", "association types", "entities" and "entity sets". The elements of these categories can not themselves be stored or processed in a computer or even be talked about without the use of names. In DIAM II three different types of names - Entity Set Names, Attribute Names and Entity Names - are used.

Five data models intended for the conceptual level of data base systems are then compared relative the number of name categories used and relative the name categories of the Entity Set Model.

The idea behind this comparison is that conceptual level data models could be discussed based on "qualitative" and "quantitative" properties. The qualitative properties concern the formalism and the precision of the model and therefore relates to which category of persons involved in the data base design that are to use the model. The quantitative properties relates to questions as "how simple is the model relative to other proposed models, how stable are programs addressed to it when changes occur in the real world" [SEN-77]. The number of

name-categories is a quantitative property of data models and "much of the character of the model depends on the number of categories and the meaning that the researcher assigns to each category" [SEN-77].

A specific aspect of information modeling approaches is comparatively discussed by Bubenko in [BUB-76C]. The specific aspect discussed is the treatment of the temporal dimension. It is claimed that ".. the majority of modeling approaches pay no explicit attention to the temporal dimension. The information model of a particual application is seen as a finit varying set of information objects normally reflecting the current (last observed) state of a model of some real-world system. The state is changed by inserting, deleting and modifying information objects. Sometimes, in these approaches, rules concerning how to define and maintain a set of information objects consistent and complete are discussed. There are, however, a few exceptions to this time-restricted and storage oriented approach". [BUB-76C]. The exceptions are the approaches proposed by Young and Kent [YOU-58], Langefors and Sundgren [LAN-66],[SUN-73]. Falkenberg [FAL-75] and Benci et al. [BEN-76]. Their approaches are then discussed from the point of view as how they incorporate and treat time in their models or methods.

In the paper "Validity and verification aspects of information modeling" [BUB-77], Bubenko suggests and discusses four "dimensions" in the conceptual base of information modeling approaches. The "dimensions" are named

- abstraction levels
- degree of integration
- scope of model
- time perspective.

In the "abstraction levels" dimension, a distinction is made between a name based and a non name based level. At the non name based level the elements of an information model correspond to classes of abstractions of real world phenomena. In this case no decisions on how to refer to individual elements in the classes have been made. At the name based level decisions on, by which names to refer to individual elements in classes of abstractions of real world have been made.
Degree of integration refers to whether or not information models are represented as complex or simple structures. In the first case, relationships between the components of the complex structure are explicitly represented. When an information model is represented by simple structures, it is represented by a set of irreducible components. Relationships between the

irreducible components are implicitely described when the same element occurrs in more than one irreducible component.
Scope of model refers to redundancy in information models. The redundancy concerns information elements which can be inferred from other elements contained in an information model.

In summary, relatively few approaches to classification and/or comparison of data models and design methods have been presented. Among approaches relevant to this study only a few are specifically concerned with models and methods proposed for the conceptual level of data base systems.

1.4 Model or method.

This study is primarily concerned with methods for conceptual level data base design.
However, the distinction between a model and a method for conceptual level data base design is not always clear.

By a data model - a conceptual data model or an information model - is here meant a set of concepts and rules intended for representation and organization of information. (The concept "data model" will be discussed in chapter 3.)
Several data models intended for the conceptual level of data base systems have been presented in recent years. Most papers presenting data models define the concepts - the types of elements - included in the model and the rules by which elements of the different types may be combined to form structures. In examples, occurrences of types of elements are shown. Often, however, nothing is said about how to arrive at a specific set of occurrences of the element types defined, or to a specific combination - structure - of such occurrences.
In an attempt to use any of these data models the data base designer is left with the problem of indentifying the design decisions needed to arrive at the structure, the order in which such design decisions could be made, criteria for making the decisions, criteria for analysis of the structure etc. etc. Approaches to conceptual level data design that - at least to some degree - suggest solutions to any of these problems are here regarded as "methods".
The etymology of the word "method" tells that it comes from Greek and "first meant the path of a person who follows or persues another person, and later came to mean generally a path, a road to something and then the way of

doing something" [KOT-66]. Within philosophy and elsewhere, definitions of the concept "method" could be found. Here the definition given by Kotharbinski is adopted:

- "A method is a mode of action used with the consiousness of repetition of its application in similar cases". [KOT-66].

Thus, data models, presented or proposed without discussions, guidelines or rules for the process of arriving at a data structure for an application are not considered as "methods".

When a conceptual level data base design method is used, the result is an, in some respect consistant, structure of information expressed in terms of some data model. This means that the methods must either comprise a proposal for a data model or refer to an elsewhere proposed data model. Therefore, a study of conceptual level data base design methods must necessarily involve data models.

During the very last years, some approaches to conceptual level modeling have been proposed. In some cases it could be debated whether these approaches correspond to models or to methods.
Here we have chosen not to make this an important issue. In this study, modeling approaches that comprise something which with a generous intepretation could be regarded as guidelines for the process of modeling are regarded as methods.
The reason is simply the - to our knowledge - relatively small number of proposals for conceptual level data base design methods, and our wish to include methods of different characters.

1.5 A "semantical aspect"

Semantics is a concept which can not be defined, in a meaningful way, by a few statements.
It is far beyond the scope of this study even to attempt to define the concept of semantics. An interested reader is referred to relevant litterature within philosophy and linguistics (as for example [KAT-72],[REG-58],[JOH-46].
However, as in this study, different approaches to conceptual level data base design are to be discussed from a "semantical" point of view an explanation is required.

Generally, in this study, a "conceptual schema" or a "conceptual structure" is regarded as a model of an enterprise (some part of the real world). Conceptual level data base design is subsequently regarded as the process of modelling an enterprise into a conceptual structure (fig 1.1).

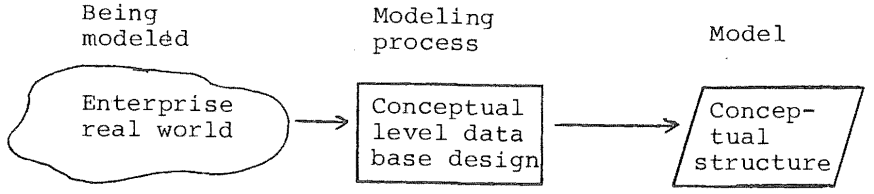


Fig. 1.1

Different methods for conceptual level data base design, identify different intermediate models and transitions within the modeling process (fig 1.2).

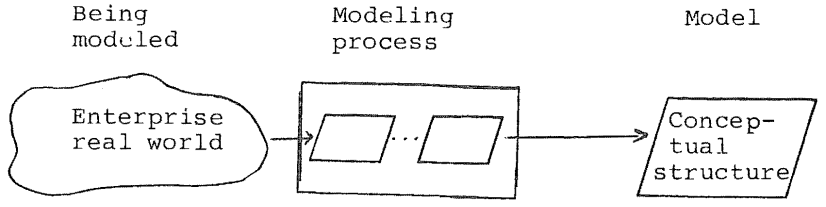


Fig. 1.2

Different authors may give different names to seemingly the same intermediate models or the same name for seemingly different intermediate models. Some examples of names for intermediate models are user views, information requirements, information structures.
The different models - intermediate as well as the resulting model - have to be expressed in terms of some

modeling concepts. For example one method may be said to require as input user views expressed in terms of generic hierarchies while another method may be said to require a user view expressed in terms of functional dependencies. The resulting model may in one approach be expressed in terms of (Codd-) relations. In another approach the resulting model may be expressed in terms of an entity diagram etc.
The different sets of modeling concepts used for expressing the models can be regarded as "languages".
It has been expressed by many authors and in various ways, that semantics concerns the relation between language and reality.
In this study, the "semantical aspect" concerns the relationship between the enterprise/real world and the models and transitions within the process of designing a conceptual structure for a data base.

1.6 Methods included in the study

Abstract data models and design methods have existed long before data base management systems. Already in the late fifties such models and methods were presented within the area of Information Systems design. These early approaches were presented as means for abstract formulation of information processing problems. Classical examples are the approaches of Young and Kent [YOU-58], CODASYL's Information Algebra [COL-68] and Langefors' Theoretical Analysis of Information Systems [LAN-66].

Methods and models for information systems design, to some degree, deal with the same problems as methods and models for conceptual level data base design. However, some differences can be identified.

Information Systems Design methods often cover problem areas which are not considered in methods suggested for conceptual level data base design.
Identification of the information needs of an organization, crude design of an organization's information system including business administrative aspects, partitioning of an information system and decisions concerning computerization of certain subsystems are examples of problem areas which typically are not considered in conceptual level data base design but which may be considered in Information Systems Design.

Abstract formulation of information processing problems is the common area for Information Systems Design and conceptual level data base design.
Characteristic of more recent approaches to such abstract formulation is the focusing on integrated systems.
In the integrated systems, data is to be shared by different users and is to be used for different purposes. In this case the importance of a semantical description of the data is increased. Also, criteria for consistency and semantical integrity becomes more urgent.

Earlier, applications were implemented as tools to support comparatively simple operational functions within an enterprise. The traditional example is a payroll routine. Only a small part - few classes of entities, properties and relationships - were needed in an abstract formulation of such applications. In an integrated data base situation, the scope of the real world being modeled at the same time, is very much enlarged. The problem of defining relevant classes of entities, properties and relationships will be increased. Complex profiles of the intended usage of the information contained in the data base must be considered.
Recent data base applications tend to support not only operational functions but also more complex functions within an enterprise as for example planning and decision-making. Complex functions often require highly aggregated information. Aggregated information in combination with large number of classes of entities, properties and relationships will require systematic ways to analyze and define consistency conditions.

The number of people involved in the design of information systems and of data base systems have increased. In information systems design of today, all people affected by the system being implemented should take part of and influence its design. The design level that is of most importance to influence by the non-technicians is the conceptual level. Although the responsibility of the conceptual level design may be in the hands of an "enterprise administrator" (as suggested by ANSI/SPARC) the methods and models ought to be understandable and adapted to be used by persons of different background, knowledge and interest.
It should also be kept in mind that the conceptual level data description is an important means for communication between the different persons involved in the design and later between the different persons using the systems.

Recent methods for conceptual level data base design do - to a varying degree - focus on the aspects of data base systems design indicated above. Often these recent

methods are based on ideas earlier presented within the more general methods for Information Systems Design. We therefore have chosen, not to include in this study the more general methods but to focus on recent methods specifically suggested for, or applicable to, conceptual level data base design.

The purpose of this study is not primarily to compare different approaches to conceptual level data base design, but to identify and discuss semantical aspects of conceptual level data base design. Therefore, the aim in choosing methods has been that the methods together should cover current and important ideas within the area. The methods included are the following (in alphabetic sequence of authors name):

Benci; 1976 Benci, Bodard, Bogaert and Cabanes presented the paper "Concepts for the design of a conceptual schema" [BEN-76]. This paper identifies separate steps within conceptual schema design and proposes "data models" for two different levels in this design. This approach belongs to those that may be debated whether it is a model or a method. The reason for including the approach is that it recognizes and explicitely includes "dynamic aspects" of the real world.

Brodie and Tsichritzis; In an article "Data Base Constraints" by Brodie and Tsichritzis [BRO-77] a "procedure for construction a schema" is proposed. The procedure is rather summarily presented but is nevertheless included here as it represents an approach to conceptual data base design based on the notation of Abstract Data Types. Some aspects of the design procedure is more exhaustively described in the report "Specification and verification of data base semantic integrity" [BRO-78]

Bernstein; Proposed conceptual schema design by "Synthesizing Third Normal Form Relations from Functional Dependencies" in [BER-76]. Bernstein's approach is already a classical example of so called "synthetic approach" as opposed to the "decomposition approach" to data base design (see [FAG-77]). The classical example of the "decomposition approach" i.e. Codd's normalization [COD-70] [COD-71] is however not included. The reason it that the ideas of Codd's decomposition approach, i.e. normalization are used in other conceptual level design approaches included in this study (for example in Bubenko and Solvberg).

Bubenko; "IAM - An Inferential Abstract Modeling

approach to design of conceptual schema" [BUB-76A]. Characteristic of this method is that it considers and includes inference and derivability aspects of information and that it explicitely considers the temporal dimension of information.

Hubbard and Raver ; presented "Automating Logical File Design" in 1975 [HUB-75]. This paper is a summary of the DBDA - Data Base Design Aid - program product which is a design tool for DL/1 of IMS [IBM-73]. Some parts of the logical file design correspond to what here is called the conceptual level design. These parts are included in this study.

Kahn ; has in the reports "A method for describing information required by the data base design process [KAH-76B] and Novak & Kahn "A framework for logical data base design" [NOV-76B]outlined steps in the procedure of designing a data base system which concerns the conceptual level.

Sheppard-Rund ; A different and an information systems design view of data base design is presented in the "Data Base Design Methodology Part I and Part II" [SHE-75] . In part I a method for definition of keys and attributes is presented and this part is here regarded as conceptual level data base design.

Smith & Smith; presented 1976 the very well known paper on "Data Base Abstractions, aggregation and generalization" [SMI-76]. Abstraction, aggregation and generalization are important activities within the conceptual level design and therefore, although the paper might be regarded as proposing a model rather than a method, it is included in this study.

Sølvberg ; presented together with Auerdal in 1976 a paper "A multi-level procedure for design of file organizations" [AUE-76]. The first two steps of this procedure - specification of the information processing problem and transformation of the specified information structures into a logical model of a file organization - have been elaborated by Sølvberg and are presented in the report "A model for specification of phenomena, properties and information structures" [SØL-77A].

1.7 Outline of the presentation

In chapter 2, the expression "conceptual level data base design" is put into a context and its evolution is indicated. The context of "conceptual level data base design" is regarded as concerning "Information systems design", "Data base management systems" and "Data Models".
Chapter 3 presents the frame of reference of this study. In this frame of reference a scope and context of "conceptual level data base design" is suggested. The scope and context is described by five problems areas identified within "conceptual level data base design".
In chapter 4, nine approaches to conceptual level data base design are informally analyzed from a semantical point of view. First, referring to problem areas identified in chapter 3, the main problems attacked by each method are identified. Thereafter, for each method, characteristic semantical properties and/or problems are informally analyzed.

Chapter 5 is a summary and a discussion of semantical properties and problems identified in the informal analysis of the different approaches to conceptual level data base design. In the last section of the chapter, conclusions are presented.

2. BACKGROUND

The purpose of this chapter is to put the expression "conceptual level data base design method" into a context and to indicate its evolution.

Conceptual level data base design can be regarded as an extension of ideas within the information systems design area, directed towards the design of integrated data base systems. Very early, the need for abstract formulation of information processing problems was recognized within the information systems design area. It is, however, not until recently that models for abstract specification of data base systems have become an integrated part of data base systems and that methods for abstract formulation have become recognized as an important part of data base design.
The evolution of data base management systems is naturally an important component in the context of conceptual level data base design.
Data base management systems include languages for data definition. Such languages are based on some explicitly or implicitly specified data model. Data models are necessary components of conceptual level data base design.
Thus, important to a context for "conceptual level database design" are ideas within information systems design, data base management systems and data models.

2.1 Information systems design

As early as in the end of 1950's, methods for design of computerized information systems began to appear. The evolution of such methods and an exhaustive survey of methods is presented in "Systems Analysis Techniques" by J. Couger and R. Knapp [COU-74]. To give an idea of the kinds of problems attacked by the early methods, three classical examples are here summarily presented.

2.1.1 Young and Kent

In 1958, Young and Kent presented a notation for abstract formulation of data processing problems.

The process of designing a computerized information

system was regarded as consisting of three stages: "1) Systems Analysis - the task of determining what is to be done, 2) Programming - a statement of how it is to be done, 3) Coding - a translation of this statement into machine language." [YOU-58]. The notation for an abstract formulation of data processing problems is intended for the first of these stages. The purpose for the notation is to provide a precise and abstract way of specifying the informational and time characteristics of data processing problems. A basic assumption was that such an abstract formulation could be done without the specification of file- and program structures. Further the approach assumes that the output from the data processing system to be designed have been stated in advance and is not to be changed and that input e.g. its information content but not necessarily its structure is known.

The basic concepts used in the abstract formulation were Information Sets, Documents, Relationships and Operational Requirements. Relationships between Information Sets and characteristics of the relationships were defined and described as well as relationships between Documents and/or Information Sets. Time aspects are considered and described in terms of intrinsic and extrinsic time points, which makes it possible to describe elapsed time relationships. The methodological approach is to start from the given output requirements and to describe and analyze how elements in the output requirements could be produced from each other or from input.

2.1.2 The Information Algebra

The CODASYL Development Committee presented in 1962 "An Information Algebra" intended as a framework for description of data processing problems.

The Information Algebra was intended as a step towards a theory of data processing on which development of new programming languages could be based. Couger describes the concepts upon which the Algebra is based as concepts that have been implicitly understood for years namely that "An information system deals with objects and events in the real world that are of interest. These real objects and events, called "entities" are represented by data. The data processing systems contains information from which the desired outputs can be extracted through processing.
Information about a particular entity is in the form of

"values" which describe quantitatively or qualitatively a set of attributes or "properties" that have significance in the system." [COU-74].

Thus, the basic concepts are entities, properties and values. Each property has a set of values associated to it, and there is only one value associated with each property of each entity. A coordinate set is a finite set of distinct properties. A property space is the Cartesian product of the property value sets associated to the properties of the coordinate set. An entity is represented by a Datum point which is a point in a property space. Set operations (union, intersection, difference) are defined for subsets of the property space (areas) and more complicated set oriented operations (corresponding to, for example, arithmetic operations) are defined for ordered points of the property space.

2.1.3 Langefors' approach

In 1966 Langefors presented a "Theoretical Analysis of Information Systems" (some parts had been presented in papers in 1963). The basic concept used in an abstract formulation and in abstract design of an information system is an "e-message" (elementary message). An "e-message" is defined as the smallest unit of information carrying a semantic meaning. "E-messages" are described by their "e-message type". An "e-message type" can be illustrated as the intention of a 3NF relation with a fixed meaning of the attributes of the relation. Two kinds of "e-message types" are defined. In the first kind of "e-message type" the attribute of the relation refers to one object, one property of that object, one property value and one time point. In the second kind of "e-message type" the attribute of the relation refers to one object, one relationship, and the object - or possibly - objects - related to the first object by the relationship and a time point.

"E-messages" can be related by precedence relationships. An "e-process" (elementary process) is a process that produces one elementary message from a set of precedent "e-messages".

A matrix algebra for description of precedence relationships is introduced.

There is an explicite design method presented. The

systems analysis starts with identification and definintion of basic functions within an enterprise or organization. For each such basic function defined, its information requirements are specified. Each function may require two kinds of information, operative information and directive information. Operative information is the information needed to monitor the basic function and directive information is the information needed for decision concerning the control of the functions. The information requirement of each basic function (operative or directive information) is defined in terms of "e-message types". Time intervals and periods during which the information is valid or at which the required information is needed are described. Through precedence analysis of required "e-message types" the set of "e-message types" required in the information system is defined.
The work presented in 1966 has later been extended by Langefors and by Sundgren and presented in several papers as for example in [LAN-75], [SUN-73].

2.2 Data base management systems

The evolution of data base management systems can be viewed from a hardware and from a software point of view. In [SEN-72] the important steps of the hardware evolution are described as the step from card processing and wired program processing to stored program machines using magnetic tape, and next the steps from magnetic tape processing to the use of direct access storage devices.
From the software point of view, important steps in the evolution of data base management systems, which took place in parallel with the development of magnetic tape processing were:

"1. Program representation in terms of binary bits or decimal digits or punched cards for direct reading into the computer.
2. Symbols for storage references and machine operations to be translated by assamblers.
3. Parameterized assembler language subroutines for processing files.
4. Compilers for the translation of procedures described in English-like (COBOL) or mathematical (FORTRAN-ALGOL-PL/I) terms into machine code" [SEN-72].

Connected to the introduction of direct access storage devices were development of methods for direct accessing and file organizations.

The evolution of data base management systems is closely related to the control of data [FRY-76].
In the fifties and sixties, programmers had the control and custody of the data. Generally, datawere not shared between applications and not even between programs. The possibility to let multiple applications share the same data and thus to let multiple programmers access the same data was an important step towards data base management systems. The COBOL-Language [JOD-60] was important as it introduced separate descriptions of data and of the processing of data. This separation of data and processing descriptions was a step towards a centralized data description which in turn was needed to allow several users to share the same data.

The CODASYL DBTG Proposals [COL-69],[COL-71], were the first widely recognized approaches to centralized data description. In the CODASYL DBTG Proposal [COL-71], it is explicitely stated that the responsibility and the definition of shared data is moved from the programmers to a centralized function called "the data base administrator". The CODASYL Proposal represents a data base system architecture of two-levels. Later, in ANSI/X3/SPARC [ANS-75] and in DIAM I [SEN-72] the two-level architecture was expanded and three or four levels of data descriptions within a data base management system were proposed.

2.2.1 Two-level systems

The CODASYL Proposal was important as it introduced two levels of data descriptions within a data base management system. The two levels are referred to as "logical" level and "physical" level. At the "logical" level the CODASYL DBTG used SCHEMA and SUB-SCHEMA definitions. The SCHEMA is the centralized data description and it constitutes a compromise of the data structures required by the different application programs. SUB-SCHEMAS are subsets of the SCHEMA and correspond to the individual data structures of the application programs. The data description of the SCHEMA and of the SUB-SCHEMAS are expressed in terms of fields, records, sets and areas.
At the "physical" level, units of stored data and their allocation on secondary storage is described. The purpose of the separation of a "logical" and a "physical" level of data description was to acchieve data independence in the sense that data could be "physically" reorganized without impact on the "logical" data description and on the application programs. 1)

---

1) The CODASYL DBTG Language Proposal has been criticized for not having achieved enough data independence as the separation of the "logcial" and "physical" levels defined by SCHEMA DDL is not regarded to be satisfactory. An example is the AREA concept which is specified in the SCHEMA DDL, i.e. at the logical level. Another example is the specification of access methods at the logical level.

The most commonly used data base management systems of today (for exampel IMS, IDS, DBMS-1100) are examples of two-level systems.

2.2.2 Three-level systems

Three levels of data description within a data base management system was proposed by the ANSI/X3/SPARC Report [ANS-75]. The three levels are called conceptual-, external- and internal level.

The three levels are said to correspond to different views of the data in the data base. The conceptual level, expressed in the conceptual schema, is the enterprise's "real world" view of the data in the data base. This view is shared among the various users who agree on the abstractions and classifications of real world phenomena represented by the enterprises "real world" view.

Different users will operate on different, possibly overlapping subsets of the enterprise's real world model. The application programmers might want these subsets of the real world model to be structured and described in ways which are adequate to their intended operations and to the programming languages that they use. These, possibly differently structured and described, subsets or mappings of the real world model are called external schemas.

The internal schema, represents the "machine view" of the data in the data base, and describes how data is stored and accessed in the data base.

It should be noticed that the conceptual schema, in the ANSI/X3/SPARC Interim Report, is regarded as a "real and tangible item" which is proposed to exist in machine readable form.

Three administrator roles have been identified, each with the responsibility of describing and maintaining the schema corresponding to their view. Figure 2.1 shows a subset of the data base management system architecture proposed by ANSI/X3/SPARC.

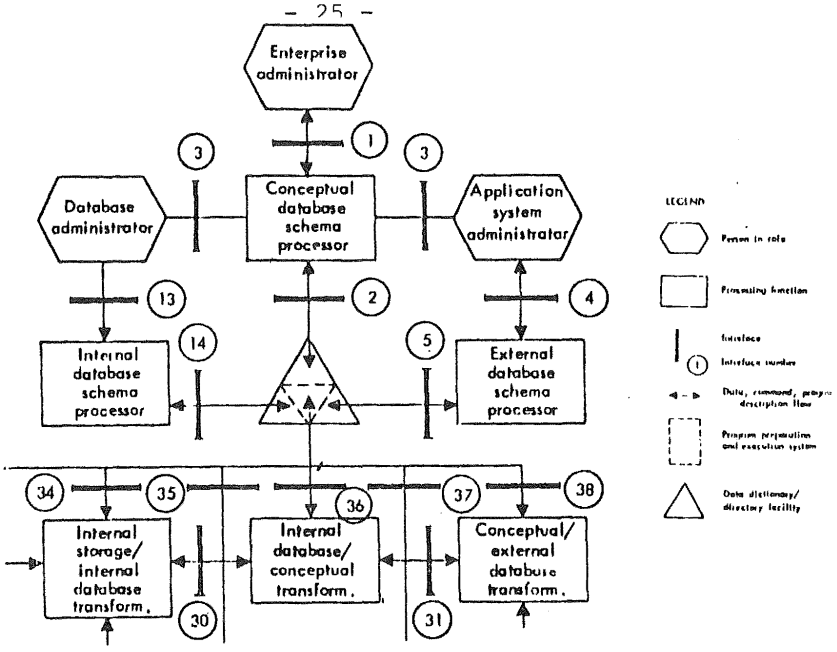


Fig. 2.1

So far, - to our knowledge - there does not exist any commercially available three level data base management system. An experimental three-level system has been implemented by Nijssen [NIJ-76].

2.2.3 Four-level systems

Already 1972 Senko presented the DIAM I Model. DIAM stands for Data Independence Accessing Model and could be regarded as an overall architecture model for data base management systems as well as specifications of models to be used at the different levels.

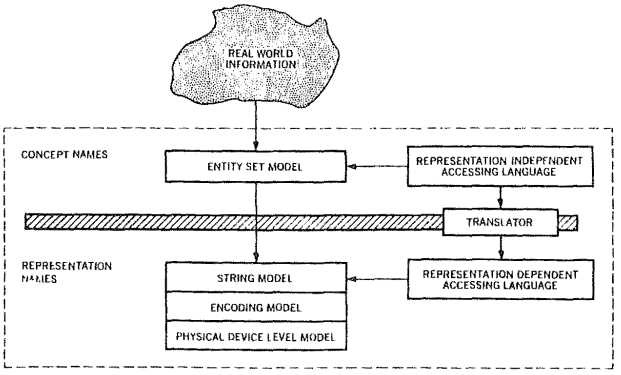


Fig. 2.5 from [SEN-72]

The overall architecture consists of a hiearchy of four models. At a "representation independent level" there is the Entity Set Model (see 2.3.1.2) which is accompanied by a Representation Independent Accessing Language. At the lower, representation dependent, levels the String Model, the Encoding Model and the Physical Device Model are proposed.

The String Model is intended as a means to describe a representation of the Entity Sets, defined in the Entity Set Model, in terms of access paths. Design of a String Model concerns efficiency of representation, search and maintenance, i.e. use of indexes, serial v.s. direct search, etc.

The Encoding Model is a model describing bit-pattern encoding of the string paths. At the lowest level, there is the Physical Device model which concerns space and overflow-handling rules for physical subdivisions of the devices. The DIAM I Model has later been revised and extended by Senko [SEN-75].

2.2.4 Hierarchical, Network and Relational systems

Data base management systems are often classified as being either hierarchical-, network- or relational systems.
This classification is relevant for two level systems and refers to the data model used at the "logical level". The data model determines which types of elements and which type of structure that can be described as a logical data structure of a data base to be managed by a data base management system.
The data model of a data base management system effects the complexibility of the language(s) used for accessing the data in the data base.

Examples of hierarchical data base management systems are IBM's Information Management System - IMS [IBM-73] and System 2000 [MRI-72].
Data base management systems based on the CODASYL-DBTG proposals are typical representatives of network systems. Examples are UNIVAC's DMS-1100 [UNI-73] and Cullinan's IDMS [CUL-75]. Relational data base management systems are based on the Relational Data Model proposed by Codd [COD-70], IBM's System R [AST-76] and the INGRESS System [STO-75] are examples of implemented relational systems.

2.3 Data Models

By data model is here meant a set of modeling concepts and a set of rules for combining these concepts to form structures.
The hierarchical, network and relational data models are often referred to as the three principle data models [DAT-75], [TSI-77].
When hierarchical and network data models are discussed

they are most often represented by the data structuring facilities of IBM's data base management system IMS and by data structuring facilities in data base management systems based on the CODASYL-DBTG proposals.

In a network data model, as proposed by CODASYL, information is represented in terms of records and sets. Records and sets are generic concepts which in turn represent sets of record occurrences and sets of relationships between record occurrences. Sets may be "information carrying" as well as "non-information carrying" [MET-75]. In a general network data model sets may represent relationships of any of the types 1:1, 1:M, M:1 or M:M. However, in the CODASYL-proposals only relationships of the types 1:1 and 1:M may be represented by a set. In order to represent M:M relationships extra records may have to be introduced in a CODASYL network. Another restriction of a CODASYL Network compared to general networks is that sets may only be defined between records of different types.

A hierarchical structure is a special case of a network structure. In a hierarchical data model records are related by links so that tree structures are formed. Each link correspond to a 1:1 or 1:M relationship between record occurrences, and thus, each record occurrence can be related to only one record occurrence at the next higher level in the tree.

The relational model proposed by Codd [COD-70] is based on the concept of time varying relations.
A relation is defined as a subset of the Cartesian product $S_1 \times S_2 \times S_3 \times \ldots S_n$ where $S_1, S_2, S_3 \ldots S_n$ are domains. Relations are represented by tables. Each row in such a table correspond to a tuple and each column of a table correspond to a domain of the relation. The column head contains names of domains or names of the roles in which the domains are used in the relation. A domain in a specific role is called an attribute. The column head, i.e. the names of domains and/or attributes of the relation is called the intension of the relation. The rows or tuples of the table constitute the extension of the relation. It is the extension of a relation that is time varying.
Each relation must have a primary key defined for it. A primary key is an attribute or a combination of attributes whose values uniquely identify the tuples of a relation.
The concept of "functional dependency" is of importance in the relational data model.
Given a relation R, the attribute B of R is functionally dependent on the attribute A of R if and only if, each

A-value in R is associated with precisely one B-value in R at any one time.
Functional dependencies are used in the process of "normalization" of relations. The choice of a specific set of relations for an application will be of importance to the semantics of the operations on the data. To avoid deletion, insertion and updating anomalies Codd proposed relations to be in (3NF) third normal form. "Normalization is a step-by-step reversible process of replacing a given collection of relations by successive collections in which the relations have a progressively simpler and more regular structure" [TSI-77][COD-71]. The normal forms and the process of normalization are described in Codd's papers [COD-70],[COD-71],[COD-72] and in most tutorial textbooks on data base systems (for example [DAT-77],[TSI-77]) and in many other papers.

Both CODASYL's network model and the relational model have been proposed as conceptual level data models. Criticism against the use of these models at the conceptual level has been conveyed, for example by Senko [SEN-75] and by Kent [KEN-76]. As an example, in the article "New Critiera for the Conceptual Model" [KEN-76] Kent points out how several properties of the CODASYL model are inherited from the record technology and how the characteristics of the record technology differs from the characteristics and requirement of information representation. In his outlook for future data models Kent concludes: "The popular models of today are driven by computer technology. This is appropriate for the external model, which must be primarily concerned with effective and efficient algorithms for processing data and for internal model, whose primary concern it to provide efficient storage structures and access paths. ... What has not yet been generally realized is that the criteria for a good conceptual model may be quite different from the criteria at the other levels. The priorities are reversed: the conceptual model is primarily a model of the information used in the enterprise" [KEN-76].

2.3.1 Conceptual level data models

In recent years, several data models have been presented as models specifically intended for the conceptual level of data base systems. Examples of such models are "Data Semantics" [ABR-74], "Semantics of data bases: the semantics of data models" [BIL-76], "The Binary Logical Association Model" [BRA-76] "The Entity Relationship Model" [CHE-76], "The Entity Set Model [SEN-72], "The Infological Model" [SUN-75].

Data models proposed for, or applicable to, the conceptual level of data base systems originate from different areas of computer science and/or information processing.
Here, we have chosen three models - The Semantic Network Model, The Entitiy Set Model and The Entity Relationship Model - to be summarily presented as examples of conceptual level data models. The reason for choosing these three particular models is that they together cover many of the concepts and ideas within conceptual level data models and that they may be regarded as representatives of three different areas of origin. The three areas are artificial intelligence, file organization and design, and information systems analysis and design.

Within artificial intelligence, research concerned with "Knowledge and belief systems, Semantics, Natural language interpretation and Conversation" [NEV-73] have proposed and used several different models for representation of knowledge (for an overview see [NEV-73] or [WIN-74]). Examples of such models are the conceptual model proposed by Schank [SCH-73], the procedural model proposed by Winograd [WIN-73] and the family of conceptual models called semantic networks as for example Simmons [SIM-73] and Quillians [QUI-68].

Strangely enough, it is not until very recently, that the ideas and models for representing knowledge developed within the artificial intelligence field have been applied in conceptual level data modeling for data base systems. Examples of this, however, are the modeling approach proposed by Smith&Smith [SMI-76] and the data model "The Semantic Network Model" proposed by N.Roussopoulos and J.Mylopoulos [ROU-75].

The Semantic Network Model is presented as a model for representing knowledge about the real world, to be used as a definition of the meaning of the data in a data base. The semantic network is a labeled, directed graph. The nodes correspond to the basic modeling concepts concept, event, characteristic or value nodes. The edges correspond to predefined cases and have a number of associated semantic properties.

Concept nodes represent physical or abstract objects of the real world, event nodes represent events or actions which occur in the real world. "Their representation is based on a case-grammer model (Fillmore) and consists of an event node and several nodes that specify who plays the roles (or fills the

cases) associated with an event. For example, fig 2.3 illustrates a representation of an instantiation of the event "supply" with "western united" playing the role of "agent" and "source", "eastern.co" playing the role of "destination" and "part.#,7395" being the supplied part" [ROU-75].

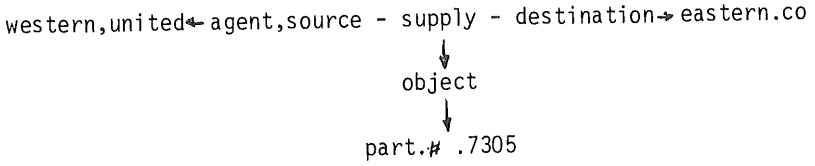


Fig. 2.3

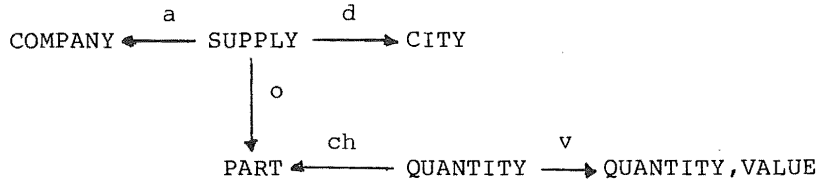
Characteristic nodes are used to modify other nodes or to represent states. Value nodes represent values of characteristics. The labels of the nodes are used for reference purpose.

The cases or roles defined are: agent (a), affected (aff), topic (t), instrument (i), result (r), source (s), destinaton (d) and object (o).

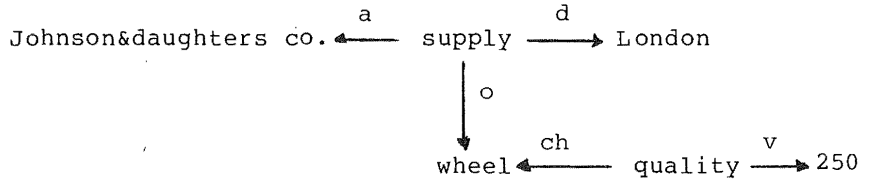
A very characteristic property of the Semantical Network approach is that the nodes may correspond to types as well as to tokens. Nodes corresponding to types are called generic nodes and nodes corresponding to tokens are called instantiations. Semantic nets may consists of only generic nodes, only instantiations or a mixture of both types of nodes. Structures consisting of only instantiations must, in order to be meaningful, be matched by structures of generic nodes.

Example:

1) Generic nodes:



2) Instantiations:



The authors propose that most isolated phenomena of the real world can be represented by semantic nets as described above. However, when "larger chunks" of knowledge are to be represented, so called "scenarios" are used. A scenario is a collection of events, characteristics and mathematical predicates related through causal connectives such as "prerequisite" ("prereq") and "effect". One may regard a scenario as a pattern or template which when matched by a structure, causes various kinds of inference and predictions to be made. Moreover, only structures which are matched by some of the scenarions on the semantic net are meaningful to the systems." [ROU-75].

Scenarios or instantiations of scenarios may be semantically related into semantic networks. Examples of such semantical relations between scenarios are the "axes" or "dimensions" named "SUB" (subset) and "DEF" (definitional) and "PART" (part of).

The "SUB" axis is used to create hierarchies along which semantic properties can be inherited. The "DEF" axis is used to define semantics of event and characteristic

nodes. The "PART" axis defines part-of relationships between nodes.

Semantic Networks have been used as models for knowledge representation for natural language processing. The Semantic Network data model as proposed by Rousopoulos and Mylopoulos may also be used as a data model for the conceptual level of data base systems. Further, the data model has been used by Rousopoulos as a base for the CSDL language. The CSDL language is intended for conceptual schema definition in the design of data base applications [ROU-78].

One major problem of file organiaztion and file design is to acchive "data independence". The separation of a "logical" and a "physical" level of files or data bases was an attempt to acchieve data independence, in the sense that it should be possible to physically reorganize a file or a data base without effects on the programs that used the files or the data base. The data models that were (and still are) used in the two-level data base system, for exampel the CODASYL network and the IMS hierarchies, however, did not acchieve a satisfactory separation of the logical and the physical levels. In describing the background of the DIAM model, Senko writes "we decided that networks and hierarchies both had obvious access paths dependence built into their naming structures and that they required quite complex data description facilities" [SEN-77].
The purpose of the DIAM model is to acchieve a separation between a representation and access path independent and a representation and access path dependent level within a data base system.

The Entity Set Model is a part of the multilevel model DIAM (see fig 2.4). The DIAM model - Data Indenpendent Accessing Model - can be regarded as an architecture for a data base management system.

Figure 1 Data independent accessing model

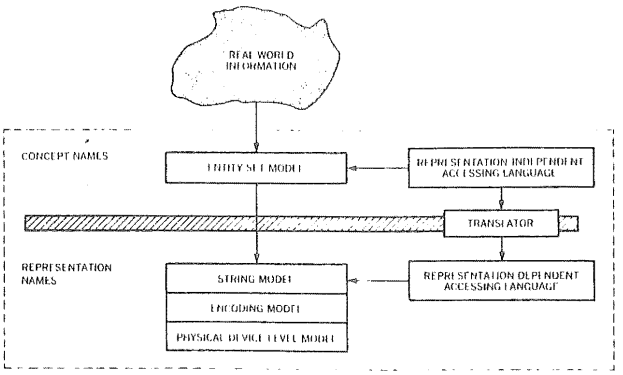


Fig. 2.4 from [SEN-72]

The Entity Set Model is intended for a representation and access path independent data description in a data base system.
In the Entity Set Model, the difference between entities and their representation in terms of names is stressed. An entity is defined as "anything that has reality and distinctness of being in fact or in thought. e.g. objects, associations, concepts and events" [SEN-72].
A name is a symbol or a combination of symbols by which an entity is known. An entity may have several names. Names can be expressed in alternative coding schemas. A name representation is a name expressed in a selected coding schema. Unique names for entities are called Entity Names.

Entities with similar properties are grouped into Entity Sets. The Entity Sets are given unique names - Entity Set Names. Set of unique entity names are called Entity Name Sets and these sets are given unique Entity Name Set Names.

An Entity Set model for an application is built up by Entity descriptions. Entities are described by other entities used in descriptive roles. The roles are given Role Names. An example is: "The Entity named PART NAME/GEAR may be described by the Entity named COLOR NAME/BLUE in the descriptive role, COLOR OF PART" [SEN-72]. A characteristic property of the Entity Set Model is that all real world phenomena are regarded and represented in a uniform way, and thus that no distinction is made between for example entities, properties and relations. This uniform way of regarding real world phenomena is based on the observation that an association between two entities, as for example between the entities represented by the Entity Names PART and BLUE is descriptive in both directions.

An entity description consists of triplets of names that together form a description of the identified entity. (See fig. 2.5).

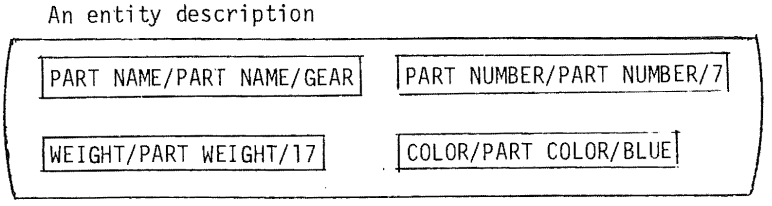


Fig. 2.5

Entity Descriptions for entities from the same Entity Set are collected in (Entity) Description Sets. These Description Sets constitute the data definition of an application in terms of the Entity Set Model. The Description Set is also a basis for a "data directory" and for definition of the representation dependent data models of DIAM, e.g. the String Model, the Encoding Model and the Physical Device Level Model.

The DIAM model has later been modified and extended in DIAM II. DIAM II has the same three models at the representation and acccess path dependent level as DIAM. The Entity Set Model level is called the Information level. On top of the Information Model an End User Level is introduced. The End User Level is introduced in order to support various user views. A representation and access path independent language named FORAL has been specified and implemented. [SEN-76], [SEN-77].

From an information system design point of view, a conceptual level data model is intended for modeling information from an enterprise's point of view. The modeling concepts attempt to be adapted to a "natural" view of the real world. Although it can be debated what a "natural" view is, and although these models can be regarded as developed for traditional business administration applications they do attempt to avoid modeling concepts inherited from the way in which computers traditionally work. An example of such a model is the Entity-Relationship model proposed by Chen [CHE-76].

The Entity-Relationship Model is based on a "multilevel view of data". The levels identified are:

"(1) Information concerning entities and relationships which exist in our minds.
(2) Information structure - organization of information in which entities and relationships are represented by data.
(3) Access-path-independent data structure - the data structures which are not involved with search schemas, indexing schemas, etc.
(4) Access-path-dependent data structure.

The Entity-Relationship Model is intended for the two first of these levels. At the first level, the basic

modeling concepts used are entities (entity sets), attributes and relationships. Entities are things that may be distinctly identified and relationships are associations among entities.

Entities that have common properties are classified into (not necessarily disjoint) entity sets. An entity set has a predicate associated to it, by which entity membership can be tested. A relationship set is a mathematical relation among n entities taken from Entity sets, $\{[e_1,e_2,\ldots,e_n]e_1 \epsilon E_1\ e_2 \epsilon E_2 \ldots e_n \epsilon E_n\}$.
Each tuple of entities $e_1,e_2,\ldots.e_n$ is an relationship. An entity may play a role within a relationship. Enterprise relevant information about entities and relationships is expressed as attribute - value pairs. Values are classified into value sets. Attributes are formally defined as a function which maps from an entity set or a relationship into a value set or a Cartisean product of value sets:

$$f: E_i \text{ or } R_i \longrightarrow V_i \text{ or } V_{il} \times V_{i2} \times \ldots. \times V_{in}$$

An example of attributes defined on a relationship set is shown i figure 2.5.

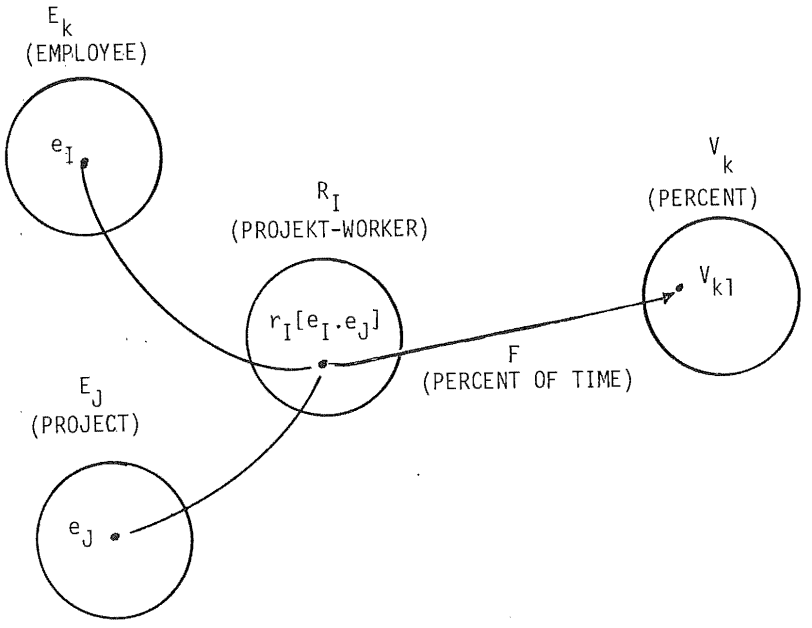
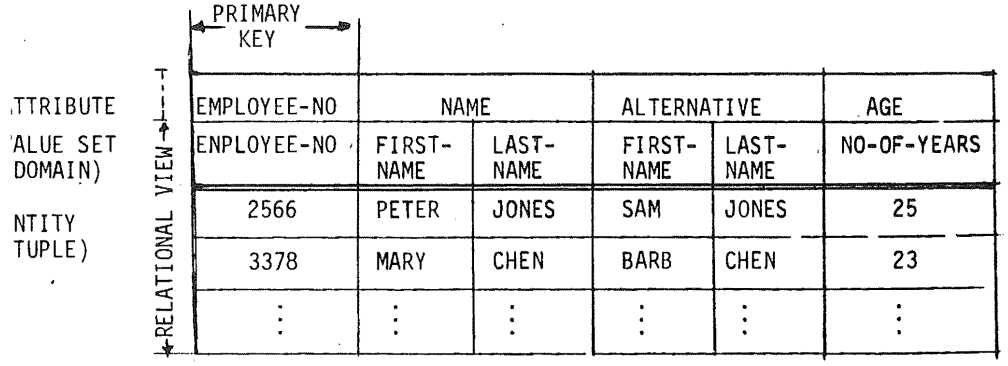


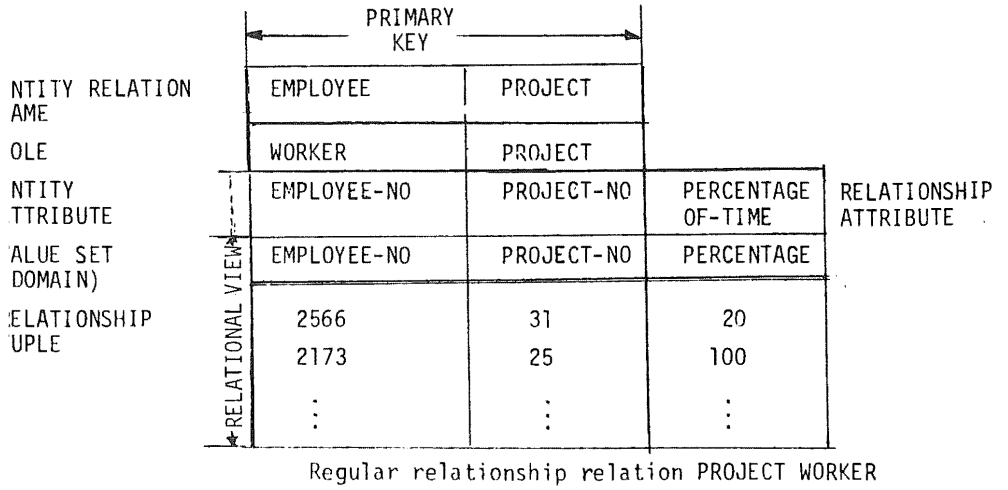
Fig. 2.6 from [CHE-76]

The entities and the relationships of the first level do only exist in peoples minds.
At the second level - the information structure level - entities, relationships and information about these are represented as names in relations. Information about entities of an entity set is organized into an entity relation where each row is an entity tuple (see figure 2.7). Information about relationships of a relationship set is organized into relationship relations where each row is a relationship tuple (se figure 2.8).
The primary key of an entity relation correspond to an entity primary key, i.e. one of alternative attributes or groups of attributes such that the mapping from the entity set to the corresponding value set(s) is one-to-one.



Regular entity relation EMPLOYEE

Fig. 2.7 from [CHE-76]



Regular relationship relation PROJECT WORKER

Fig. 2.8 from [CHE-76]

Sometimes, entities of an entity set can not be uniquely identified by any (set of) attribute(s) defined for that entity set. In such cases, the entities are identified by a relationship to entities called an ID dependency.
Relationships between entity sets may also be of the type "existence dependency", i.e. entities of the dependent entity set will not exist unless the related entities exist.
Entity relationships of the type "existence dependency" are called "weak relations" to distinquish from regular entity relations.
ID dependencies are automatically existence dependencies while existence dependencies not necessarily have to be ID dependencies.
A graphical notation for description of entity relationships is introduced. Fig 2.9 shows an example of an entity-relationship diagram for analysis of information in a manufacturing firm.

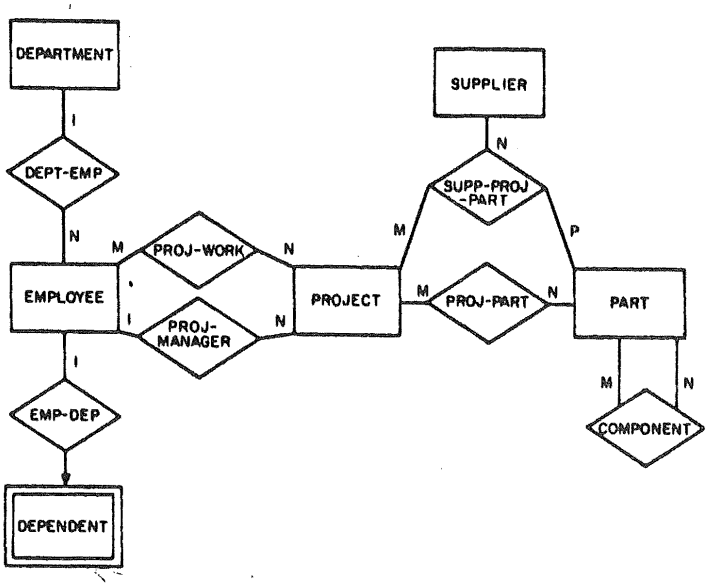


Fig. 2.9

Procedures for transformation of entity-relationship structures into data structures based on the Relational Data Model as well as on a Network (CODASYL) Data Model

have been outlined by Chen [CHE-79].
Languages for conceptual schema definition as well as for data manipulation are also outlined.

2.4 Summary

This chapter describes a background and a context for "conceptual level data base design".
The main components in this context are information systems design, data base management systems and data models.
As a background, examples of early approaches to information systems design are summarily presented. The evolution of data base management systems is indicated.
Data Models and especially data models intended for the conceptual level of data base systems are exemplified.

3. CONCEPTUAL LEVEL DATA BASE DESIGN

The purpose of this chapter is to present a frame of reference for this study.
The area of conceptual level data base design is not well defined. Different authors have different opinions about the scope and the contents of the area. No commonly accepted or commonly used terminology exists. An example of this are the names used for the methods included in this study, as for example "logical data base design", "conceptual schema design", "information structure design".

First in this chapter, the concepts "data model" and "data structure" are discussed. Second, these concepts are related to the data base management system architecture proposed by ANSI/SPARC. Third, the author's view of "conceptual level data base design" is described. The scope of conceptual level data base design is described by six problem areas. The contents described in detailed discussions of these problem areas.

The scope and contents of conceptual level data base design as presented in this study is heavily influenced by other authors within the area. Exact references to where different ideas have been picked up are hard to give. Therefore, references are generally not given in this chapter. However, two authors by whom this study is much influenced should be mentioned. Very early, Langefors presented works on information requirements and messages which have had an impact on this and many other scandinavian works (for example [LAN-66],[LAN-68]). Bubenko's work on inference analysis in information structure design has contributed substantially to the frame of reference here presented (for example [BUB-76] [BUB-79]).

3.1 Data Models and Data Structures

The concepts "data model" and "data structure" have different meaning in different contexts. For example, some authors associate the term "data structure" to the way in which data are organized in main or secondary memory of a computer, while others associate "data structure" to constitute a model of a part of the real world. Some authors regard the data stored in a data base as a "data model" of the real world, while others regard a "data model" as a computer independent, intellectual tool for representing and organizing

information about a part of the real world.

Thus, in order to describe what here is ment by "data model" and "data structure" a context must be specified. Here, the context is data base systems for administrative applications of an enterprise. (The term enterprise should be understood in a wide sense as in [DAT-75] , i.e. "as a convenient generic term for any reasonably large-scale commercial, scientific, technical or other operation").
Data base systems, however, can be regarded from different points of view.
Most often they are described and discussed from an implementation point of view. When this is the case, the focus is placed on different functions performed by the data base management systems.
Here, the data base systems are regarded primarily from an information system point of view.

An information system has been defined, by Langefors, as "a system of information sets needed for decision and signalling in a larger system (of which it is a subsystem) containing subsystems for
- collecting
- storing
- processing
- distribution

of information sets" [LAN-66].

In analogy to this definition of an information system, a data base system can be regarded as consisting of
- functions to accept/process input transactions
- a data base
- application programs
- functions to accept/process output reports and answers to queries

These components of a data base system can be described and analyzed from different points ov views. For example how they interact, the order in which they are designed, their relations to the data base management system, etc., etc.
Here, one specific aspect - a model aspect - i.e. how the components model the enterprise is focused on.

Now, in order to discuss the concepts "data model" and "data structure" within this context, a definition of the concept "model" would be appropriate. Such a definition, relevant to our context and purpose, is not easily found. A number of books from diciplines as philosophy, cybernetics and business administration where surveyed. The experience, however, was that the concept "model" was

discussed on a - for our purpose - too general or too application oriented level. (Examples are [BAD-71], [KLI-65], [RIV-72]).

Within computer science, the concept "model" is often discussed in the simulation field. For example, in Svoboda [SVO-76], a model is defined as: "A model is an abstraction containing only the significant variables and relations".
Models used at various stages in performance evaluation are, by Svoboda, classified into three general classes of models:

(1) Structural models
(2) Functional models
(3) Performance models

"A strucutral model describes individual system components and their connections.... A functional model describes how the system operates.... A performance model formulates the dependencies of performance on the system workload and the system structure" [SVO-76].

Zeigler, in [ZEI-76], is concerned with concepts as real systems, models and computers, where modeling is said to deal with the relationships between real systems and models, while simulation deals with relationships between computers and models. A real system is some part of the real world which is of interest. The real system may be natural, artificial, in existence or planned for the future. A real system is a source of behavioral data. A model is basically a set of instructions for generating behavioral data. Although, a model itself does not generate data one may speak of model generated data or model behavior. Validity of a model concerns the extent of agreement between real system data and model-generated data [ZEI-76].

The concept of "model" is also frequently discussed within the artificial intelligence area. The definition of "model" which we find most appropriate for our purpose is one from this field, namely the one used by Raphael in [RAP-68]. Raphael writes about the model concept:
"The term "model" has been grossly overworked, and it does not seem to have any generally agreed-upon definition. For purposes of this paper I present the following definition:
A model for an entity x has the following properties:

a) Certain features of the model correspond in some well-defined way to certain features of x.
b) Changes in the model represent, in some well-defined way, corresponding changes in x.
c) There is some distinct advantage to studying the model and effects of changes upon it in order to learn about

> x, rather than studying x directly. The term x may be any of wide class of entities, such as an object, a statement in English, or a mathematical concept". [RAP-68].

It is important to stress - as have been made by several authors - that a model is always a simplification and that the simplification is done for some specific purpose. This is expressed for example in [WEI-76]: "A model is always a simplification, a kind of idealization of what it is intended to model. The aim of a model is of course precisely not to reproduce reality in all this complexity. It is rather to capture in a viewed, often formal, way what is essential to understanding some aspect of its structure and behavior. The word "essential" as used in the above sentence is enormously significant, not to say problematical. It implies first of all purpose".

Referring to the definition of model given by Raphael and to the context of this study, the components of a data base system can be regarded as components of a model of an enterprise.

This model consists of data and programs. The data exist in input transactions, in the data base and in output reports and answers to queries.
The data in the input transactions, in the data base and in the output consist of data elements and structural relationships between data elements. The programs describe inferential relationships between data elements in input transactions, in the data base and in the output reports and query answers.

In this study, the data elements, the structural relationships between data elements and the inferential relationships between data elements is called a data structure.
A data structure can be seen as a model of an enterprise. In this model, the data elements and their relationships in the data structure, are representations of abstractions of phenomena in the enterprise.

In most of the data base management systems which are commercially available today, data structures appear at two abstraction levels.
The two levels can be called "type level" and "instance level" or "occurrence level". Synonymously the levels can be called "intentional level" and "extentional level".

The data structure at the type (or intentional) level can be regarded as a meta model of the data structure at the instance (or extentional) level.

The data structure at the instance (or extentional) level can be regarded as a model of phenomena in an enterprise. However, this model may be very complex. The instance level data structure may contain millions of data elements and relationships. The instance level data structure, its structure and behavior, may be so complex that abstraction and simplification - a model - is needed in order to describe, understand and manipulate it.

The data structure at the type (or intentional) level can be seen as a meta model, i.e. as a model of the model represented by the instance level data structure.
Typically, the elements of the data structure at the type level are generalizations of the elements in the instance level data structure. The elements of the type level data structure are fewer and less time varying than the elements of the instance level data structure.

Data structures at the type level as well as at the instance level are here seen as models of an enterprise. Both these models - independent of the abstraction level - satisfy the properties of a model as defined by Raphael [RAP-78].

The type level data structure, at the same time, constitutes a model and a meta model of an enterprise.
Some changes in an enterprise will affect only the instance level data structure while other changes in the enterprise will effect both instance and the type level data structures.
For example, assume that some elements in an instance level data structure, correspond to customers of an enterprise and that one element of a type level data structure correspond to a generalization of customers. If then, the enterprise, being modeled by the data structures, gets a new customer, this change will effect only the instance level data structure. If, however, a change occurrs so that customers of the enterprise may also at the same time be suppliers to the enterprise, and this has not been the case earlier, such a change may effect both the instance level and type level data structures.

Instance level data structures as well as type level data structures have to be expressed in terms of some modeling concepts.

An example of frequently used modeling concepts for an instance level data structure are record occurrences and set occurrences. Another example is tuples of relations. The corresponding modeling concepts for a type level data structure are record types and settypes or alternatively relations (intensions of relations).
These examples indicate that data structures at instance level as well as at type level could be expressed in terms of alternative sets of modeling concepts.
Such a set of modeling concepts will here be called a data model. Thus, a data model is not in itself a model of an enterprise. A change in the enterprise will not effect the modeling concepts of a data model.

A data model is a set of modeling concepts and a set of rules for relating the modeling concepts to form structures.
An often referenced data model is the one proposed by CODASYL-DBTG. Examples of modeling concepts in the CODASYL-DBTG data model are data item, data aggregate, record types and settypes. The ways in which these modeling concepts can be combined to form structures are implicitly defined in general rules and in the syntax of the SCHEMA-DDL as for example:
"- a data item is the smallest named unit of data
- a data aggregate is a named collection of data items and/or data aggregates,
- a record type is a named collection of data items and/or data aggregates,
- a settype is a named collection of recordtypes
- each settype must have one record type declared as its owner and one or more recordtype declared as its members,
- a recordtype may not be declared as both the owner and the member of the same settype
- etc., etc. " [COL-71].

The CODASYL-DBTG data model is intended for specification of a type level data structure. As an example, a type level data structure, which is a model of (a part of) a university department could be expressed as fig 3.1.

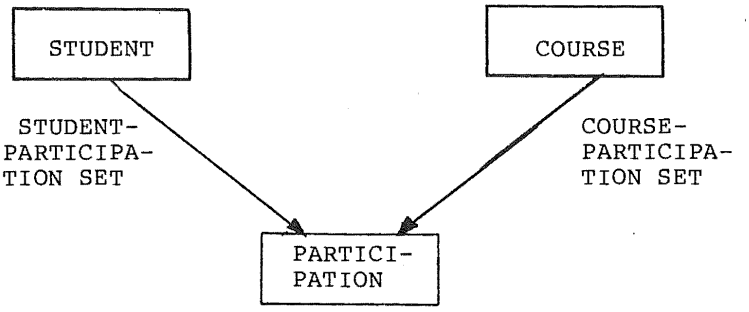
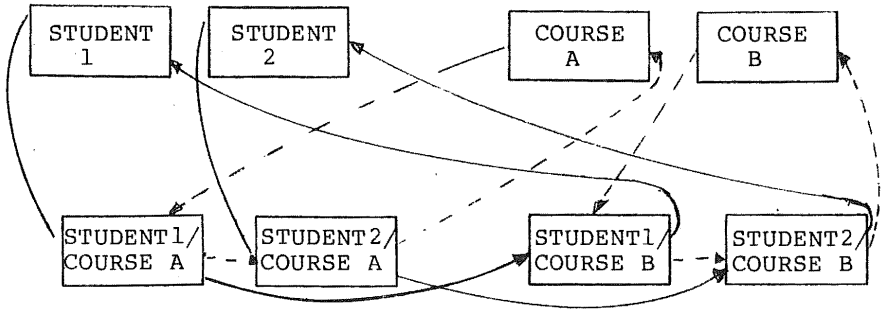


Fig 3.1.

The recordtypes (STUDENT, COURSE, PARTICIPATION) and the settypes (STUDENT-PARTICIPATION, COURSE-PARTICIPATION) are occurrences of modeling concepts defined in the data model. The record types and settypes are combined according to the structuring rules of the data model into a type level data structure. This type level data structure can be regarded as a model of (a part of) a university department (an enterprise) and at the same time as a meta model of an instance level data structure. In the instance level data structure - which also is a model of the same (part of) a university department - instances of the recordtypes and settypes representing individual students, courses and participations will exist. An example of an instance level data structure is shown in fig 3.2.



Instances of records and sets

Fig 3.2

The two levels of data structures - the type and instance levels - as examplified above, are common in most data base systems today. Exceptions do however exist. For

example, sometimes within the area of artificial intelligence, data base systems with only one level data structure exist. In these cases, the data structure contains elements corresponding to representations of individual real world phenomena as well as to representations of generalizations of real world phenomena, In these cases, general knowledge is mixed with specific knowledge in an one level model of the real world/enterprise.

So far, the type level data structure has been described as a meta-model of the instance level data structure. The elements of the type level data structure have been described as generalizations of elements in the instance level data structure.
It should, however, be pointed out, that the order in which the data structures at the type and the instance levels are designed is the opposite to generalization, i.e. the type level data structure is designed before the instance level data structure.

In summary, in this study, a data structure is regarded as consisting of a set of data elements, a set of structural relationships between the data elements and a set of inferential relationships between the data elements.
A data structure is seen as a model of some part of an enterprise.
Data structures can appear at different abstraction levels. In the commonly used data base management systems of today, data structures appear at a type level and at an instance level. A type level data structure is at the same time a model of some part of an enterprise, and a meta model of the instance level data structure.
Data structures have to be expressed in terms of some modeling concepts. A set of modeling concepts and rules for combination of these concepts is here called a data model. Referring to a definition of the concept model given by Raphael, a data model is not a model of an enterprise.
A data structure - independent of abstraction level - can be expressed by alternative data models.

3.1.1 Data models and data structures for different purposes within a data base management system

The ANSI/SPARC Interim Report [ANS-75] proposes an architecture for a data base management system. Such a data base management system is said to handle data structures at three levels. The three levels are called

internal-, external- and conceptual levels.
Very crudely, the data structure at the internal level describes how data elements in the data base are encoded, stored and accessed. The external level data structures describes data structures as seen by application programs. The external level data structures are adapted to the programming languages used for application programs and describe application relevant subsets of the data in the data base. The conceptual level data structure describes the information content and structure of the data base.

In section 3.1 two levels of data structures, a type level and an instance level were discussed. A type-level data structure could be regarded as a meta model of an instance level data model. The meta model was in itself a model of an enterprise.
When a type level data structure is regarded as a model of an instance level data structure, the type level data structure is an abstraction and generalization of the instance level data structure.
As has been pointed out, abstraction and generalization is always done for some specific purpose.
Models of an instance level data structure could be built for different purposes. For example, one model could be built for the purpose of describing how elements of the instance level data structure are encoded, stored and accessed in a data base. Another model could be built for the purpose of describing the information content and structure of a data base etc., etc.
In the frame of reference of this study, an ANSI/SPARC type of data base system is regarded as handling and using data structures at two levels, the type level and the instance level. In an ANSI/SPARC type of data base system however, there exist three different type level data structures used for three different purposes.
The here described view of the three levels within an ANSI/SPARC type data base management system is not common. The ANSI/SPARC terminology concerning "levels" is widely known and used, and therefore as a concession to commonly used terminology the ANSI/SPARC notion of "level" will be used in the rest of this study.
As the "levels" of an ANSI/SPARC type data base management system have different purposes it is here assumed that appropriate modeling concepts will be different for the three "levels".
At the internal level, the modeling concepts should be adapted to description of the way in which data are encoded, stored and accessed in the data base. At the external level, the modeling concepts have to be adapted to the various programming and query languages that are to be used in manipulation of the data base. At the conceptual level, the modeling concepts should be adapted to description of the semantics of the data stored in the data base.

As the modeling concepts are assumed to be different at the different "levels" it is practical to distinguish between internal data models, i.e. data models intended for an internal data structure, external data models, i.e. data models intended for external data structures and conceptual data models, i.e. data models intended for a conceptual data structure.

The terms information structure and information model are sometimes used as synonyms to conceptual data structure and conceptual data model.
In this study, a distinction is made between an information structure and a conceptual data structure.

In the meta model perspective, an information structure is seen as a meta model of data elements and relationships in a data base system.
A conceptual data structure is seen as a meta model of the data elements and relationships in a data base.

Thus, the elements of an information structure describe not only data elements and relationships in a data base, but also data elements and relationships in input transactions, output reports and in application programs.

A conceptual data structure corresponds to the data structure which in an ANSI/SPARC type of data base management system is described in a conceptual schema.

3.1.2 Schemas

The term schema will here be used to denote a description of a data structure in terms of some declarative language. As an example, the data structure in fig 3.1 could be described in a schema using CODASYL-DBTG SCHEMA DDL

```
SCHEMA NAME IS DEPARTMENT
RECORD NAME IS STUDENT
01  STUDENTNUMBER PIC 9(5)
01  NAME          PIC X(35)
.
.
.
RECORD NAME IS COURSE
01  COURSENAME   X(15)
.
.
.
RECORD NAME IS PARTICIPATION
01  SEMSETER     9(4)
.
.
.
SET NAME IS STUDENT-PARTICIPATION
OWNER IS STUDENT
MEMBER IS PARTICIPATION  MANDATORY, AUTOMATIC
.
.
.
SET NAME IS COURSE-PARTICIPATION     1)
.
.
```

Analogous to the distinction between internal-, external- and conceptual data models and data structures there exists an internal schema, a set of external schemas and a conceptual schema in an ANSI/SPARC type data base management system.
The conceptual schema is a distinguished property of an ANSI/SPARC type data base management system. Independent of the data base management systems used to implement an information system, the designers will need and use an implicit or explicit conceptual data structure. When a data base management system of ANSI/SPARC type is used, the conceptual data structure is not only made explicit but also made known to and used by the data base management system in its operation.

"The Study Group contends that in the data base area it must be made explicit and, in fact be known to the data base management system. The proposed mechanism for doing so is the conceptual schema. The other two views of data, internal and external, must necessarily be

---

1) Only a few clauses of the SCHEMA definition is shown in this example.

consistent with the view expressed by the conceptual schema". [ANS-75].

3.2 Data base design

By data base design is here meant design of information-, conceptual-, external and internal structures.
As different types of design decisions are made in the design of the different structures, it is practical to distinguish between conceptual level data base design, external level data base design and internal level data base design.

In this study, conceptual level data base design is regarded as consisting of two parts, the information structure design and the conceptual data structure design.
The information structure describes the information contents of a data base system. Typical design decisions within information structure design concern abstraction and classification of real world phenomena. Specification and analysis of statements about the real world phenomena is an important task within the information structure design.
The conceptual data structure describes the content of a data base. In order to design the conceptual data structure, the data base system must have been crudely specified. In this specification, data and processes within the data base system must have been determined. Once the data and processes of the data base system have been determined the structure of the data base can be designed in an appropriate way.

External level data base design concerns design of data structures which are consistent to the conceptual data structure and which meet the information requirements of different applications. The elements of the external data structures must correspond to data types which can be handled by the programming and/or query languages which are to be used. Definition of the mappings between the conceptual and the external data structures is regarded as a part of the external level data base design.
Internal level data base design concerns decisions on what elements of the conceptual data structure to explicitly store and what elements to compute/derive when requested. Storage principles, access methods and physical allocation of stored data are all important issues within this design area. Definition of the

mapping between the conceptual and the internal data structures is regarded as a part of the internal level data base design.

3.2.1 "Logical data base design"

The term "logical data base design" has been and is still used to denote any one or all of the activities that take place in a process which starts with the identification of an information problem within an enterprise and ends with a data base management dependent data structure, for example a SCHEMA expressed by CODASYL SCHEMA DDL.
With the separation of a "physical" and a "logical" level of data within a data base management system as introduced by the CODASYL Proposals [COL-69], [COL-71] the process of designing a data base could also be split up into "logical" and "physical" data base design.
According to the CODASYL Proposal, both the "logical" and the "physical" design is the responsibility of the Data Base Administrator. The emphasis of the logical design is on how to represent the information to be contained in the data base system in terms of the modeling concepts of the CODASYL data model. The Data Base Administrator should: "emply a data structure(s) that models the business or problem provided in the SCHEMA DDL in terms of areas, records, sets, data items and data aggregates" [COL-71].
The "logical" and "physical" levels do to some extent correspond to the two levels of models, the type and the instance level data structures as described in section 3.2. However, in most "two level" data base management systems as for example in the CODASYL-DBTG systems, the modeling concepts of the data model used to express the "logical" data structure, are adapted to fulfillment of several purposes of the data structure. For example, it describes how the elements at the enterprise model level are encoded, stored and accessed in the data base at the same time as it attempts to model the "semantics" of the enterprise model.
In an ANSI/SPARC type data base management system different data models are used to express the different type level data structures which are built for different purposes.

In this study, the expression "logical data base design" will not be used (except in references). The ANSI/SPARC terminology, i.e. conceptual, external- and internal data base design will be used instead.

3.3 The scope of conceptual level data base design

Terms as "conceptual level" and "conceptual structure" have been used within the computer area [SCH-73] before it was made an everyday expression within the data base area by the ANSI/SPARC Interim Report. It has also a correspondence in the "infological level" as proposed by Langefors and Sundgren [SUN-73].
The ANSI/SPARC Report suggested the conceptual level as a part of the data base management system, and describes it by a number of purposes:

"- It should provide a description of the information of interest to the enterprise.
- It should provide a stable platform to which internal schemas and external schemas may be bound.
- It should permit additional external schemas to be defined or existing ones to be modified or augmented, without impact on the internal level. It should allow modifications to the internal level to be invisable at the external level.
- It should provide a mechanism of control over the content and use of the data base."
[ANS-75], [ANS-77].

These different purposes indicate that there are several problems and research areas related to the conceptual level of a data base system.
For example, one research area is related to the conceptual level as a means to achieve data independence in a data base management system. This area concerns, for example, strategies for binding external data required by an application program to its correspoding internal data. Concepts and languages for definition of mappings between external-conceptual and conceptual-internal data structures is another important issue.

In this study, the purpose of the conceptual level data structure, to constitute a description of the information of interest to the enterprise is focused.
Within this somewhat limited perspective of the conceptual level of a data base system, there is still a number of different problems and research areas, and there still does not exist any commonly accepted or used terminology, concepts, models or methods.
An increasing interest in the design aspect can be noticed. Some years ago, several data models intended for the conceptual level of a data base system were presented. However, hardly any of these presentations discussed how to arrive at a data structure for an application system, in terms of the data model proposed. In the latest years, however, papers discussing design

aspects of the conceptual level have appeared. Most of the approaches analyzed in chapter 4 are such recently presented design methods.

In the ANSI/SPARC Interim Report, 1975, design of a conceptual schema was relatively summarily described as the task of the enterprise administrator to decide what entities, properties and relationships in the enterprise to represent in the data base.
In a later version [ANS-77] the task of designing a conceptual schema has approached the area of information system design.
"Prior to undertaking, the development and creation of a data base, it is necessary to have an understanding of the environment which that data base is to serve. Consequently, a very important step is the characterization or synthesis of the information needs within an enterprise. This includes determining the information flows through the enterprise and how it is used. This systems synthesizing function, in the context of the many applications utilizing the data base, is one function of the enterprise administrator. The enterprise administrator serves as the focal point for identification of information use in an enterprise", [ANS-77].

The ANSI/SPARC description of conceptual level data base design has the advantage of being independent of any data model used to express a conceptual data structure or any specific method for design of conceptual data structures. However, a somewhat more detailed specification of tasks or problems included in the conceptual level design is required in this study.

Several authors within the area implicitly define conceptual level data base design by defining the input required by their method and/or by describing the result expressed in terms of their data model. However, neither the starting point nor the result of conceptual level design is expressed in a method/model independent way.

As a part of the frame of reference of this study, the scope of "conceptual level data base design" is here described by five problem areas.
This description in terms of problem areas is an attempt to describe the scope in a general, i.e. method/model independent way.
The five problem areas are:
1. Concepts/models for description of users' views.

2. Concepts/models for description of users information requirements.
3. Methods for design and analysis of information structures.
4. Concepts/models for description of conceptual data structures.
5. Methods for design and analysis of conceptual data structures.

The different methods discussed in chapter 4 are concerned with any or all of these five problem areas. However, approaches concerned primarily with the first three problem areas will be focused. In sections 3.3.1 to 3.3.5 the problem areas will be summarily described. As the problem areas are neither obvious nor commonly recognized they will be described more thoroughly in the sections 3.4 to 3.5.

3.3.1 Concepts/models for description of users' views

The term "user" should be understood in a broad sense, and to mean any person involved in design and/or use of a data base system.
In integrated data base systems, the data are to be shared among several different users and to be used for different purposes.
In this study, the data in a data base system are regarded as representations of statements about real world phenomena. In order to achieve semantical integrity in a data base system it is necessary that the different users have common interpretations of the statements in the system.
The statements refer to abstractions, classifications and assumptions about real word phenomena. A persons' - or a group of persons' - abstractions, classifications and assumptions about real world phenomena is here called a user view.
In order to achieve a common interpretation of a statement, the statement must refer to a common view. Therefore, it is here assumed to be essential to conceptual level data base design that users' views are made explicit, and that they are described as precisely as possible.
Precise descriptions of users' views require concepts or models in terms of which the views can be expressed.
Thus, concepts/models for description of user views, i.e. of abstractions, classifications and assumptions about real world phenomena constitute an important problem area within conceptual level data base design.
Concepts and models for formal description of user's views are strongly related to models used in psychology,

linguistics and knowledge representation in artificial intelligence.
Terms as "views", "local views", "user views" etc. are frequently used within the conceptual level data base design area. Although the signification of these terms is not very clear, they often seem to differ from what here is meant by "user's views".
"Views", "local views", "user views" are sometimes used to denote data structures required by an application program, sometimes used to denote a specification of information required by some user(s). Data structures required by an application program are in this study called external data structures. Information required by some user(s) is called "user's information requirement".

3.3.2 Concepts/models for description of user's information structures

Specificaton of information requirements is necessary in order to design purposeful and efficient data base systems. Information requirements correspond to input to and output from the data base system being designed.
Knowledge about information requirements is necessary in design of data structures at all levels in a data base management system. For example, information requirements constitute the base for design of external data structures and they are necessary in order to design an efficient internal data structure.
In the conceptual level data base design, information requirements are here seen as specifications of types of statements that users wish to obtain from, or that they intend to provide to, a data base system.
Information requirements and users views are here seen as the input to and the base for information structure design and thus as the input to and the base for conceptual level data base design.
As a distinction is made here between users views and information requirements (this distinction will futher be discussed in 3.4) a distinction between concepts and models for formal description of user's views and of users' information requirements is also made. Thus, concepts and models for formal description of users' information requirements is considered as an important problem area within the conceptual level design. It may be that the same set of concepts is suitable for both these types of formal descriptions. However, as long as this is not clear we wish to separate the two types of formal descriptions.
Users' information requirements are here regarded as an interface between information systems design and data base design. Methods for determination of users' information requirements are not considered as a part of

the data base design but as an essential part of information systems design. What information a person or a group of persons within an enterprise needs and/or wants in order to fulfill his/hers tasks in line with personal and enterprise goals is not considered a data base design problem, but an information system design problem. We are aware of the importance of good methods for identification of information requirements and of the disastrous consequences of designing a data base system on bad, unsuitable or irrelevant information requirements. However, we have here chosen to define conceptual level data base design not to include methods for determination of users' information requirements but to include concepts/models for formal description of - elsewhere determined - users' information requirements.

3.3.3 Methods for design and analysis of information structures

By an information structure we mean a specification of all types of statements that are to be contained in a data base system. The information structure also includes description of inferential relationships between statements. The types of statements specified in an information requirement may imply or may be inferred from statements in other information requirements. Such relationships between statements have to be analyzed and described in the information structure.

Information requirements can not be formally described without references to concepts defined in users' views. On the other hand, it may be discussed whether or not it is practical to specify users' views without knowledge about users' information requirements. Users' views and users' information requirements are mutually dependent. Independent of the order in which users' views and information requirements are specified it is, however, necessary to explicitly define the correspondance between the views and the requirements.
In the analysis of implication and inference relationships between types of statements, it is necessary to analyze relationships between the elements in user views, referenced by the statements. For example user views may or may not concern overlapping parts of the real world, or in other words, they may partly concern the same real world phenomena. When users' views do concern the same real world phenomena they may still be different as they may correspond to different abstractions, generalizations and/or classifications of real world phenomena and they may be based on different

assumptions about the phenomena.
In summary, design and analysis of information structures includes:

- For each type of statement, specification of the, by each type of statement referenced, elements and relationships between elements in user views.
- Integration of user views. The integration of user views implies that users views must be consistent i.e. the abstractions, generalizations and assumptions about real world phenomena in different user views must be consistent.
- Analysis and specification of inferential relationships between statements. This analysis must verify that statements required as output from the data base system correspond to or can be inferred from statements specified as input to the system.

The resulting information structure is seen as specification of types of statements to be contained in a data base system, and of inferential relationships between these types of statements.
For each type of statement, its referenced generalizations, classifications and assumptions of real world phenomena are explicitly described. The generalizations, classifications and assumptions of real world phenomena which are referenced by the different types of statements are - in order to achieve semantical integrity - verified to be consistent.

The information structure is here regarded as the base for design of the conceptual data structure of a data base.

3.3.4 Concepts/models for description of conceptual data structures

An ANSI/SPARC type data base system will have one conceptual data structure declared in the conceptual schema. The conceptual schema is used by the data base management system in its operations.
The data model used to express the conceptual data structure need not be the same as the one(s) used to express user's views and/or information requirements. The data model must, however, support the purposes of the conceptual schema to constitute an interface between the external and internal data structures of the data base system. This indicates that the data model used for the conceptual schema must be suitable for the definitions of external-conceptual and conceptual-internal mappings.

What constitutes a set of good and relevant modeling concepts for the conceptual data structure has been and is still a much debated issue within the data base area. Several authors propose the Relational Data Model as a candidate for the conceptual data structure. Also, the CODASYL data modeling concepts have been proposed as a candidate. Some authors (for example [KEN-76]) stress the need of new concepts for the conceptual level as they find the so far proposed ones too much adapted to the way in which computers work, and they prefer the modeling concepts to be adapted to the way in which human beings think and work.
In any case, concepts and models for the conceptual data structure is obviously an important research area within the conceptual level data base design.

3.3.5 Methods for design and analysis of conceptual data structures

Within this study, an information structure is seen as a specification of the types of statements that are to be contained in a data base system, and of inference relationships between statements.
The design of the information structure is regarded as a part of "conceptual level" data base design.
An information structure is independent of any specific data base management system.
A conceptual data structure is here regarded as a data base management system dependent data structure. It is dependent in the sense that it has to be expressed in terms of, and be consistent with the constraints of the data model provided by the DBMS for the conceptual level. The conceptual data structure is to be declared by a Data Definition Language in order to be used as the conceptual schema for a data base.

The transition of an information structure into a conceptual data structure involves several types of design decisions.
One important type of decision is the decision on which of the types of statements specified in the information structure, to include in the conceptual data structure and which types of statements to derive from and support to the data base. This implies the design of a "logical" file and process structure for the data base system.

The types of statements defined in the information structure may refer to time in an unrestricted way. In order to design the conceptual data structure, however,

it is necessary to give exact specifications of the time points or intervals to which statements of the different types may refer.

In the information structure, types of statements may be described by references to types or classes of entities. At the information structure level, decisions on how to refer to individual elements in the classes of entities/values may not have been made. This however is necessary in the conceptual data structure, and corresponds to decisions on how to represent entities/values.

Depending on the data model used to express the conceptual data structure types of statements may have to be grouped together. For example, if the Relational Data Model is used, it has to be decided which types of statements to group and represent by which relations.

Also, depending on the data model used for the conceptual data structure, it will be required to conform to certain conditions. For example, a conceptual data structure expressed in terms of the Relational Data Model may be required to contain only 3NF relations. A conceptual data structure expressed in terms of the CODASYL Data Model is requested not to allow settypes corresponding to M:M relations or to use settypes to relate records of the same recordtypes etc.
The analysis of the conceptual data structure aims to assure that restrictions on the data structure, imposed by the data model, do hold. The conceptual data structure may have to be reorganized as a reslut of this analysis.

In summary, the scope of "conceptual level" data base design, as defined in this study, has been described by five problem areas. The five problem areas have been summarily introduced.
In order to describe the content of "conceptual level" data base design, the five problem areas will be discussed in more detail in the following sections (3.4 and 3.5). In this discussion, the first three problem areas:

- Concepts and models for description of users views
- Concepts and models for description of users' information requirements
- Methods for design and analysis of information structures

are treated together as information structure design.

The following two problem areas:
- Concepts and models for description of conceptual data structures
- Methods for design and analysis of conceptual data structures

are treated as conceptual data structure design.

3.4 Information structure design

In this study, an information structure is regarded as a specification of the types of statements that are to be included in a data base system. A conceptual data structure describes the types of statements included in a data base.
"Conceptual level" data base design is here regarded as including information structure design as well as conceptual data structure design.

Within information structure design, a distinction is here made between users' views and users' information requirements. Most authors within the area are concerned with either views or information requirements. The distinction between views and requirements is not commonly made and is not obvious and therefore requires some elaboration.
In order to describe the distinction between users' views and users' information requirements typical concepts used to express users' views and information requirements are presented. Thereafter the information structure design is discussed.

3.4.1 Users' views

A user view is seen as a person's or a group of persons' abstractions, classifications and assumptions about real world phenomena.
Most authors within the area, explicitly or implicitly assume some basic modeling concepts in terms of which perceptions of real world pheneomena are expressed. The basic modeling concept in most approaches is the concept of "object" (or synonymously "entity") An "object" is an undefined concept often described as something about which information is required.
Some authors percieve real world phenomena in terms of "objects and associations" between objects.
A representation of "objects" and an "association" could be illustrated as in fig 3.3.

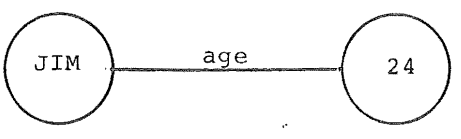


Fig. 3.3

As with the concept of "object", the concept of "association" is not defined. Sometimes the distinction between an "object" and an "association" is described in terms of existence. Associations are said not to exist unless the associated objects exist while objects may exist independent of its associations.
Associations may be directed (fig 3.4).

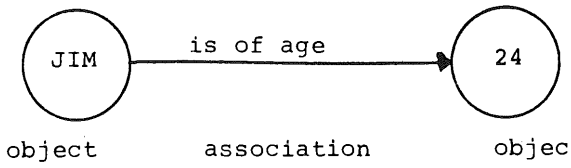


Fig. 3.4

When associations are directed, alternativ views of the same real world phenomena may be expressed by the same modeling concepts (fig. 3.5).

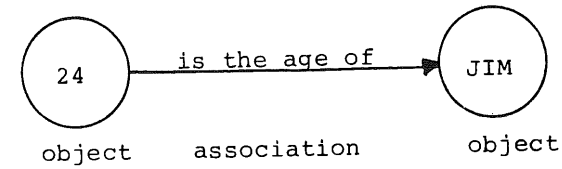


Fig. 3.5

Some authors restrict associations to be binary, while others use n-ary associations.
In some cases, a destinction between "object" and "value" is introduced. In these case an association between an object and a value is a directed assocaition which is called "attribute" or "property" (fig 3.6).

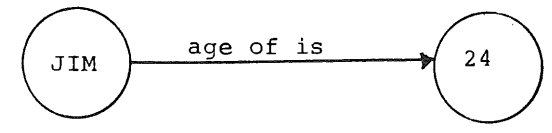


Fig. 3.6

"Object", "value", "association" and "attribute" are often the only concepts used to express perceptions of real world phenomena.
There do however exist "data models" with more modeling concepts. Semantical networks are examples of such "data models". For example, in the "Semantical network model" [ROU-75] real world phenomena are classified as being objects, values, characteristics, events or associations of different types as "agent", "topic", "instrument", "result", "source" etc.

Now, individual real world phenomena are most often not described in users' views. Only types or sets or classes of phenomena are described. (The word "set" and "class" will here be used as synonymes. For a discussion of their meaning see for exampel [KOT-66]. Thus, frequently used modeling concepts for expressing users' views are classes of objects and/or classes of values.

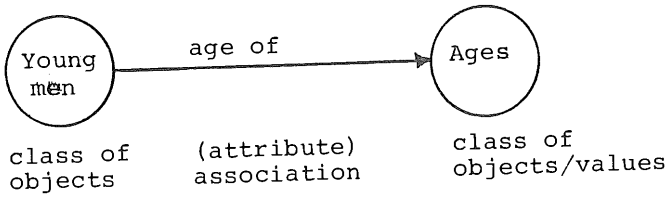


Fig. 3.7

In some approaches, classes of objects and classes of values are called domains. A domain - just as a class of objects or values - may appear in several associations. The different associations in which a domain appears are sometimes called roles (fig 3.8).

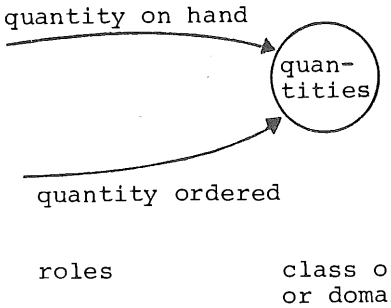


Fig. 3.8

In some modeling approaches the distinction between individual phenomena and classes of phenomena is not made. Instead, a distinction is made between types and tokens, i.e. between types of phenomena and individual phenomena. In this case, individual phenomena are not classified into classes of objects but generalized into (single) generic objects. Generic objects, as well as individual objects, may in turn be generalized to "higher level" generic objects so that hierarchies of generic objects are formed (see fig. 3.9). Users' views may in this case be expressed in terms of objects, generic objects and attributes.

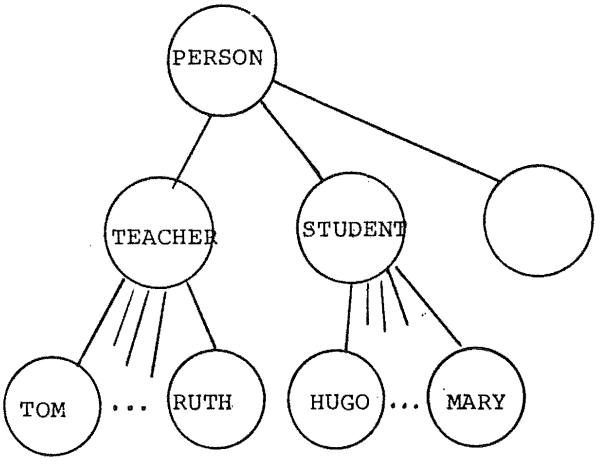


Fig. 3.9 Hierarchy of objects and generic objects.

By some authors (as for example by Senko [SEN-71B$], a distinction is made between phenomena and names of phenomena and thus between classes of abstract phenomena and classes of names of phenomena.

In most "graph theoretical" data models (see [KER-76]) individual phenomena as well as classes or types of phenomena are (as for example in fig 3.4 and 3.9) represented by names. In some approaches where user views are expressed in terms of classes or types of phenomena, the decision on, by which names, to represent individual phenomena is postponed until after the information structure design.

Modeling concepts as objects, values, associations, classes or types of objects and values are here regarded as modeling concepts for expressing users' views, i.e. a person's or a group of persons' abstractions and classifications of real world phenomena.
Some assumptions about the real world phenomena are implicitly described in the classification of real world phenomena into objects, values and associations. Further assumptions may be expressed as properties of classes of objects,as for example the number of members of a class and membership conditions for a class. Properties of associations may be described by specification of types of mappings (1:M, M:1, M:M, 1:1) and/or whether or not mappings correspond to total, partial, bijective, surjective, injective functions etc.
As an example, the association "age of is" between object classes "Young men" and "Ages" (fig. 3.10) may be described as a total injective function.

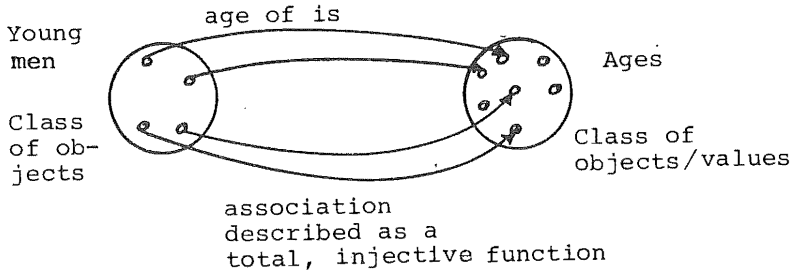


Fig. 3.10

In this study, concepts and models for explicit specification of users' views is suggested as one of five problem areas within conceptual level data base design. By this, the importance of explicit formulation of users' views is here stressed. The data in a data base system is regarded as representations of statements about real

world phenomena. In order to achieve a, to all users, common interpretation of these statements it is here regarded as necessary to describe, as precisely as possible the real world phenomena being referenced.
User views can be seen as specifications of a semantical aspect of statements in a data base system.

A user view is here regarded as a description of abstractions, classifications and assumptions about real world phenomena which are of interest to a user (or to a group of users). The real world phenomena and the abstractions and classifications which are of interest to a user are those, about which the user wants to make or obtain statements.
The same real world phenomena may be of interest to several users. However, different or partly different abstractions and classifications of these phenomena may be of interest to different users. Therefore, it is important that abstractions and classifications of real world phenomena are expressed in a way which makes it possible to determine, whether or not, consistent views of real world phenomena are held by different users.
A graphical notation may not be precise enough to describe a user view. As an example a user view described as in fig. 3.11 may be interpreted in at least three different ways.

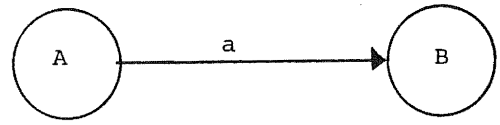


Fig 3.11

In one interpretation A and B may be regarded as names of classes of objects. In this case the association "a" may denote a mapping of elements in "A" into elements in "B", as illustrated in fig 3.12.

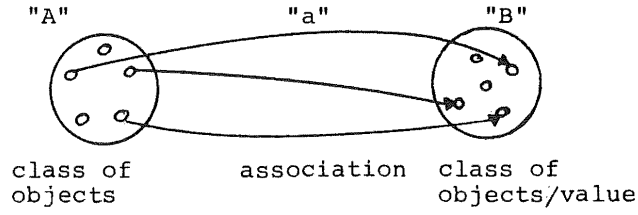


Fig. 3.12

A second interpretation of a user view as described in fig 3.11 may be to interpret "A" and "B" as names of classes of objects and to interpret "a" as an association between a class ("A") and elements of another class ("B"), as illustrated in fig 3.13. In this case the class "A" is regarded as an object, i.e. the class is the object.

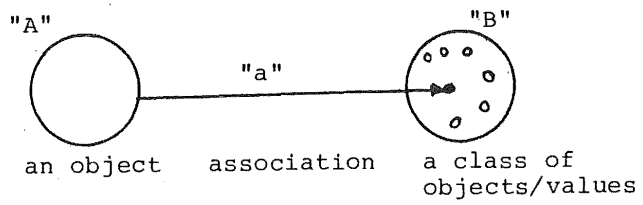


Fig. 3.13

In a third interpretation of the user view in fig. 3.12, "A" and "B" may be interpreted as names of (single) generic objects and the association "a" as a relationship between these two generic objects as illustrated in fig 3.14.

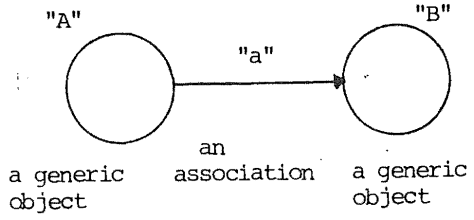


Fig. 3.14

Then three different interpretations of the user view in fig. 3.12 are shown to illustrate the importance of explicit specifications of the assumptions on which abstractions, classifications and associations are based. Quantitative information, as for example the number of elements in classes and specification of mappings, are by some authors regarded as necessary only in the design of computer efficient data structures. Here, quantitative information is seen as a part of the assumptions about real world phenomena which are of importance in descriptions of users' views. The quantitative information is important in determination of whether or not users' views are consistent.

The distinction between classes of objects, classes seen as objects and generic objects is not frequently made. Graphical representation of user's views may contain a mixture of these as for example in fig. 3.15.

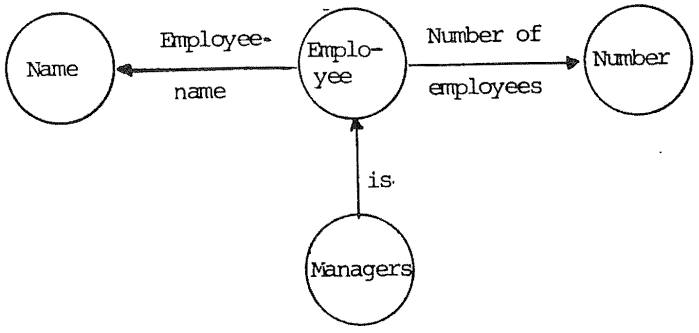


Fig. 3.15

One indication of a mixture of classes of objects, classes seen as objects and generic objects is a mixture of plural and singular forms for the names of the nodes in the graph. However, the distinction between classes of objects, classes seen as objects and generic objects, would be clear if assumptions as quantitative properties were specified.
The user view in fig 3.15 could be specified as in fig. 3.16.

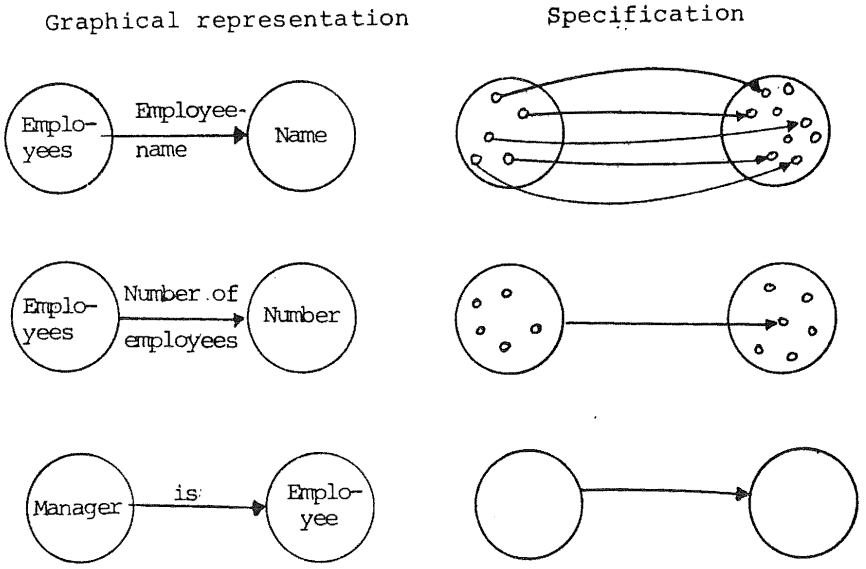


Fig. 3.16

Precise descriptions of users views makes it possible to identify inconsistencies in different users' views.
An example of inconsistency in abstraction could be if "Employee" in one user view is assumed to denote only full time employees while "Employee" in another view denotes full time employees as well as part time employees. Classificatios of phenomena may be "overlapping". For example, the people employed by an enterprise may, by some user be classified as "mangagers", "clerks" and "workers". Another user may classify the same employees as female employees" and and "male employees". Still another user may classify the employees as "union members" and "non-union members".
Overlapping classifications may, if not identified lead to inconcistency.
Assumptions about associations may be inconsistant. For example, one user may assume that an employee is associated to one and only one address while another user may assume that an employee can be associated to several addresses.
User views must be analyzed and eventually reformulated in order to achieve consistency. Consistent, overlapping users' views can be integrated into a, to all users, common "global" view of real world phenomena of interest. This "global" view can be regarded as a semantical frame of reference for a data base system. The data base system will contain statements which refer to the elements in the "global" view.

3.4.2 Information requirements

The concept "information requirements" seems to originate from the very early approaches to information systems design (see chapter 2).
In these early approaches, the information to be contained in the input files, in stored files and in output reports of an implemented information system, was specified in terms of information sets.

Comparatively few authors within the data base area are concerned with or discuss "information requirements", and very few authors make a destinction between users' views and users information requirements.

In this study, an information requirement is seen as a specification of types of statements which a user wants to support to or obtain from a data base system.

Types of statements refer to types or classes of real world pheneomena.
Information requirements can not be formulated without at least an implicit view of the referenced real world phenomena. As has been stressed in section 3.4.1, it is, in this study, considered to be important that the referenced view of the real world phenomena is explicit and precisely described.

The distinction between statements and phenomena being referenced by statements which is made in this study, is influenced by work on "messages" by Langefors.
The "message" concepts is an example of a concept for description of information requirements.

Langefors defines information sets as sets of messages. Messages can be "consolidated" or "elementary". An "elementary message" or "e-message" is regarded as the smallest meaningful unit of information.
An "e-message" corresponds to information about an observation of a state of a thing in a system (enterprise). An "e-message" consists of references to the thing (a phenomena) being observed, to the aspect (of this thing) being observed (a variable), to the time at which the observation was made (a time point) and to the observation made (a value). For example, "the quantity on hand of an article" may be described by an "e-message" containing the references : "Sisters & Sisters Ltd", "bicycle", "15:th of October 1978", "quantity on hand" and "694".

The references are assumed to uniquely identify the things being referenced and an "e-message" is defined as references to
- A system
- A time point
- A system point
- A variable
- A value

The definition of an "e-message" does not imply any restrictions on the references and their representation as data. For example, referring to the example above, if "bicycle" is not a unique reference to the thing being observed the "e-message" could contain a system point reference as "Sweden, Stockholm, Southern warehouse, Storage Place 559, Bicycles".
The difference between an e-message and a "consolidated message" is the number of different aspects - variables - being observed and thus "consolidated messages" can always be decomposed into "e-messages".
In a specification of the information content in an information system types of messages are described. A

message type describes e-messages which have references to the same system, the same type of system point and the same type of variable(s).
The message concept,as described above, is one example of concepts for description of statements and thus for description of information requirements.

The relationship between users' views and information requirements as seen in this study, can be illustrated as in fig 3.17.

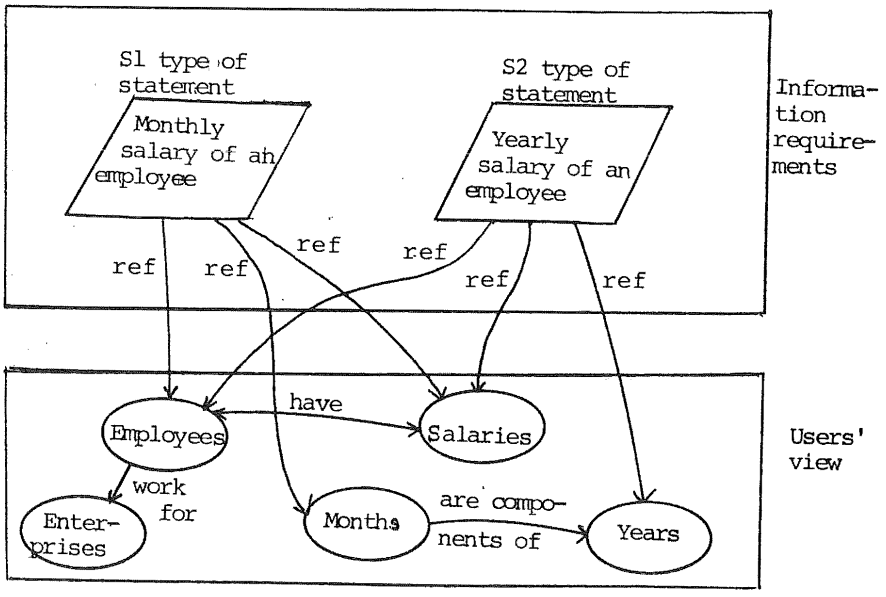


Fig 3.17

The types of statements of information requirements refer to elements in users views. 1)

---

1) In fig 3.17 the users' view is expressed in terms of classes of objects and associations between classes of objects. As has been stressed (in section 3.4.1) a user view should also describe assumptions as for example quantitative information about classes of objects and about associations.
This is not shown in fig. 3.17.

3.4.3 Information structure

In this study, users' views and information requirements are regarded as a base for information structure design. Users' views correspond to users' perceptions and assumptions about real world phenomena. Information requirements correspond to statements about real world phenomena.
Concepts and models for description of users' views and information requirements have here been pointed out as important problem areas within "conceptual level" data base design.
Information requirements and users views are mutually dependent. Abstraction and classification of phenomena is always done for the purpose of expressing some information about the abstractions and/or classes, i.e. in this context, to make statements. Information requirements can not be formulated without at least an implicit view of the phenomena being referenced.
Thus, information requirements and users views must be specified in an iterative process.
In the information structure design, information requirements, i.e. statements, are analyzed.
This analysis aims at identification and description of inferential relationships between (types of) statements. For example, the statement "Ruth's average monthly salary the first quarter of 1978 is $2500" may be inferred from the statements "Ruth's monthly salary for January 1978 is $2000", "Ruth's monthly salary for February 1978 is $3000" and "Ruth's monthly salary for March 1978 is $2500".
Knowledge about inference relationships is necessary in order to determine that statements required as output can be obtained from statements specified as input.
In order to determine inference relationships between (types of) statements the user views, referenced by the statements, must be precisely described. Users' views may describe implications which in turn may be used in identification of inference relationships between (types of) statements. For example in an user view (as in fig 3.18) where "Person" as well as "Teacher" correspond to generic objects, the fact that a teacher has a social security numer may be implied.

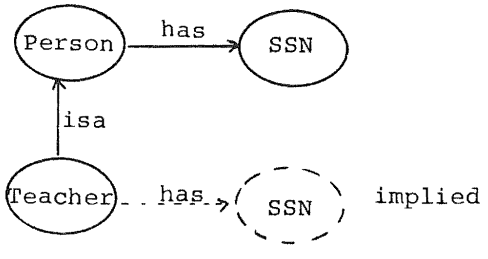


Fig. 3.18

The implied fact can be used to identify an inferential relationship between a statement about a teachers social security number and a statement about the social security number of a person.
As another example, in a user view where "employees", "departments" and "managers" denote classes of phenomena which are related as in fig 3.19 an implicit association exist.

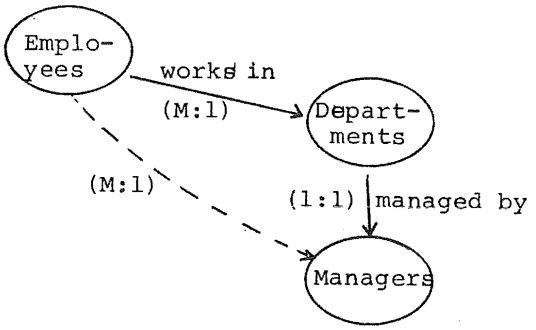


Fig. 3.19

If we assume that the manager of the department is also the manager of the employee, a statement about the manager of an employee may in this case be inferred from statements about the department in which the employee works and a statement about the manager of this department.

Analysis of inference relationships may be an extensive task in a practical application where the number of (types of) statements may be large. A systematic approach to identification of inference relationships is an essential part of the information structure design.

An information structure is in this study seen as a specification of (types of) statements to be contained in a data base system.
This specification of types of statements include description of a consistent, global user view of real world phenomena as well as descriptions of inference relationships between (types of) statements.
The information structure is seen as the base for conceptual data structure design.

3.5 Conceptual data structure design

The conceptual data structure describes the (types of) statements contained in a data base.
In the design of a conceptual data structure decisions on how to organize and represent statements must be made.
However, before the conceptual data structure can be designed, a design of the data base system must have been made.
The crude design of the data base is here regarded as a part of the conceptual data structure design.
In the design of the data base system, decisions on which data to be contained in which "files" and which processes that are to operate on which "files" in order to meet the information requirements, must be made.
The design of the data base system (or data processing system) can be seen as a transition from the information structure into a conceptual data and process structure.
An extremely simplified example may illustrate this transition.
An information structure consisting of two types of statements S1 and S2 and an inference rule I have been specified (see fig. 3.20).

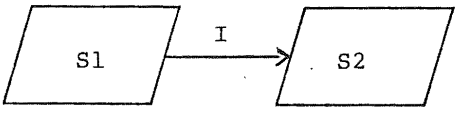


Fig. 3.20 1)

The S1 type of statement is specified as initial and denotes "Monthly salary of an employee". The S2 type of statement is required as output and denotes "Yearly salary of an employee". The inference relationship I can be described as:

$$S2(EMP,SAL,YEAR) = \sum_{MONTH\ K=1}^{MONTH\ K=12} S1(EMP,SAL,MONTH_K)$$

Even in this extremely simplified case we can design alternative data base systems. Fig 3.21 illustrates three such feasable alternatives. Notice that the alternatives lead to different content (in terms of type of statements) of the data base.

1) The user view referenced by the S1 and S2 types of statements is illustrated in fig 3.17.

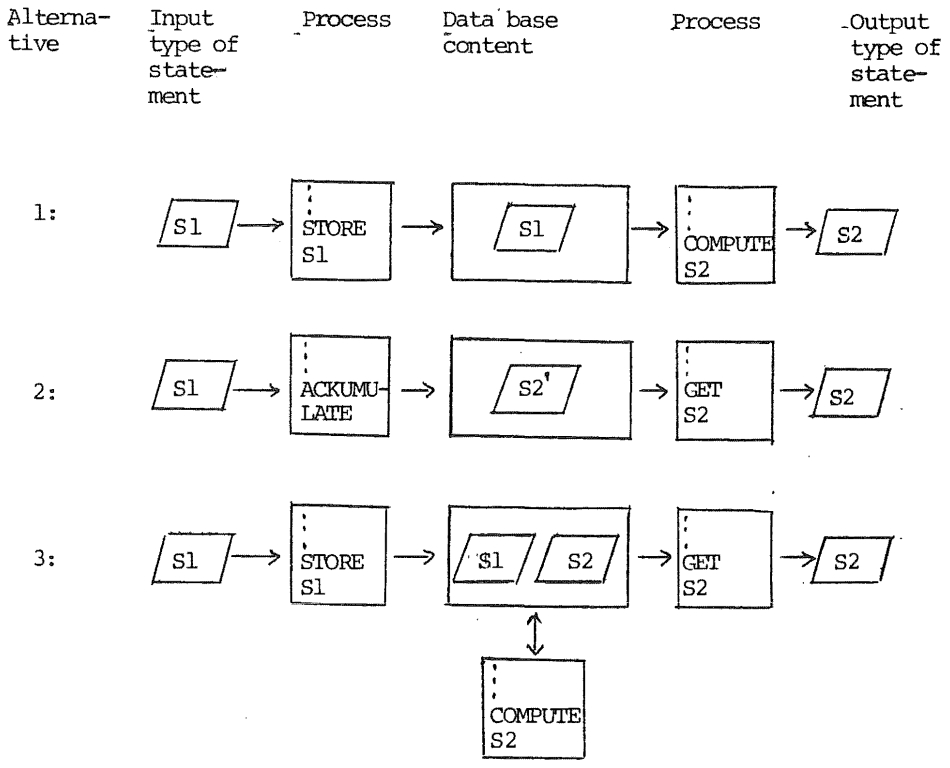


Fig 3.21 Alternative data base systems

In the design of the data base system decisions of types of statements and on restriction of the instances of the types of statements to be contained in input transactions in the data base and in output reports have to be made. In fig 3.21 only types of statements in the "files" of the "alternative data base (data processing) systems are shown. However, specifications of possible instances of the different types of statements are also needed.

Generally, the possible instances of a type of statement correspond to subsets of the Cartesian product of the classes of objects/values referenced by the types of statements.
For example, if the S1 and S2 types of statements in fig 3.20 are described as in fig 3.17 the possible instances of S1 and S2 correspond to subsets of the Cartesian products "Employees"x"Salaries"x"Months" and "Employees"x"Salaries"x"Years".

The specification of the global user view determines the possible instances of the different types of statements.

However, in the specification of a global user view the classes of objects/values may have been described in a general way. For example, the class of objects/values named "Salaries" in fig 3.17 may have been described as real number in the interval 1000-100.000. In order to determine relevant instances of each type of statement, restrictions on the classes of objects/values relevant for each type of statement can be made. S1, for example, may be restricted to refer only to "Salaries" corresponding to real numbers in the interval 1000-10.000 while S2 may be restricted to "Salaries" in the interval 10.000-100.000.
As another example, the "Months" referenced by S1, may when it appears in an input "file" be restricted to "Current month". "Years" being referenced by S2, when it appears in output files, may be restricted to "the year before current year". Restrictions on possible instances of types of statements in input and output "files" will determine the necessary instances of statements in the data base.

Thus, the design of the data base system is here regarded as determination of "files" and processes of the system. The "files" are specified in terms of types of statements and restrictions on instances of the types of statements. The specification of "files", in terms of statements, is independent of the "data model" used by the data base management system.

Also independent of any "data model" are decisions concerning representation of statements in the data base (system). Decisions on representations have to be done before the conceptual data structure can be designed and described. As an example, statements of the types S1 and S2 in fig 3.17 refer to "Employees". In order to represent these statements, decisions on how to represent the references must be made. A reference to an employee, for example, may be represented as the name of the employee or may be represented as an employee number etc.

The design of the conceptual data structure is dependent on the data model for the conceptual level provided by the data base management system to be used.
The data structure has to be expressed in terms of the modeling concepts of this data model. For example, if the Relational Data Model is provided by the data base management system, the conceptual data structure has to be expressed in terms of relations. If a CODASYL-like data model is used the data structure has to be expressed in terms of record types and settypes etc.
In design of relations (or record types) decisions have to be made on how to group types of statements into

relations. For example, a type of statement concerning the monthly salary of an employee may be grouped with types of statements as the name and the address of an employee, into one relation (or record type).
In the grouping of types of statements, eventual restrictions imposed by the data model have to be considered. For example, relations may be required to be in 3NF and thus, the grouping of types of statements into relations must consider this restriction.

Analysis and eventual restructuring of the conceptual data structure in order to achieve a structure which does not violate restrictions imposed by the data model is seen as the final task within the conceptual level data base design. When this task has been performed, the resulting conceptual data structure can be described (by some DDL) and used as the conceptual schema of a data base.

3.6 Summary

The purpose of this chapter was to present a context for this study. The frame of reference constitutes a specification of the scope and the content of "conceptual level" data base design.
The scope of "conceptual level" data base design was described as five problem areas:

1. Concepts/models for description of users' views.
2. Concepts/models for description of users' information requirements.
3. Methods for design and analysis of information structures.
4. Concepts/models for description of conceptual data structures.
5. Methods for design and analysis of conceptaul data structures.

In the descriptions of the context of "conceptual level" data base design the three first problem areas were considered as relevant to "information structure design". The two following problem areas were considered as concerning "conceptual data structure design".
An "information structure" was described as a specification of the information contained in a data base system. Description of user views and information requirements were examplified and their relevance to the design of the information structure were illustrated.
A "conceptual data structure" was described as a specification of the information content and its representation in a data base.

An important idea to this study is that the conceptual data structure of a data base can not be designed until a data and process structure for the data base (or data processing) system has been designed. The data and process structure of the data base system determines the information content of the data base.
After decisions on how to represent the information contained in the data base have been made, the content can be structured in terms of the data model provided by the data base management system to be used.

4. INFORMAL ANALYSIS OF CONCEPTUAL LEVEL DESIGN METHODS

In this chapter different methods for conceptual level data base design are analyzed from a semantical point of view.

In the previous chapter, the scope and content of conceptual level data base design, as seen in this study, was described. According to this description, conceptual level data base design could be partitioned into information structure design and conceptual data structure design. An information structure was regarded as consisting of a global user view, a set of types of statements (referring to elements of the global view) and a set of inference relationships between types of statements. A conceptual data structure was seen as a set of types of statements, including constraints on instances of the types of statements and of specification of representation and organization of statments.
The information structure as well as the conceptual data structure constitute models of an enterprise (some part of the real world).
Within the data base design context, the purpose of the information structure model is to constitute a base for design of a data base system. The purpose of the conceptual data structure model is to constitute a specification of the organization and representation of information contained in a data base.
Conceptual level data base design could be illustrated as in fig 4.1.

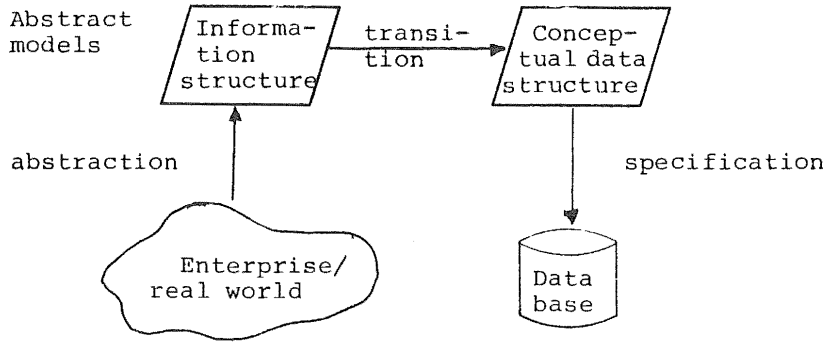


Fig 4.1

Nine different methods are analyzed below (ref. chapter 1). First in the analysis, models and transitions within the different methods are identified. Thereafter, semantical aspects of models and transitions are identified and informally analyzed. It should be noticed that the purpose of the analysis of the methods is not to compare different approaches to conceptual level data base design. The purpose is to analyze different methods in order to identify semantical aspects and problems

within conceptual level data base design. Therefore, an attempt is made to, for each method, identify some characteristic semantical property/problem. Semantical properties/problems which have beeen analyzed for one method may very well exist in other methods without being especially commented.

4.1 Benci, Bodart, Bogaert and Cabanes

In Benci's et.al approach, conceptual level data base design is the design of a "conceptual organization". The "conceptual organization" consists of a conceptual data structure, of integrity constraints and of evolution rules.
The base for the design of a "conceptual organization" is a, to all users common, view of real world phenomena called a "real world perception".
The to all users common "real world perception" in turn, is an integration of local users perceptions of real world phenomena.
The "conceptual organization" models three different aspects of a data base system, namely its data, its integrity and its evolution.
The models and transitions identified in Benci's approach can be described as in fig 4.2.

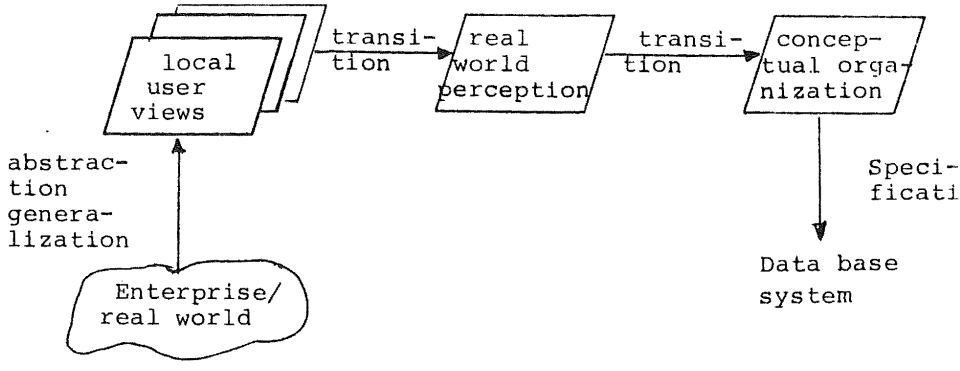


Fig 4.2

Different users are assumed to perceive real world phenomena in terms of different models or concepts as for example: "mathematical models, logical models, analogical models, descriptive models etc".

The transition of local users views into a to all users common real world perception is not described in the reference. However, the content of each local view is said to be described by the "real world perception model".
A set of basic modeling concepts in terms of which the "real world perception model" is to be expressed is proposed. The authors assume that the information content of each local view can be expressed in terms of these basic modeling concepts.

Enterprise or real world, phenomena are seen as objects, properties and associations, and the elements of the "perceived real world model" are described as abstract types of objects, properties and associations.

Guidelines for identification of the elements in the enterprise are suggested and are based on identification of temporal and spatial localization and dimension of the objects, properties and associations.
The modeling concepts proposed for the "perceived real world model" are types of objects, properties and associations. The so called types are here interpreted as sets/classes of objects, values and associations.

The conceptual organization consists of a "conceptual structure" for the data base, integrity constraints and evolution rules of the data base.

As a formalism for expressing the "conceptual structure" of a data base, an extended version of the Relational Data Model is proposed.
A conceptual structure is made up by three types of basic elements:
- the type of data
- the type of entity
- the relation

The transition of a "real world perception model" into a conceptual organization is not described.
However, a correspondence between elements of the "perceived real world model" and the "conceptual structure" is described as

Perceived real world	Conceptual Structure
- type of property	- one or more type of data
- type of association	- relation(s)
- type of object	- relation(s)

Integrity constraints describe properties of the data in the data base.

The integrity constraints are expressed as predicates on the elements in the conceptual data structure or as predicates on data elements in the data base, i.e. as predicates on the type level as well as the instance level elements. Examples of constraints are definitions of possible values and formats for values of a data type, definitions of realtionships between data types within a relation.
As an example, in a relation ORDERLINE:

ORDERLINE (ORD-NR,PROD-NR,Q,P,LV) where

ORD-NR	:	ordernumber
PROD-NR	:	product number
Q	:	quantity
P	:	price
LV	:	value of an orderline

an integrity constraint as LV=QxP may be defined.
Integrity constraints may be used to specify different users permitted access to data in the data base.

The transition from a "perceived real world model" into integrity constraints is not described. A comment, in the reference, indicates that those integrity constraints are based on properties of real world phenomena. However, it is not clear how such properties are identified and whether or not the properties are described in the "real world perception model" and if this is the case, how the properties are described in terms of the modeling concepts proposed for the "real world perception model".

Evolution rules are said to model dynamic aspects of the data base system.
Evolution rules are descriptions of, under which conditions integrity constraints are to be enforced. Enforcement of integrity constraints is triggered by so called events. Events are described as any decisions which triggers the processing of an operation on the data base. Events can be "at rest" or "activated". The

activation of events may depend on events in the enterprise or on a particular state in the data base.

The extension of the Relational Data Model proposed by the authors is made for the purpose of describing the evoultion rules.
Again, the transition from a "perceived real world model" into evolution rules of the conceptual organization is not described. Thus, it is not clear whether or not the phenomena in the enterprise, as for example "events", which in the conceptual organization are represented as evolution rules, are represented and described in the "perceived real world model".

In relation to the frame of reference of this study, the "real world perception model" corresponds roughly to a "global user view" and the "conceptual organization" corresponds roughly to a conceptual level specification of a data base system.
As seen in this study, however, a problem in Benci's approach is that the scope of the "conceptual organization" is wider than the "perceived real world model". The "perceived real world model" is said to be the base for design of a "conceptual organizaton". However, it seems doubtful that modeling concepts proposed for the "perceived real world model" are able to express and model such real world phenomena which in a "conceptaul organization" are represented as integrity constraints and/or evolution rules.
An information structure (including a global user view), as described in chapter 3, intends to model and describe all relevant aspects of real world phenomena which are needed in the design of a data base system and which are represented as a data structure as well as integrity constraints. In Benci's et al approach it seems as if the "real world perception model" describes only those real world phenomena which in a data base system are to be represented as a data structure, but not those which are to be represented as integrity constraints and/or evolution rules.

4.2 Bernstein

In Bernstein's approach, a data base is seen as a set of relations. The intensions of these relations constitute a conceptual data structure for the data base.
As a base for arriving at a specific set of relations, a universal set of attributes and functional dependencies between sets of attributes are assumed to exist. This set of attributes and functional dependencies corresponds to what in this study is called a global user view.
In Bernstein's approach the origin of this global view is not discussed. Whether or not the global view is a result of a design process is not indicated. Thus, models and transitions in Bernstein's approach can be illustrated as in fig 4.3.

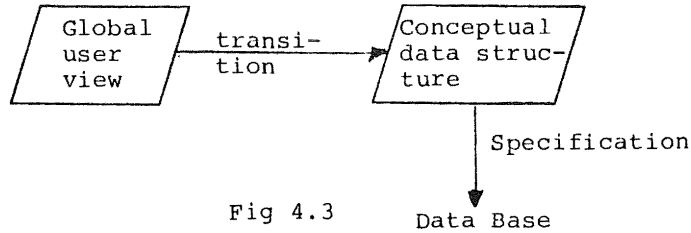


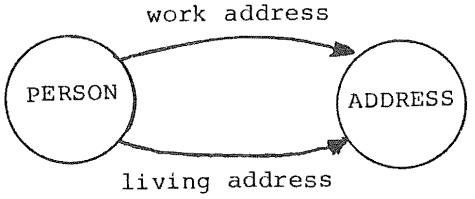
Fig 4.3

The global user view is expressed in terms of attributes and functional dependencies.
Attributes correspond to sets of objects or values or rather to sets of names of objects and values.
Functional dependencies are defined: "Let A and B be attributes, let DOM(A) be the domain of A and DOM(B) be the domain of B and let f be a time-varying function such that $f:DOM(A) \longrightarrow DOM(B)$. f is not a function in the precise mathematical sence because we allow the extension of f to vary over time in the same sense that we allow extensions of base relations to change over time. That is, if f is thought of as a set of ordered pairs $\{(a,b) | a \in (DOM(A) \text{ and } b \in (DOM(B)\}$, then at every point in time for a given value of $a \in (DOM(A)$ there will be at most one value of $b \in DOM(B)$". [BER-76].
In the notation used in this study, a functional dependency (FD) is an "element to element" relationship between two sets of entities/values and the relationship is either of 1:1 or M:1 type.

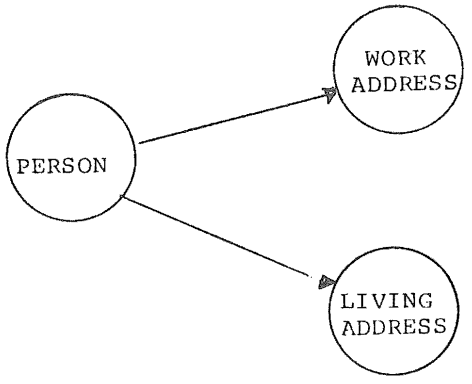
A non functional association, existing in the real world, is in Bernstein's approach represented as a functional dependency.

"A non functional connection f among a group of attributes A_1, A_2 ....A_n will be represented as the following FD: $f: A_1, A_2 \ldots A_n \rightarrow \Theta$, Θ is an attribute that is unique to f, and it does not appear in any other FD. Each FD representing a nonfunctional relationship has its own private Θ attribute. The underlaying domain for all these Θ attributes is the set $\{0,1\}$. For each element $(a_1, a_2, a_3 \ldots a_n) \in (DOM(A_1) \times DOM(A_2) \ldots DOM(A_n), f(a_1, a_2 \ldots a_n) = 1$ if and only if $(a_1, a_2 \ldots a_n)$ is related under f. Thus the extention of f completely defines a nonfunctional relationship among $A_1 \ldots A_n$. For example a nonfunctional relationship between a DRIVER and AUTOMOBIL, where each AUTOMOBIL can be driven by more than one DRIVER and each DRIVER can drive more than one AUTOMOBILE is represented by the FD: DRIVER, AUTOMOBIL $\longrightarrow \Theta 2$.

In Bernstein's approach (as in other synthesizing approaches) there may exist at most one functional dependency between any sets of attributes. If, for example, more than one functional dependency exist between the attributes A and B, these functional dependencies would be regarded as semantically equivalent and thus as the "same" functional dependency.
In order to represent semantically different functional dependencies between sets of attributes, new attributes have to be introcduced. For example, to represent



the attribute ADDRESS has to be represented as two attributes.



The transition from the global user view to the conceptual data structure is exactly defined by and algorithm. This algorithm for generating a data structure from a set of functional dependencies is based on Armstrongs Axiomatization of Functional Dependencies [ARM-74]. The axiomatizations used in the algorithm are

A1 (reflexivity) $X \rightarrow X$

A2 (augmentation) if $X \rightarrow Z$ then $X+Y \rightarrow Z$

A3 (pseudotransitivity) if $X \rightarrow Y$ and $Y+Z \rightarrow W$ then $X+Z \rightarrow W$.

Bernstein stresses and points out that application of these axioms may lead to semantically ambigous inferences. For example, "Let f_1:DEPT# →MGR# and f_2:MGR#,FLOOR →NUMBER_OF_EMPLOYEES. One interpretation of f_1 and f_2 is that f_1 determines the manager of each department and f_2 determines the number of employees working for a particular manager of a floor. By applying pseudotransitivity to f_1 and f_2 we obtain f_3:DEPT#, FLOOR → NUMBER_OF_EMPLOYEES, which determines the number of employees of the manager of a particular department on a particular floor. If a manager can manage more than one department then f_2 is not the same as the syntactically identical FDg_1:DEPT#,FLOOR → NUMBER_OF_EMPLOYEES, which determines the number of employees of a particular department on a particular floor. To make g_1 distinct from f_3 one has to change an attribute name to make the FDs syntactically different. For example, one could change f_2 and g_1 such that f_2:MGR#,FLOOR → NUMBER_OF_EMPLOYEES_OF_MANAGER and g_1:DEPT#,FLOOR → NUMBER_OF_EMPLOYEES_OF_DEPT. Now g_1 is distinct from the composition of f_1 and f_2" [BER-76].

Another example presented: "let f_6:STOCK# → STORE and f_7:STOCK#, STORE# → QTY . Since the composition of f_6 and f_7 is g_2:STOCK# → →QTY, it must be (by our assumption) that the attribute STORE in f_7 not is needed. But suppose f_6 maps a STOCK# into a STORE# of the store that is in charge of ordering that item, and f_7 maps the STOCK# of an item and the STORE# of the store in which it is being sold, into the quantity on hand. In this case g_2 does not imply that STORE# is extraneous in f_7. To prevent this syntactic inference from taking plance, we must change an attribute name (e.g. f_6:STOCK# →ORDERING_STORE)" [BER-76].

Bernstein points out that in these examples "a syntactic inference was either errorneous or misleading". In each case we solved the problem by renaming an attribute to distinguish it from another attribute. This renaming essentially moves some semantic knowledge that we have about an FD into the syntactic level, where it can be used by the algebra of FD's". Further, it is stated that "if we are to make use of a formal algebra of FD's we must make the assumption that all syntactic inference are valied". [BER-76].

From a semantical point of view, the application of the

algebra of FD's is uncertain.
The essence of Bernstein's statements are here interpreted as:
- In order to use the algebra of FD's only FD's that will lead to semantically valid inferences are assumed to exist
- Problems of errorneous or misleading inferences are solved by renaming attributes.

Bernstein's approach to avoid "semantical errors or ambiguities" in inference of functional dependencies is to create unique attribute names when needed as for example in

	Problem	Solution
1)	PERSONS→→ADDRESSES PERSONS→→ADDRESSES	PERSONS→→WORKADDRESS PERSONS→→HOMEADDRESS
2)	DEPT#→→MGR# MGR#,FLOOR→→NUM.OF.EMP DEPT#,FLOOR→→NUM.Of.EMP.	DEPT#→→MGR# MGR#,FLOOR→→NO.OF.EMP1 DEPT#,FLOOR→→NO.OF.EMP2
3)	STOCK#→→STORE# STOCK#,STORE#→→QTY STOCK# QTY	STOCK#→→STORE# STOCK#,STORE#→→QTY1 STOCK#→→QTY2

Practically, Bernstein achieves unique attribute names by moving the semantics of the association to the attribute names as for example:

PERSONS $\xrightarrow{work}$ ADDRESSES	PERSONS →→WORKADDRESSES
PERSONS $\xrightarrow{home}$ ADDRESSES	PERSONS →→HOMEADDRESSES

or generally

X $\xrightarrow{a_1}$ Y	X →→a_1Y
X $\xrightarrow{a_2}$ Y	X →→a_2Y

Bernstein states that "Specifying a set of FD's that can lead to no invalid syntactic inferences is clearly a difficult problem".
However, there is a very simple way of specifying such FD's, namely to use unique attribute names for each time an

attribute appears as the dependent attribute in a functional dependency.

For example:

$X \rightarrow Y$	is changed to	$X \rightarrow Y1$
$Z \rightarrow Y$		$Z \rightarrow Y2$

or practically:

$X \xrightarrow{a_1} Y$	is changed to	$Z \rightarrow a_1 Y$
$Z \xrightarrow{a_2} Y$		$Z \rightarrow a_2 Y$

or generally:

$f_1: X \rightarrow Y$	is changed to	$X \rightarrow f_1 Y$
$f_2: Z \rightarrow Y$	is changed to	$Z \rightarrow f_2 Y$

If unique names are used for attributes each time an attribute appears as the dependent attribute in a functional dependency, semantical errors and ambiguities which refer to the use of non-semantical functional dependencies are eliminated. However, if unique names of attributes are systematically applied, the axiomatizations and thus the applicability of the syntactical inference in the algorithm is not predictable, as in

$f_1: X \rightarrow Y$	but	X may be $\neq f_1 X$
$f_2: X \rightarrow Z$ $f_3: Z + Y \rightarrow Z$		$f_2 Z$ may be $\neq f_3 Z$
$f_4: X \rightarrow Y$ $f_5: Y + Z \rightarrow W$		$f_4 Y$ may be $\neq Y$ in f_5

Thus, in summary, the algorithm proposed by Bernstein can not be proved to produce semantically correct or valid relations unless the input set of FD's can be guaranteed not to lead to invalid syntactic inference.

In relation to the frame of reference of this study, Bernstein's assumed input, i.e. a set of attributes and a set of functional dependencies correspond to a to all users common shared view of real world phenomena.
In many of Bernstein's examples the attributes refer to sets of names of phenomena as for example STOCK#, STORE#, etc, thus indicating that not only decisions on what to represent in the data base but also on how to represent it is initially assumed by Bernstein.

The conceptual data structure as described in the frame of reference corresponds to the data structure generated by Bernstein's algorithm.

4.3 Brodie and Tsichritzis

In relation to our frame of reference, Brodie and Tsichritzis are primarily concerned with specification of user views and with integration of users views into a global user view. Summarily, their approach can be described as in fig 4.4.

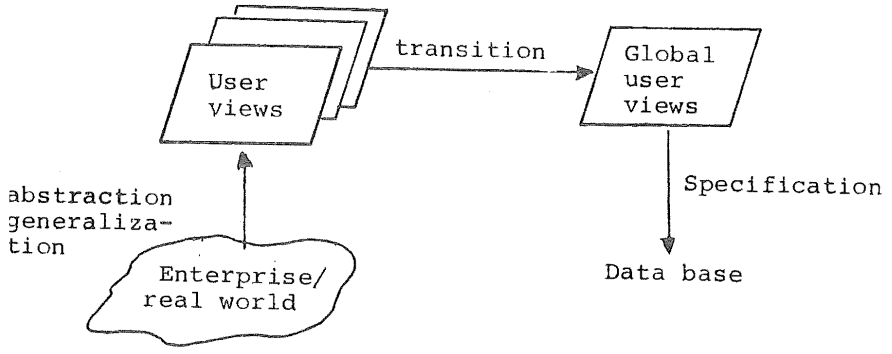


Fig 4.4

A local user view is described as a collection of objects and relationships, as seen by a given user or class of users. According to this description, a (local) user view consists of objects and relationships. However, from a semantical point of view it is not quite clear what the objects and relationships of a user view denote.
In chapter 3, concepts frequently used for expressing user views were described and discussed. Among these concepts were generic objects and classes/sets of objects. According to the distinction between generic objects and class/sets of objects, the elements (objects) of a user view as described by Brodie and Tsichritzis seems to (initially) correspond to single generic objects.
In our frame of reference, associations between generic objects - as well as associations between classes/ sets of objects - correspond to abstractions of real world phenomena. For example, in fig 4.5 the association "teach" is seen as an abstraction and generalization of

some real world phenomenon, in this case of an activity which we call "teach".

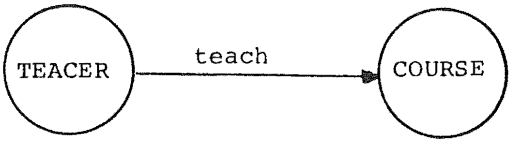


Fig 4.5.

Other associations in a user view correspond to abstractions and generalizations of other real world phenomena for example as in fig 4.6

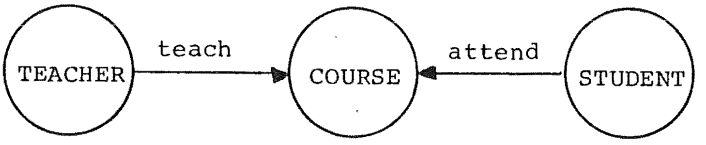


Fig 4.6

However, the associations in a user view as described by Brodie and Tsichritzis do not seem to correspond to genralizations of real world phenomena as in our frame of reference. Instead, all associations in a user view seem to describe how objects described in a user view can be represented in a data base in terms of other objects. User views are expressed in terms of (generic) objects and "is used to compose" associations as if fig 4.7.

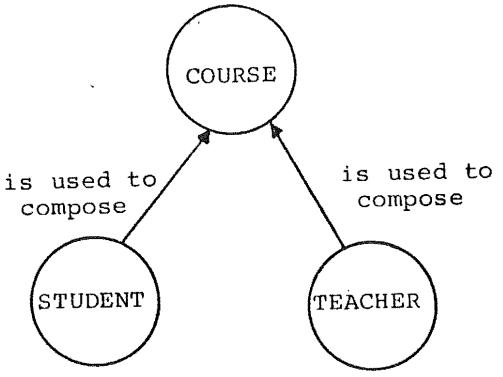


Fig 4.7

The process of specifying user views is described as: "...decompose each distinct object into its component

objects. Then, decompose each component object into components and so forth until the final components are the fundamental domains". [BRO-78]
The fundamental domains assumed are STRING and NUMBER. Here, we regard this procedure not as a stepwise decomposition of objects, but as stepwise decisions on how to represent objects in the data base. Referring to fig 4.7, the next step according to this procedure would be to decide how STUDENT and TEACHER are to be represented in the data base in terms of other objects.

As pointed out in the reference, a particular hierarchy of objects presents only one way of viewing the objects or, in our terminology, that it presents one possible way to represent objects in a data base.
Different views, i.e. different decompositions of an object are combined into a so called abstract view of an object. A schema for a data base is a combination of the schemas for the abstract views.
Abstract data type specifications are suggested as formalism to describe the objects in a hierarchy or network of "component objects". "The structural properties are the component objects and their composition rules. The behavioral properties depend on the semantics of the objects; they are based on query, update, insert and delete operations. The resulting network of abstract data types forms the schema". [TSI-77]

A conceptual data structure consists of abstract data type specifications and of data base constraints. Brodie and Tsichritzis describe semantic integrity rules to be properties of abstract data types which cannot be expressed in the data definition. Semantic integrity rules are expressed as data base constraints. "A constraint asserts that a (set of) data value(s) exhibits a given property or objects, a given relation with respect to other data values".

Some examples of constraints given in [TSI-77] are: "The assertion that every full-time student take at least four and at most six courses. The assertion that a tutor must be a student who has received at least B+ for the course being tutored. The assertion that there may be one tutor for every 25 students enrolled in the a course." In our terminology, the constraints , in the above examples, can be regarded as descriptions of user views where the objects correspond to sets of elements and the associations correspond to associations between elements in sets. In this case, the integrity rules correspond to detailed specifications of associations. For example,

the constraint that every full time student takes at least four and at most six courses can be described by an association as in fig 4.8.

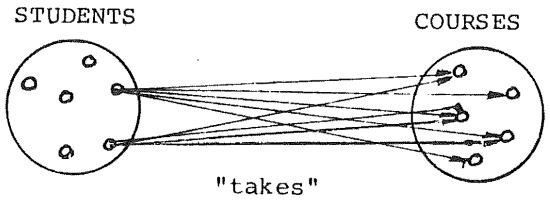


Fig 4.8

Thus, in the determination and description of semantical integrity rules the objects in the user views seem to be regarded as sets of objects and the associations as a set of associations between elements in the two sets of objects. In this case, the associations are semantical associations, i.e. they are abstractions and generalizations of real world phenomena.
This way of interpreting the objects and associations within a local user view, i.e. as sets of objects and as sets of semantical associations between elements in the sets of objects is implicitly assumed also when hierarchies of objects are combined into user view schemas. In order to combine hierarchies of objects, there must be no conflicting properties of the objects in the hierarchies. "An example of a logical conflict is that one abstract view may permit one or more professors to be associated with a given course while another view requires that exactly one professor teaches a course" [TSI-77]. This example implies an interpretation of COURSE and PROFESSOR as sets of objects and a semantical association between these two sets.
Thus in summary, users' views as described by Brodie and Tsichritzis seem to consist of generic objects which are related by associations which describe how the objects are intended to be represented in the data base. However, in order to combine users views and in order to formulate and describe semantic integrity rules, user views in terms of sets of objects and semantical associations between elements in the sets of objects are implicitly assumed. In the presentation of the design approach the authors stress the possibility of allowing different users to maintain alternative views of the real world. Here this possibility could rather be expressed as the possibility for different users to represent objects in different ways. However, in order to assure semantical integrity, a to all users common view in terms of sets of objects and semantical associations between sets of objects is required.

In relation to the frame of reference of this study, Brodie's and Tsichritzis' approach is regarded as concerned with design of a global view of real world phenomena, i.e. of determination of a part of an information structure and with the transition of this part of the information structure into a conceptual data structure. In this transition the problem of how to represent real world phenomena as data stored in the data base is focused.

4.4 Bubenko

The IAM-approach is presented as a method for design of a conceptual schema. Conceptual schema is informally defined as: "a conceptual schema describes the information content and relationships of a data base. The information content is an integration of different local "needs" and "views". Redundancy exists in the sence that some information "elements" can be inferred from others. The conceptual schema acts as a basis for i) definition of external schemata and queries and ii) design of data structure schema (storage oriented) where performance, efficiency and limited storage issues have to be considered". [BUB-76]

This informal definition of "conceptual schema" corresponds to what in this study is called an information structure.
The design of a "conceptual schema" as described by Bubenko does not include or persume the design of a data base (or of a data base system). The "conceptual schema" as described by Bubenko does not restrict the types of statements within a data base system that are to be contained in the data base, it does not specify constraints on instances of types of statements in the data base or specify an organization of types of statements in the data base.
Therefore, Bubenko's "conceptual schema" is here regarded as an information structure.
The first three steps within the IAM method concerns modeling of real world phenomena in terms of local users information requirements and in terms of a to all users common shared view of real world phenomena.
The local information requirements correspond to anticipated input transactions, output reports and queries. They are assumed to be initially stated in narrative terms.
The local information requirements and the global users view together constitute a base for a systematical design of an information structure as in fig 4.9.

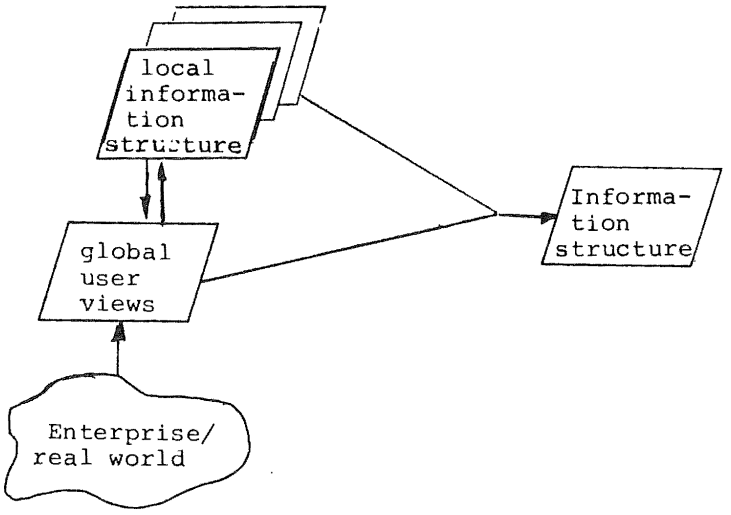


Fig. 4.9

The design approach is primarily concerned with the transition of a global user view and a set of local information requirements to an information structure.

The process of designing or determine a global user view is not elaborated in the IAM report [BUB-76B]. However, the three first steps of the design method are said to concern analysis of anticipated input transactions and output requirements in order to determine relevant classes of objects and associations between classes of objects.

The, to all users common shared view of real world phenomena, is expressed in terms of "concept classes" and functional dependencies between concept classes. A "concept class" corresponds to what in this study is called an object class, and thus, to an abstraction and a classification of real world phenomena. Real world phenomena are required to be classified into disjoint concept classes.
Associations between concept classes are expressed in terms of functional dependencies.
Functional dependencies between sets of objects (concept classes) correspond to "semantical associations" of the types 1:1 or M:1 between sets of objects. By "semantical associations" is meant that an association is an

abstraction of some real world phenomena. The associations are assumed to be named. For example:

EMP,D—lives→ADR

which denotes a functional dependency between the concept classes EMPLOYEE, DAY and ADDRESS and which states that a particular employee and a particular day uniquely determine the employees living address.
By using semantical associations several functional dependencies between the same set of concept classes may be defined.

The information requirements correspond to (types of) statements about real world phenomena. As such they refer to concept classes in the global user view.

However, initially, the information requirements are local, i.e. the referenced classes of real world phenomena may not be consistent. Different users may have abstracted and classified the real world phenomena differently. The analysis of the anticipated input transactions and output requirements (i.e. the information requirements) in the first steps of the IAM procedure aims at identification and elimination of such inconsistencies.

The process of designing an information structure from information requirements and from the common shared view starts with a formal representation of information requirments and functional dependencies in terms of "abstract objects". "An abstract object of class A_i is an abstract construct which conceptually serves to represent information about some associations or relationships in the real world. The structure of an abstract object class A_i can be described as:

$$A_i=[<a:\Theta_a>,<b:\Theta_b>.... <s:\Theta_s>]$$

where a,b... are associations names, unique within A_i, and $\Theta a, \Theta b$... denote concept classes or abstract object classes (sub-objects to A_i)" [BUB-76].
Graphically an abstract object class can be illustrated as in fig. 4.10.

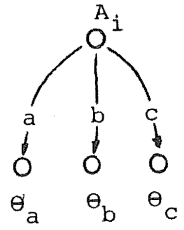


Fig 4.10

When information requirements and functional dependencies have been represented as abstract object classes, these abstract object classes are "normalized" to "self-contained", 1-level objects.
"A 1-level abstract object has the properties that all its associations are functional ((1:1) or M:1)) and that one or more subsets of its associations uniquely identify it". [BUB-76].

However, from a semantical point of view, it is not clear what is meant by a "self-contained" abstract object and it is not clear how "normalization" is carried out.
"Normalization" and "self-contained" abstract objects are described by the following example in the report [BUB-76].
"Assume the following request:
Q1 For each plant and each month list the total number of employees and total-salary, then list for each employee name, hours-worked and salary".

Disregarding the fact that the formulation of this request is ambigous, we assume that it leads us to the set of concept classes {PLA,M,NR,$$,EMP,NAME} and to the following abstract object construct:

Ax {< plant:PLA>,<month:M>,<tot-emp:NR>,<tot-sal:$$>, < sub-report:Ax' >} where
Ax' {< empl:EMP>,<e-name:NAME>,<sal:$$>,<hours-w:NR>}

We assume that the associations 'plant' and 'month' together uniquely identify objects of class Ax and that the associations 'empl' and 'sub-reportc' (where the subscribt 'c' denotes the converse of an associaton) uniquely identify Ax-type objects. This implies 1:1 correspondences between PLA,M ⟷ Ax and Ax,EMP ⟷ Ax', making it possible to normalize Q1 Ax to the two 'self-contained' abstract objects Ax and Ay:

Ax = {<plant:PLA>,<month:M>,<tot-emp:NR>,<tot-sal:$$>}
Ay = {<empl:EMP>,<month:M>,<plant:PLA>,<name:NAME>,
<hours-w:NR>,<sam:$$>} " [BUB-76].

The described normalization process could be interpreted as:

assume: PLA,M ⟶ Ax
and assume: Ax,EMP ⟶ Ax'
then : PLA,M,EMP ⟶ Ax'

which indicate that some "syntactical inference" of functional dependencies is applied (see 4.2).

However, this should not be the case in "IAM-normalization" as the associations and thus the functional dependencies are here required to have a "semantical meaning".
In terms of functional dependencies, the result of "normalization" in the example above can be described as:

$$EMP,M,PLA \xrightarrow{empl,month,plant} Ax'$$

However, in order to infer this functional dependency from the initially assumed functional dependencies

1) $PLA,M \xrightarrow{plant,month} Ax$

2) $Ax,EMP \xrightarrow{sub\text{-}report^{G},empl} Ax'$

general assumptions about inference of "semantical, functional dependencies" must have been made. Also, assumptions which are specific to the application, about semantical equivalence between associations (in this application probably between plant, month and subreport^{C}) must have been made.
As none of these assumptions are described we conclude that from a semantical point of view, it is not clear how "normalization" is carried out.

The meaning of the expression "a selfcontained abstract object" is not described. It seems to have something to do with a "semantical context".
For example in

Ax' ={<empl:EMP>,<e-name:NAME>,<sal:$$>,<hours-w:NR>}

the "semantical context" is not wide enough for interpretation of the semantics of the associations "sal" and "hours-worked".
However, in the "self-contained" abstract object:

Ay = {<empl:EMP>,<month:M>,<plant:PLA>,<name:NAME>,
<hours:NR>,<sal:$$>}

the "semantical context" is widened and the associations could be interpreted as "hours-worked at a specific plant during a specific month" and "salary for work performed at a specific plant during a specific month". The "semantical context" is one possible interpretation of the meaning of "self-contained" abstract object, other interpretations may exist.

Referring back to the IAM-method, the initial abstract objects are "normalized" to "self-contained" 1-level abstract objects. Foreach association within each 1-level abstract object it is to be indicated whether the association is regarded as identifying, implied or derivable.
The next step within the IAM-method is the so called "implied association analysis". If an association is specified as "implied" within an abstract object, the analysis and the determination of the implication rule may lead to identification of (by users) not specified information requirements or functional dependencies which are needed in order to produce the required output from the specified input. Such "new" abstract objects are specified and analyzed iteratively. After the "implied" association analysis is carried out, "derivation analysis" is performed. In this analysis, derivation rules for associations specified as derivable are determined. As in the analysis of implication rules, this analysis may lead to identification of "new" abstract objects which are necessary in order to derive "requirements" formulated as associations specified as derivable.

The result achieved after iteration of implication and derivation analysis is an abstract model expressed in terms of abstract objects, implication and derivation rules. Characteristic of this model is that it is complete in the sense that abstract objects specified as input are sufficient to produce abstract objects specified as output.
In the last step of the IAM-method the abstract model is transformed to a name-based model, i.e. for each concept class decisions on how to represent its elements in terms of names are made.
In relation to the frame of reference of this study, the

IAM-method is concerned with determination of an information structure.
The information structure describes the information to be contained in a data base system in terms of abstract objects and inference relationships between abstract objects.
The determination of the information structure is based on, a to all users common shared view expressed in terms of concept classes and functional dependencies, and on information requirements expressed in terms of abstract objects associated to concepts classes of the user views.

4.5 Hubbard and Raver

Hubbard's and Raver's design method is primarily intended for design of data bases to be managed by IBM's data base management system IMS. The design process can, in the terminology of this study, be described as:
- Specification of users' views
- Design of a data base management system dependent data structure.

A user view corresponds to the data elements and associations between data elements that are required by an application. The global view, which is called "the associative model" constitutes an integration of the local views. The integrated view is said to be "non-redundant" in the sence that only associations which can not be inferred from other associations are included. The integrated view can be regarded as a data structure for a data base. To be used as a data structure, the associative model has to be expressed in terms of a data model used by some data base management system, and thus be adapted to structuring restrictions imposed by that data model. In the reference [HUB-75]design of "logical" data structures for IMS and CODASYL data base management systems is discussed. However, these DMBS's are examples of two level systems based on "logical" and "physical" data structures,where the logical data structure to some extent is concerned with storage and access aspects of the data. Therefore, the part of Hubbard's and Raver's design method concerned with design of IMS or CODASYL structures are not included in this discussion.
In terms of models and transitions, the to this study relevant part of the design method can be described as in fig. 4.11.

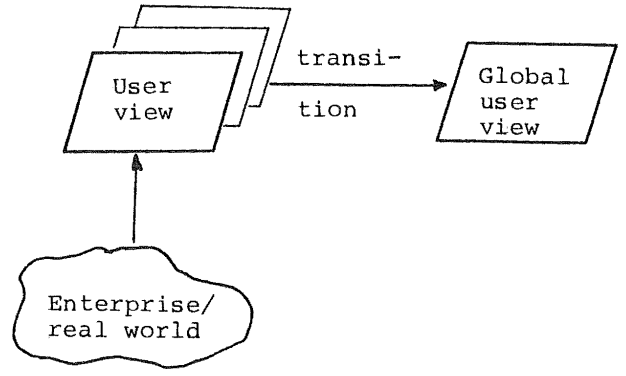


Fig. 4.11

There are some distinctions between local user views as described by Hubbard and Raver and user views as described in our frame of reference. First, the elements in local user views described by Hubbard and Raver are seen as data itmes and associations between data items. In the frame of reference, the elements of local user views are seen as abstract objects, i.e. as abstractions and generalizations of real world phenomena. In the case, when the elements are seen as data items, decisions on how to represent abstract objects in terms of names must have been made. In the frame of reference, user views might corespond to "overlapping" classification of real world phenomena and to inconsistent assumptions about the real world phenomena. Such inconsistencies between local views are not considered or discussed in Hubbard's and Raver's approach. The data items in the local views are assumed to refer to consistent classifications of the real world phenomena.
Thus Hubbard's and Raver's local user views are "overlapping" only at the type level, and the types are assumed to be consistent. Homonyms and synonyms among the names of the classes are considered and discussed.
The basic modeling concepts proposed for specification of local user views are "data elements" and "associations". A "data element" corresponds to an IMS "field" or a CODASYL "data item", i.e. is the smallest nondivisible data unit. Data elements are to be interpretated as specifications of sets of data values. In Hubbard's and Raver's terminology: "A 'data element name' is the symbolic reference to all occurrences of that data element and a 'data element occurrence' is a specific value" [HUB-75].

Two different kinds of data elements are identified: "A key is a data element whose occurrences uniquely

identify the entity or concept being described. An attribute is a data element that adds further description but does not by itself provide unique identification" [IBM-76]. If necessary for unique identification of entities more than one data element can be used as a key. Such keys are called compound keys. Associations are described as "from-to" relationships between two data elements. "The "from" element serves as a key, and each of its occurrences identified one or more (or no) occurrences of the "to" element". [IBM-76] .
For each association in a local user view its "mapping" is to be described. "Mappings" are combinations of foreward and backward associations. Each directed association is described as either "simple" ("1"), "complex" ("M") or "conditional" ("C"). "A simple association is one in which every occurrence of the "from" element identifies one and only one occurrence of the "to" element. A complex association is one in which each occurrence of the "from" element can identify any number (including zero) of occurrences of the "to" element. A conditional association is similar to a simple association, except that the "to" occurrence may or may not exist. Each occurrence of the "from" element identifies either one or no occurrence of the "to" element." [IBM-76]
Five possible "mappings" i.e. pairs of associations and their inverses are identified.
In Hubbard's and Raver's design approach local user views are integrated into a global user view. During this integration "redundant" associations are identified and eleminated. "Redundant" associations are such associations that can be inferred from other associations. The resulting global view - the associative model - consists of a "minimum" set of associations from which all associations specified in local users views can be inferred.
It should be noted that "inference" and thus "redundancy" in Hubbard's and Raver's approach refers to "syntactical inference".
Two important tasks within the process of integrating local views are:
- generation of implied associations
- identification of implied associations.
When, in local users views, compound data elements have been specified, these compound data elements imply additional associations. As an example, suppose the compound data element (AxBxC) has been identified. This compound data element implies the 12 (syntactical) associations:

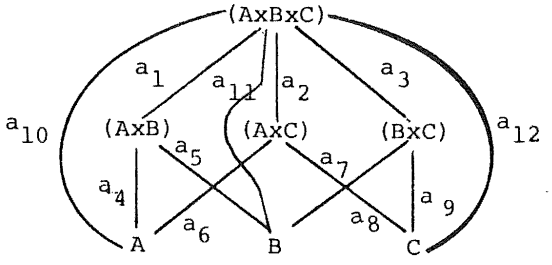


Fig. 4.12

Syntactically implied associations are "automatically" generated from compound data elements specified in the local user views. The number of implied associations increases with the number of elements in the compound data elements. A combination of three elements leads to 12 implied associations, a combination of four elements leads to 50 implied associations, a combination of five elements leads to 180 implied associations etc. etc.
As all syntactically implied associations may not be of value to the users, a possibility to restrict the number of generated associations is suggested. In this case, only associations which involves data elements included in local views are generated. For example, referring to fig 4.12 and assuming that in a set of local views, the data elements (AxC), A, B and C have been specified, a restricted generation of implied associations would identifiy six associations:

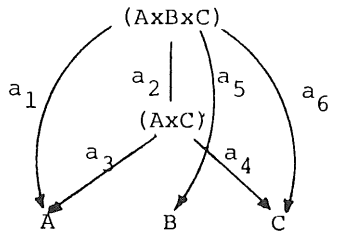


Fig. 4.13

Simple associations, specified in local users views or implied by compound data elements are structured into a network. As an example, suppose the following simple associations have been specified in local views.

EMPLOYEE ⟶ MANAGER
EMPLOYEE ⟶ DEPARTMENT
MANAGER ⟶ DEPARTMENT

When the local views are integrated into an associative network, the association EMPLOYEE⟶DEPARTMENT is considered as implied - redundant - as it is possible to determine the department of an employee via the simple associations EMPLOYEE⟶MANAGER and MANAGER⟶DEPARTMENT. Implied associations can be identified "automatically" from the specifications of associations in the users views. However, a warning against automatic identification of implied associations is issued: "An implied association may yield a different occurrence of the "to" element than is obtained by following the alternate path of simple associations. Consider the data elements Employee, Manager and Phone-No connected by simple association as shown. The association (EMPLOYEE, PHONE-NO) will yield the phone number of the employee whereas the associations (EMPLOYEE, MANAGER) and (MANAGER, PHONE-NO) will yield the phone number of the employees manager". [IBM-76P].

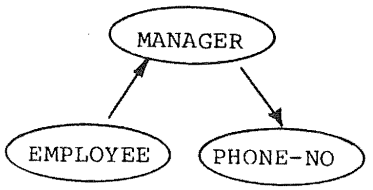


Fig. 4.14 (from [IBM-76].)

Although expressed in another way, this describes the same semantical problem in inference and implication of functional dependencies as has been pointed out in 4.2 and 4.3.
Hubbard and Raver suggests a practical solution to this semantical problem. All automatically identified "implied" associations are listed and presented to the designer. The designer decides whether or not "implied" associations can be regarded as implicable also from a semantical point of view. If this is not the case, the designer redefines such associations as being "essential" instead of implied. After the implied associations have been inspected and eventually redefined by the designer, all essential, simple associations are integrated into a "non-redundant" global view, i.e. the associative model.

Compared to the frame of reference of this study, Hubbard and Raver are concerned with determination of a global user view. The global user view is name-based, i.e. decisions on how to represent phenomena in terms of names is assumed to have been made.
Information requirements in the sence of statements about phenomena are not considered. However, statistical information about local user views as for example frequency and response time requirements are collected and used in the design of storage oriented data structure.

4.6 Kahn

Beverly Kahn's design approach is intended for "logical data base design". Logical data base design is regarded as consisting of three main activities, named:
- Information Requirements Analysis
- Information Structure Design
- Information Structure Implementation

Kahn's approach is based on the idea that the information content of a data base system can be described from two different points of view, or from two different perspectives: the information structure perspective and the usage perspective.
The information content of a data base system to be designed is specified in terms of "local" requirements. These local requirements are analyzed and the information to be contained in the data base is described from a "semantical" point of view, i.e. the information structure perspective and from a usage perspective.
"The information structure perspective depicts the natural and conceptual relationships of all data in the information system as viewed by the entire user community and is determined by the users of the system. It is not bound to any application and it represents the natural clustering of data (information). Therefore, this perspective will provide the mechanism for a flexible data base design and the basis for expressing the basic semantic properties as well as other facts about the entities in the information system and the relationships between these entities. The representation of natural relationships and facts in the data base provides the basis for handling unstructured and unanticipated queries or requests for information." [KAH-76B].

The "Information structure" corresponds to what in the frame of reference of this study has been called a global user view. This global user view is the result of an analysis and design process.
After the "information structure" has been designed, the usage perspective of the data, i.e. of elements of the information structure, can be determined. The information structure and the usage perspective together constitute a base for design of a data base management system dependent "logical" data structure as for example a CODASYL data structure. The design of the "logical" data structure is not elaborated in the references. Kahn's approach as seen in this study can be illustrated as in fig 4.15.

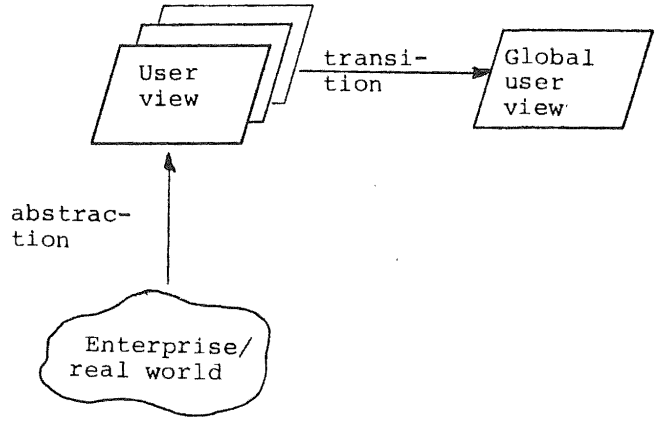


Fig. 4.15

User views are expressed in terms of entities, properties and relationships. "An entity is a person, place, thing, concept or event, real or abstract, of interest to the enterprise (organization). A property is a characteristic of an entity. A relationship in the real world is a connection between two or more collections of entities, individual entities or properties of entities. A relationship involves the objects connected, the kind of connection, and the direction of the connection". [KAH-76B]. Entity diagrams (as in fig 4.16) are suggested as representations of views.

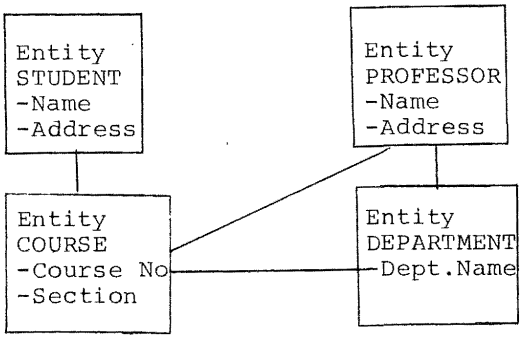


Fig. 4.16

The interpretation of nodes and arcs in an entity diagram is not quite obvious. For example, the entity diagram in fig 4.16 could be interpreted as a graphical representation of general knowledge about real world phenomena. In this case, the nodes of the entity diagram would correspond to single, generic objects and the arcs

would corresepond to binary associations between single generic objects. The entity diagram in fig 4.16 would then describe the general knowledge that:
- a student has a name and an address and is related to course,
- a professor has a name and a degree and is related to course and department,
- a department has a department name and is related to professor and course,
- a course has a course number and a section and is related to student, professor and department.

However, in the reference, Kahn summarily describes how the relationships in the entity diagram may be specified. This description indicates that the entities in the entity diagram could be interpreted as sets of entities and that the relationships could be interpreted as relationships between elements in sets.
For example: "A relationship between two entities describes how they are logically connected. The first entity occurrence form the basis for the domain of the relationship while the second form the basis for the range of the relationship" [KAH-76B]. It is also suggested that relationships between two entities could be classified as one-to-one, one-to-many, many-to-one or many-to-many mappings. How to, in a entity diagram, describe an association between a class of entities and an element of another class of entities (as illustrated in fig 4.17) is not clear.

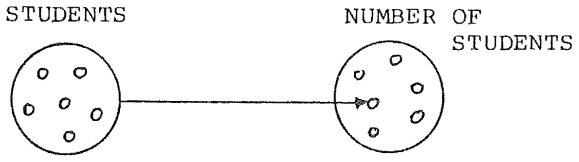


Fig. 4.17

The processes of arriving at a global user view from (local) users views is summarily described as:
"After the basic description of all of the pertinent system entities and relationships are collected and recorded by each participating member of the user community, these individual descriptions can then be refined. The refinement process includes detecting redundant and duplicate information, creating "relational" entities, analyzing relationships and derived elements and normalizing entities. When all of the individual information structure perspectives have completed this refinement process, they can be aggregated and consolidated into a global, consistent and non-redundant information structure perspective for the

entire organization through the use of the same refinement process" [KAH-76B].
Although, several steps in this process are not described in detail, we interpret Kahn's approach as similar to what in our frame or reference has been described as determination of a global view of real world phenomena based on local views.

The meaning of "non-redundancy" in an entity diagram is not clear. As an example, it is stated that: "All elements that belong to more than one entity (i.e. are included in more than one entity definition) should be determined. The information structure perspective should be as non-redundant as possible, therefore, a given element should only occur in one entity. To minimize the number of unique elements represented in the information structure perspective, all relational and derived properties should be separated from their associated entites". [KAH-76B].

First, it is not obvious why the information structure perspective should be as "non-redundant as possible".
Second, it is not clear why the number of unique elements in the information structure should be minimized and whether or not this concerns "redundancy". Furhter, does "separation" of relational and derived properties from their associated entities imply that these properties are excluded from the information structure?

The "usage perspective" is a definition of the systems processing requirements. In Kahn's approach, the information structure is seen as an application independent model of real world phenomena. The elements of the information structure are to be represented in the "logical schema" for a data base, however, the logical schema is application dependent as the structuring and grouping of the elements of the information structure is adapted to the applications processing requirements, i.e. to the usage perspective.

Identification and description of the usage perspective indicates a design of a data and process structure for the data base system.
"The first task in building the usage perspective is to identify all processing functions and then subdivide these functions into moduls (processes). The next task is to determine all of the data that each process uses to perform its designated functions" [KAH-76B].
In the detailed specification of the usage perspective the operations of each identified process is described .

The operations are described in terms of the primitive operators FIND, ADD, DELETE and MODIFY.
Thus, in summary, the usage perspective describes the data and process structure of a data base system. Each process is described by its primitive operations on data elements of the information structure.
The information structure and its usage perspective together constitute the base for design of a "logical" data structure. The design of the "logical" data structure is called implementation of the information structure. Within this step, "optimization" of the logical data structure relative the specified usage perspective is performed. Kahn does not suggest any specific method or algorithm for this optimization but refers to existing techniques as for example the ones proposed by Mitoma and Irani [MIT-75] and Hoffer [HOF-75].

4.7 Sheppard-Rund

A method for "logical data base design" is proposed by Donna Sheppard-Rund. This method differs from the other methods in this study with respect to its perspective of the result of the logical data base design. "The product of logical design can best be described as a model that depicts all pieces of data and how each relates according to its many uses across the organization" [SHE-76].
This perspective has an impact on the design method in the sence that it is primarily concerned with analysis of tasks and their information requirements rather than with abstractions and generalizations of real world phenomena. The result of the design process is a model of real world phenomena expressed in terms of of data elements and relationships between data elements. However, this model is determined as a result of an analysis of how the data elements are used in different tasks of the organization and the frequencies at which they are used.
"Logical data base design" is described as consisting of
- definition of scope
- identification of data elements
- identification of relationships of data elements
- identification of the organization's operating rules and their implication on data elements.

In the definition of scope, the functional areas of the organization which are to use and to be served by the data base are identified. Crude sets of data used by these functional areas are identified. The result is called an "Information Plan".
In the next step, functional areas, identified in the

Information Plan, are analyzed and described. The analysis includes
- identification and description of tasks performed within each functional area
- identification and description of data associated with performing each task
- identification and description of rules associated with when and how the task is performed.

Systematical interviews with top managers and middle managers are proposed as a "method" for identification of tasks and their associated data requirements. Flow chart technique is used in the description of tasks and data processing requirements.

The data requirements of each task are analyzed and organized into a "logical" data structure of a data base. In summary, the method approach may be described as in fig 4.18.

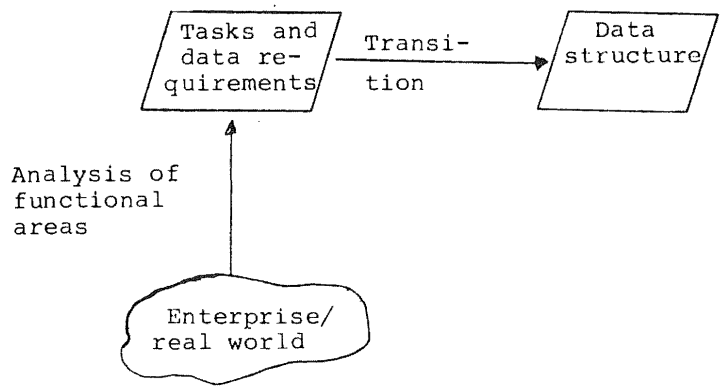


Fig. 4.18

The Data Requirements identified in the analysis of tasks within an enterprise may be regarded as corresponding to information requirements at a name based level. However, Sheperd-Rund's approach differs from our frame of reference with respect to the aspect of an enterprise that is primarily modeled.

In our frame of reference, the enterprise phenomena primarliy being modeled, (and referred to by statements) are those which are objects in the activities of an enterprise as for example ARTICLES, CUSTOMERS, ORDERS. In Sheperd-Rund's approach, the enterprise phenomena primarily being modeled are the activities (tasks) themselves, as for example the ORDER PROCESSING, the SALES FUNCTION.

This distinction has an impact on the approach taken in the transition of Data Requirments into a "logical" data structure.

The "logical" data structure is expressed in terms of data items and relationships between data items. Data items are classified as keys or attributes. The classification of data items as keys or attributes is based on the frequencies by which the data items are used in the different tasks of an enterprise. The classification of data items into keys or attributes is said to be based upon the "hypothesis" that:

- A data element will most likely be a key if it is used:
  - in a large number of tasks
  - with a large number of other data elements
  - with each individual data element a low percentage of the total time it is used
- A data element will most likely be an attribute referenced and owned by one key if it is used:
  - in a relatively small number of tasks
  - with a few data elements
  - with each data element a high percentage of the total time it is used.
- A data element will most likely be an attribute referenced by many keys but owned by one key if it is used:
  - in an average member of tasks
  - with an average number of other data elements
  - with each data element an average number of times.

Thus, a main activity within the transition of Data Requirements into a "logical" data structure is to created a frequency distribution of data elements across tasks (fig 4.19). Ranges corresponding to low, average and high frequencies are thereafter determined.

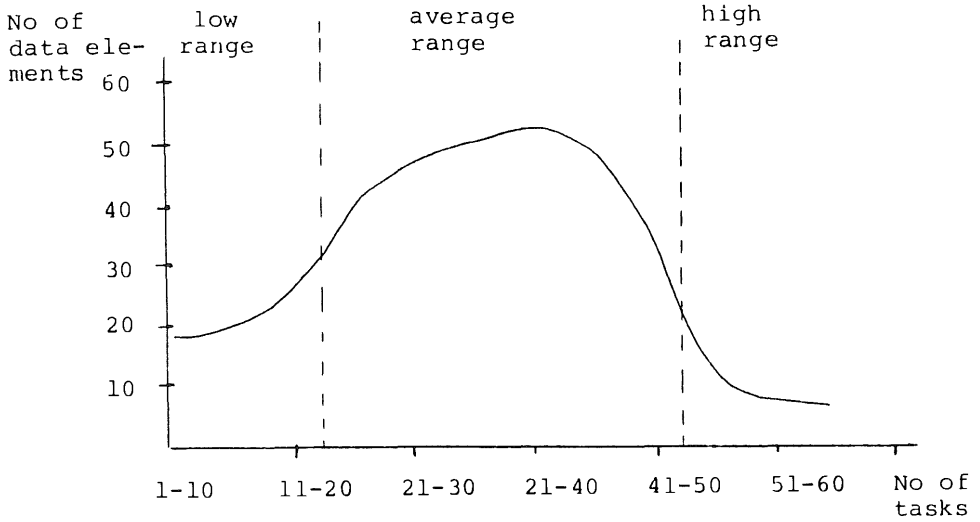
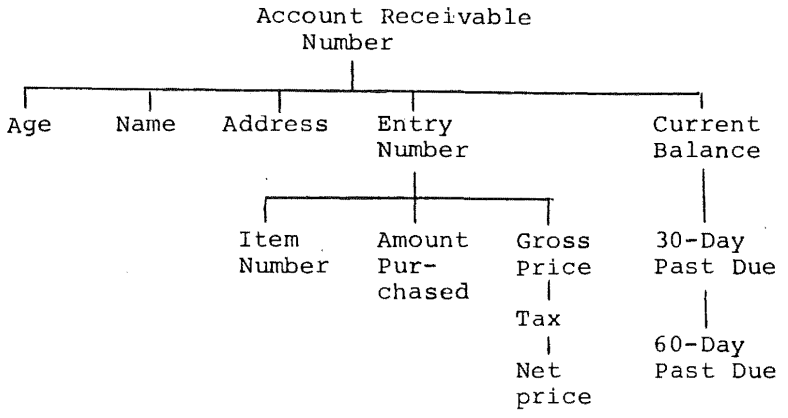


Fig. 4.19

The high, average and low ranges determine whether a data element is classified as a key, an attribute owned by one key and referenced by several keys or as an attribute owned and referenced by one key. After data elements have been classified, attributes are assigned to keys. Assignment of attributes to keys is accomplished by identifying the individual data elements that are related to the attribute being assigned, and assigning the attribute to the data element that has been identified as a key. Criteria for choosing one, out of several possible keys, in assignement of attributes are described:

"- if there is a wide range in the number of times the attribute is used with each key, assign it to the one with which it is most frequently used
- refer back to the list of task/data relationships and identify the task that created and deleted the attribute. Assign the attribute to the key present in those tasks.
- if different keys were used in the creation and deletion tasks, assign the attribute to the key present in the task in which it was deleted".
[SHE-76]

The next step in the design of the "logical" data structure is to identify dependencies between attributes assigned to a key. In this steps, the tasks which create the attributes of one key are identified and ordered according to their occurrence. The order in which the tasks occur will determine a hierarchical structure of attributes assigned to one key (fig. 4.20)
-



A hiearchial structure of attributes assigned to one key.

Fig. 4.20

In the last step of the "Logical" data structure design, relationships between keys are determined. Again, a to our frame of reference different aspect of real world phenomena constitutes the base for determination of relationships between keys. Relationships between keys are considered as reflecting the following aspects:

- owner relationships, i.e. existence relationships between keys,
- status relationships, i.e. relationships between keys, caused by the fact that groups of data change their meaning over time.
- regulation relationships, i.e. relationships between keys caused by regulations on the way in which the business is conducted, enforced by agencies outside the enterprise,
- policy relationships, i.e. relationships between keys caused by policies which regulated how different areas of the business must function either separately or together.

The identification of relationships between keys is considered as the most difficult and the most crucial task within the "logical" data structure design. The "logical" data structure design is considered as complete after all key relationships have been identified and reflected in the "logical" data structure.

It is difficult to relate Sheppard-Rund's approach to data structure design to the frame of reference of this study. The reason for this is the perspective of an enterprise which is represented in the design approach. The data elements of a "Logical" data structure are not primarily considered as representations of abstract objects and properties of abstract objects which in turn model phenomena in an enterprise. Instead, the data elements are primarily considered as components of the tasks that are performed in the enterprise. The organization of data elements into a "logical" data structure is based completely on how data elements are used in different tasks and on relationships between tasks. The relationship between data elements and what they represent is not explicitly considered in Sheppard-Rund's design approach.

4.8 Smith and Smith

Diane and John Smith present a method for design of a relational data base. The method is based on the concepts abstraction, generalization and aggregation. A conceptual data structure for a data base consists of representations of abstractions, generalizations and aggregations, of real world phenomena. Abstractions of real world phenomena - generalizations as well as aggregations - are represented in terms of relations. An extended version of Codd's Relational Data Model is proposed as a formalism for this representation. In relation to the frame of reference of this study, the design approach can be illustrated as in fig 4.21.

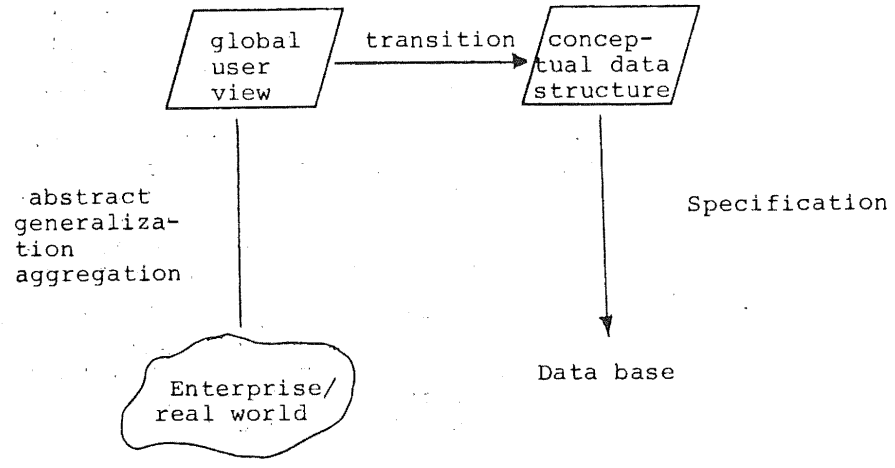


Fig. 4.21

The elements of a global user view correspond to abstractions of real world phenomena. Two kinds of abstraction - aggregation and generalization - are identified. "Aggregation refers to an abstraction in which a relationship between objects is regarded as a higher level object". "Generalization refers to an abstraction in which a set of similar objects is regarded as a generic object." [SMI-76].
The here referenced article is primarily concerned with generalization.
The term "generalization" is said to be used in the following way: "a generalization is an abstraction which enables a class of individual objects to be thought of generically as a single named object". [SMI-76].
However, examples and discussions of "generic objects" indicate an interpretation of "generic object" not as a single object but as a class/set of individual objects.

Attributes of generic objects are described as information which "summarizes" attributes of the individual objects being generalized. Two examples of attributes of generic objects are described. The first example concerns the single generic object "dog". "Since all dogs have "sharp teeth" and "four legs", this information can be attached as an attribute of the generic object dog. This information could be attached redundantly as attributes of individual dogs. However, this disguises the fact that dogs in general have these attributes rather than the individuals mentioned". [SMI-76].
This example describes very clearly the concepts of a single generic object and of attributes of single generic objects. In the second example however, this description does not seem quite as clear.
"As another example, we might generalize a class of truck-drivers into the generic object "trucker". We may not be interested in the pay-rate of each individual truck-driver, but only in the average (maximum, minimum) pay-rate of truck-drivers in general. This pay-rate information belongs as an attribute of the generic object "trucker" [SMI-76].
This second example indicates that "trucker" is seen not as a single generic object, but as a class/set of individual objects. According to the first example, an attribute of a generic object could be redundantly attached to the individual objects being generalized. This however, does not seem applicable in the second example. An individual trucker has no average pay-rate. Also, average pay-rate can hardly be regarded as something that a trucker in general has. Attributes corresponding to an average, a maximum, or a minimum are typical examples of attributes of classes/sets of objects rather than of single generic objects.

A basic construct in the design approach is a hierarchy of generic objects. An example from [SMI-76] of a generic hierarchy is shown in fig 4.22.

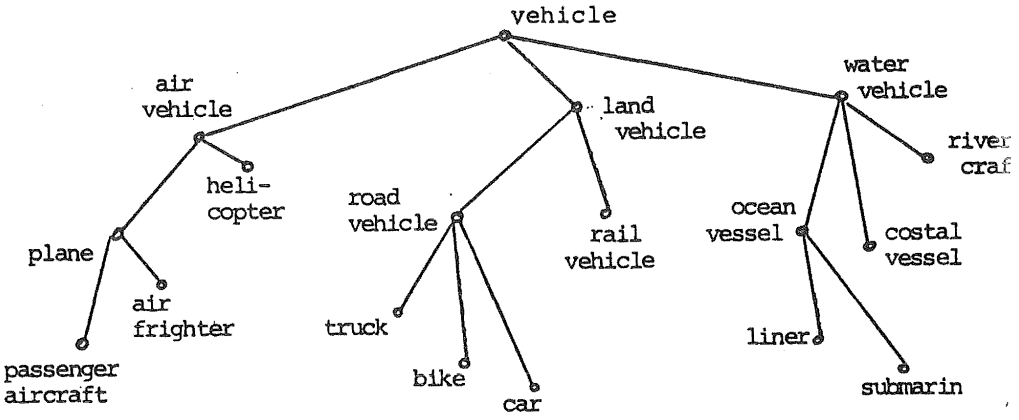


Fig. 4.22

The interpretation of generic hierarchies as described by [SMI-76] is not quite clear.
In a discussion of properties of generic hierarchies the hierarchy in fig 4.22 is said to indicate, for example, that "truck", "bike" and "car" are generalized to the notion of "road vehicle" that "road vehicle" and "rail vehicle" are generalized to the notion of "land vehicle" e.t.c.
This description seems to indicate an interpretation where the nodes (in the graphical representation) denote single, generic objects and where the edges denote "is" relations, as for example
a "truck" is a "road vehicle" a "bike" is a "road vehicle" a "road vehicle" is a "land vehicle" etc.
An implication of this interpretation - as understood in this study - would be that all properties of a generic object are valid for all related objects at lower levels. For example, referring to fig 4.22, all properties of a "vehicle" are valid for a "river craft" as well as for an "air frighter".
For each node, at lower levels in the hierarchy further attributes may be associated. These attributes need not be valid for nodes at higher levels but again, must be valid for each node at lower levels in the hierarchy. For example, an "air vehicle" may have attributes which are not valid for a "vehicle", but which are valid for a "plane", a "helicopter", a "passenger aircraft" and an "air freighter".

However, often in the article [SMI-76] the generic objects are referred to as classes of individual objects. For example, "In general, an individual object will have more relevant attributes the lower the generic level of the class in which it appears. We will call the attributes of an individual object that are relevant to a class G the G-attributes of that object". This implies another interpretation of a "generic hierarchy". In this other intepretation the nodes (in a representation of a hierarchy) will denote classes of objects and the edges may denote "sub-set" associations. For example, the node named "vehicle" may be interpreted as the class of vehicles the node named "land vehicle" as the class of land vehicles and the association between these classes as denoting a subset relation i.e. that the elements of the class "land vehicle" is a subset of the elements of the class "vehicle". The interpretation of the nodes as classes of objects implies that the rules for "inheritage" of attributes along the hierarchy will be more complicated. For example, a distinction between attributes of elements of classes and attributes of the classes themselves ought to be made. Attributes of a class will not be valid for elements in subclasses and attributes of elements of a class will not be valid for the subclasses themselves, as examplified above by "trucker" and "average pay-rate".

Relations are used to represent classes of individual objects. A class of objects is said to be defined by a generalization abstraction and by an aggregation abstraction.
A conceptual data structure is expressed in terms of intentions of relations. An intention of a relation defines explicitly an object. In this definition, names of other objects being aggregated to the defined object as well as names of other objects being generalized to the defined object are explicitly described. It is required that the object being defined by the intention of a relation is explicitly named. A consequence of defining an object in terms of other objects being generalized and to represent this in terms of relations is that names of relations will appear as values in domains.

In the design of a conceptual data structure a systematical, stepwise decomposition of objects is proposed. Generalization and aggregation are considered as independent of each other and the decomposition of objects is carried out alternatively in a generalization and an aggregation plane. Decompositon in the generalization plane consists of identification of objects being generalized to the decomposed object, i.e.

identificaion of objects at the immediate lower level in a generic hierarchy. Decomposition along the aggregation plane consists of identification of relevant attributes of the object being decomposed.

In relation to our frame of reference, Smith and Smith are concerned with design of a global user view and with representation of this user view in terms of relations as a conceptual data structure. The user view is expressed in terms of generic and aggregate objects structured into a generic hierarchy. The concepts of generic objects and generic hierarchies are used as means to achieve a systematical approach to the process of designing a conceptual structure, and as means to achieve semantically meaningful relations in a relational data base.

4.9 Sølvberg

Sølvberg's appraoch consists of specifications of models and relationships between models within "logical" data base design.

A characteristic property of Solvberg's approach is the distinction between phenomena and information about phenomena.
Real world phenomena are modeled in a "phenomena model". Properties of phenomena are modeled in a "descriptor structure". Information requirements, i.e. information about phenomena are modeled in an "information structure". The "information structure" is a part of an information system. An information system consists of an "information structure" and a "process structure". An information system specification, i.e. specifications of an information- and a process structure, constitutes the base for design of a "logical" data structure and a program structure for a data base system. This design process is partitioned into two phases "cracking" and "synthezis". In the "cracking" phase, the information - and process structures specified for an information system are decomposed to the smallest selfcontained units of information and processing. These smallest selfcontained units are called (as in [LAN-66]) "elementary files" and "elementary processes". The result of the "cracking" process is an "elementary system". In the "synthezis" phase, the elements of the "elementary system" are synthezised into a "logical" data structure and a program structure of a data base system. In the "synthezis" phase computer efficiency is

considered. The models and their relationships within "logical" data base design are illustrated as in fig 4.23.

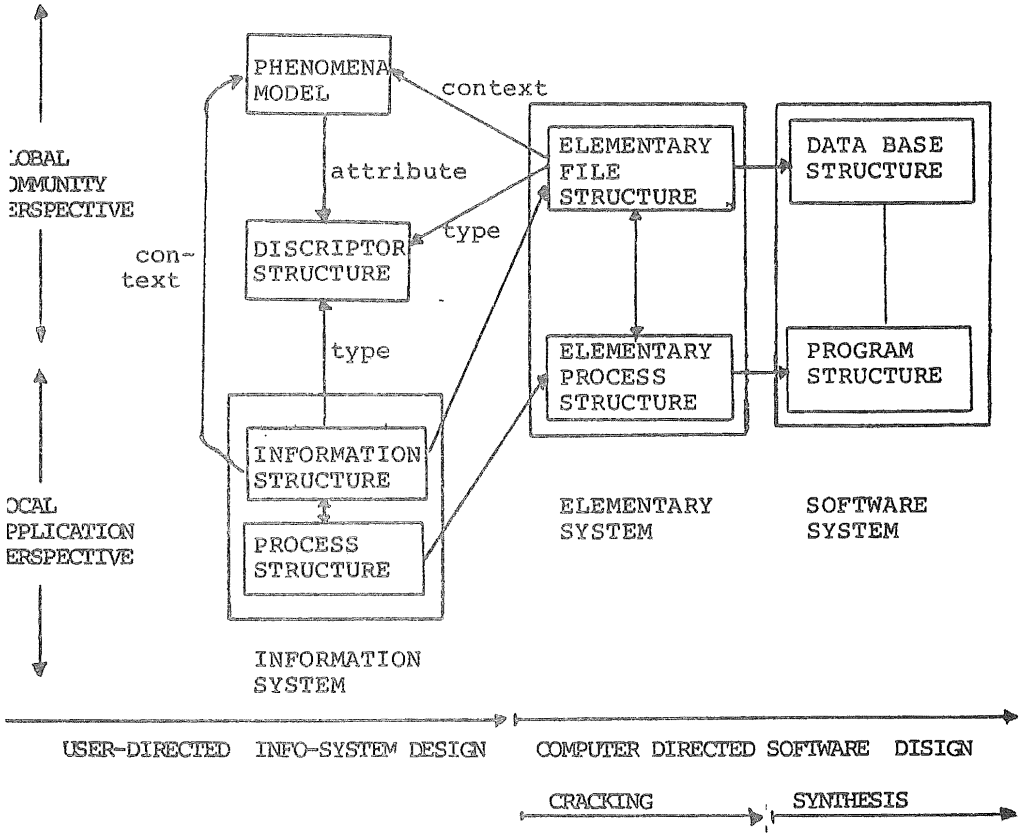


Fig. 4.23.

The "phenomena model" and the descriptor structure" together correspond to what in this study is called a global user view. Modeling concepts proposed for the "Phenomena model" are "entitysets" and "connections".
An "entityset" is a mathematical set. An "entityset" corresponds to the result of abstracting and classifying real world phenomena.
Entitysets may in turn be abstracted and classified into higher order entitysets. Higher order entitysets may be viewed as n-ary relations between the entity sets being abstracted and classified. No special modeling concept is provided for higher order abstractions and classifications except for the special case when the higher order abstraction corresponds to a binary relation. In this case the modeling concept "connection" is used. A connection is a - not necessarily named - binary relation between entitysets.

Detailed descriptions of quantitative properties of entitysets and connections are required in the specification of the phenomena model.
A distinction is made between phenomena, and data which convey information about properties of phenomena. Property values are classified in value sets. A value set is called a "descriptor". A specification of a descriptor includes specifications of how the values of the value set are represented. The descriptor structure consists of specifications of descriptors.

Descriptors of the descriptor structure are related to entitysets and connections of the phenomena model by attribute and/or identifier relationships.
The distinction between values sets - descriptors - and entitysets is not obvious. One difference seems to be that descriptors are described at a name-based level while entitysets are not.
Also, and consequently, the distinction between a connection between two entitysets and an attribute relation between an entityset and a descriptor is not obvious.

An information system is described as consisting of information structures and information handling processes.
An information structure is a tree structure, expressed in terms of information sets, groups and items. The elements of an information structure are related to elements of the phenomena model by "contextual" relationships, and to elements of the descriptor structure by "type" relationships.
The process structures describes processing requirements - relative a given information structure - of an application. Processes are related to elements of an information structure by relations as "target", "result" etc.
As seen in this study, the base for specification of a data- and process structure is knowledge about inference relationships between statements. A data- and process structure represents one - out of several - possible data processing systems by which statements required as output can be produced from statements specified as input. This implies, that data- and process structures can not be specified unless inference relationships are known. In Solvberg's approach a data- and process structure represents the data and processing requirements of an application. As seen in our frame of reference, an application or a user does not have requirements of processes, only of information/data. Identification of processes and process structures is not seen as a result of an analysis of information/data requirements, but a result of a data processing system design procedure. This design procedure requires as input, knowledge about inference relations.
In Solvberg's approach, the inference relationships on which the information - and process structures are based are not explicitly described.

In order to design a data structure and a process structure of a data base systems, the application dependent information - and process structures are first decomposed to "elementary files" and "elementary processes". These elementary elements correspond to the smallest selfcontained units of data and of processes. The result of the decomposition ("cracking") is an "elementary system". In the synthezis phase, elements of the "elementary system" are grouped into a "logical" data structure and a program structure. In the synthezis phase, computer efficiency is considered.
Though not elaborated, the synthezis phase indicates, as in our frame of reference, the mutual dependency between the logical structure of a data base and the process structure of the data base system.

4.10 Summary

In this chapter, a number of approaches to conceptual level data base design have been informally analyzed. The purpose of the analysis has been to identify characteristic semantical properties/problems in the different approaches.
Each method approach has first been described in terms of models and transitions between models. These descriptions are intended to indicate which problem areas within conceptual level data base design that are primarily attacked by each method. The reason for this is the lack of a common understanding of the scope and content of "conceptual level data base design".
After identification of models and transitions, characteristic semantical properties/problems within models and/or transitions of a method approach are identified. The purpose of the identification of semantical properties/problems is not to compare the different approaches but to identify semantical properties/problems within conceptual level data base design. In the next chapter, the analysis of the different method approaches will be summarized and discussed.

5. SUMMARY AND DISCUSSION.

In this chapter, the analysis of different methods for conceptual level data base design is summarized. Characteristic semantical properties and problems are pointed out and discussed.

5.1 Scope of conceptual level data base design.

One of the problems in comparison of (conceptual level) data models and design methods is the difference in their purpose. The purpose of data models and/or design methods are often summarily presented in very general terms. The purpose of a data model may be to provide concepts in terms of which an enterprise or a problem can be modelled. The purpose of a method or of a step within a method may be to decompose, analyze and describe components and their inter relationships etc. Still, however, the purpose of the Relational Data Model and the purpose of a Semantical Network Model does not seem to be the same. The purpose of "inference analysis" and of "normalization" is not the same.

The purpose of this study is not primarily to compare different approaches to conceptual level data base design, but to identify relevant semantical properties and problems within design methods. However, the problem of different purposes for different method approaches is apparent also in our attempt to identify relevant semantical properties and problems.

The approach taken in this study is to identify the problems primarily beeing attacked by each method. In the frame of reference (presented in chapter 3) the scope of conceptual level data base design was informally defined as five problem areas. In the description of these problem areas different steps within a design procedure were described. In summary, problem areas and design steps within conceptual level data base design can be described as:

- concepts for description of users' views
- integration of users' views into a global user view
- concepts for description of information requirements/ /statements
- design and analysis of an information structure
- crude data base system design, i.e. design of a file and process strucuture
- concepts for description of conceptual data structures

- design and analysis of a conceptual data structure for a data base.

In the analysis of different methods (see chapter 4), models and transitions between models within each method were identified. Below, these models and transitions will be related to the problem areas and design steps (described in chapter 3) summarized above. The result describes, for each method, the problem area(s) primarily attacked.

In Benci's et.al. approach, the starting point for the design process is local users' views. The first step in the design procedure is to express these local users' views as a common "real world perception", for which a set of basic modeling concepts is proposed. As interpreted here, the integration of the local users views is not considered. The "real world perception" is the base for design of a "conceptual organization". The "conceptual organization" consists of a data structure, a set of integrity constraints and a set of evolution rules. Concepts for expressing a data structure are proposed (an extended version of the Relational Data Model). Integrity rules and evolution rules are exemplified. The process of designing a conceptual organization from a "real world perception" is not described. However, the evolution rules indicate that the conceptual data structure is seen as a part of a data base system consisting of a data- and a process structure.

Bernstein's design approach starts with a global user view of real world phenomena, expressed in terms of attributes and functional dependencies. The problem attacked by Bernstein is to group attributes into a set of 3NF Relations. As concepts for the conceptual data structure, the Relational Data Model is assumed. Referring to problem areas and design steps in our frame of reference, this approach is regarded as proposing a set of concepts for expressing a global user view and is concerned with design of a conceptual data structure.

Brodie and Tsichritzis are here interpreted as primarily concerned with specification and integration of local users views. Modeling concepts for descriptions of local views as well as of a global user view are proposed. When the global user view is described in terms of abstract data types it constitutes a conceptual data structure.

Bubenko's design method is here seen as a method for design of an information structure. The most important problems attacked by the IAM method are formal specificaion of information requirements and systematical analysis of inference relationships between abstract objects, i.e. between elements of information requirements. Concepts for description of users views are here regarded as indirectely considered by the modeling concepts proposed for information requirements.

Hubbard and Raver are concerned primarily with integration of users' views into a global view. The global user view is intended to be "nonredundant" and in order to acchieve this, implied associations between attributes are identified and analyzed. The resulting global view or "associative model" is intended as a base for design of "logical data" structures.

Kahn's approach to "logical" data base design is here considered as primarily concerned with design of a global user view. The global user view is seen as a result of integration of local views. The global user view describes the information to be represented as data in a data base. The global user view is said to be independent of any application. The processing requirements of different applications, however, determine the grouping and organization of the data in the data base. The processing requirements (the so called usage perspective) is here interpreted as corresponding to a process structure.

Sheppard-Rund's design approach is different from all the other methods here studied in that it is not concerned with views of real world phenomena. The initial steps in Sheppard-Rund's approach can be interpreted as concerned with identification of information required by different functions and tasks within an enterprise. Identification of information requirements is, in this study, regarded as an important task within information systems design, but not as a problem specific to data base design. The result of these initial steps, however, can be interpreted as a data- and process structure. The parts of Sheppard-Rund's approach, relevant to this study, concern the determination of keys and attributes among data elements and the assignment of attributes to keys. This part of Sheppard-Rund's approach is here regarded as concerning design of a conceptual level data structure.

Smith and Smith's design approach is concerned with design of a conceptual data structure. The conceptual

data structure is expressed in terms of an extended version of the Relational Data Model. Though not expressed by Smith and Smith in this way, we here interpret generic hierarchies as a concept for description of a user view.

Sølvberg is primarily concerned with concepts and notations for different models identified within "logical" data base design. Relationships between elements of the different models are also identified and described. Thus, Sølvbergs approach is not primarily concerned with the process of logical data base design. Concepts and notation for description of a phenomena model and of a descriptor structure which correspond to a global user view are proposed. Also proposed are concepts and notation for description of information- and processing requirements. A procedure for design of a data and process structure of a data base system is outlined. This procedure consists of the two main steps: decomposition ("cracking") and synthesis of information and process requirements.

The problems attacked by the different approaches to conceptual level data base design are summarized in fig 5.1. It should be noticed that the purpose of the table in fig 5.1 is not to "evaluate" the different methods or to define any specific method in terms of problem areas concerned. The informal analysis on which fig. 5.1 is based is not exact enough for such purposes. The intention is to show that also within a seemingly narrow area as "conceptual level data base design" different method approaches may be concerned with varying problems, which in turn causes problems in an attempt to identify properties which are common to and/or specific of the different methods.

	Benci et.al.	Bernstein	Brodie et.al.	Bubenko	Hubbard Raver	Kahn	Sheppard-Rund	Smith & Smith	Sølvberg
concepts for description of views	X	X	X	(X)	X	X		X	X
integration of views			X		X	X			
concepts for description of information requirements				X					X
design and analysis of an information structure				X					
file- and process structure design	(X)					(X)	X		X
concepts for the conceptual data structure	X		X					X	
design of a conceptual data structure		X					X	X	

Fig. 5.1

As seen in this study, an important distinction between different approaches concerns the scope of the input to the method. For some methods, the input, i.e. a set of local views or a global view corresponds to the information to be contained in a data base. The problems primarily attacked concern representation of elements of the view(s), grouping of elements and eventually analysis and elimination of "redundant" elements. Examples of this type of methods are the approaches by Bernstein, Brodie and Tsichritzis, Hubbard and Raver, and Smith and Smith. For other methods, the input in terms of views as well as information requirements corresponds to information to be contained in a data base system. In this case, the determinaton of which information to represent in the data base is not assumed to have been made in advance, but is a task of the design process. Characteristic of such methods are problems concerning inference relationships, the temporal dimension and data- and process structures. As methods of this type cover many more and diverse problems, these methods are less uniform.

5.2 Views and information requirements

All except one of the methods, informally analyzed, require as input "a global user view" (or a set of local views) i.e. models of real world phenomena. However, the concepts in terms of which these views or models are expressed vary among the methods.
In all approaches, the concept of "entity" can be identified. In some approaches, a distinction is made between an entity and a value. Fig 5.2 describes for each method its name for the entity concept and eventually its name for a value.

	Brodie	Bern-stein	Brodie Tsichrit-zis	Bubenko	Hubbard Raver	Kahn	Smith & Smith	Sølvberg
tity	object	attribute	object	entity	data item	entity	object	entity
lue	property value	value			value	property value	attribute value	property value

Fig 5.2

Users views are expressed in terms of abstractions of entities/values, and associations between these abstractions.
An abstraction can be described by its connotation and by its denotation. The connotation corresponds to the intention of the abstraction while denotaton corresponds to the extention of the abstraction. In some apporaches, elements of user views clearly describe the denotations of abstractions. In other approaches it is not obvious whether elements of a view correspond to denotation or connotation of abstractions. Names of abstractions and their descriptions are summarized in fig. 5.3.

abstraction	Benci et.al.	Bern-stein	Brodie Tsichrit-zis	Bubenko	Hubbard Raver	Kahn	Smith & Smith	Sølvber
connotation			object			entity property	generic object attribute	
denotation	object class property value set	attribute		concept class	data item			entityse property value se

Fig. 5.3

When elements of views correspond to denotations of abstractions, the denoted elements may be entities or names of entites. For example, in Bernstein's and Hubbard and Raver's approaches only names of entities are denoted. This implies that decisons on how to represent entities (in terms of names) is assumed to have been made before the conceptual level data base design starts. In other approaches, decisions on representation of entities (in terms of names) are considered as part of the conceptual level data base design.

The associations between abstractions of entities/values in the "views" vary between the methods. Two crude types of associations are here identified:
- "semantical" associations which models real world phenomena.
- "structural" associations which models structural relationships between abstractions.

The types of associations used in the "views" of the methods are summarized in fig. 5.4.

sociations	Benci et.al.	Bern-stein	Brodie Tsichrit-zis	Bubenko	Hubbard Raver	Kahn	Smith & Smith	Sølvberg
semantical"	X	X		X	X	X		X
tructural"			X				X	

Fig. 5.4

However, also among "semantical" associations as well as among "structural" associations there are differences. For example, the structural associations in Brodie and Tsichritzis' approach correspond to "component" relationships while structural associations of Smiths and Smiths approach (seem to) correspond to "is" relations. The "semantical" associaions can be further chategorized according to:

i) - explicite or implicite semantics
ii) - functionality

By explicit semantics is meant that the associations are named. Implicit semantical associations are not named. The use of implicit semantical associations implies that whenever more than one association exists between the same abstractions these associations are considered as semantically equivalent.
The use of explict and implicit semantical associations is summarized in fig. 5.5.

semantics	Benci	Bernstein	Bubenko	Hubbard Raver	Sølvberg
explicite	X		X	X	X
implicite		X		X	

Fig. 5.5

That associations correspond to "functions" is required by some approaches as summarized in fig. 5.6.

	Benci	Bernstein	Bubenko	Hubbard Raver	Sølvberg
functionality required		X	X		
functionality not required	X			X	X

Fig. 5.6

The concepts "view" and "information requirements" can be regarded in two perspectives.
In the first perspective (which is the most frequently used perspective within the data base area) a (global) "view" corresponds to a model of some part of the real world. Elements of the model correspond to abstractions, generalizations and classifications of real world phenomena. The purpose of the model is to specify types/classes of objects and associations which are to be represented in the data base. The data base is seen as containing representations of objects, values and associations. In this perspective, information requirements are seen as specifications of subsets of the objects, values and associations of the (global) view. Information requirements are specified by the same concepts as the (global) view. Analysis of information requirements aims at verifying that objects, values and associations specified in the requirements do exist or have correspondance to objects, values and associations in the (global) view. In this perspective, conceptual level (as well as "logical") data base design is primarily concerned with grouping/structuring of the elements of the (global) view into modeling concepts of the data model used by the data base management system. Criteria for this grouping/structuring seem to be of two kinds; 1) frequencies at which objects, values and associations are required for update and retrieval (as for example in Kahn's approach) or 2) avoiding of insertion, deletion or update anomalies (as for example in Bernstein's approach). When the first kind of criteria is used, the goal seems to be minimization of data storage and access. The second kind of criteria indicates a goal in which "semantically" simple and/or independent units are searched for.

Semantical integrity in the data base refers to restrictions on associations between objects represented in the data base or to restrictions on values of attributes of objects represented in the data base. In some cases (as for example in Benci's et.al. approach) relationships between attribute values may also be specified as semantical integrity constraints.
In this perspective, there is no difference between a (global) user view and an information structure.

In the second perspective, the global user view is also seen as a model of some part of the real world. Also, elements of the model are seen as abstractions, generalizations and classifications of real world phenomena. However, the purpose of this model is to constitute a "semantical context" of the information represented in a data base system. A distinction is made between information and the "things" being informed about. The data base is seen as containing representations of (units of) information. These (units of) information refer to or inform about elements of the (global) "view".
In this perspective, information requirements specify (units of) information required by some user. The concepts in terms of which information requirements are expressed are different from the concepts used to express the (global) "view". Specification of information requirements corresponds to specification of units of information. Examples of units of information are "messages" and "statements". An "abstract-object" as suggested by Bubenko may also be interpreted as an unit of information.
Units of information may be related in different ways. For example, they may be semantically equivalent, they may imply other units of information, they may be inferred from each other, etc.
In this perspective, an information structure is a specification of units of information and their interrelationships. In the design of a conceptual data structure, information about properties of units of information required as output and specified as input are needed. Important properties are the frequencies at which the units of information are required or supplied, and the actuality required for output information.
Thus, in this second perspective, there is a difference between a (global) "view" and an information structure, and views and information requirements are expressed in terms of different concepts.

Among the design approaches included in this study, only the ones by Bubenko and Sølvberg can be seen as representatives of the second perspective.
The difference in perspective on "views" and information

requirements will have an impact on aspects such as "inference", "redundancy" and "temporal dimension".

5.3 Inference

Inference relationships are discussed and/or used within different phases in conceptual level data base design depending on the perspective of "views" and "information requirements".

For approaches representing the first perspective, inference relationships seem to appear primarily in the conceptual data structure design phase i.e. when elements of the (global) view are to be grouped according to modeling concepts used for the conceptaul data structure. Inference relationships also appear in specification of semantical integrity constraints for the data base.
In this first perspective, inference relationships refer to relationships between attribute/property values, represented in the data base.

In the second perspective, inference relationships are analyzed and described in the design/specification of the information structure. In this case, the inference relationships refer to relationships between units of information.

Indenpendent of perspecitve, three different kinds of inference relationships can be identified in the here studied design methods. The three kinds may be called:

i) - calculation
ii) - inheritance
iii) - implication

Caluclation refers to inference relationships where a value is calculated from other values. For example, "average salary of employee" may be calculated from a set of "salary of an employee" and "number of employees".

Inheritance relationships refer to the case when attributes/properties of an objects in a generic hierarchy are inherited by objects at lower levels in the hierarchy (for example see fig 3.18).

Implication refers to inference relationships between associations in cases where an association between two

sets of entities/values is implied by other associations (for example see fig 3.19).

In Benci's et.al. design approach, inference relationships of the kind "calculation" appear as semantical integrity contraints on values of data in the data base. From the presentation of Benci's approach, inference relationships do not seem to be discussed at the "percieved real world" level, i.e. in the real world model.

Bernstein is concerned with inference of the "implication" kind, and the algorithm presented is based on this kind of inference. In his approach, the inference relationships are used in the design of the conceptual data structure. However, as pointed out (in section 4.2) in Bernstein's approach, the semantics of inferred associations is not specified. Other kinds of inference are not considered in Bernstein's approach.

In Brodie and Tsichritzis' approach, inference relationships are not discussed. The formulation of views indicate that inference relationships of the kinds "inheritance" and "implication" are not applicable. Relationships of the kind "calculation" are not explicitely mentioned, but are applicable in specifications of semantical integrity constraints, i.e. as constraints on data values.

In Bubenko's IAM-method inference relationships of the kinds "calculation" and "implication" are applied. In this approach, both these kinds of inference are analyzed and specified between "abstract objects", i.e. between units of information. In this case the determination and specification of inference relationships is a part of the information structure design and specification process.

Hubbard and Raver's design approach is based on "implication" relationships. Here, these inference relationships are used in the design of an "associative network". In Hubbard and Raver's approach, as distinct from Bernstein's, the semantics of the inferred associations is not autmatically considered as relevant, but has to be checked by the data base designer. Other kinds of inference are not considered in this approach.

In Kahn's approach, inference relationships of the kind "calculation" are considered. In the specification properties (of entities) "derived" properties are

identified. Specification of derived properties corresponds to identificaion of what here is called "calculation" relationships. Inference of the other kinds are not considered.

Smith and Smith's approach is the one in which "inheritance" relationships are identified. This is the only approach, in this study, in which this kind of inference is discussed. From a semantical point of view this kind of inference seems powerful. It originates from ideas within the AI field and has, strangely enough, not been adopted in any of the other methods of this study. In Smith and Smith's approach, inference relationships exist between properties of objects in a generic hierarchy and the inference relationships are specified in the (global) user view. Other kinds of inference are not considered.
As Sølvberg's approach is primarily concerned with identification of models within "logical" data base design and with specification of formal notations for these models, analysis of inference relationships as a part of a design procedure is not applicable. However, in the specification of the phenomena model, a possibility to define connections in terms of other connections is indicated. Such a definition may correspond to specification of "implication" relationships.

In summary, inference relationships are treated very differently within the here studied approaches to conceptual level data base design. Depending of the perspective of user views and information requirements, inference relationships may be applicable either to "entities" and "property values" or to "statements" or units of information.

5.4. Redundancy

In several of the here analyzed methods, the conceptual (or logical) data structure which is the result of applying the method, is said to be "non-redundant". In many cases, the concept "non-redundancy" is not described in detail, and its interpretation is not obvious.

The requirement of "non-redundancy" appears for different models within the conceptual level data base design, depending on the perspective of users views and information requirements.
In approaches representing the perspective where

information requirements are seen as subsets of a "global view" and where the data base is seen as containing representations of objects, values and associations, non-redundancy is required also in the global view. Often, in this case, integration of local views into a global view concerns elimination of redundant elements. Characteritic of the perspective in which information requirements are seen as units of information referring to elements in a global view, is that an information structure is redundant in the sense that units of information specified in the information structure may imply or be inferred by other units of information.

In Benci's approach, redundancy is assumed to exist in the "real world perception" model, in the sence that a class of entities or a type of property appears only once in the model. Redundancy or non-redundancy in the conceptual data structure is not discussed.

A conceptual schema designed by application of Bernstein's algorithm contains a minimal number of 3NF relations. In the synthesis of functional dependencies into 3NF relations, extraneous attributes as well as transitive dependencies are eliminated. The conceptual schema can be regarded as nonredundant with respect to associations - functional dependencies - which can be (syntactically) inferred from other associations.

In Brodie and Tsichritzis' approach, there is no requirement of non-redundancy in the global view/conceptual data structure. On the contrary, a characteristic idea of their approach is to permit different (and overlapping) users views to coexist. However, in combination of hierarchies of objects into a common schema, redundant decompositions, i.e. decompositions in terms of "component objects" which are common to two (or more) objects are eliminated.

In Bubenko's approach, an information structure which is the result of applying the IAM-method, is explictly described as being redundant. "Redundancy exists in the sense that some information "elements" can be inferred from others" [BUB-76A]. In this case redundancy refers to units of information and to inference of the types "implication" and "calculation".

In Hubbard and Raver's approach, an associative network which is the result of applying (a part of) their method, is said to be "non-redundant". In this case, non-redundancy refers to elimination of associations

which are implied by - can be inferred from - other associations.

In Kahn's approach, the entity diagram which correspond to a representation of a global "view" is explicitely requiered to be "non-redundant". Requirements of non-redundancy are expressed as

"- each entity must appear only once in the diagram (entity non-redundancy)
- each relationship must appear only once in the diagram (relationship non-redundancy)". [NOV-76B].

Concerning redundancy of properties, it is clear that non-redundancy is aimed at. How this non-redundancy is acchieved is however not quite obvious. "All elements that belong to more than one entity (i.e. are included in more than one entity definition) should be determined. The information structure perspective should be as non-redundant as possible; therefore, a given element should only occur in one entity. To minimize the number of unique elements represented in the information structure perspective, all relational and derived properties should be separated from their associated entites". [KAH-76B].

In Sheppard-Rund's approach, non-redundancy is considered for data elements required by the different tasks. "In order to thoroughly analyze the flow charts developed during the interviews, it is necessary to develop a list of unique data elements and definitions. The problem most often experienced creating such a list is trying to determine which data elements are or are not redundant" [SHE-76]. The meaning of non-redundancy of data elements is however not discussed. As, in this approach, inference relationships between data elements are not considered redundancy may refer to different names for "the same" things.

In Smith and Smith's design approach, the conceptual data structure is not required to be "non-redundant". "It is clear that a great deal of information occurrs redundantly in a relation hierarchy. This is perfectly acceptable provided there is some way to implement a relation hierarchy such that

i) storage space is not wasted due to data duplication
ii) consistency of redundant information can be maintained" [SMI-76].

The redundancy of a relational hierarchy makes it possible for different users to access the data base at different levels of abstraction and thus to allow coexisting user views.

In Sølvberg's approach, an information structure is explicitly described as being redundant in the sense that information objects may represent the same information. One of the purposes of decomposition ("cracking") of an information structure is to identify and eliminate such redundancy, so that the input to the synthesis phase is a non-redundant specification of information to be contained in the data base.

As can be seen from the above descriptions, requirements on "redundancy" or "non-redundancy" vary among the here studied design methods.
Characteristic is that methods, representing the perspective of views and information requirements where information requirements correspond to subsets of a global view, require non-redundancy in the global user view. This is natural, as non-redundancy is required in the conceptual data structure, and as the conceptual data structure is seen as a representation of the global user view.
Methods representing the perspective where information requirements correspond to units of information referring to elements of a global view characteristically do not require "non-redundancy" in the information structure.

From a semantical point of view, the meaning of "redundancy" is not clear. For example, suppose "non-redundancy" is required in a global user view in the sence that unique names are required for classes of entities/values and for associations. Does this imply that the "same" real world phenomena may be abstracted and classified as an element of only one class of entities/values/associations, or may a real world phenomenon be an element of several classes? If disjoint classes of phenomena are not required, does this imply redundancy? This kind of redundancy is not discussed in any of the method approaches discussed in this study.

"Non-redundancy" in a global user view means different things in the different approaches. Characteristically, "non-redundancy" is related to the type(s) of inference considered by the methods, i.e. what is regarded as redundant depends on the type(s) of inference identified/used in the design method. For example, in Bernstein's and Hubbard and Raver's approaches, redundancy refers to associations which are implied by other associations. In these two approaches, inference relationships between attributes/data items corresponding to "calculation", i.e. that an attribute/data item value can be calculated from other attribute/data item values is not considered as redundancy.

Disregarding the fact that "non-redundancy" means different things in different approaches, none of the approaches requiring "non-redundancy" discusses or indicates advantages of this "non-redundancy".
As seen in this study, "non-redundancy" in a global user view as described by several of the here studied methods, restricts the possibility to allow different users to have alternative views of real world phenomena. The alternative views allowed by a "non-redundant" global view concern only alternative, eventually overlapping, subsets of the elements of the global view. Redundancy in a global user view may allow users not only to specify subsets but also to specify alternative views of the real world phenomenon. An example of this possibility is described in Smith and Smith's approach, where "non-redundancy" is not required. "There are two ways to view a generic hierarchy. We will call one way a "levelled view" and the other way a "direct view". In a levelled view one thinks of a generic object as being a generalization of the class which contains its immediate descendants. For example, "water vehicle" is viewed as a generalization of the class {ocean vessel, costal vessel, river craft}; and "road vehicle" is viewed as a generalization of the class {truck, bike, car}. In a direct view one thinks of a generic object as beeing a generalization of the class which contains all its descendant individuals. For example, "water vehicle" is viewed as a generalization of the class containing all individual liners, submarines, coastal vessels, caonoes and sailboats: and "road vehicle" is viewed as a generalization of the class containing all individual trucks, bikes and cars." [SMI-76]. This example (which refers to fig 4.22 in our study) describes very clearly the possibility to allow different users to view "the same" real world phenomena in different ways.

5.5 A temporal dimension

A temporal dimension in conceptual data modeling is explicitly considered in two of the methods included in this study. The two methods are the one presented by Benci et.al. and the IAM-method presented by Bubenko.

In Benci's approach, two aspects of a temporal dimension can be identified. One aspect concerns a temporal dimension of objects and properties. The other aspcet concerns events and evolution of the data base.
The temporal dimension of objects and properties is here interpreted as related to the existence of objects and to the need of qualifying some properties by time references. The qualification of properties may be achieved by the introduction of domains corresponding to

sets of time points or time intervals in the relations of the conceptual data structure. The existence of objects seems to be represented by the existence of n-tuples in the data base.
The other aspect of the temporal dimension in Benci's approach concerns the evolution of the data base. The evolution is described as depending on events external to the data base i.e. events in the organization as well as on events internal to the data base. Events are modelled by evolution rules specified in the "conceptual organization".

In Bubenko's approach, the temporal dimension relates to a "time-unrestricted" perspective of information requirements and of the information structure.
A "global user view" is, in Bubenko's approach, built up by "concept-classes" which correspond to classes of abstract entities and to property value sets. Time points and time intervalls are considered as entities. Entities are described by their associations to other entities or values. From an application point of view, some associations are considered as "time stable" while other associations are considered as "timevarying". The temporal dimension is described in the following way: "Information about entity associations is represented by abstract objects (AOs). The set of all abstract objects correspond to stated output information requirements and transactions. Abstract objects which represent time-varying associations are related to time-entities. In fact, we assume that every abstract object, which represents a time-varying association is coupled to a time point or interval indicating the time when the (true) assertion about an association was issued. This assumption makes each abstract object's "truth-value" time-invariant once the object has been created". [BUB-76C]

An information structure, as described by Bubenko, consists of abstract objects and inference relationships between these. The information structure represents a "full-time perspective" model as opposed to "snapshot type" models. A "snapshot type" model implicitly or explicitly presumes processes for update/maintenance of the information. In the IAM-method, the transformation of a "full-time perspective" model into a conceptual data structure corresponding to a "snapshot" model is identified as a problem, but is not included as a step within the method.

5.6 Summary

The analysis of a number of design methods shows that these methods are concerned with varying problems within the conceptual level data base design area. In order to discuss the different methods and the problems by which they are concerned, a frame of reference is suggested. In this frame of reference, conceptual level data base design is regarded as consisting of five problem areas. The problem areas are:

- concepts/models for description of user's views
- concepts/models for description of user's information requirements
- design and analysis of an information structure
- concept/models for conceptual data structures
- design and analysis of a conceptual data structure.

The suggested problem areas have been used to describe the problems primarily attacked by the different methods. For eight out of nine methods, the suggested problem areas turned out appropriate for such a description. The method for which the problem areas were not quite relevant - the method by Sheppard-Rund - can be considered as primarily concerned with identification of information requirements, i.e. with a problem which , in this study, is not considered as a data base design problem, but as an information system design problem.

The analysis of the design methods indicates that methods for conceptual level data base design can be regarded as comprising models and transitions between models. When the methods were described as models and transitions two "schools" can be identified. In one "school" models corresponding to a "global user view" and a "conceptual data structure" could be identified. In this case, the data base was regarded as containing representations of objects, properties and associations specified in the "global user view". In this "school" there was no distinction between a "global user view" and an "information structure".
In the second "school" a distinction is made between a model of real world phenomena, i.e. a "global user view" and information about real world phenomena, i.e. an "information structure". The data in a data base is, in this case, regarded as representation of information which refers to elements of the "global user view". This second "school" is represented by the Scandinavians, i.e. in this study by Bubenko and Solvberg.

Determination and use of inference relationships is identified as an important semantical aspect of

conceptual level data base design.
Analysis of inference relationships in the different methods has led to identification of three kinds of inference which have been called:
- calculation
- inheritance
- implication

Most methods are concerned with only one of these kinds of inference. In the Scandinavian "school", inference relationships are considered to exist between units of information. Here, inference relationships are used in the design of an information structure and in the design of a conceptual data structure. In the other "school", inference relationships are considered as existing between elements of a global "user view" as for example between associations or between property values. In some of these methods, inference relationships are used in order to design a "non-redundant" data structure, in other methods the inference relationships appear as semantical integrity constraints.

"Non-redundancy" in the global "user view" and in the conceptual (or logical) data structure is required in some of the design methods. Characteristically, no reasons for the requirement of "non-redundancy" are presented. Further, the meaning of "redundancy" varies among different approaches. What is ment by "redundancy" is often related to the type of inference identified in a design method. In this study, the requirement of "non-redundancy" in models at the conceptual level is questioned.

Very few authors within the area consider a temporal dimension in conceptual level modeling. Most conceptual data structures correspond to "snapshot type" models. A "snapshot" type model preassumes processes for update/maintenance of the model. However, the design of the preassumed process structure is, in most cases, not considered as a part of the conceptual level design process.

In the analysis of a number of design approaches, semantical aspects have been identified. In many cases, semantical problems have been critically discussed. The purpose of these discussions has <u>not</u> been to critisize individual design approaches, but to point out and stress semantical aspects and problems within conceptual level data base design.

6. CONCLUDING REMARKS

In this study, an attempt is made to identify and discuss semantical aspects of conceptual level data base design. A number of methods are analyzed and discussed. The purpose of this analysis has been to identify relevant aspects and problems. The result of the analysis constitutes a base for future developement of design methods and for classification of approaches to conceptual level data base design.

The conceptual level of data base design has only recently been identified and accepted. Since approximately five years , research work within this area has been reported. So far, however, there does not seem to exist any common aggreement on scope and contents of the area. In this study, an attempt is made to informally define the scope of conceptual level data base design in terms of problem areas. The analysis of a number of design approaches has emphasized the need of some aggreement of scope, contents and stucture of the conceptual level problem area. Such an aggreement must aim at faciliating communication between researchers within the area without beeing an obstruction to the expansion of a new field.

The conceptual level design results in a data structure for a data base. Traditionally, syntactical aspects of such specifications have been emphasized. The distinction between syntax and semantics is not obvious. However, the concepual schema level as suggested and described by the ANSI/SPARC Interim Report can be interpreted as emphasising semantics in the information specification. The analysis carried out in this study stresses that further efforts should be dedicated to the semantical area. A semantically oriented terminology would constitute a progress from today's situation.

Within the data base area, semantics has been approached mainly via physical and logical data representation. This is a fairly straightforward line of thought. However, semantics in a data base system can be arrived at from other starting points. The data base area could profite from closer contact with other diciplines as linguistics and psychology. Also, as shown for example by Smith and Smith, research and experience in "knowledge representation" within the AI field can be proved useful.

An important semantical aspect, as seen in this study, concerns "inference". The possibility to infer some information from other information is identified and used in most of the here analyzed methods.
Very different approaches to determination of inference relationships are represented among the methods. Some methods represent a formal approach to determination of inference relationships. The formal approaches can be characterized by their formal definitions of what is inferable and by the possibility to automatize the analysis and determination of inference relationships. However, as seen in this study, formal approaches as represented by methods in this study may lead to semantically unidentified or ambigous results.
Semi-formal approaches are also represented among the methods. An example of what here is regarded as semi-formal is the case when inference is formally defined and identification of possible inference relations may be automatically performed but is complemented by manual decisions. The manual decisions are made in order to assure the semantical relevance of the atomatically identified inference relationships.
Still other methods can be seen as representing intuitive approaches to determination of inference relationships. In this case, there does not exist any formal definitions of what is inferable and the analysis and determination of inference relationships is completely manual.
A central question to the data base design area which may be raised concerns how far it is profitabl or meaningful to formalize and automatize the analysis and determination of inference relationships.

"Non-redundancy" is by some methods required in models at the conceptual level. What is considered as "redundant" is often related to the inference relations identified and used in these methods. In this study, the advantages of "non-redundancy" in a conceptual level model is questioned.

o

User influence in the design of computerized information systems is - at least in Sweden- generally agreed on as essential. So far, it seems as user influence has been emphasized in the determination of information reqirements and in decisions on how to represent information in terms of data and layouts. Information - represented as data and otherwise -

convey abstraction of and assumptions about real world phenomena. As seen in this study, such abstactions of and assumptions about real world phenomena have most often been implicitly built in the data representations without explicitly beeing subjected to user influence. It has been suggested in this study that abstractions and assumptions of real world phenomena underlying the information contents of a data base system should be made explicit. This explicit formulation can be regarded as a prerequisite fo user influence on the semantics of the data in a data base system.

REFERENCES

[ABR-74] Abrial J.R.; "Data Semantics" in [KIM-74] pp 1-60

[ACM-76A] ACM Transactions on Database Systems 1, Volume 1, No. 2, June 1976

[ACM-76B] ACM Transactions on Database Systems, Volume 1, No.4, December 1976

[ACM-76C] ACM Transactions on Database Systems, Volume 1, No.1, March 1976

[ACM-76D] ACM Computing Surveys, Vol.8, No.1, March 1976

[ACM-77] ACM Computing Surveys, Vol.9, No.4, Dec. 1977

[ANS-75] ANSI/X3/SPARC Study Group on Database Management systems, Interim Report, 1975

[ANS-77] ANSI/X3/SPARC DBMS Framework Report of the Study Group on Data Base Management Systems, edited by D. Tsichritzis, A. Klug.

[ARM-74] Armstrong,W.W.; "Dependency structures of data base relationships" Information Processing 1974, North Holland Publ.Co., Amsterdam 1974

[AST-76] Astrahan,M.M., Blasgen,M.W., Chamberlin,D.D., Eswaran,K.P., Gray,J.N., Griffiths,P.P., King,W.F., Lorie,R.A., McJones,P.R., Mehl,J.W., Putzolu,G.R., Traiger,I.L., Wade,B.W., and Watson,V.; "System R: A Relational Approach to Data Base Management" in [ACM-76A].

[AUE-76] Auerdal,E. and Sølvberg,A.; "A multilevel procedure for design of file organizations". Proceedings, National Computer Conference 1977, AFIPS, Vol.46, 1977

[BAD-71] Badiou,A; "Begreppet modell" Libraire François Maspero/Alain Badiou/ Bo Cavefors Bokförlag AB

[BEN-76] Benci,E., Bodart,F., Bogaert,H., Cabanes,A.; "Concepts for the Design of a Conceptual Schema", in [IFP-76]

[BER-76] Bernstein,Ph.A.; "Synthesizing Third Normal Form Relations from Functional Dependencies", in [ACM-76B]

[BIL-76] Biller,H., Neuhold,E.J.; "On the semantics of Data Bases: The Semantics of Data Models", Report Institut für Informatik, Universitat Stuttgart, 1976

[BRA-76] Bracci,G., Paolini,P., Pelagatti,G.; "Binary Logical Assciations in Data Modeling", in [IFP-76]

[BRO-77] Brodie,M.L., Tsichritzis,D.; "Data Base Constraints" in [TSI-77]

[BRO-78] Brodie,M.L.; "Specification and verification of data base semantic integrity"
Technical Report CSRG-91, April 1978, Computer Systems Research Group, University of Toronto

[BUB-76A] Bubenko,J.A.; "IAM - Inferential Abstract Modeling - An Approach to Design of Information Models for large shared data bases"
IBM Thomas J.Watson Research Center, Yorktown Heights, New York 10598, 1976

[BUB-76C] Bubenko,J.A.; "The Temporal Dimension in Information Modeling"
IBM, Thomas J.Watson Research Center, Yorktown Heights, New York 10598, 1976

[BUB-77] Bubenko,J.A.; "Validity and Verification Aspects of Information Modeling" in [VLD-77]

[BUB-79] Bubenko,J.A.; "On the role of 'understanding models' in conceptual schema design"
Gothenburg, Febr. 1979, Unpublished paper

[CHE-76] Chen,P.Pin-Shan; "The Entity-Relationship Model - Toward a Unified View of Data" in [ACM-76C]

[COB-73] Colby,K.M., Schank,R.C. edits; "Computer models of thought and language"
W.H.Freeman and Company, San Fransisco 1973

[COD-70] Codd,E.F. "A relational model of Data for Large Shared Data Banks"
CACM, 13, 1970

[COD-71] Codd,E.F. "Normalized Data Base Structure: A Brief Tutorial"
Proc. ACM SIGFIDET Workshop on Data Description Access and Control 1971

[COD-72] Codd,E.F.; "Further Normalization of the Data Base Relational Model" in Data Base Systems, Courant Computer Science Symposium 6th, Prentice Hall 1972

[COL-62] CODASYL Development Committee, "An Information Algebra, Phase 1 Report"
Communication of the ACM, Vol.5, No.4, April 1962

[COL-69] CODASYL Systems Committee; "Report on the CODASYL Data Base Task Group,
ACM, October 1969

[COL-71] CODASYL Systems Committee; "Report on the CODASYL Data Base Task Group,
ACM, April 1971

[COU-74] Couger,J.D., Knapp,R.W.; "System Analysis Techniques",
John Wiley & Sons, Inc 1974

[CUL-75] Cullinane Corporation: Integrated Database Management System (IDMS). 1975

[DAT-75] Date,C.J.; "An Introduction to Database Systems" Addison-Wesley Publishing Company 1975 (Second edition 1977)

[DOU-75] Douque,B.C., Nijssen,G.M. (eds); "Data Base Description"
North-Holland, Amsterdam 1975

[DEL-77] Deliyanni,A.J.; "A comparative study of semantic networks and predicate logic"
Department of Computing and Control, Computing Science section, Imperial College, University of London

[FAL-75] Falkenberg,E.; "Design and application of a natural language oriented data base language" Advanced Course on Data Base Language Procesing, Freudenstadt, Black Forest, Fed.Republic of Germany, Aug. 1975

[FAG-77B] Fagin,R.; "The Decomposition versus the Synthetic Approach to Relational Data Base Design" in [VLD-77]

[FRY-76] Fry, J.P., Sibley,E.H.; "Evolution of Data-Base Management Systems" in [ACM-76D]

[HOA-72] Hoare,C.A.R.; "Notes on data structuring" APIC Studies in Data Processing No.8: Structural Programming, Academic Press, New York, 1972

[HUB-75] Hubbard,G., Raver,N.; "Automating logical file design" in [KRR-75]

[IBM-73] IBM, IMS/360 Application Description Manual, GH 20-0767, White Plains, N.Y.

[IBM-76] IBM, Program Product, Data Base Design Aid, Designer's Guide, GH 29-1627-1, IBM

[IFP-76] IFIP-TC-2 Working Conference on Modeling in Data Base Management Systems, January 5-9, 1976. Freudenstadt, Fed.Republic of Germany, Preprints

[ISO-78] ISO/TC97/SC5/WG3 working paper: "Concepts for the Conceptual Schema", April 1978

[JOD-68] CODASYL, COBOL Journal of Development, 1968

[JOH-46] Johnsson,W.; "People in Quandaries" Haper & Row Publishers, New York 1946

[KAH-76B] Kahn,B.; "A method for describing information requiremed by the data base design process" University of Michigan, Technical Report 76 D.E. Jan 1976

[KAT-72] Katz,J.; "Semantic Theory" Harper & Row Publishers, New York 1972

[KEN-76] Kent,B.; "New criteria for the conceptual model" in [LOK-76]

[KER-76] Kerchberg,L., Klug,A. and Tsichritzis,D.; "A taxonomy of data models" in [LOK-76]

[KIM-74] Klimbie,J.W. and Koffeman,K.I.; "Data Base management",
Proceedings of the IFIP Working Conference, Corsica 1974, North-Holland/American Elsewier, 1974

[KLI-65] Klir,J. Valach,M.: "Cybernetic Modelling", Iliffe Books Ltd, Dorset House, Stanford Street, London 1975

[KOT-66] Kotarbinski,T.: "Gnosiology The Scientific Approach to the Theory of Knowledge" Pergamon Press Ltd, Headington Hill Hall, Oxford 1966

[KRR-75] Kerr,D.S. (ed); Proceedings of the international conference on Very Large Database, ACM, Framingham, Massachusets, USA, Sept. 1975

[LAN-66] Langefors,B.; "Theoretical analysis of Information Systems",
Studentlitteratur, Auerbach, Lund 1966

[LAN-68] Langefors,B.; "Introduktion till Informationsbehandling, Natur och Kultur, Stockholm 1968

[LAN-75] Langefors,B., Sundgren,B.; "Information Systems Architecture"
Petricelli/Charter, N.Y. 1975

[LOK-76] Lockeman,P.C. and Neyhold,E.J.(eds); "System for large data bases"
Preprints Brussels, Belgium 8-10 sept., 1976
North Holland Publ.Co.

[LOC-77] Lochovsky,F.H.; "User Performance Measures for Data Base Mangement Systems" in [TSI-77]

[MET-75] Metaxides.A.;"'Information bearing' and 'non-information bearing' Sets" in [DOU-75]

[MIN-68] Minsky; "Semantic Information Processing"
The MIT Press, Cambridge, Massachusetts 1968

[MRI-72] MRI Systems Cooperation, System 2000, General Information Manual, Austin, Texas 1972

[NEV-73] Nevell,A.; "Artificial Intelligence and the Concept of Mind" in [COB-73]

[NIJ-76] Nijssen; EDMS Version 1.0, DDL Reference Manual, Control Data 1976

[NOV-76B] Novak,D. Kahn,B.; "A framework for logical data base design"
Working Paper D.E. 3.1, June 1976, University of Michigan

[QUI-68] Quillian,R.M.; "Semantic Memory" in [MIN-68]

[RAP-68] Raphael,B.; "SIR: Semantic Information Retrieval" in [MIN-68]

[REG-58] Regnell,H.; "Semantik"
Scandinavian University Books, Stockholm 1958

[RIV-72] Rivett,P.; "Principles of Model Building"
John Wiley & Sons, Inc. 1972

[ROU-75] Roussopoulos,N., Mylopoulos,J.: "Using Semantic Networks for Data Base Mangement" in [KRR-75]

[ROU-78] Roussopoulos,N.; "CSDL: A conceptual schema definition language for the design of data base applications". Department of Computer Science, University of Texas, Austin , April 1978

[SCH-73] Schank,R.; "Identification of Conceptualizations Underlaying Natural Language" in [COB-73]

[SEN-72] Senko,M.E., Altman,E.B., Astrahan,M.M., Fehder,P.L. and Wang,C.P.; "A Data Indenpendent Architecture Model I: Four levels of description from logical structures to physical search structures"
IBM Research, RJ 982, February 25, 1972

[SEN-75] Senko,M.; "Specification of stored data structures and desired output results in DIAMII with FORAL" in [KRR-75]

[SEN-77] Senko,M.; "Conceptual Schemas, Abstract Data Structures, Enterprise Descriptions".
Preprints, ACM, April 1977

[SHE-76] Sheppard-Rund,D.L.; "Data Base Design Methodology - Part I and II"
Cincom Systems Inc.

[SIM-73] Simmons,R.F.; "Semantic Networks, Their Computation and Use for Understanding English Sentences" in [COB-73]

[SMI-76] Smith,J.M., Smith,D.C.P.; "Data Base Abstractions: Aggregation and Generalization"
Proceedings of the Conference on Data: ABSTRACTION, DEFINITION AND STRUCTURE, Salt Lake City, Utah, March 22-24, 1976

[STO-75] Stonebraker,M.R. and Wong,E.; "INGRESS: A relational data base system", Proceedings of National Computer Conference, Anaheim, California, May 1975

[SUN-73] Sundgren,B. ; "An infological approach to Data Bases"
Ph.D. Dissertation, University of Stockholm, 1973

[SVO-76] Svobodova,L.; "Computer Measurement and Evaluation Methods: Analysis and Applications"
American Elseview Publishing Company Inc. 1976

[SØL-77] Sølvberg,A.; "A Model for specification of phenomena, properties and information structures"
IBM Research Laboratories, San José, California

[THA-77] Thaggert,W.M., Tharp,M.O.; "A Survey of Information Requirements Analysis Techniques"
in [ACM-77]

[TSI-77A] Tsichritzis,D. (ed); "A Panaché of DBMS Ideas"
Computer Systems Research Group, University of Toronto, Technical Report CSRG-77, Febr. 1977

[TSI-77B] Tsichrtizis,D., Lochovsky,F.; "Data Base Management Systems"
Academic Press 1977

[UNI-73] Data Base Management System (DMS-1100) Schema Definition. Data Administrator Reference.
Sperry Rand Cooperation,

[VIA-77] Brandt,P., Gustafsson,M.R., Hohansson,L-Å.; "Att jämföra systemeringsmetoder", Delrapport från VIA-projektet, Inst för Inf.-beh-ADB, Göteborg, 1977

[WEI-76] Weizenbaum,J.; "Computer Power and Human Reason", W.H.Freeman and Company 1976

[WIN-73] Winograd,T.; "A Procedural Model of Language Understanding" in[COB-73]

[WIN-75] Winograd,T.; "Five Lectures on Artificial Intelligence" Computer Science Department, Stanford University 1975

[ZEI-76] Zeigler,B.P.; "Theory of Modeling and Simulation" John Wiely & Sons Inc, 1976

[YOU-58] Young,J.W., Kent,H.K.; "Abstract formulation of data processing problems" in [COU-74]

SUMMARY OF METHODS

The purpose of this appendix is to summarily characterize each of the approaches that are discussed in chapters 4 and 5. Each method is presented without comments or discussions. The presentations are based on the authors' terminology, and references from the original papers are frequently included, in order to give examples from the original texts. The presentation of each method is structured into:

- 1. The context in which the author presents the method
- 2. Characteristic ideas in the method approach
- 3. Modeling concepts used or proposed in the approach
- 4. Summary of the method.

Some of the methods have names, other do not. For practical reasons, each method approach is presented under the name(s) of the author(s).

1. Benci, Bodart, Bogaert and Cabanes

This summary is based on the report "Concepts for design of a conceptual schema" [BEN-76]

As indicated by the title, concepts to be used at different steps in the process of designing a conceptual schema are proposed.

1.1 The Context

The authors introduce the approach by regarding an ANSI/SPARC definition of the concept "conceptual schema" as consisting of two views of the conceptual schema:

"- the conceptual schema as an integrated view and a bridge between the external schema and the internal schema,
- this integrated view as the result of an analysis and design process". [BEN-76]

According to the authors, it is the second view of the conceptual schema which is focused in their approach.

The authors propose that the processes of designing a conceptual schema starts with the design of a conceptual organization.
Different users are supposed to percieve the real world in terms of different models for example "mathematical models, logical models, analogical models, descriptive models, etc,". There is a basic assumption made by the authors, that the information of any such model can be described by a suggested set of "basic modeling concepts".
Using these basic modeling concepts, real world modeling is performed and the information contained in the different users perception models are structured into a shared view of the real world. This common shared view is the conceptual organization.

The conceptual organization consists of two parts, the conceptual structure and the evolution rules.
"The conceptual structure contains the description of the data sets (conceptual data base) and their properties (integrity constraints). The evolution rules describe the set of operations that modify the state of the data base: in this respect they describe the interaction of the information systems with the "environment" [BEN-76].

A formalism - modeling concepts - for the description of the conceptual data structure is proposed. The formalism is an extension of the relational data model. A conceptual schema is the description of the conceptual structure in terms of a declarative language.

1.2 Characteristic ideas

Some characteristic ideas in the article are:

- The identification of two levels of models and of modeling within conceptual schema design.
- The proposal of a set of basic modeling concepts for the conceptual organization level model.
- The proposal of an extension of the relational data model to be used as a formalism for description of a conceptual data structure.
- The distinction between evolution rules and integrity constraints.

1.3 Modeling concepts

A "conceptual organization" is a, to all users, common model of the percieved real world. The basic modeling concepts proposed for the conceptual organization are objects, associations and properties.

"-Object: An object is what an individual or group (e.g.: an organization, a firm, a group of scientists, ...) sees as a whole. (As a system or element of a system, an object is characterized by a set of quantitative and qualitative properties, and a permanent behavior being function of the level of resolution.)

- Association: Association is a set of two or more objects where every one plays a given role. A relation may possess different properties. The existence of an association is contingent on the existence of the objects it relates.

- Property: A property value belonging to an object or an association is a quality that the individuals attribute to this object or this association. The existence of the attribut property values is contingent on the existence of the object or the association concerned."

A conceptual organization can be used as a definition of the meaning of data in a data base. When the conceptual organization is used for this purpose the relational model is proposed as a basic formalism. However, as the relational model can only define statical structures of data in the data base, extensions to the relational data model are proposed for expressing evolution rules and integrity constraints.

"Although the concept of relation is useful to describe a static state of a conceptual data base, this is not the case when describing its evolution. To this purpose, the notation of relation family is introduced $R_1^1 \ldots R_1^i$ which is a series of relations defined on the same components, indexed $\in I$ and writted $(R_1^i \mid i \in I)$. IF R^i is defined on the sets A, B, C it is quite clear that the families $(A^i \mid i \in I)$ $(B^i \mid i \in I)$ and $(C^i \mid i \in I)$ have to be defined.

An operation on a relation is a function f_1^j such that $R^{i+1} = f_j^i (R^i)$ and R^{i+1} are obtained

- by addition of a n-tuple to R^i
- by suppression of a n-tuple from R^i
- by modification of n-tuple of R^i
- by equating with R^i (identity operation).

A conceptual data base R^i is a set of relations idexed by the same value $i \in I$

$$B^i = \{R_1^i, R_2^i \ldots R_q^i\}$$

A conceptual data base R^{i+1} is optained by applying to each relation $R_n^i \in B_i$, one of the possible operations.

$$B^{i+1} = \{R_1^{i+1}, R_2^{i+1}, \ldots R_q^{i+1}\} = \{f_1^i(R_1^i), f_2^i(R_2^i), \ldots f_p^i(R_q^i)\}$$

Note: a relation R_n^i can be empty. We will call type of relation, of set... a family of relations, sets... We will call evolution rules the definition of the allround operations on a base B^i" [BEN-76]

1.4 Method Approach

The design of a conceptual schema is separated into the two parts
- design of a conceptual organization
- representation of the conceptual organization by a formalism which is an extension of the relational model.

The conceptual organization design is described by the two tasks
i) identification of basic elements of the perceived real world,
ii) definition of temporal and spatial context of the basic elements.

Elements of the perceived real world are identified, and people in the enterprise or organization decide whether an element is to be regarded as an object or as a property. Elements of the real world are represented by names in the conceptual organization.

When objects, properties and associations have been identified their spatial and temporal context are determined. Spatial and temporal context are described as localization in space and in time and as a temporal dimension.

Localization in time can have two different natures:
"The existence duration or validity duration of a basic element"
"The date at which something or some events are observed in the real world".

Concerning the temporal dimension it is pointed out that quantitative property values are defined by this dimension.

The conceptual structure is a formal representation of a part of the conceptual organization:
"The conceptual structure is made up by three types of basic elements:
- the type of data
- the type of entity
- the relation
They correspond to basic elements of the perceived real world in the following way:

Perceived real world		conceptual structure
- a type of property	is represented by	- one or more type of data
- a type of association	is represented by	- relation(s)
- a type of object	is represented by	- relation(s)
	is identified by	- a type of entity identifies is only in the conceptual structure
- a value of property	is represented by	- a data "

[BEN-76]

Integrity constraints are regarded as properties of data sets. The purpose of the integrity constraints are to assure semantically correct data in the data base. Semantically correct data are those which conform to properties of the real world. A definition of integirty constraints is given:
"In the conceptual structure, the integrity constraints are predicates relative to the data of the base, defined on the conceptual structure elements in order to provide the user

with correct data as a result of his application programs insofar as those are syntactically and semantically correct." [BEN-76]

Characteristic of some frequent types of integrity constraints are desribed. Examples of such integrity constraints are:
- constraints concerning the possible values of a type of data or a type on entity,
- specification of the format(s) of a type of data or a type of entity.

The conceptual structure and the integrity constraints describe statical properties of the real world beeing modeled. Evaluation rules model dynamic aspects of the real world. Evaluation rules are regarded as procedures related to integrity constraints and to events that trigger the execution of operations on data in the data base.

2. Bernstein

Bernstein's approach to conceptual structure design is described in the report "Synthesizing Third Normal Form Relations for Functional Dependencies" [BER-76]

2.1 The Context

In this approach the conceptual structure is regarded as a definition of the meaning of data in a data base. "The purpose of any data model, relational or otherwise is to allow the user of the model to describe and manipulate those relationships among objects in the real world that he intends to store in the data base".

"A relational schema consists of data base relations and for each relation the specification of one or more keys" [BER-76].

The method proposed for design of a relational schema is an algorithm which derives a set of relations from a set of functional dependencies. The set of derived relations can be proved to contain the fewest possible number of relations and each relation can be proved to be in Third Normal Form.

The input to the algorithm is a set of functional dependencies between sets of attributes. Thus a real world model is implicitly assumed. In this assumed model, the real world phenomena are regarded as sets of attributes between which there exists a set of (functional) dependencies.

2.2 Characteristic ideas

As has been shown by Codd, relations in first or second normal form have properties that will cause problems when tuples of the data base are to be inserted, deleted or updated and that these problems are avoided when relations are in third normal form. In Codd's approach third normal form relations are achieved through the process of normalization of relations in first or second normal form. This process of normalization requires information about functional dependencies.
In Bernstein's approach, relations in first or second normal forms never exist. Attributes are "directly" grouped into third normal form relations. This process, called "synthesizing", is based upon knowledge about functional dependencies between sets of attributes.

The suggested process of synthesizing is based on the assumptions that all relationships between attributes can be represented as functional dependencies. "The approach of building a relational schema for FD's rests entirely on the ability to represent all data relationships as functional dependencies. Clearly, though, not every logical connection in the world is functional. Nevertheless, we claim that all connections among attributes in a data base description can be represented by functional dependencies" [BER-76]

The process described further asumes that there exists at most one functional dependency between any one set of attributes.

Based on these assumptions an algorithm for deriving a "conceptual schema" in terms of a set or relations is proposed. The conceptual schema which is the result of applying the algorithm is proved to contain the fewest possible number of relations and to contain only 3NF relations.

2.3 Modeling Concepts

The conceptual framework of this approach is the relational data model as proposed by Codd [COD-70][COD-71]. A restriction to the relational model - that between any two sets of attributes there is at most one functional dependency - is introduced by Bernstein.

A conceptual schema for a data base is seen as a set of data base relations and for each relation the specification of one or more keys. "A functional dependency X--A is embodied in a relation R if X is a key of R and A is any other attribute or R. The set of FD's embodied in a schema is the union of the FD's embodied in all of relations of the schema".

This is a modification of Codds formulation, where functional dependencies are given as information additional to the relations and their keys.

2.4 Method Approach

The method consists of an algorithm by which 3NF relations are generated. The input to the algorithm is a set of functional dependencies between sets of attributes.

The algorithm:

"1. (Eliminate extraneous attributes.) Let F be the given set of FDs. Eliminate extraneous attributes from the left side of each FD in F, producing the set G. An attribute is extraneous if its elimination does not alter the closure of the set of FD's.

2. (Find covering.) Find a nonredundant covering H of G.

3. (Partition.) Partition H into groups such that all of the FD's in each group have identical left sides.

4. (Merge equivalent keys.) Let $J = \emptyset$. For each pair of groups, say H_1 and H_2, with left sides X and Y, respectively, merge H_1 and H_2 together if there is a bijection $X \leftrightarrow Y$ in H^+. For each such bijection add $X \rightarrow Y$ and $Y \rightarrow X$ to J. For each $A \in Y$, if $X \rightarrow A$ is in H, the then delete it from H. Do the same for each $Y \rightarrow B$ in H with $B \in X$.

5. (Eliminate transitive dependencies.) Find and $H' \subseteq H$ such that $(H'+J)^+$ and no proper subset of H' has this property. Add each FD into its corresponding group of H'.

6. (Construct relations.) For each group, construct a relation consisting of all the attributes appearing in that group. Each set of attributes that appears on the left side of any FD in the group is a key of the relation. (Step 1 garanties that no such set contains any extra attributes.) All keys found by this algorithm will be called synthesized. The set of construced relations constitutes a schema for the given set of FDs". [3ER-76]

Proofs of the minimality of the number of relations in a schema and of all relations of a schema being in 3NF are presented in the article.

3. Brodie and Tsichritzis

This summary is based on the ref. "Data Base Constraints" within the report "A panache of DBMS Ideas" [TSI-77A] and to some extent on the report "Specification and verification of Data Base Semantic Integrity" [BRO-78]. None of these papers are primarily concerned with a method for conceptual level data base design. However, an approach to conceptual schema design is summarily presented in "Data Base Constraints".

3.1 The Context

A Data Base Management system is regarded as a system which should present each user with an abstract view and abstract operations appropriate to the user. A Data Base Management system must maintain each view both structurally and behaviorally with respect to the limited evolving real world that is models.
The authors regard an abstract view as the collection of objects and relationships seen be a given user (class of users), and abstract operations as means of modifying an abstract view.
The authors point out that some of the problems concerning data base management systems concern the lack of concepts and rigorous methods for the specification, description and maintenance of coexisting user vies.

Four important DBMS problems are stressed:

"1. Specification and representation of an appropriate abstract view for each user in such a way that views may coexist without interference over the same data base.

2. Verification that each and every abstract view satisfies stated requirements.

3. Implementation and enforcement of the properties of each abstract view.

4. Evolution of any or all abstract view through the alteration of view properties or global data base requirements" [TSI-77].

The authors propose formalization of the problem specification as a step towards solution of the problems. Abstract data types and constraints are introduced as means to achieve such a formalization.

3.2 Characteristic ideas

The reference proposes an approach to conceptual schema design. The most characteristic property of a schema designed by this approach is that it is not a to all users common view, but rather a schema which permits the coexistence of different user's views.

"First a description of the schema is constructed in terms of objects. Then, the schema is specified in terms of abstract data types. It is assumed that the data model/data language provide the fundamental domains string and number. It is also assumed that there has been informal description of the data base in terms of its abstract views which, in turn, are described in terms of the objects and relationships in the views. The problem is to specify the objects in terms of the fundamental domains" [TSI-77].

In this approach, each user's view of an object will be represented by a tree. The root of the tree is the object being viewed by the user, the other nodes are the component objects that the user (in this specific view) regards the objects as being composed of. Thus the nodes of the tree represent objects and the edges represent and "is used to compose" association between the objects.
Several trees might represent the same object, but each tree represents a particular way of viewing that object. Each tree expresses a set of properties that the data base system must obey.

A schema for a specific user view (an abstract view) is the result of combining all trees involving objects from that user view. A data base schema is constructed from the abstract view schema descriptions. More precisely "A data base schema is a collection of data types. A data base schema specification is a collection of data type specifications. A data base schema specification may be used to verify that types are specified consistently and to validate the type and manipulation of data values. A data base is a collection of data values stored in instances. A data base schema comprises the type level corresponding to the data base which comprises the token level. Variables and constants exist only in the programs used to access data base values. Data bases consist entirely of instances" [BRO-78].

3.3 Modeling concepts

The basic concept used in the schema definition is the "abstract data type". The concepts of an abstract data type is described by the stages of the abstraction process as proposed by Hoare [HOA-72]. The stages are:
"(1) Abstraction: the generic/aggregate type that results from abstraction over certain tokens, perhaps via types. (2) Representation: the choice of a set of symbols to stand for the abstraction. (3) Manipulation: rules for transforming the representations. (4) Axiomatization: rigorous statements of those properties that have been abstracted from tokens, perhaps via types."

The data type concept is used within the data base area for three purposes
"(1) for abstraction and relating data values via generalization, aggregation, and more general binary relations, (2) for generating or reconizing particular data values and binding those values to variables and instances of the type and (3) for specifying that certain properties be maintained for data values." [BRO-78].

3.4 Method Approach

A procedure for constructing a schema is presented in [TSI-77A]. It is assumed that a model of the "real world" application, or informal descriptions of the users different abstract views, exists before the schema design can start.

When the objects of interest have been identified the procedure starts with decomposition of each distinct object into its component objects, as viewed by a user.

Example: Suppose the objects STUDENT and COURSE have been identified.

A STUDENT might (by some user) be regarded as composed by the objects STUDENT , COURSE, PERSON, FEE.
A Course might (by some user) be regarded as composed by the objects STUDENT, TUTOR and PROFESSOR.

The decompositions of the objects STUDENT and COURSE can be illustrated:

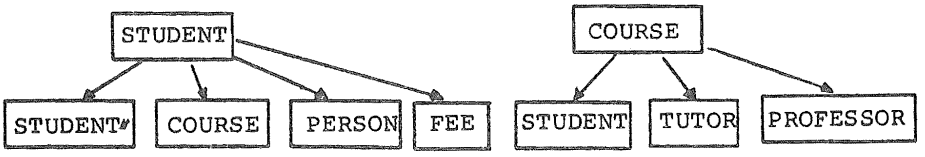


Fig. A.1.

The directed edges in the hierarchies, X→Y, denote that y is used to compose X.

Next, each component object is decomposed, and so forth, until the fundamental domains STRING or NUMBER is reached. The decomposition of component objects will result in a forest of trees of component objects. The trees can be simplified by recognition of semantically similar (sets of) components (e.g. Students, professors and tutors have common properties that pertain to their being people) and by representing them in a single object type (e.g. person data type).

Example:

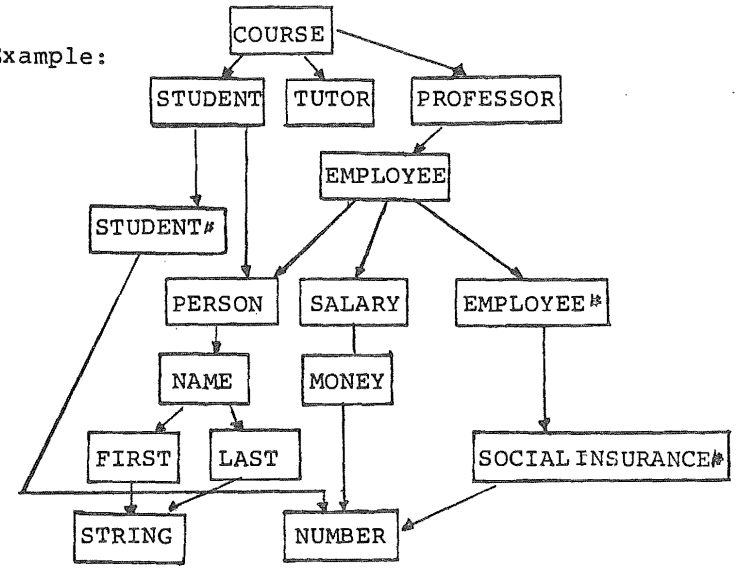


Fig. A.2.

The components of the STUDENT object as seen in fig A.1 can be decompsed in the same way, until the fundamental domains are reached.

A particular hierarchy of objects represents one way of viewing the objects. For each object, several alternative views, represented by hierarchies can be specified. Such hierarchies, representing alternative views of the same objects, are requested not to express inconsistent properties.

A schema for an abstract view is the result of combining all hierarchies involving objects from that abstract view. For example, in fig A.3 the hierarchies representing STUDENTS and COURSES are combined into a schema for an abstract view.

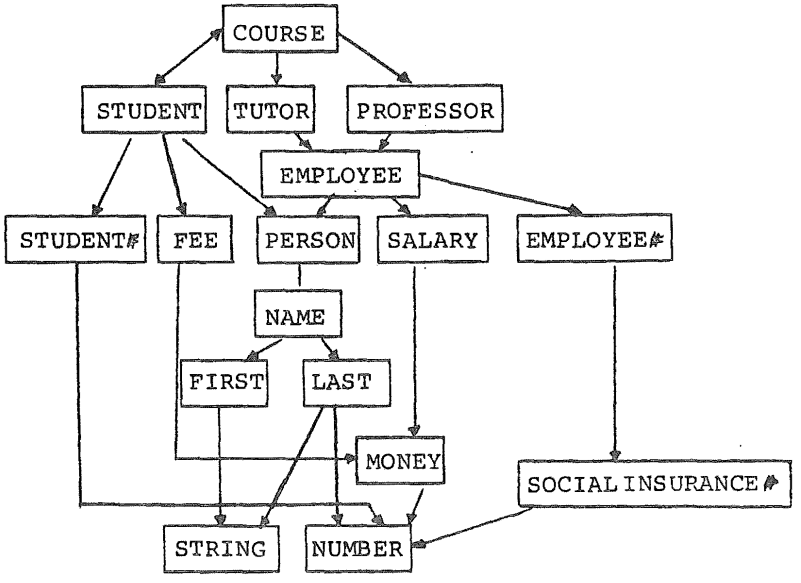


Fig. A.3.

Once the properties of each component object are determined, each object is defined as an abstract data type. The structural properties of the abstract data types are the component objects and their composition rules. The behavioral properties depend on the semantics of the objects; they are based on query, update, insert and delete operations.

4. Bubenko

This summary is based on the report; Bubenko J: "IAM: Inferential Abstract Modeling - An approach to Design of Information Models for Large Shared Data Bases" [BUB-76A].

4.1 The Context

IAM is presented as a method for design of a conceptual schema. An informal definition of a conceptual schema is given: "A conceptual schema describes the information content and relationships of a data base. The information content is an integration of different local user "needs" and "views". Redundancy exists in the sense that some information "elements" can be inferred from others. The conceptual schema acts as a basis for

i) definition of external schemata and queries and
ii) design of data structure schema (storage oriented) where performance, efficiency and limited storage issues have to be considered." [BUB-76A]

4.2 Characteristic Ideas

A common shared view of the real world is assumed and it is expressed in terms of entities, events and associations. Application relevant entities are classified into disjoint concept classes, and functional dependencies between concept classes are identified. Characteristic of this part of the IAM approach is that time points or time periods are regarded as entities and that every event is associated to at least one time-entity.

Information requirements correspond to anticipated input transactions and output requirements of the data base system to be designed. It is assumed that the set of output requirements is complete while the input transactions are assumed incomplete. Output requirements as well as input

transactions are all assumed to be ambigous and inconsistent. Characteristic of the IAM approach is that it is concerned with analysis and determination of relationships between information requirements. Information requirements and known functional dependencies are formally described as "abstract objects". Abstract objects represent information about associations between concept classes. An abstract object refers to a set of concept classes. However, as the concept classes explicitely referenced in an abstract object may - according to the functional dependencies identified - be functionally related to concept classes not explicitely referenced in the abstract object, the abstract object may imply information about not explicitely referenced concept classes.
Analysis and determination of such implied associations is performed and may result in new abstract objects and in identification of further functional dependencies. Thereafter, derivation analysis is performed and for each abstract object which is not implied it is determined whether or not it can be derived from initial abstract objects. When this is not the case, new initial abstract objects are created and thus, in the resulting information model, all information required as output can be guaranteed to be obtainable from the information specified as initial.

In the last step of the IAM approach decisions on how to refer to concept classes are made. In this step abstract objects are transformed to information objects.
"The end product of the procedure is a specification of a set $I = \{I_D, I_O\}$, where I_D denotes the set of derivable and I_O the set of initial information object classes. The set I corresponds to the total set of information requirements and the set I_O to input transactions (signalling events or stating time-invariant facts) needed to maintain the data base consistent with Q (output requirements). I_O is sufficient to generate all stated output requirements". [BUB-76A].

4.3 Modeling concepts

The basic concepts used in the abstract model and in the analysis are: concept classes, functional dependencies and abstract objects.
Application relevant entities in the real world are classified into disjoint concept classes. The abstract model concerns a set $C = \{C_1, C_2 \ldots . C_n\}$ of disjoint concept classes.

Functional dependency exists between concept classes if a given a c-tuple, $c > 0$, of entities from concept classes

$C_1, C_2, \ldots C_n$ it may uniquely determined another entity in a concept class C_f.

Functional dependencies and other associations between concept classes are in the abstract model represented as abstract objects. An abstract object is described as "an abstract construct which conceptually serves to represent information about some associations or relationships in the real world".

Abstract objects can be described by the notation:

$A_i = \{<a:\Theta_a>, <b:\Theta_b> \ldots . <s:\Theta_s>\}$

where a, b are association names that are required to be unique within A_i and Θ_a Θ_b denote concept classes or abstract object classes (sub object to A_i).

Graphically an abstract object can be illustrated:

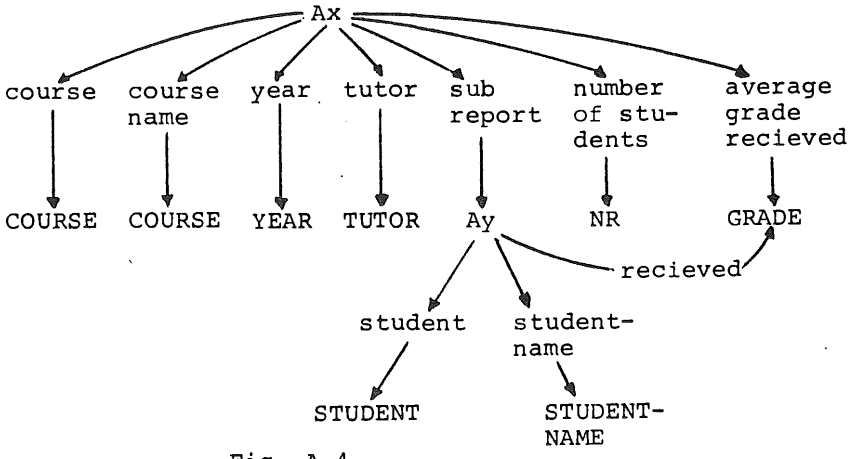


Fig. A.4

4.4 Method Approach

It is assumed that systems analysis has been carried out in the organization and that is had led to identification of user information requirements, expressed in narrative terms. The user information requirements correspond to input or output transactions of the planned data base system.

In the first of the seven steps of the procedure, existing information requirements are determined. In the second

step, users information requirements are analyzed and entities which are referred to in these requirements are classified into the set C of disjoint concept classes. In the third step, functional dependencies between elements of concept classes are determined.
Neither of the three first steps can be performed mechanically and they require thorough knowledge about the real world and the application.

In the fourth step, information requirements and functional dependencies are represented as abstract objects. The abstract objects are "normalized" to 1-level abstract objects. In a 1-level abstract object, all its associations are functional ((1:1),(1:M)), and one or more subsets of its associations, uniquely identify the abstract object.

Abstract objects representing input transactions are decomposed to "elementary abstract object classes". An abstract object is in "elementary form" if it can not be decomposed further into separate, selfcontained abstract objects, such that the original abstract object can not be restructured again.
In the fifth step, implied association analysis is performed.
An association within an abstract object is said to be implied if it is functionally dependent on a subset of a set of identifiying associations for that abstract object. When implied associations are found within an abstract object, a search among other existing abstract objects is made in order to determine if there exists any abstract object which has a set of identifying associations which correspond to the subset of identifying associations on which the implied association is functionally dependent. If no such abstract objects are found, the (partial) functional dependencies are expressed as additional abstract objects. For the additional abstract objects, the steps 4 and 5 of the procedure are iteratively applied.

In the sixth step, derivation analysis is performed. An association within an abstract object is called derivable if it is fully dependent on a set of identifying associations of that abstract object and if it is not initial. Rules for determination of potential precedents of derivable associations are presented. These rules make it possible to mechanically determine potential precedent abstract objects. Examination of whether or not the potential precedent abstract objects are also feasible precedents and the determination of the derivation rules can then be performed.

The examination of potential precedent abstract objects will

lead to one of three cases:

(1) One set of feasible precedents is found and a derivation rule can be specified.
(2) Several, alternative sets of abstracts object classes are found and several alternative derivation rules can be specified.
(3) No feasible precedent abstract objects are found and new abstract objects has to be introduced.

When - as in the third case - new abstract objects are introduced, the steps 4-6 of the procedure are iteratively applied. When the sixth step is ready, the result is a complete specification of the application problem, expressed as an abstract model. The model consists of abstract objects. The initial abstract objects of the model can "generate" abstract objects corresponding to all user specified output requirements.

In the last, seventh step of the procedure the abstract model is transformed to an "external-name-based information model".
For several of the steps in the procedure, matrices for documentation are defined.

5. Hubbard and Raver

This summary is based on the article "Automating Logic File Design" [HUB-75] and to some extent on "Data Base Design Aid" [IBM-76]. The automatic file design was primarily developed for design of IMS data base systems. However, the initial steps of the procedure correspond to what here is called conceptual level data base design.

5.1 The Context

An integrated data base is regarded as a system which supports a network structure of data and which can generate local data structures required by different applications. The authors regard data to occur in at least three levels within an integrated data base. The three levels are:

"(a) External - The external level is data as presented on output reports, displays etc and, therefore, correponds to the structure of data as it appears to a user at a

terminal or a user reading a report printed by the system. Note that the application program generates (or absorbs) the external view.
(b) Local view - this is the data structure needed to support a particular application. The local view or LVIEW is used to generate the external view or in the case of update, absorb the external view by the application program (AP).
(c) Associative model - this is the data structure maintained by the system to generate the multiple LVIEWS. The term associative model is used to describe the internal network or integrated data base." [HUB-75]

By "logical design" is meant the procedure which results in an associative network for an integrated data base system. The associative network should make it possible to generate all local views required by the applications. The associative network is restrained by the datamodeling possibility of the data base management system used to implement the system.
Some parts of the logical design, i.e. integration of local views into an associative network involves computation which can be automated. The authors present a tool - concepts and procedures - for such an automated design. As different data base management systems imply different restrictions on the associative network, the tool must be adapted to different data base management systems. The IBM program product DBDA (Data Base Design Aid) is an adaption of the tool, for deign of IMS data base system.
Logical design is regarded as consisting of three steps. The first step concerns specification of LVIEWS and is regarded as the task of a systems analyst. The second step concerns integration of LVIEWS into an associative network and adaption of this network to the restrictions imposed by the data base management system. In the third step the LVIEWS are modified so that they satisfy the restriction of the data base management system. A LVIEW which satisfies such restrictions is called a restrained LVIEW.

In relation to the frame of reference used in this study, unrestrained LVIEWS and their integration into an associative network correspond to design and analysis of the conceptual level data structure. The adaption of the associative network and the LVIEWS to a specific data base system is here not regarded as concerning conceptual level data base design.

5.2 Characteristic ideas

Local user views are described in terms of data elements, associations and mappings. Analysis of local user views result in identification of "essential" associations, i.e. associations which occur as input associations and which can not be implied from other associations.

The "essential" associations determine how data elements can be clustered into segments. An associative network is created where the nodes represent the segments and the arcs represent the "essential" associations. The associative network is unrestricted in the sense that it is independent of any restriction imposed by a data base management system.

5.3 Modeling Concept

The basic concepts used in this approach are data elements and associations. A data element is the smallest nondivisable unit of data. A data element corresponds to an IMS or DBTG data item. A data element name is the symbolic reference to all occurrences of the data element and a "data element occurrence" is a specific value.

Two kinds of data elements are recognized - keys and attributes - where keys are data elements whose occurrences are unique,while the values of attributes need not be unique.
Data elements can be clustered or grouped into segments. Each segment must contain a key and thus each occurrence of a segment is uniquely identified by that key. A key can consist of one or more elements. A key which consists of more than one data element is called a compound key. A "full compound key" which contains a data element which is not a key in itself is called a "qualified compound key".

An association is a from-to relationship between two data elements. The notation for an association is (A,B):LABEL. The labels are optional and can be used in "multiple-meaning" associations, i.e. when there exists more than one association between the same pair of data elements. Three basic types of associations are identified.

A '1' association is an association where: "Each occurrence of the data element A has a single associated B occurrence. Note that every B occurrence need not be involved."

A 'M' association is an association between A and B where: "Each occurrence of the A can have multiple occurrences of B. Note that the number of multiple B occurrences can take on the values of 0,1,2,3... etc. so that there can be an A occurrence which has no B occurrence".

In a 'C' association between A and B "not every A occurrence, has a single B occurrence but that for those A occurrences that do, each of these occurrences has only one B occurrence".

"Mappings" are defined by combinations of backward and foreward associations between two data elements:

	Inverse '1'	'M'	'C'
'1'	1:1	1:M	1:C
'M'	M:1	M:M	M:C
'C'	C:1	C:M	Not possible

"Mapping table"

Fig. A.4

5.4 Method approach

The method approach is regarded as comprising the six steps that are described for the DBDA tool. The six steps are

1) Data Capture and Edit
2) Association Analysis
3) Mapping Analysis
4) Path Evaluation
5) Hierarchical Determination
6) Structure Model.

The first four of these steps are relevant for the conceptual level data base design while the two last steps are specific for design of IMS data base systems.

Step 1, Data Capture and Edit: A System Analyst is assumed to identify and document local views, i.e. the data structures needed to support the different applications.

DBDA has specific forms for documentation of the local views. The local views are the input to the data base design procedure. In step one the local views are transformed into a set of data elements and a set of associations. Data elements which are keys are identified and local views transformed into associations by pairing each key field with all fields identified by the key. Each association contains thename of the "from" field, (data element), the name of the "to" field, (data element) and the association type.
Information about the frequencies of use of the associations are collected. When compound data elements are found in the input specifications it is checked whether or not the elements of the compound data element are specified as data elements elsewhere in the input local views. If not, the non specified elements of compound data elements are generated as data elements.

Step 2, the association analysis, takes as input the associations from step 1. The first function performed within step 2 is decomposition of compound data names. The compound data names imply not only the simple data elements identified in step 1. but also additional associations not explicitely specified in input local views. Such additional associations are automatically generated. Within step 2, inconsistency between association specifications is checked. The type of inconsistency checked, concerns the case when the same association between the same two elements is specified several times in the input specification. In such cases the association must have the same association type specified for it at the different places where it occurs. If this is not the case this inconsistency is automatically reported.

Attribute analysis is performed within step 2 and 3 concerns identification of attributes identified by more than one key. Multi-keyed attributes may represent errors in input specification or may represent valid design decisions.

Within the Association Mapping evaluation the mappings specified for the associations are analyzed and evaluated. Incomplete mappings, i.e. when only one association type is specifed between a pair of data elements, are identified. The evaluation procedure determines whether or not a mapping should be modified by the DBDA. There are three types of mappings that are modified by the DBDA. The (C:1) mapping is changed to the (M:1) mapping, the (C:M) mapping is changed to the (M:M) mapping and incomplete complex association mappings are eventally completed by simple associations created by DBDA. An associative network consisting of the associations specified is created.

In the third step, this network is analyzed. First, loops - connected sets of simple associations that double back onto themselves - are detected. Second, paths comprising more than 15 associations are identified as the DBDA does not handle paths of a length greater than 15 associations. Third, implied associations are identified. "A given association is said to be an implied association if its "from-to" elements are connected by some other path of essential, simple associations. A simpel association is an essential association if it can not be implied" [IBM-76]. A network of essential associations represents the non-redundant information content of the data base. Implied associations are decision points for the data base designer as implied associations may be implemented for performance reasons.

In the fourth step complex mappings in the associative network are identified and analyzed.

In the fifth step, candidates for secondary indexes are identified and hierarchical levels are established.

In the sixth step, a structural model containing suggested segments, parent/child relationships, candidates for secondary indexes is proposed. After this structural model is created the unrestrained local user views (LVIEWS) that was the original input to the design can be adapted to the model and replaced by restrained local user views.

Figure A.5 is a summary of the basic computations performed by DBDA:

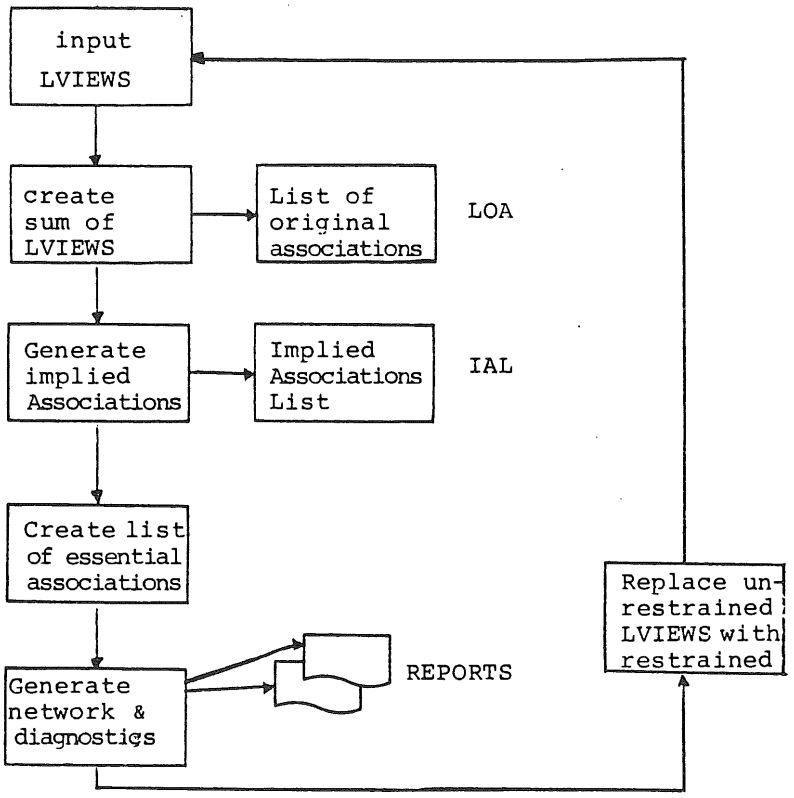


Fig. A.5.

6. Kahn

The summary of Beverly Kahn's approach is based on the reports "A method for describing information required by the data base design process" [KAH-76B] and "A framework for logical data base design" [NOV-76B].

6.1 The context

The overall data base design is regarded as structured into seven, sequential, iterative phases: 1) Information requirements Specification and Analysis, 2) Logical Data Base Design(s), 3) Evaluation of Logical Database Design(s), 4) Physical Data Base Design(s), 5) Evaluation of Physical Database Design(s), 6) Database Construction and Initialization, 7) Performance Evaluation.
Relevant for this study, i.e. for conceptual level data base design, are the two first phases.

The first phase is illustrated as in fig A.6,

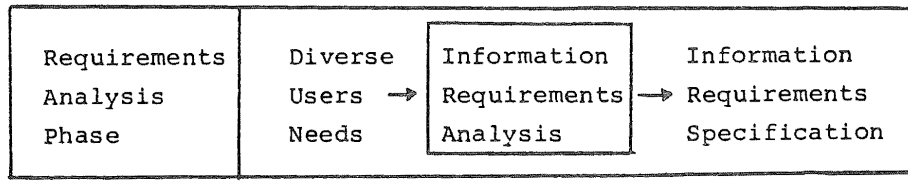


Fig. A.6. from [NOV-76B]

The first phase is not elaborated in the work presented. However, the importance of this phase is stressed. The accuracy of the information requirements is essential to the succeding phases.

The presented method approach concerns the second phase. This phase is further divided into two separate steps as illustrated in fig A.7.

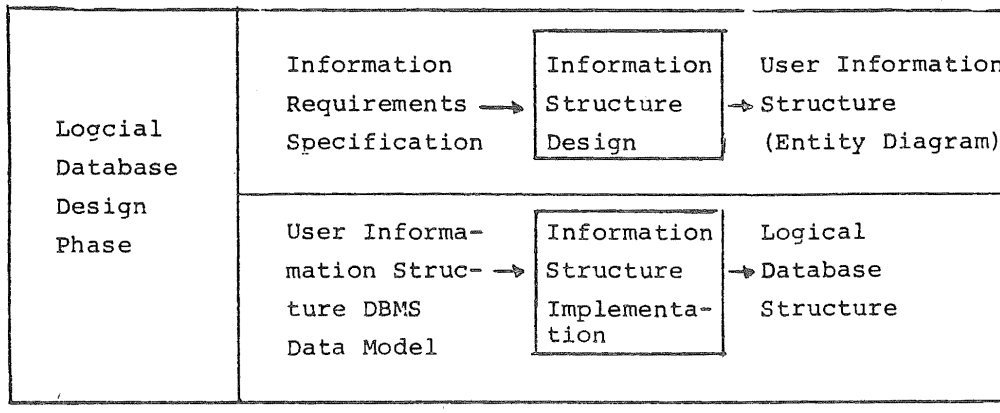


Fig. A.7. from [NOV-76B]

The first step, in the second phase, is the design of an Information structure, i.e. a specification of the information (to be) contained in the data base. In the second step, the information structure is organized and adapted according to the Data Model used by the Data Base Management System to be used for implementation.

6.2 Characteristic ideas

The author identifies and distinguishes between two views of the data base system. These two views represent two kinds of information used in the logical data base design. The two kinds of information are described as "two diverse and independent perspectives of the organizational information requirements to construct the logical data base design. The information structure perspective depicts the natural and conceptual data relationships in the organization. The usage perspective describes how data is used in the system to faciliate and to satisfy the organization's processing requirements." [KAH-76B]. According to the author, existing methods for logical data base design are (almost) only based on the usage perspective, i.e. on process oriented information. This use of only process oriented information may result in data bases that are inflexible. The approach proposed aims at complementing the existing methods with information from the information structure perspective.

6.3 Modeling concepts

The basic concepts used to express an information structure are entities, properties and relationships. Definitions of these basic concepts are the same as in the ANSI/SPARC Interim Report [ANS-75].

Each entity must have a name and certain attributes or properties. "A property plays a role in describing an entity, a property identifies it, characterizes it etc. A property has a set of eligible values referred to as a value set. A given property may be a characteristic of more than one entity. An element is the smallest named object (unit of information) to which processes, such as an application program, can refer. With respect to an entity, properties and elements can be described with a similar structure " [KAH-76B].

6.4 Method Approach

The information structure perspective is created through three sequential activities:

1) The collection of information from various system users to describe entities and relationships.

An Entity description consists of the name of the entity and of a list of attributes or properties for that entity. Each property must be classified into one or several of five roles. The property roles are:
- description of an entity
- unique identification of an entity
- representation of a relationship or a connection between entities (relational or structural property role)
- derived properties
- information necessary to operate in a specific database management system environment for example properties which describe the ordering of entity instances or describe entity security.

Relations are discussed only between pairs of entities. A description of a relationship includes a unique name, a probability of existence and a type of mapping.

In this first step entities and relationships are collected and described by each participating member of the user community. In the next step, these individual descriptions can be refined.

2) Aggregation, consolidation and analysis of each user's perspective of the system's entities, facts and relationships into a global, consistent and nonredundant state.

Three criteria that must be satisfied in order to meet the objectives of having a consistent entity diagram are specified:
- each entity myst be related to at least one other entity,
- each entity must appear only once in the diagram (entity non-redundancy),
- each relationship must appear only once in the diagram (relationship non-redundancy).

A primary goal for the information structure perspective is that it should be "as non-redundant as possible" and therefore "a given element should only occur in one entity".

To achieve this,"normalization of entities" is proposed. Normalization of entities is described as "a step by step reversible process of replacing a given set of entities by succesive collections of entities which have a progressively simpler and more regular structure."

3) Collection of volume and other statistical information about entities and relationships.

Examples of statistical information about entities is cardinalily and about attributes, length, size of value set, probability of existence, cardinality and repetition factor.

The result of the information structure design step is a user information structure which can be represented by an entity diagram:

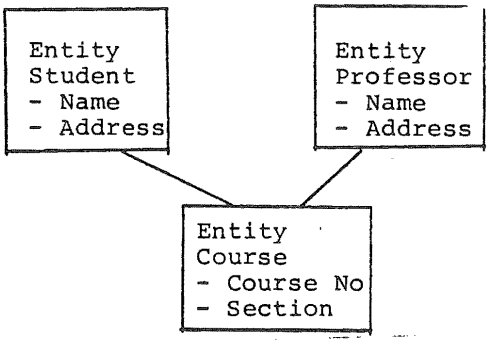


Fig. A.8

An Entity Diagram is said to "represent all of the information necessary to model the organization's information needs as documented in the user requirement specifications".
The Entity Diagram is described as "an implementation independent, general, undirected network which is the most general form of relationship representation and which can be mapped to a logical data base structure that uses any of the DBMS organizations, i.e. hierarchical, network, relational or inverted, to implement the general entity network".

When an Entity Diagram is designed according to the three steps above, the entities and relationships included in the Diagram are regarded as "the natural and conceptual relationships of all the data in the information system". It is regarded as independent of any application and as a representation of "the natural clustering of data (information) and it will therefore provide a basis for expressing basic semantic properties".

7. Sheppard-Rund

This summary is based on the report "Data Base Design Methodology" [SHE-76]. The report contains two parts; the logical and the physical data base design. Here, it is part one - the logical data base design approach - that is of interest.

7.1 The Context

Data Base Design is regarded as consisting of two major phases: Logical Data Base Design and Physical Data Base Design. The Logical Data Base Design is said to be the planning and analytical phase of the design, and to include:

- Definition of scope
- Identification of data elements
- Identification of the relationship of data elements
- Identification of the organization's operating rules and their implications on data relationships.

The logical design is independent of the Data Base Management System that is to be used in implementing the system. "The product of logical design can best be described as a model that depicts all pieces of data and how each relates according to its many uses across the organization. As a by-product of the model, a list is produced that outlines the capabilities and limitations of the data base in terms of data dependencies. These rules describe how an organization conducts its business and should be carefully verified" [SHE-76] .

The definition of the scope of the logical design comprise decisions of which functional areas of the organization that should be included. The base for such a decision is a so called Information Plan. An Information Plan should contain definitions of the organization's current and future information requirement as well as diagrams of the dependencies between major systmems (both manual and automated) and groups of data.

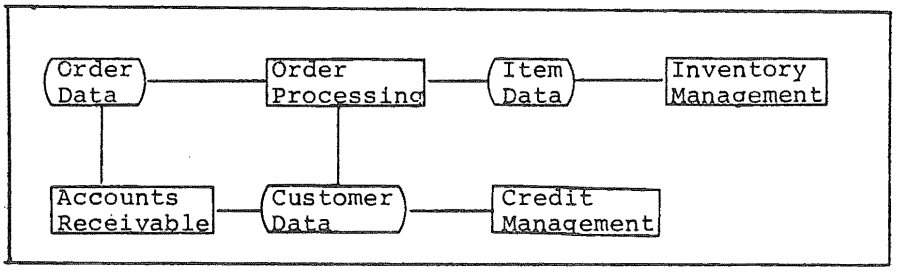


Fig. A.9. Information Plan - Data and System Dependencies from [SHE-76]

When the scope of the logical design has been decided the collection of information about data can start. Three types of information about data are required in the logical design. "(1) The first type of information describes the business basic functions and their related sets of data, (2) The second type of information identifies the explicite and implicite operational policies that determine when and how basic functions are performed. (3) The third type of information identifies additional data elements required to forecast, measure or maintain a history of the data utilized by basic functions" [SHE-76]. Identification of basic functions and their associated data is accomplished by interviews with persons within the operational area of the business. The interviews should result in a complete understanding of the tasks performed within each operational area, the data associated with performing each task and the explicit or implicit rules associated with when and how the task is peformed.

7.2 Characteristic ideas

Three characteristic ideas of Sheppard's approach are here identified.
First, the "over all" approach to logical data base design is characteristic. Sheppard stresses that the logical design of a data base system should be separate from any application development effort and that it should be based on a plan over all functional areas of the organization. Thus, also when a data base is to be designed for only one functional area or for a specific application the information plan, i.e. data and system dependencies of the whole organization must be known. Second, the data elements that are to be contained in the data base are identified and analyzed relative their associations to the specific operational tasks where they are used. Thus, the tasks performed within the operational areas of the organization are important parts of the logical data base design. Third, the approach in categorizing data elements as keys or attributes based on their relative usage in the identified tasks is specific for Sheppard's logical data base design approach.

7.3 Modeling concepts

The major concepts used in the logical structure are; keys, attributes and relationships.
"Keys are data elements used throughout the organization to identify objects, create other data or reference other data. There are two types of keys:
- Unique keys are data elements that identify a particular and distinct thing or object (e.g. social security number, order number, customer number etc.)
- Nonunique keys are collections of data elements unique in their use, that receive their identity through two or more unique keys (such as an inventory that is identified through the unique keys, item number and warehouse location).

Attributes are data elements that must be qualified by a unique key in order to have any meaning (e.g. address and quantity received have no useful meaning unless they are respectively qualified by the keys: customer number and item number).

Relations represent dependencies between two or more data elements and are devided into three categories, Key Relationships, Key-Attribute Relationships and Attribute-Attribute Relationships" [SHE-76].

7.4 Method Approach

In the interviews with people involved in the operational areas, the tasks performed within each operational area and the data elements associatied to each task are identified and documented in "Flow Charts of Operations". These flow charts are the input to the logical data base design.

First, a list of all unique data elements is created. To resolve redundancy it is recommended that data elements should be categorized into generic elements (e.g. dates, names, amounts etc.). Unique data elements are given unique identifiers and descriptions.

Second, relationships between tasks and data elements are specified. Tasks are defined as: "A unit of work, that consists of a set of performed steps, all of which are directed toward a common goal, and all of which utilizes one common set of data". A set of rules for dividing the operations described by the flow charts, into tasks are given.

When all tasks have been defined, they are described by their frequency, volume, usage of data (e.g. create, delete etc.), functional area (i.e. where it is performed) and data element usage. The task/data element relationships are input to the analysis which result in categorization of data elements into keys or attributes. This analysis are based on the assumptions:
"A data element will most likely be a key if it is used: - In a large number of tasks, - With a large number of other data elements, - with each individual data element a low percentage of the total time is used. A data element will most likely be an attribute referenced and owned by one key if it is used: -In relatively small number of tasks, - with each data element in high percentage of the total time it is used. A data element will most likely be an attribute referenced by many keys but owned by one key if it is: - In average number of tasks, - With an average number of other data elements, - With each data element an average number of times" [SHE-76].

In the analysis of task/data element relationships, the first step is to analyze the number of times that a data element is used together with other data elements. For each data element its usage and relative usage with other data elements is determined.

Data Element	Total Tasks	Total Data Relationships	Relative Usage with Data
1	75	120	20%
2	40	97	45%
3	140	241	1%
4	10	50	97%
5	65	101	30%
6	140	211	2%

Fig. A.10 from [SHE-76]
Summary of Data Element Usage

Frequency distributions of Data Elements across tasks are the created. The frequency distributions are used to define which data elements that are to be categorized as keys and as attributes.

An exampel:

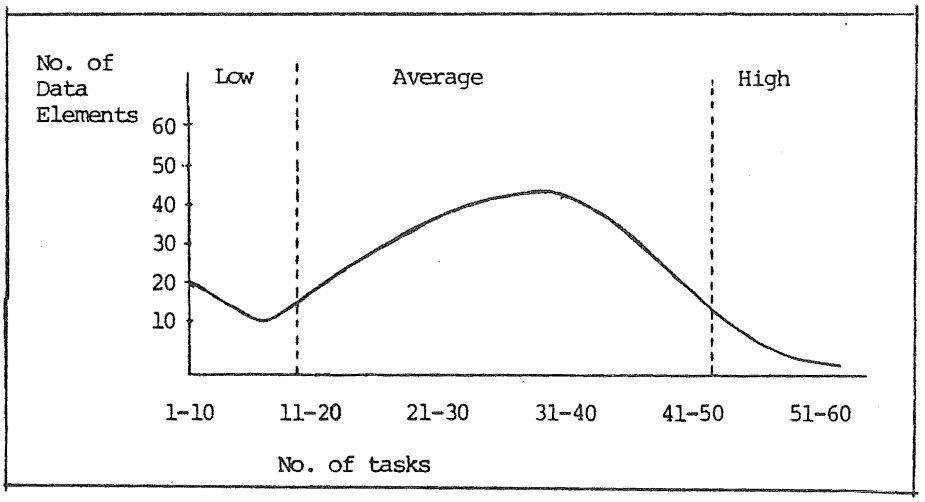


Fig. A.11 from [SHE-76] Frequency Distribution of Data Elements across Tasks

In this example, a data element will be regarded as a key if its task usage >49, as an attribute owned by one key and referenced by many keys if its task usage is >10<50 and a data elements will be an attribute referenced and owned by one key if its task usage is <11.

After data elements have been categorized as keys and attributes, the attributes are assigned to their owning keys. "Assignment of attributes to keys is accomplished by identifying the individual data elements that are related to the attribute beeing asssigned, and assigning the attribute to the data element that has been identified as a key." [SHE-76]. When an attribute is related to more than one key, the following criteria for which key to use are proposed:
"If there is a wide range in the number of times the attribute is used with each key, assign it to the one with which it is most frequently used. Refer back to the list of task/data relationships and identify the task that created and deleted the attribute. Assign the attribute to the key present in those tasks. If different keys were used in creation and deletion tasks, assign the attribute to the key present in the task in which it was deleted." [SHE-76].

When all attributes has been assigned to keys, the next step is to analyze relationships between attributes assigned to the same key. Attribute-Attribute relationships represent unqualified dependencies between attributes. For example

the attribute "Total Balance Owned" may own another attribute "30-Days Past Due Balance" which implies that the attribute "30-Dat Past Due Balance" will never exist unless "total Balance Owned" exists. In order to determine relationships between the attributes of a key, the tasks that created the attributes are identified and ordered in the sequence in which they are performed. From such a ordered sequence of tasks, the order in which the attributes are created and the unqualified dependencies between the attributes can be determined. When the dependencies are identified they are documented in hierarchical diagrams:

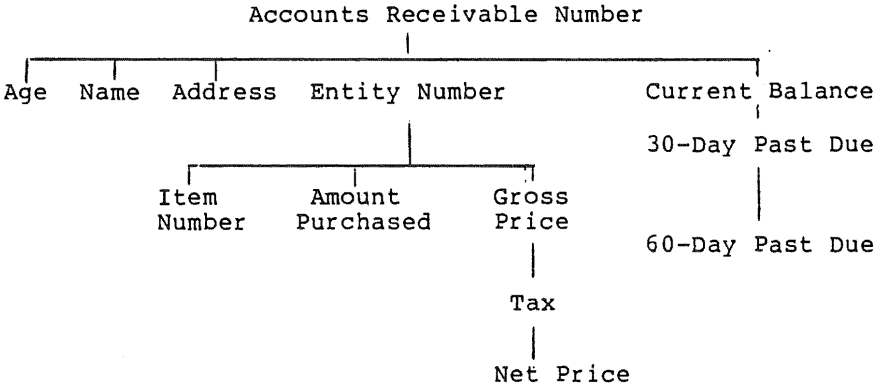


Fig A.12 Hiearchical Diagram of Key's Attributes from [SHE-76]

After attribute-attribute relationships have been identified and documented, key-key relationships are determined. Four types of relationships between keys are identified:

"Owner: In many situations a key may be totally meaningless unless it is related to another key. This type of key relationships is termed Owner, because the dependent key must always be related to the owning key in order to exist in the data base.

Status: Groups of data may take on different characteristics because their use and meaning change over a period of time. Under these conditions, a group of data may be owned by one of several different keys, depending upon its current status.

Regulations: Oftentimes, outside regulatory agencies determine the way in which part of a business is conducted. The rules enforced by such agencies normally create relationships between keys. These relationships are termed regulatory relationships.

Policies: The policies of an organization (i.e. major rules that regulate how different areas of the business must function either separately or together) represent yet

another type of relationship between keys. Policy statements usually fall into two categories: organizational and functional. Functional policies tend to be fairly static over a period of time while organizational policies are not usually included in developing key relationships." [SHE-76].

Identification of key relationships is stressed as the most important part of the logical data base design. and it is proposed that each of the four types of relationships are identified separately.

In the very last step of the logical data base design the limitations and capabilities implied by the logical design of the data base are verified with the users.

8. Smith and Smith

This summary is based on the report "Data Base Abstraction: Aggregation and Generalization" [SMI-76]. The approach presented is primarily a model, based on an extended relational data model, for representing a conceptual data structure. However, the process of aggregation and generalization, suggested in the report, is here regarded as essential to conceptual level data base design and therefore motivates that the approach is included in this study of method approaches.

8.1 The Context

The conceptual data structure corresponds to an abstraction. "An abstraction of some system is a model of that system in which certain details are deliberately omitted. The choice of the details to omit is made by considering both the intended application of the abstraction and also its users" [SMI-76].

There are two fundamental types of abstraction; aggregation and generalization. "Aggregation is an abstraction which turns a relationship between objects into an aggregate object. Generalization is an abstraction which turns a class of objects into a generic object". When models, i.e. abstractions, are comprehensive they may be structured as hiearchies of abstractions. Such a hierarchy of abstractions makes it possible to control the number of

details included at each leve. An advantage of this is the possibility to let different users interact with the model at different levels of the hierarchy and thus ignoring the abstractions and details at higher levels in the hierarchy.

In this approach, an extention to Codd's relational data model is proposed. This extention makes a relational schema appropriate for representing hierarchies of aggregation and generalization abstractions. A structuring dicipline for arriving at such a relational data base schema is outlined.

8.2 Characteristic Ideas

The authors point out that data base research almost exclusively has been concerned with aggregation abstractions while artificial intelligence research has been concerned with generalization abstractions. The approach taken is to combine aggregation and generalization into a structuring dicipline and to extend the relational data model so that it can represent hiearchies of abstractions.

The term generalization is used in the following way:
"a generalization is an abstraction which enables a class of individual objects to be thought of generically as a single named object".

The importance of generalization in data base design is stressed, referring to its importance in conceptualizing real world phenomena and as a basis for natural language. In a data base it is important that generic objects can be explicitely represented. Explicit naming of generic objects allows the following capabilities:

" i) the application of operators to generic objects;
ii) the specification of attributes of generic objects; and
iii) the specification of relationships in which generic objects participate".

In order to represent hierarchies of generalization and aggregation abstractions in the relational data model, this model is proposed to be extended so that domains may consist of relation names.

8.3 Modeling Concepts

"A generalization is an abstraction which enables a class of individual objects to be thought of generically as a single named object".
Generic objects may have attributes attached to them. Such attributes may be "summaries" of attributes of the objects being generalized, e.g. attributes that all objects being generalized into a generic object have. Attributes of generic objects may also be specific for the generic object and not exist for the individual objects. Generic objects may participate in relationships with other objects. Hierarchies of generic objects can be created as generic objects may be generalizations of other generic objects.

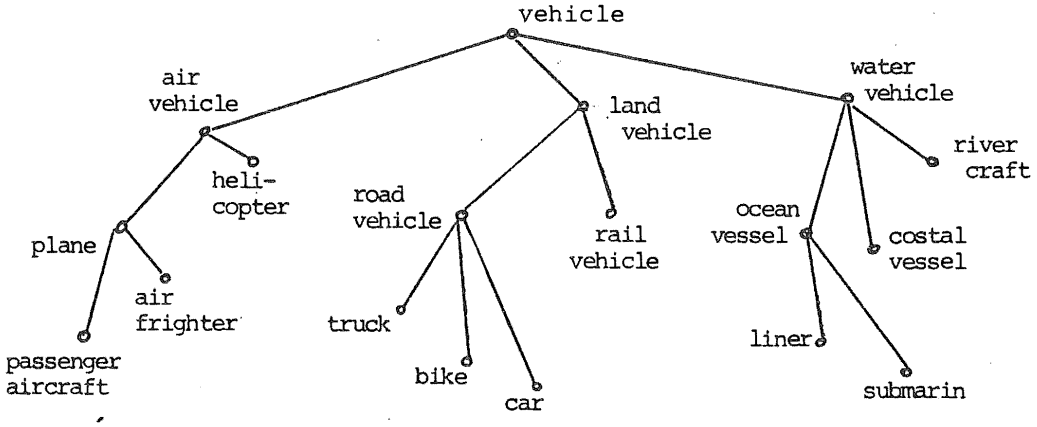


Fig. A.13 "A generic hierarchy over vehicle from [SMI-76].

Generally, generic hierarchies do not necessarily have to be trees, i.e. a generic object may be an immediate descendant of more than one generic object. In such cases, the descendent generic object has been generalized in more than one way.

When the generic hiearchies are to be represented by relations, it is required that the intermediate descendents of any node are partitioned into groups. "Each group must contain generic objects which have mutually exclusive classes. In practice, this grouping can usually be made quite easily from semantic considerations." [SMI-76].

Mutually exclusive groups sharing a common parent are called clusters. A cluster is said to belong to its parent generic object. Each cluster must hav a semantically meaningful name, that describes the generic objects included in the cluster.

A generic hierarchy is represented as a hierarchy of Codd relations; "We will create one relation for each generic object in the hierarcy. Assume G is a generic object such that:

i) I is the class of individual objects generalized to G,
ii) $A_i,\ldots,A_n$ are the G attributes, and
iii) $C_i,\ldots,C_m$ are the names of clusters belonging to G.

G is represented by the Codd relation:

G:

A_1	....	A_n	C_1	....	C_m
...	...	...	...	...	...
v_1	...	v_n	v_{n+1}	...	v_{n+m}
...	...	...	...	...	...

where:

i) there is one and only one tuple for each individual in I,
ii) if an individual has a value v_i for attribute A_i, then its tuple contains v_i in domain A_i,
iii) if an individual is generalizable to generic object v_{i+j} in domain C and
iv) if an individual is <u>not</u> generalized to any generic object in cluster Cj, then its tuple contains blanks(-) in domain Cj". [SMI-76].

A structuring primitive for representing generalizations as well as aggregations is proposed. The primitive which is called <u>the relation structure</u> is defined:

var R : relation key (selectorlist)
S_1 : {key of} R_1;
...
S_n : {key of} R_n;
S_{n+1}: {key of} R_{n+1} = [R_{11},.....,R_{jK_1}] ;
...
S_{n+m}: {key of} R_{n+m} = [R_{m+1},....,R_{mk_m}] ;
end

where:

i) R_1 ($1<i<n+m$) is either a relation identifier (in which case "key of" must appear) or a non-relational type identifier (in which case "key of" must not appear),
ii) selectorlist is a sequence of s_i's separated by commas,
iii) each R_i is a relation identifier,
iv) for each R_{ij}, its key domains must be the same as the key domains of R (possibly excluding S_{n+1}).

Five semantic conditions are added to the four syntactic requirements:
i) each R-individual must determine a unique R-individual ($1<i<n+m$);
ii) no two R-individuals determine the same set of R-individuals for all R_i named in selectorlist;
iii) each R_{ij}-individual must also be an r-individual;
iv) each R-individual classified as R_{ij} must also be an R_{ij}-individual;
v) no R_{ij}-individual is also an R_{ik}-individual, for $j \neq k$.

A relation R is said to be well-defined if its definition satisfies all the syntactic and semantic conditions. The naming of the relation is stressed as the most crucial part of the definition as a relation which can not be given a name which is in common usage in the application area is not a well-defined relation.

8.4 Method Approach

The basic activities in real-world modeling are aggregation and generalization. An important assumption is that aggregation and generalization are independent activities. A graphical representation for relation definition is proposed. In this representation, aggregation and generalization are represented as two orthogonally planes. Aggregation is represented in the plane of the page and generalization in the plane perpendicular to the page.

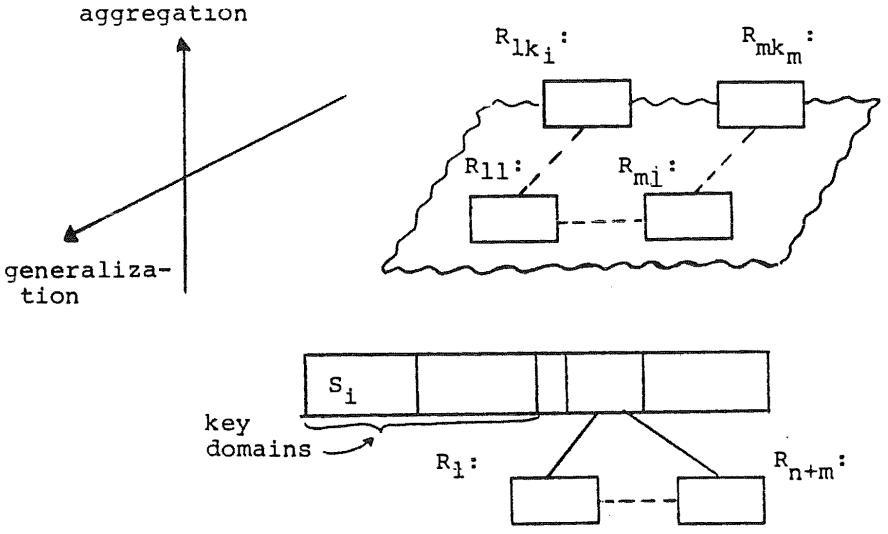


Fig A.14: "A graphical notation for the relation structure" from [SMI-76]

"High level aggregate objects will appear towards the top of the page and low-level aggregate objects towards the bottom. Aggregation therefore occurs up the page. High-level generic objects will appear (in a simulated three dimensional space) in the surface of the page and low-level generic objects will appear below the surface. Generalization therefore occurs out of the page" [SMI-76].

The modeling approach is to start with a high level abstract object and then to stepwise decompose this object along the generalization and the aggregation planes.
An example: suppose the initial model is an abstract object "employee"

Fig. A.15 from [SMI-76]

The abstract object "employee" can be decomposed along the generalization plane into the generic objects "trucker", "secretary" and "engineer":

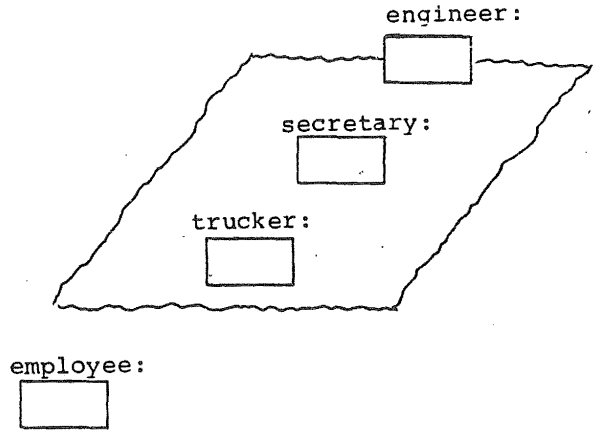


Fig. A.16 from [SMI-76]

Along the aggregation plane, the abstract object employee can be decomposed into other (eventually) aggregate objects as for example into "employee ID*", "name", "age", and "employee type".

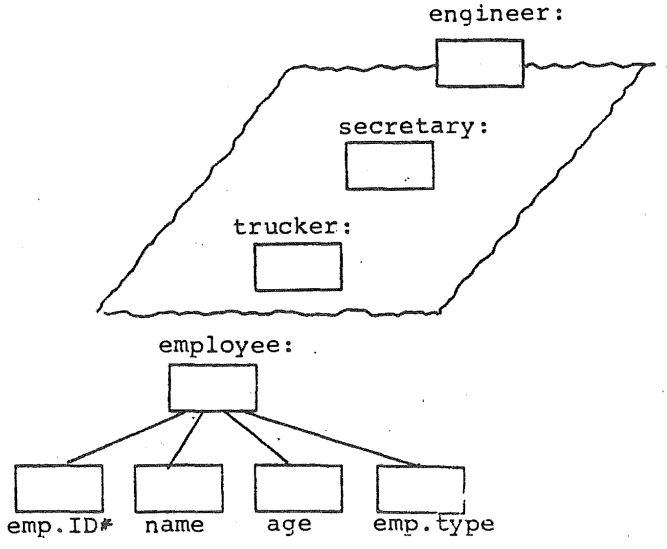


Fig. A.17 from [SMI-76]

The decomposition is continued alternatively along the generalization and the aggregation planes. Along the aggregation pane, the decomposition ends when each component at the lowest level is thought of as a whole. Along the generalization plane the decomposition ends when there is no - from an application point of view - need for, or interest in any sub-types of the components at the lowest level.

9. Sølvberg

In the report "A multi-level procedure for design of file organizations" [AUE-76], Auerdal and Sølvberg have presented an approach to data base design. The first steps of the procedure have been elaborated by Sølvberg in the report "A model for specification of phenomena, properties and information structures" [SØL-77A]. This report does concern the conceptual level data base design.

9.1 The Context

The multilevel procedure for design of data base systems contains as main steps:

"- Specification of the information processing problem.
- Transformation of the specified information structures to a data structure which satisfy retrieval requirements.
- Modification of the data structure to fit a particular data base management system.
- Physical implementation of the modified data structure using the chosen data base management system.
- Evaluation of the final solution using performance analysis".

The model for specification of phenomena, properties and information structures, proposed by Sølvberg, concerns these two first steps in this procedure. The specification model consists of interrelated models for specification of:

"- relevant phenomena
- descriptors, which represent ranges of phenomena - properties
- information objects, which convey information about particular properties about particular phenomena
- processes, which transmit and transform information about relevant phenomena".

The overall specification model is illustrated:

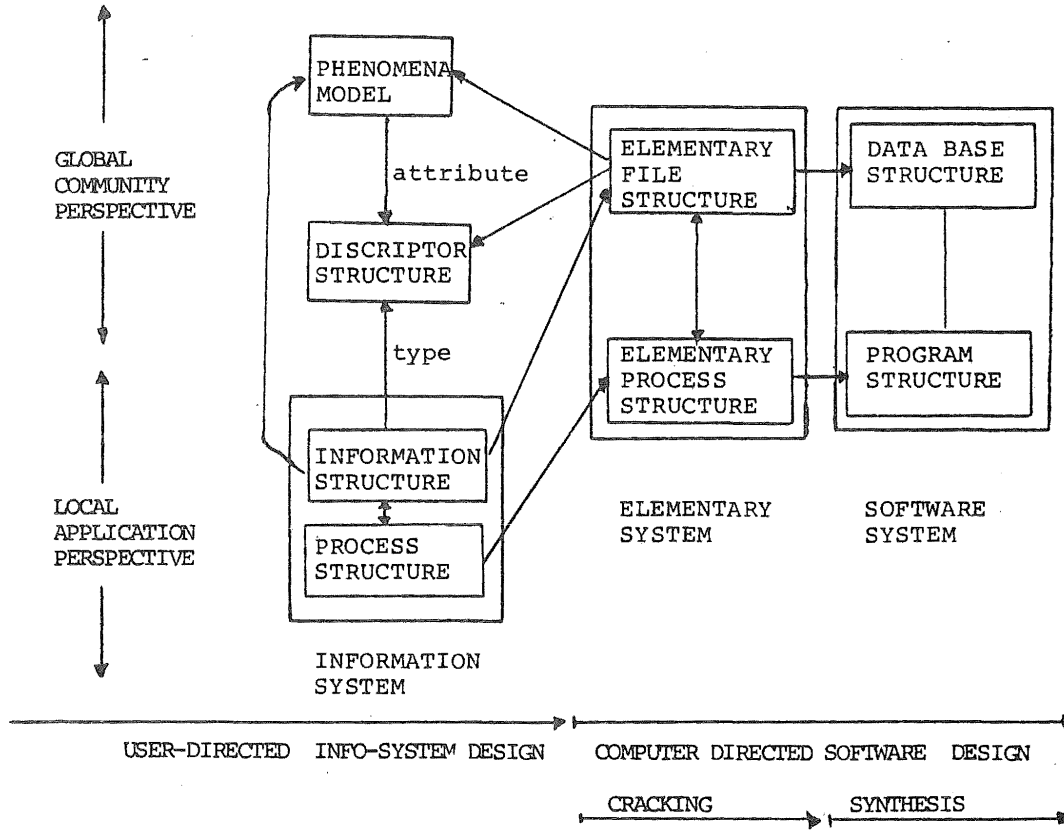


Fig. A.17

"Features of a system specification model"
from [SØL-77A].

9.2 Characteristic ideas

The same piece of information can be represented in different ways. The over-all model contains different representations of the same information. The information system (see fig A.17) containing the information structure and the process structure, corrsponds to a representation of the information (to be) contained in the data base system made from a user point of view, i.e. with the purpose of easy user comunication. The data base structure and the program structure together contain the same information as the information system, but the information is here structured from the data base management system point of view, i.e. with the purpose of optimal computer processing. In order to transform the information system representation

into a data base system representation in a way that retains the semantics of the information represented by the two different structures, a common "semantical" model, to which both structures are related is introduced. This "semantical" model consists of the phenomena model and the descriptor structure (see fig. A.17).

The transformation from information systems structure to the data base management system structure is made in two main steps. First the information structure is decomposed ("cracked") into atomic parts. Next, these atomic parts are synthesized into the data base management system structure. The elementary file structure and the elementary process structure constitute the intermediate structures, i.e. the structures consisting of the atomic parts.

Modeling concepts are proposed for the phenomena model, the descriptor structure and for the information and process structures.

9.2 Modeling Concepts

The basic concepts used in the phenomena model are entitysets, and a connection is a (user defined) binary relation between entitysets, and does therefore represent an abstracted set of phenomena, each individual phenomenon being that one phenomenon in the connections domain is related to one phenomenon in the connections range. The formation of connections is the result of a process of abstraction of relations between phenomena, based on a recognition of similarities between these relations". [SØL-77A]. [SØL-77A].

Quantitative aspects of entitysets and connections can be specified. For entitysets, the size, i.e. the number of members, are specified. For connections, for example, the correspondence, i.e. the number of elements in the range which are related to one element in the domain can be specified. A formal notation for the specification of entitysets and connections is introduced.

In the descriptor structure, properties of the phenomena in the phenomena model are specified. Descriptors can represent value configurations of varying complexity. A descriptor may be a value set, a Cartesian-product between value sets. A member of a descriptor may be a member of several alternative descriptors or each member of a descriptor may be a set of members of another descriptor.

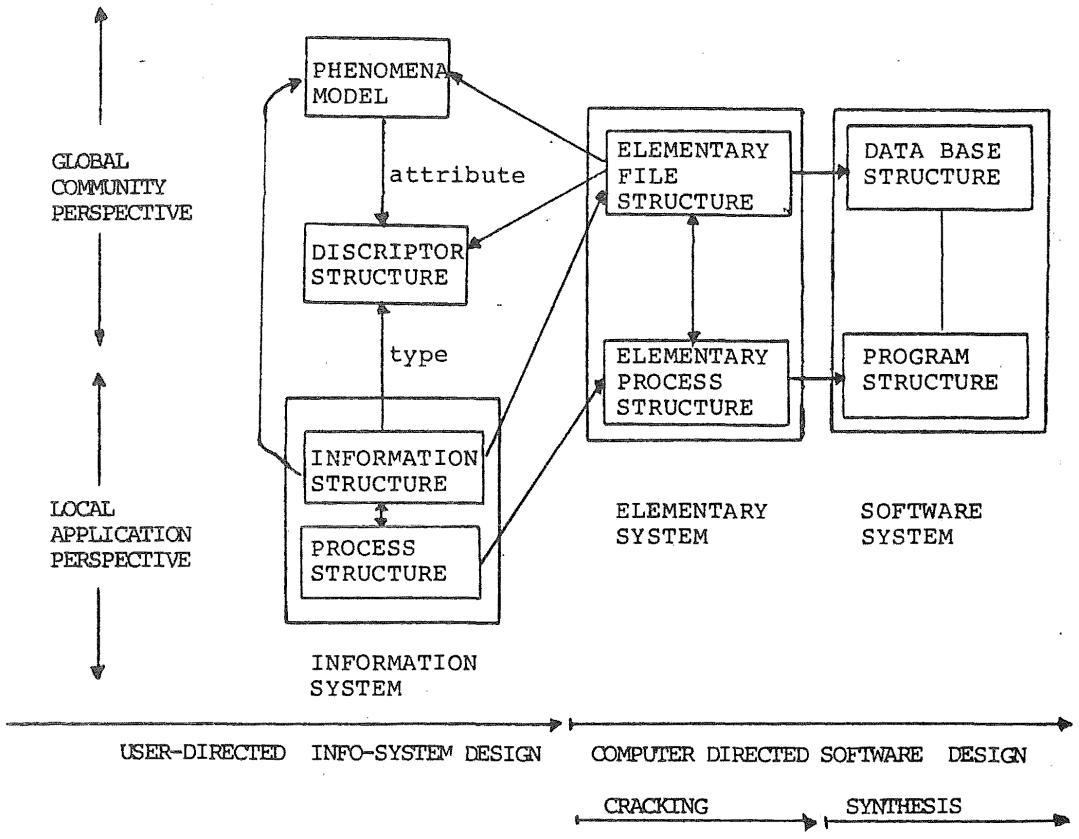


Fig. A.17

"Features of a system specification model"
from [SØL-77A].

9.2 Characteristic ideas

The same piece of information can be represented in different ways. The over-all model contains different representations of the same information. The information system (see fig A.17) containing the information structure and the process structure, corrsponds to a representation of the information (to be) contained in the data base system made from a user point of view, i.e. with the purpose of easy user comunication. The data base structure and the program structure together contain the same information as the information system, but the information is here structured from the data base management system point of view, i.e. with the purpose of optimal computer processing. In order to transform the information system representation

into a data base system representation in a way that retains the semantics of the information represented by the two different structures, a common "semantical" model, to which both structures are related is introduced. This "semantical" model consists of the phenomena model and the descriptor structure (see fig. A.17).

The transformation from information systems structure to the data base management system structure is made in two main steps. First the information structure is decomposed ("cracked") into atomic parts. Next, these atomic parts are synthesized into the data base management system structure. The elementary file structure and the elementary process structure constitute the intermediate structures, i.e. the structures consisting of the atomic parts.

Modeling concepts are proposed for the phenomena model, the descriptor structure and for the information and process structures.

9.2 Modeling Concepts

The basic concepts used in the phenomena model are entitysets, and a connection is a (user defined) binary relation between entitysets, and does therefore represent an abstracted set of phenomena, each individual phenomenon being that one phenomenon in the connections domain is related to one phenomenon in the connections range. The formation of connections is the result of a process of abstraction of relations between phenomena, based on a recognition of similarities between these relations". [SØL-77A]. [SØL-77A].

Quantitative aspects of entitysets and connections can be specified. For entitysets, the size, i.e. the number of members, are specified. For connections, for example, the correspondence, i.e. the number of elements in the range which are related to one element in the domain can be specified. A formal notation for the specification of entitysets and connections is introduced.

In the descriptor structure, properties of the phenomena in the phenomena model are specified. Descriptors can represent value configurations of varying complexity. A descriptor may be a value set, a Cartesian-product between value sets. A member of a descriptor may be a member of several alternative descriptors or each member of a descriptor may be a set of members of another descriptor.

In the descriptor structure, attributes and identifiers of phenomena classes are specified.

"An attribute is a partial function from a phenomena class (i.e. entityset or connection) into a descriptor". "An identifier is a partial, one-to-one, function from an entityset into a descriptor. One entityset can have more than one identifier." [SØL-77].
The phenomena model and the descriptor structure are of a global nature. The model is shared by all users and exists independent of any specific applications.

The information system consists of information structures and process structures. "Information structures are normally user-oriented, such that information that "naturally belongs together" is represented in one and the same structure. What "naturally belong together" is dependent upon is the intended use of the information.
Retrieval and updating processes are specified relative to the user-oriented information strucures, so the form of the information system can be quite different from what one would expect by looking at the phenomena model. The normal situation is that information about several phenomena of different classes, is integrated in one information structure, because this integrated information "belongs naturally together" for the intended application." [SØL-77A].

Basic modeling concepts proposed for the information structures are information sets, groups and items.
"An information structure is a tree structure, the leaves of the tree structure being items. An information object is analogue to a program varible, in that it is a class of (structured) values, ordered in a time sequence. Information objects represent the values that are stored and transmitted in the information system at one particular point in time". In parallel with the specification of the information structures, the process structures are developed. "The information object which is retrieved or updated is called the target of the retrieval/update process while the transaction which is the reason for performing the retrieval/updating, is called the process' request". Four different classes of processes can be specified; retrieve, insert, change and delete.

The information structure and the process structure are representations of the information system. The information structure conveys information about properties and behavior of phenomena, and thus the information structure is related to the phenomena model and the descriptor structure. This relation is called the "contextual relationship" and

concepts for explicite description of the contextual relationship are proposed. The main "contextual" systems relations are called foreach, foreone and selector, where:

foreach is a one-to-one function from an information

foreach is a one-to-one function from an information set or repeating group, to a phenomena class (i.e. entityset or connection),

foreone is a relation which connects one group or item, thus implying that the group or item convey information about only one phenomenon of this class,

selector relates an information object, which is not the root node of an information structure, to a connection. The selector related connection represents the operator which determines which phenomena in the connections range, the information is about.

9.4 Method approach

The over all specification model (fig A.17) illustrates the different models which are the results of a number of steps within a method approach. First the phenomena model and the description structure are designed. "In general, several interest groups and persons, of different background and of different ability, are involved in the design of information systems. Those different persons and groups, the "user community", have an obvious right to have their say, when decisions about systems features are made. Through a discussion process one strives to agree on a classification of phenomena, based on decisions of ignoring differences between phenomena within the same class". Thus, the phenomena classes are designed on the base of the relative importance of phenomena and their properties.

After the phenomena model and the descriptore structure have been specified, the information structures and the process structures appropreate for the individual applications are determined and formally specified. The information structures are contextually related to the phenomena model and the descriptor structure.

The following steps of the design procedure are summarily described as "cracking" and synthesis. "Developement of an elementary information systems sturcture, first proposed by Langefors, is introduced as one step of the data system design procedure. An elementary information object is the smallest information structure that can be unambigously interpreted with regard to its meaning, within the contextual framwork provided by the phenomena model. An

elementary process retrieves and updates the content of one elementary information object. The data systems design procedure can be seen as consisting of a cracking of the user oriented system structure to its elementary, "atomic", parts, followed by a synthesis of these elementary parts, aimed at creating a performance-optimal data base and program system." [SØL-77A]