ISO/TC97/SC5/WG3 78.09/01

A FRAMEWORK FOR DISCUSSION IN ISO/TC97/SC5/WG3 AND COMMENTS ON 78.04/01 AND 78.05/03

G.M. NIJSSEN
Control Data
Brussels

September 1978

Summary

In this paper we describe an overall conceptual framework for information systems with the aim to provide a platform for more precise discussions on the topics of the program of work of ISO/TC97/SC5/WG3. This is covered in chapter 1 through 6. In chapter 7 we present our comments on the working paper 78.04/01 and in chapter 8 we give answers to the discussion points of working paper 78.05/03. Chapter 7 and 8 assume the reader is acqainted with the preceding chapters.

TABLE OF CONTENTS

1. INTRODUCTION

It has been stated that the database research community is entering a second great debate. The topic of the second great debate is: which is the best set of concepts for the conceptual schema?

However, before one can start to have a useful discussion on this topic, it is necessary that one has a clear answer to the question: what is a conceptual schema?

But before one can answer this last question, it is necessary to specify the environment of a conceptual schema. It is our opinion that one can specify this environment by describing the overall architecture of an information system.

In section two we will "derive" the three major components of an information system, one of which turns out to be the conceptual schema. In that section we will give a precise definition of our notion of conceptual schema. The framework in section two gives quite some emphasis on effectiveness in information analysis.

In section three we will extend the information systems framework of three components such as to include programming effectiveness and machine effectiveness. The result is an architecture for database management systems, consisting of three schemata, the conceptual schema (or the "what"), the internal schema (or the "how to the machine") and external schema (or the "how to the program"), and two transformations, one between conceptual and internal, and one between conceptual and external schema.

Given this three schema architecture, and our assumption that any computerised information system can be considered as a special case of human communication, we will describe in chapter four that the basic (or manipulatable) component of a conceptual schema is the elementary sentence; to this concept needs to be added the concept of "object" to which the elementary sentences refer,

and the concept of "constraint" which places arbitrarily limitations on the population of elementary sentence instances in which we are interested.

However, in section five we will consider the question why some respectable persons want to have binary associations and others want to have elementary sentences. In section five we will describe that this discussion (not debate) between the binary association proponents and the elementary sentence proponents can be finished fairly soon, once we see that one has to distinguish between two concepts of conceptual schema, namely one in which one takes into account how names are assigned to objects (called significational schema) and one in which one abstracts from the names assigned to objects (called ontological schema). We then propose to use binary associations as the basic component in the ontological schema and elementary sentences as the basic component in the significational schema.

In section six we will describe an architecture for database management systems which takes the distinction between significational and ontological schema into account.
This architecture contains four schemata:

- ontological (conceptual)
- significational (conceptual)
- internal
- external

and three transformations between

- ontological and significational
- significational and internal
- significational and external.

Motto. A computerised information system can be considered as a system for a special kind of human communication.

2. MAJOR CONCEPTUAL COMPONENTS OF FUTURE INFORMATION SYSTEMS

In this chapter we will start with a model of an information system understandable to users, managers and experts in information systems, databases and programming, in short, to all who make use of, or are involved in the analysis, design, implementation use and maintenance of an information system. Based upon this solid ground, we will "derive" the three major components of an information system. The major assumption on which our entire philosophy of information systems and databases is based can be stated as follows:

A COMPUTERISED INFORMATION SYSTEM CAN BE CONSIDERED AS A SYSTEM FOR A SPECIAL KIND OF HUMAN COMMUNICATION.

What precisely this special kind of human communication is, will be discussed later; for now it is sufficient to state that a computerised information system can be considered as a system, which has as input and output a certain kind of natural language sentences. This means that this approach is based on deep structure concepts in natural language. Putting such an information system on a computer requires some computer and programming oriented aspects which will be discussed in section 3. In that section we discuss a gross architecture for the next generation database management system. This architecture is "derived" from the just mentioned general model of an information system by classifying the various aspects of an information system into three classes, namely the "what" of the information, the "how" to the programmer and the "how" to the machine.

Remark. Our approach is not governed by the data independence syndrome. (ANSI, ref. 2; Date, Ref. 14).
Later on, when we have introduced enough concepts and have discussed major aspects of current database management systems, I think it is worth expanding upon the

difference between the data independence syndrome approach (which is an attempt to free ourselves from some limitations of existing database management systems implementations) and the approach we will describe hereafter (which is a top-down approach mainly based on linguistics).

With the help of this gross architecture one may clearly show what the relation is between a database, a database management system, an information system, a data dictionary and the conceptual schema.

The axioma that all input to and output from an information system can be considered to consist of a set of sentences results in an approach to information systems which can much more easily be taught to users and managers than the conventional computer-centered approaches like CODASYL or mathematically based approach like the Normalized Relational model. Experience of teaching this approach during the last three years to several hundreds of persons brings me to the provisional conclusion that users and managers can be given an effective understanding of information systems in less than half the time required by teaching one of the conventional approaches.

Furthermore, experience with designing, implementing, using and maintaining information systems with this approach during the last three years unveals the following startling results:

a. overall information system design and implementation in approximately half the time as compared to conventional approaches

b. overall costs for information systems analysis, design and programming requires less than half the costs as compared to conventional approaches

c. maintenance costs are less than one fifth compared to conventional approaches.

The preceding statements will become more acceptable after having read all the following chapters.

2.1 THE BASIC ASSUMPTION OF THIS KIND OF INFORMATION SYSTEMS

Information systems can be distinguished into three categories namely:

- formatted
- unformatted
- combination of formatted and unformatted.

The question may arise: what is a formatted information system? A precise answer to this question is however only possible when we have described some of the essential concepts of an information system. For now, we may define very loosely that a formatted information system is an information system in which one does not treat text. Examples of such formatted information systems are payroll, accounting, inventory control, planning, air line reservation etc.

An unformatted information system treats texts and specifically in such systems one tries to "understand" the text. Examples of unformatted information systems are text retrieval systems like KWIC, where one searches the title (text) of publications for certain keywords.

From now on, we will restrict ourselves to formatted information systems and we will use the term information system in this sense. As a first approximation of an information system we assume that human beings make use of an information system in order to facilitate a certain kind of communication among each other. This kind of communication can be characterized by the existence of a set of rules which explicitly governs the communication and can be considered a simplified case of interhuman communication. We want to emphasize that in our point of view, any (mechanical or computerized) information system can be considered as a simplified case of such interhuman communication. It is simplified in many aspects; one dominant aspect is that an information system is usually subject to a set of well-specified rules, while another aspect is the fact

that homonyms (*) are not used in information systems. This point of view takes advantage of the results of 2500 years of endeavours of mankind to understand the phenomenon "language". This is contrasted to the more generally used approaches in information systems, which mainly take advantage of the last 25 years of computer science. In other words, this is a natural language based, or linguistic approach, which means that, in principle, each human being can be made aware of and understand. Experience has shown during the last three years of teaching this approach to users that users indeed do not have the feeling they have to learn something isoteric or artificial. It is very essential in this approach that the point of origin is a certain kind of natural language communication, something familiar to users, while in the more conventional approaches the computer is the point of origin.

At this point, the reader may come to the question: but what is meant by *information* in this paper? We may answer this as follows:

Information is a set of one or more related sentences, which satisfy a certain set of rules or grammar.

Let us give a few examples of information in this sense:

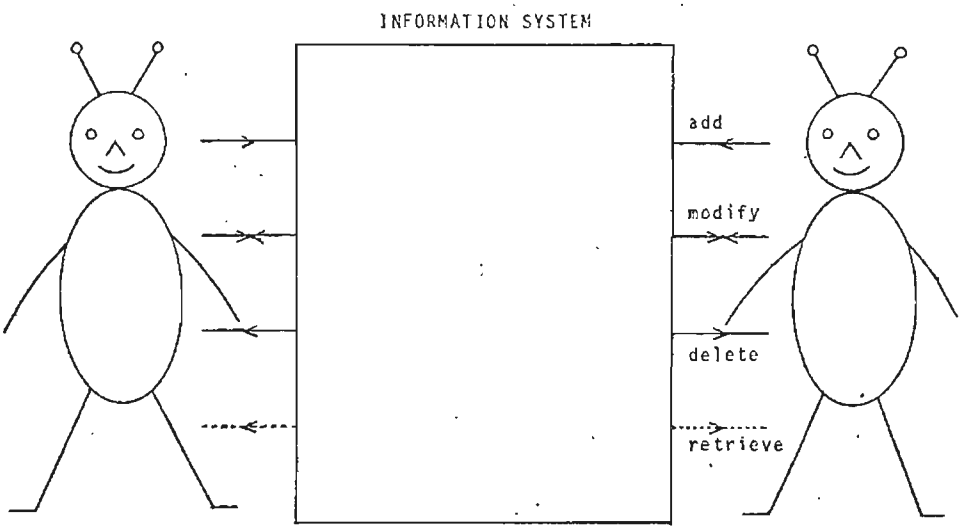
a. President Kennedy is born in Massachusetts.
b. President Eisenhower is born in Texas.
c. Department D1 has a current-budget of $ 60.000.
d. Employee E01 works for department D1 and has a salary of $ 20.000.
e. Employee E02 works for department D1 and has a salary of $ 15.000.
f. Professor Miller gives at 9o'clock a lecture in hall L51, and at 10o'clock in hall L40.
h. President Kennedy married Jacqueline in 1953.

We may then consider an information system as a black box (see figure 2.1.a) which helps to facilitate communication among two or more persons.

(*) A homonym is a name referring to two or more things.

The contents of a black box may be studied in more detail by analyzing the results of the interactions with it. What are the possible kind of (basic) interactions users can have with such an information system? Our answer is

a. a person may *add* information to the information system
b. a person may *modify* information in the information system
c. a person may *delete* information from the information systen
d. a person may *retrieve* information from the information syste



first approximation

Figure 2.1.a

It is essential to our philosophy that all traffic between a human being and the information system consist of information. If we then take the definition of information as given before in this section, we may say that all traffic consists of a set of sentences. Without any loss we may consider a sentence to consist of a set of atomic sentences. We then may say that all the traffic between the human beings and the information system consist of a set of atomic sentences. If we now abstract for a minute of authorization aspects, it may become clear that we need a set of rules which define, in terms easy to understand by the user, which kind of traffic is considered of interest, or said otherwise, which sets of atomic sentences may enter and reside in the information system.

At this point in the discussion it is good to specify very clearly how the reality comes into the picture. In figure 2.1.b we want to illustrate that the things we can record in an information system are the (atomic) sentences expressed by the user. We cannot record in the information system the direct observation of the reality, but only the "transformed" observation, where transformed means here that the user transforms a certain phenomenon in the reality into one or more atomic sentences.

For the discussion it is not essential that the observer and transformer is a human being or a device which replaces the human being as is illustrated in figure 2.1.c. In figure 2.1.c we want to record the temperature of the water of a certain basin at each hour of the day; we therefore use a device consisting of a clock (date and time) and thermometer, which are programmed in such a way that the traffic between the mechanical observer and transformer and the information system can be considered as atomic sentences expressing the temperature at a certain hour. Examples of such atomic sentences issued by the programmed clock-thermometer are given in figure 2.1.d.

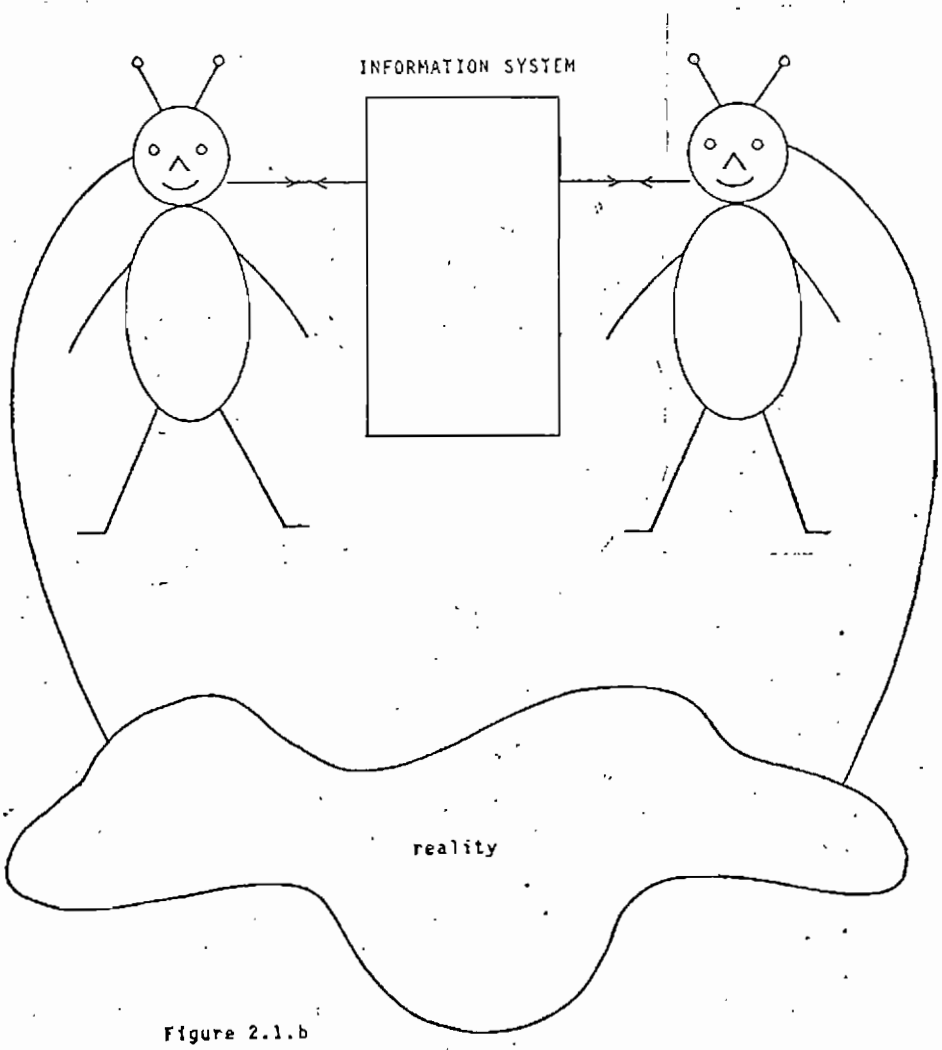


Figure 2.1.b

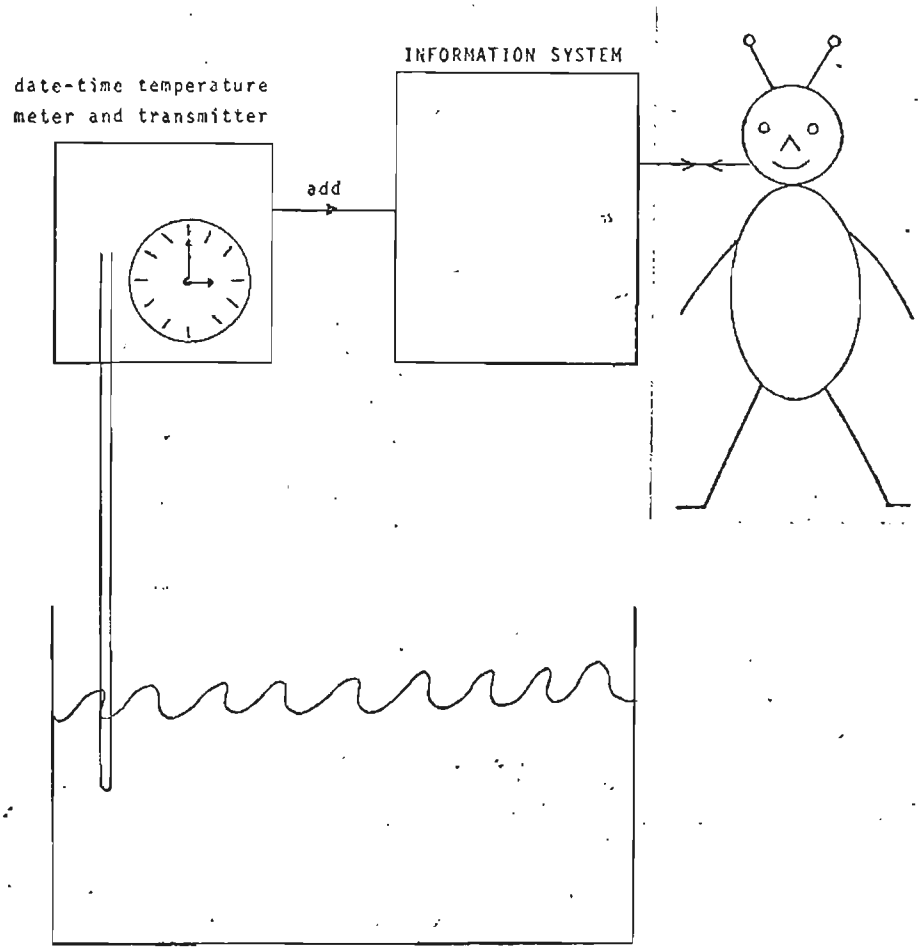

INFORMATION SYSTEM
date-time temperature
meter and transmitter
add

Figure 2.1.c

TIME		TEMPERATURE
1 SEP 1978	12.00	18.5
1 SEP 1978	13.00	18.5
1 SEP 1978	14.00	18.5
1 SEP 1978	19.00	17.9
1 SEP 1978	23.00	17.2
2 SEP 1978	02.00	16.9
2 SEP 1978	12.00	18.4

Figure 2.1.d

If there would be no programmed clock-thermometer but a user with a watch and a thermometer, the user would communicate to the information system the following:

At 12.00 o'clock of September 1, 1978, the temperature was 18.5
At 13.00 o'clock of September 1, 1978, the temperature was 18.5
At 19.00 o'clock of September 1, 1978, the temperature was 17.9

2.2 THE THREE MAJOR COMPONENTS OF AN INFORMATION SYSTEM

In this section we will refine the black box into a model which is less abstract by analyzing the effects of the four kinds of basic interactions, which users may have with the information system. We will then see that the black box consists of three related components.

In order to better put the emphasis on the nature of the three major components of an information system, we will start with a very simple information system. In this information system there are two users, called P1 and P2, who agree to set up an information system which will only deal with one kind of atomic sentences. Each atomic sentence instance of their interest conveys the knowledge that a certain president of the U.S.A. is born in a certain state, belonging to the United States of America.

Suppose that by some means (later it will be clear how this is done) that the current "contents" of the information system is as presented in figure 2.2.a.

The interpretation we give to the contents of figure 2.2.a is as follows:

President KENNEDY was born in the state MASSACHUSETTS
President NIXON was born in the state CALIFORNIA
President ADAMS J was born in the state MASSACHUSETTS

Remark. An interpretation which makes more aspects explicit which are supposed to be interpreted, unspoken, in the same way, would be:
The president, referred to by the name KENNEDY, was born in the state referred to by the name MASSACHUSETTS.

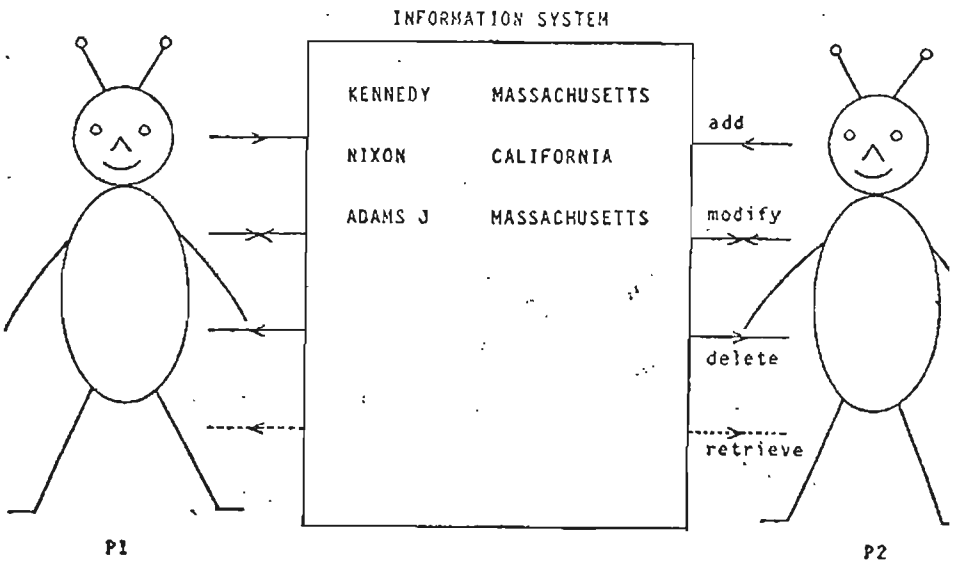


Figure 2.2.a

We will now discuss the interactions with the information system which may change the "contents" of an information system.

The reason that we start with interactions which change the contents of an information system and not with the interactions which retrieve from the contents of the information system as is done in the Normalized Relational approach, is the fact that we place the highest priority on correctness of the contents of the information system. A major source for incorrectness are the change or update interactions (although they are not the only source for errors because interpretation errors may occur at retrieval).

Person P1 may interact with the black box (of figure 2.2.a) by saying "delete from the contents of the information system that

president NIXON was born in the state CALIFORNIA" or more formatted

DELETE NIXON CALIFORNIA

The result of this interaction is that the "contents" of the information system is changed and the resulting "contents" is represented in figure 2.2.b. The "contents" of the information system could be called the "knowledge base" of the information system, or the "informationbase" of the information system. We will use the term "informationbase" to denote the "contents" or "knowledge base" of an information system.

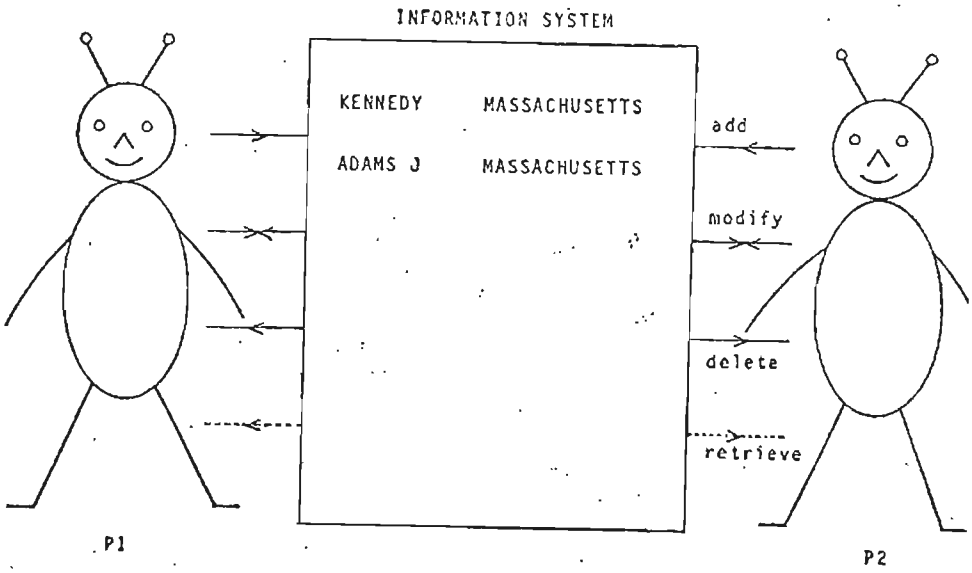


Figure 2.2.b

Person P2 may interact with the black box (of figure 2.2.b) by saying "add to the knowledge or informationbase of the information system that president EISENHOWER was born in the state GEORGIA", or more formatted:

ADD EISENHOWER GEORGIA

The result of this interaction is that the informationbase is changed and the resulting informationbase is represented in figure 2.2.c

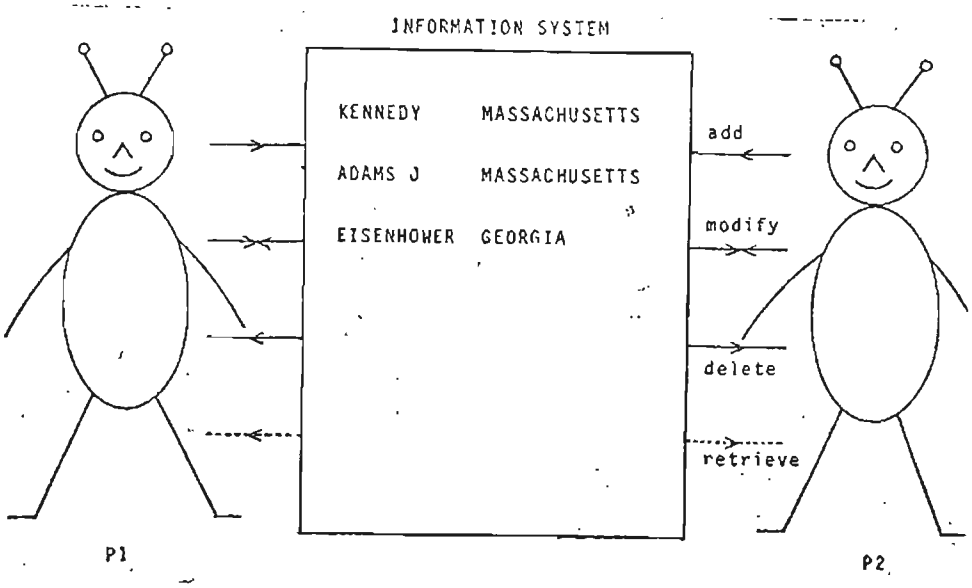


Figure 2.2.c

Person P1 may interact with the black box (of figure 2.2.c) by saying "replace in the informationbase the sentence instance that president EISENHOWER was born in the state GEORGIA with president EISENHOWER was born in the state TEXAS" or more formatted

REPLACE EISENHOWER GEORGIA INTO EISENHOWER TEXAS

The result of this interaction is that the informationbase is changed and the resulting informationbase is represented in figure 2.2.d.

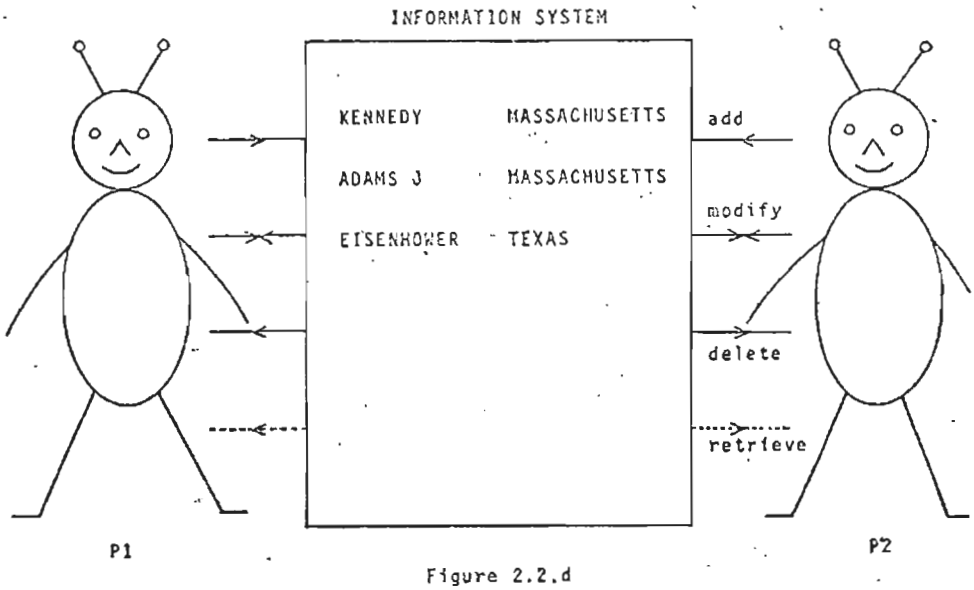

INFORMATION SYSTEM
KENNEDY
MASSACHUSETTS
ADAMS J
MASSACHUSETTS
EISENHOWER
TEXAS
add
modify
delete
retrieve
P1
P2

Figure 2.2.d

Remark. The REPLACE interaction is not a basic interaction, because it may be considered as a pair consisting of a *DELETE* and an *ADD* or replace a sentence in the informationbase with another one. However, for the time being, it may be useful to discuss three kinds of update operators until the point where we can introduce a more precise definition of an update interaction.

Up to now we have discussed examples of three of the four kinds of interactions. The common aspect of these three interactions is that the informationbase is changed as a result of the interaction.

An arbitrary combination of the three operators *ADD*, *DELETE* and *REPLACE*, is sufficient to change the contents of any information system. Please note that we have not assumed that the information system makes use of a computer or a card system.

Remark. From a conceptual point of view, it is sufficient to have update interactions which consist of an arbitrary number of ADD or DELETE operators. With this concept one can perform all necessary changes to the informationbase. However, for convenience reasons it may be useful to introduce the REPLACE operator. However, more than three update operators are neither convenient nor necessary but counter-productive. Why is it then that some of the current database management systems have eleven update operators (e.g. the CODASYL proposals)? It is clear that the majority of these update operators have nothing to do with changing the informationbase but are there for other reasons.

The remaining kind of interaction, retrieve, does not change the informationbase. The result of a retrieve interaction is that all or a part of the informationbase is presented in a specified way. E.g. Person P1 may perform a retrieve interaction on the informationbase of figure 2.2.d, and the result may be as presented in figure 2.2.e, or P2 may prefer to see it as in 2.2.f.

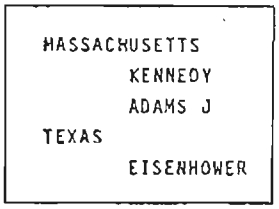


Figure 2.2.e

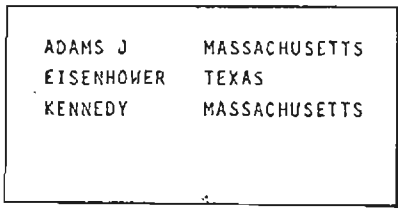


Figure 2.2.f

It is important to see that the informationbase represented by both figure 2.2.e and 2.2.f contains the same knowledge. The only difference is *how* it is represented. In figure 2.2.e the knowledge of figure 2.2.d is presented in a certain sequence and suppressing a possible echo. In figure 2.2.e the same knowledge is just presented in another sequence.

By analyzing the four kinds of interactions on the black box, we have encountered two of the three major components of an information system, namely:

- informationbase
  (or knowledge base
  or contents of the information system)
- interaction
  (or application program)
  which can be distinguished in
  - update interactions
    (the informationbase changes as a result of the update interaction)
  - retrieval interaction
    (the informationbase remains the same)

The third major component of an information system is introduced by asking some questions on some interactions and informationbases. E.g. Suppose that user P1 wants to add to the informationbase of figure 2.2.d the sentence instance that BEGIN was born in the state of POLAND.

As we have said before, in this informationbase we have agreed to collect sentence instances which tell us which president of the U.S.A. is born in which state belonging to the U.S.A. As we all know mister BEGIN is not a president of the U.S.A., nor is POLAND a state which belongs to the U.S.A. Hence, there must exist a set of rules, known to the system and the user, which eliminates unwanted sentence instances.

Another question is the following: would we accept an informationbase with the following contents (see figure 2.2.g).

KENNEDY	MASSACHUSETTS
KENNEDY	TEXAS
KENNEDY	CALIFORNIA
ADAMS J	MASSACHUSETTS
EISENHOWER	TEXAS

Figure 2.2.g

If we assigned to each pair of the informationbase of figure 2.2.g, the meaning that the first name refers to a person who has been president and who has visited the state, referred to by the second name, then the combination of the interpretation and the informationbase makes sense.

However, if we assigned to each pair of that same informationbase the meaning that the first name refers to a person who has been president and who has been born in the state, referred to by the second name, then the informationbase contains nonsense (because a president can be born in only one state). In other words, while we keep the contents the same, for one interpretation it makes sense, for another it is nonsense. It is clear that the "interpretation" is an essential part of an information system, therefore, the informationbase and interactions are not meaningful, unless we specify what the meaning is of each pair of names in the informationbase and which rules have to be followed (such as maximum one birthstate per person) by the contents of the informationbase.

The persons who work with the information system have to agree on an interpretation and a set of rules which will be enforced on each instance of the informationbase, independent of which person is performing the interaction. Such a set of rules has a close analogy with the concept of prescriptive grammar as used in linguistics, or the concept of agreement or contract used in other areas. Such a set of rules specifies which informationbase instances are permitted and which informationbase instances may follow other instances and what each element means. The name for this set of rules could be called the Conceptual Schema, or Universe of Discourse Schema, and is defined as follows:

A conceptual schema is a set of rules describing *which* information may enter and reside in the informationbase, and the *meaning* of the information in the informationbase.

This definition of the term "conceptual schema" is somewhat at variance with the meaning more commonly used in todays database management literature. To emphasize, it is our notion that the conceptual schema corresponds with a prescriptive grammar for an information system and completely describes the possible dynamic behaviour of the informationbase. In other words, it describes

the set of all permissable informationbase instances, and the set of all permissable couples
⟨informationbase instances, successor informationbase instances⟩.
In order to avoid some misunderstanding, we clearly state that our notion of grammar is not confined to grammatical aspects but includes also all the semantical aspects.

Starting from the simple model as used in figure 2.1.a and using common sense, top down, reasoning, we have arrived at the second, more detailed model of an information system, with three major components. This model of an information system, as shown in figure 2.2.h consists of:

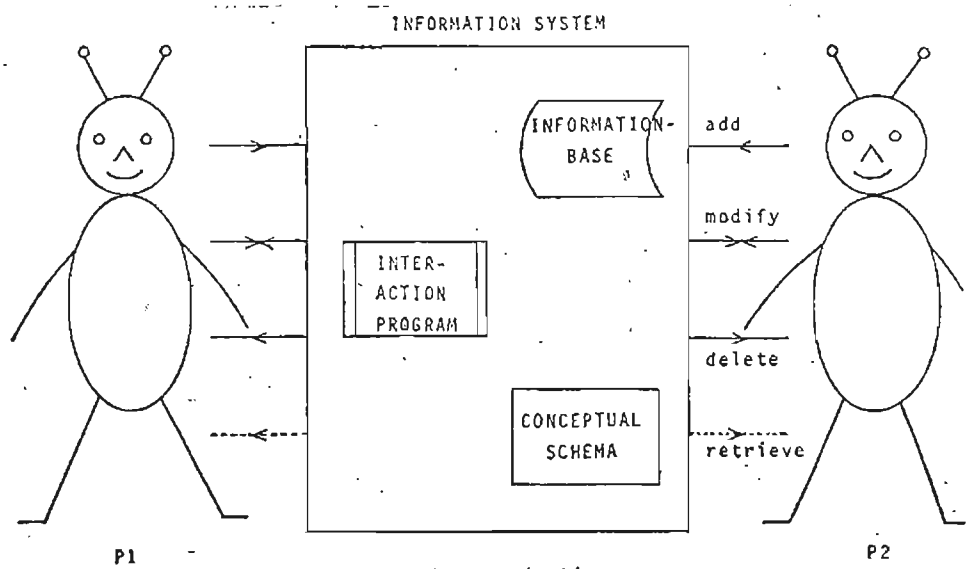
- interaction program
  which may retrieve from or change the
- informationbase,
  according to the rules described in the
- conceptual schema.

Later, we will deal with questions such as: how to describe a conceptual schema and the concepts to be used. For now, it is sufficient to repeat that a conceptual schema describes completely and formally *which* contents of the informationbase are permitted, which contents may follow a given contents, and what each element means. The conceptual schema contains no rules which describe *how* the information is stored on magnetic media and *how* it is presented to a program.

Thus far we have used the term *informationbase*. As we have seen in the example of the figures 2.2.a, 2.2.b, 2.2.c and 2.2.d, an informationbase can have different states, a different contents or status at varying times. To put emphasis on the state of an informationbase, we use the word *instantiation of an informationbase*. This term is selected because of the close analogy between this concept and the concept of tokens in natural language, in which some tokes are called a sentence instantiation (analogous with informationbase instantiation) satisfying a sentence type

(analogous with conceptual schema). Said otherwise, an instantiation of an informationbase is the state of an informationbase at a given instant in time. If we want to stay closer in line with some terms used in current database technology, we could see that an *instance of an informationbase*, or an *occurrence of an informationbase* are synonyms for an *instantiation of an informationbase*.

Before we proceed with the introduction of programming, and computer aspects, it may be good to have as an interludium some example to illustrate the three major components of an information system, and to analyse the consequences of our meaning of conceptual schema, namely a complete prescriptive grammar, for the notion of unit of update.



second approximation

Figure 2.2.h

2.3 EXAMPLES TO ILLUSTRATE THE THREE MAJOR COMPONENTS OF AN INFORMATION SYSTEM

We have seen in the previous section that a conceptual schema defines completely the *what* of an information system and the *meaning* of the information and this "*what*" and "*meaning*" must be understandable not only to the experts in information systems, databases or programming, but also to the users and managers. In short, we may say that a conceptual schema need to be

- complete
- easy to understand
- easy to formulate
- formal
- easy to change
- all constraints explicit

Our basic assumption with respect to information systems has been stated in section 2.1, namely all traffic between the human being and the information system is in terms of information, that means in terms of sentence instances. In other words, the only manipulatable (or communicatable) construct between the user and the information system is a sentence instance, which is the same as a set of one or more related atomic sentence instances. What does this mean for the conceptual schema? The answer is that the conceptual schema has to describe which sentence instances may go between the user and the information system, but this for each permitted contents or instance of the informationbase. The way to express such a "traffic regulator" is to describe the

- atomic sentence types
  and
  . the object types referred to in the elementary sentences
  and
  . constraints applied to atomic sentence populations.

Because of our basic assumption made in section 2.1 it is clear that our conceptual schema does not describe (normalized) relations, or records and links between records and structures within a record (CODASYL). Informally speaking, our conceptual schema describes which sets of atomic sentence instances may go from the user to the informationbase or may be deleted by the user from the informationbase. The example we have used in section 2.2 deals with one atomic sentence type which could be given the name PRESIDENTS-BIRTHSTATE in which two object types are referred to, namely PRESIDENT and STATE and one constraint specifying that a specific PRESIDENT can appear at most once in a sentence instantiation of the sentence type PRESIDENTS-BIRTHSTATE in any given instantiation of this informationbase. Formally, this conceptual schema could be described as on page 27.

The conceptual schema called EXAMPLE-ONE consists of three divisions. In our approach each conceptual schema consists of three divisions, which may informally be described as follows:

The OBJECT DIVISION describes the object types comprising all the possible object instances about which we are interested to have some knowledge in the informationbase.

The ELEMENTARY SENTENCE DIVISION describes the elementary sentence types covering all the possible elementary sentence instances we want to have in the informationbase.

The CONSTRAINTS DIVISION (arbitrarily) restricts the informationbase possibilities described in the ELEMENTARY SENTENCE DIVISION.

The conceptual schema called EXAMPLE-ONE defines completely which instantiations of informationbases are permitted. It is very useful, besides the conceptual schema text, to use a diagram to communicate major parts of the conceptual schema. Such diagrams are given the name "information type diagram".

```
CONCEPTUAL SCHEMA NAME IS EXAMPLE-ONE.

OBJECT DIVISION.

    OBJECT TYPE NAME IS PRESIDENT,
                NAME LENGTH IS 14 CHARACTERS.
    OBJECT TYPE NAME IS STATE,
                NAME LENGTH IS 20 CHARACTERS.

    COMMENT.  THE NAME LENGTH DECLARATION IS NOT ESSENTIAL
              IN THIS STAGE.  END COMMENT.

ELEMENTARY SENTENCE DIVISION.

    SENTENCE TYPE NAME IS PRESIDENTS-BIRTHSTATE
        REFERENCED OBJECT TYPE NAME IS PRESIDENT,
            ROLE IS NATIVE-SON-OF
        REFERENCED OBJECT TYPE NAME IS STATE,
            ROLE IS BIRTH-STATE

CONSTRAINT DIVISION.

    CONSTRAINT NAME IS AT-MOST-ONE-BIRTHSTATE
                CODE IS AB26
        ROLE NATIVE-SON-OF OF SENTENCE PRESIDENTS-BIRTHSTATE
                              IS UNIQUE.

END CONCEPTUAL SCHEMA.
```

Remark 1. It has been stated earlier that a conceptual schema describes "the *meaning* of the information in the informationbase".

The question has been raised: how can the (other) user know what the meaning is of the role "*BIRTH-STATE*"? Behind this question is the assumption that the conceptual schema describes the absolute meaning. From philosophy and linguistics we know that it is impossible to describe the absolute meaning, and the concept meaning as used in the definition of the conceptual schema is used in this sense. In other words, with a conceptual schema one tries to describe the meaning as far as possible. Consequently, we have to assume that users have a common interpretation of such things as "BIRTH-STATE", "PRESIDENT", etc.

As of today, we cannot assume that a computer has any interpretation of the names of the object type names, sentence names, role names or constraint names; however the computer has the same interpretation for such things as object types, sentences, roles, "IS UNIQUE" etc.

Remark 2. The OBJECT DIVISION can be considered as a special semantic constraint, a retrieval constraint. It namely serves to help avoid some nonsense questions.

Why do we present the OBJECT DIVISION as the first division in the conceptual schema? One would expect that the SENTENCE DIVISION would be the first division, because the sentence is the most important construct of the conceptual schema.

The reasons for first presenting the OBJECT DIVISION have to do with checking the correctness of the conceptual schema in an on-line situation. By first presenting the OBJECT DIVISION it is possible to check each declaration of a sentence type independently of any future declaration.

Figure 2.3.a contains the information type diagram corresponding to the schema called EXAMPLE-ONE.

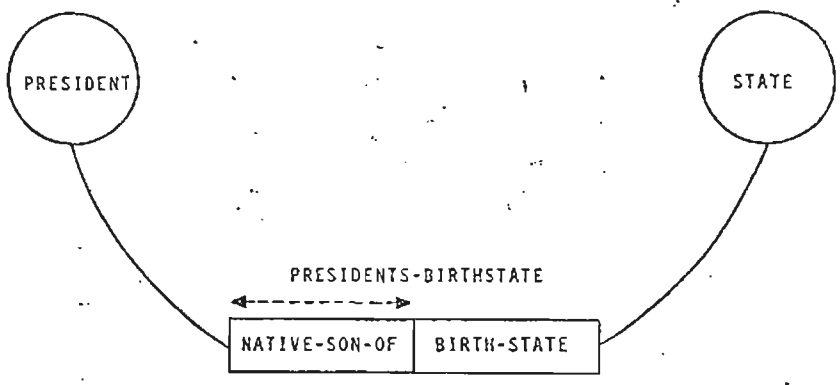


Figure 2.3.a

Please compare the elements of figure 2.3.a with the text of the conceptual schema EXAMPLE-ONE. We will later introduce in more detail the information type diagrams; very briefly, circles denote object types, and one or more attached rectangles a sentence type (please note that a sentence type consists of a set of one or more roles, and each rectangle corresponds with one role) and the dotted arrow denotes the uniqueness constraint.

The meaning of a uniqueness constraint is that the names of the objects referred to from the roles which are the subject of the uniqueness constraint appear at most once in each informationbase instance.

Another kind of diagram is very useful in discussions and that is the so-called combined information type-population diagram. The type-population diagram corresponding with the instantiation of the informationbase as presented in figure 2.2.d is as is presented in figure 2.3.b.

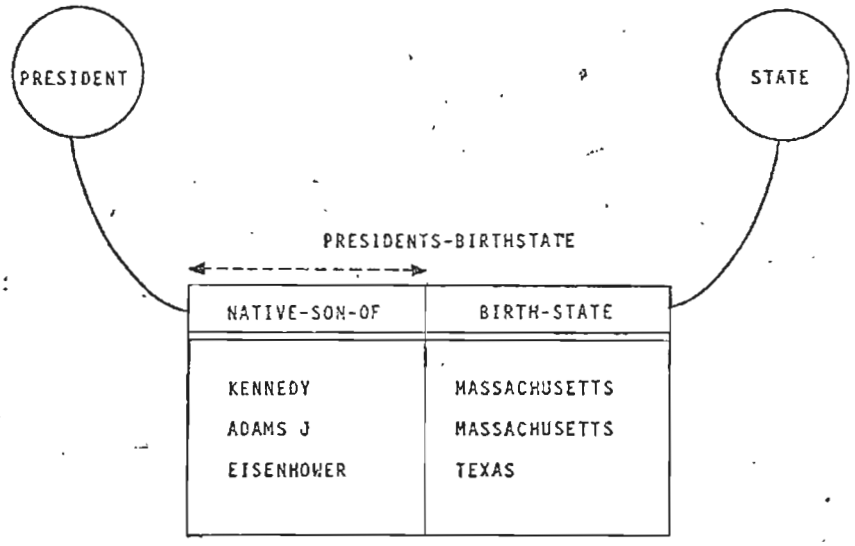


Figure 2.3.b

In figure 2.3.b, the portion above the double line corresponds with the conceptual schema and the portion under the double line corresponds with one informationbase. Two of the three major components of an information system are presented in an information type-population diagram.

If we now recall that in our approach the conceptual schema defines only and completely all the correctness aspects of an informationbase (and consequently the interaction or application program

has no effect on the correctness of the informationbase) it is clear that an information type-population diagram is a powerful communication aid among users as well as between users and information systems experts.

Sometimes it is not clear that an informationbase is dynamic and the conceptual schema is static. The conceptual schema is static as long as the agreement is not changed. To stress this dynamic aspect and at the same time to illustrate the three major components of an information system, we will simulate the build up of the instance of the informationbase as presented in figure 2.2.a and the informationbase instances of figures 2.2.b, 2.2.c and 2.2.d.

Figure 2.3.c presents an information type-population diagram denoting an empty informationbase satisfying the conceptual schema called EXAMPLE-ONE.

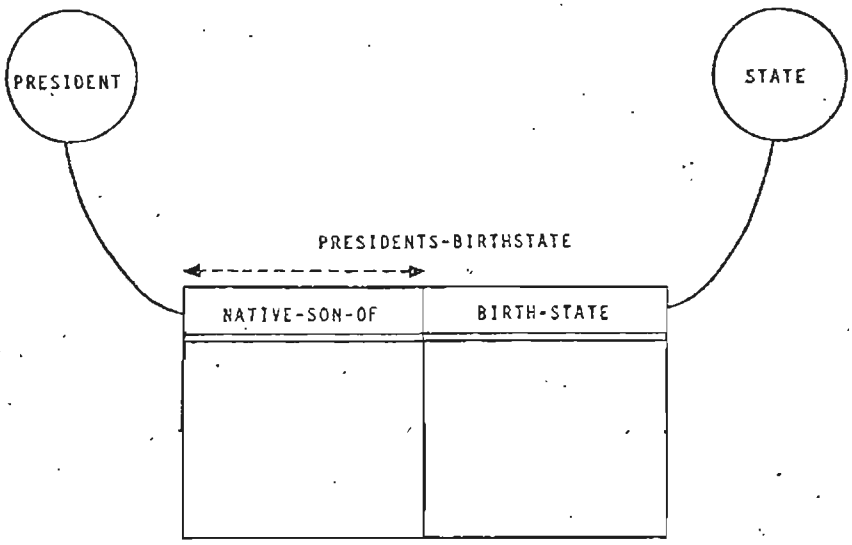


Figure 2.3.c

If we now apply the interaction

ADD PRESIDENTS-BIRTHSTATE KENNEDY MASSACHUSETTS

to the instantiation of the informationbase of figure 2.3.c, we get figure 2.3.d.

If we want to be very complete we would have to say the following:

> ADD to the informationbase instance at time t, under control of the conceptual schema called EXAMPLE-ONE, the sentence instance KENNEDY MASSACHUSETTS of the sentence type PRESIDENTS-BIRTHSTATE.

Furthermore, in the stylized interaction

ADD PRESIDENTS-BIRTHSTATE KENNEDY MASSACHUSETTS

it is assumed that we somehow have indicated to the system that we want to have interaction with the current informationbase instance satisfying the conceptual schema called EXAMPLE-ONE. We could do so by once issuing, preceding all other interactions, the statement

INTERACTION WITH INFORMATIONBASE OF EXAMPLE-ONE

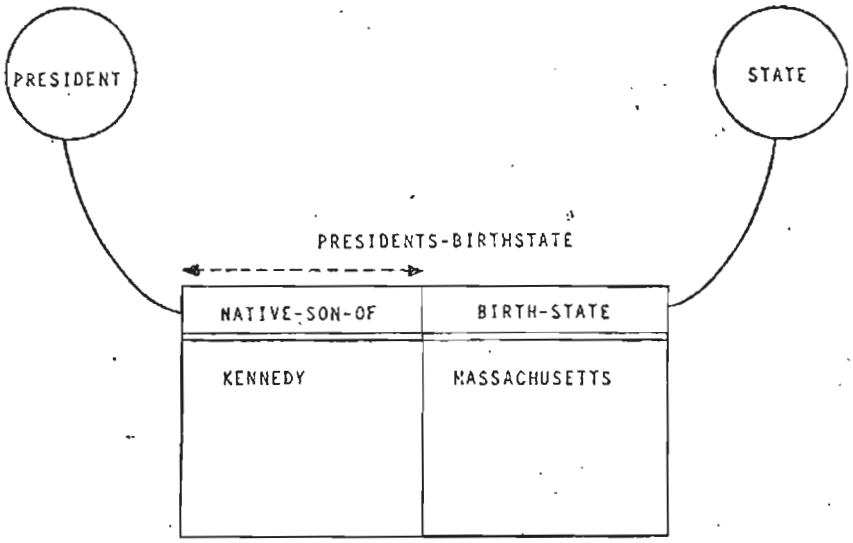


Figure 2.3.d

If we now apply the interaction

ADD PRESIDENTS-BIRTHSTATE NIXON CALIFORNIA

to the instantiation of the information base of figure 2.3.d, we get figure 2.3.e.

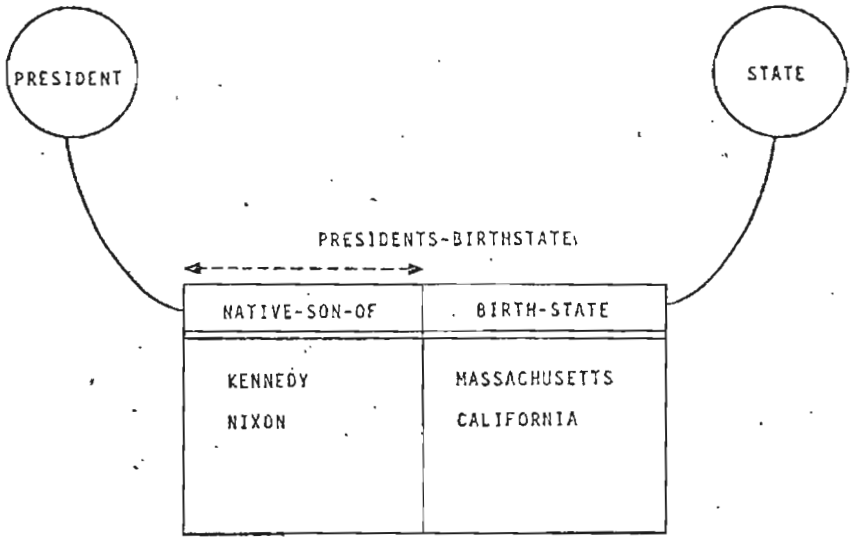


Figure 2.3.e

If we now apply the interaction

ADD PRESIDENTS-BIRTHSTATE ADAMS J MASSACHUSETTS

to the instantiation of the information base of figure 2.3.e, we get figure 2.3.f (compare with 2.2.a).

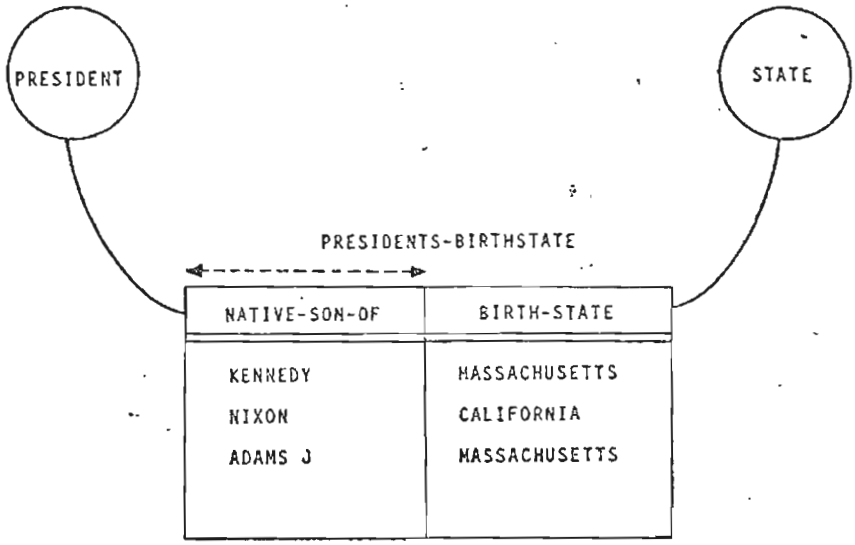


Figure 2.3.f

If we now apply the interaction

DELETE PRESIDENTS-BIRTHSTATE NIXON CALIFORNIA

to the instantiation of the information base of figure 2.3.f, we get figure 2.3.h (compare with 2.2.b).

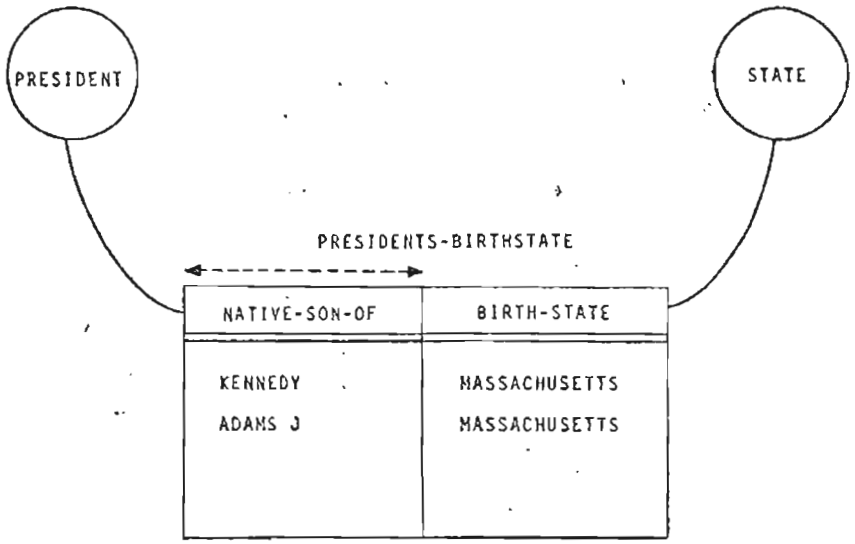


Figure 2.3.h

If we now apply the interaction

ADD PRESIDENTS-BIRTHSTATE EISENHOWER GEORGIA

to the instantiation of the information base of figure 2.3.h, we get figure 2.3.i (compare with 2.2.c).

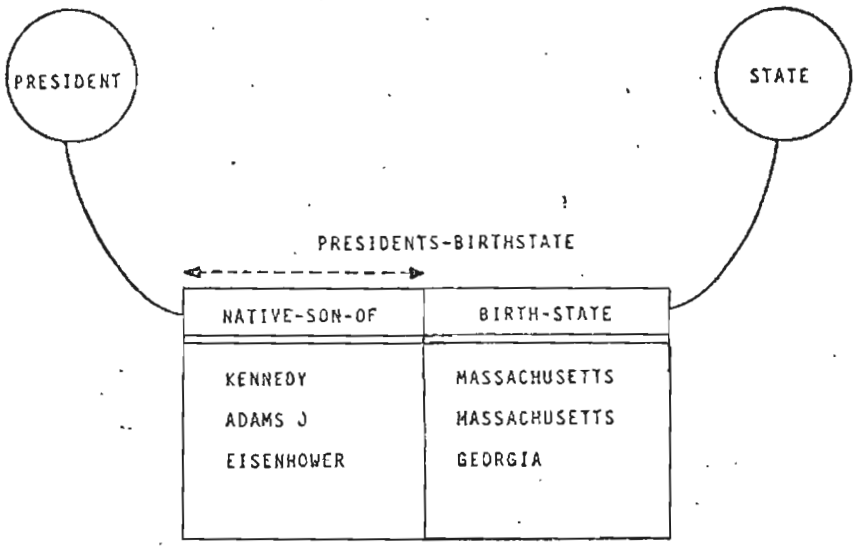


Figure 2.3.i

If we now apply the interaction

REPLACE PRESIDENTS-BIRTHSTATE EISENHOWER GEORGIA WITH
EISENHOWER TEXAS

to the instantiation of the information base of figure 2.3.i, we get figure 2.3.j (compare with 2.2.d).

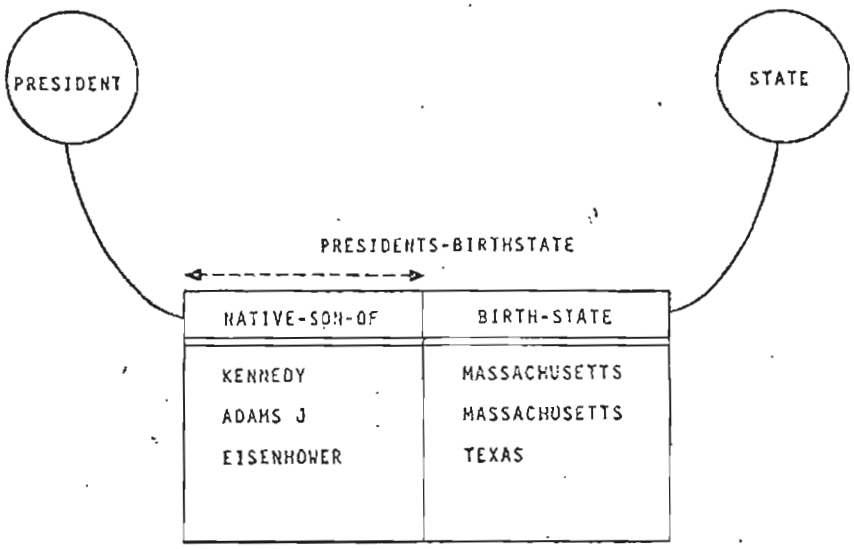


Figure 2.3.j

This series of examples has illustrated how the three major components of an information system are related to each other. Please note that in all these examples from 2.3.c through 2.3.j the conceptual schema was the same, and the instantiations of the informationbase were dynamically changed by the interaction according to the rules in the conceptual schema. This latter aspect (contract enforcement by the conceptual schema) may be illustrated with a last example.

Let us apply the interaction

ADD PRESIDENTS-BIRTHSTATE KENNEDY CALIFORNIA

to the instantiation of the informationbase of figure 2.3.j. The conceptual schema will reject this interaction. If it had accepted this interaction, then it would have resulted in an instantiation of an informationbase as in figure 2.3.k.

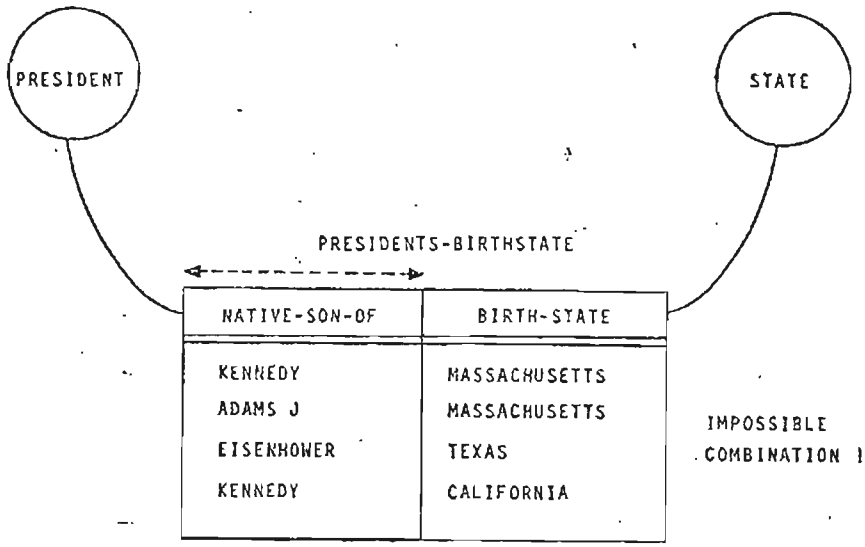


Figure 2.3.k

The informationbase of figure 2.3.k violates the constraint in the conceptual schema specifying that a president can at most have one birthstate (expressed by the constraint named AT-MOST-ONE-BIRTHSTATE in the constraints division of the conceptual schema EXAMPLE-ONE, or represented by the arrow above the role NATIVE-SON-OF) in each instance of an informationbase satisfying the conceptual schema EXAMPLE-ONE.

In this case, the system would respond to the interaction program:

INTERACTION REJECTED BECAUSE OF VIOLATION OF CONSTRAINT
CODED AB26

2.4 CONCEPTUAL UPDATE OPERATORS AND UNIT OF INTERACTION

In the previous two sections we have seen that an informationbase can be changed by an update interaction, and we have introduced three update operators

ADD elementary-sentence-instance
DELETE elementary-sentence-instance
REPLACE elementary-sentence-instance

In the previous two sections, each interaction consisted of just one ADD, or one DELETE or one REPLACE of an elementary sentence instance. One should not conclude from this that a unit of interaction consists always of only one ADD or one DELETE, or one REPLACE of an elementary sentence instance. In several realistic environments, it is even impossible to ever populate an informationbase, if one restricts the unit of interaction to just one ADD, or one DELETE, or one REPLACE of an elementary sentence instance. We may illustrate this with the following examples (see also figure 2.4.a which contains the information type diagram for this example).

We are interested in the current budget of each department, and we want to know for which department an employee works, and his salary; the following constraints apply to this example, namely a department has only one current budget, an employee works for only one department and has only one salary; if the salary of an employee is known, and, if the employing department of an employee is known, then the salary of that employee must be known; and an employee can only work for a department if the department has a budget, and (for our discussions now the following constraints is most important) the sum of the salaries of all the employees in a given department is more than 30 percent and less than 70 percent of the current budget of that department.

The conceptual schema which formally describes this example is given at page 42.

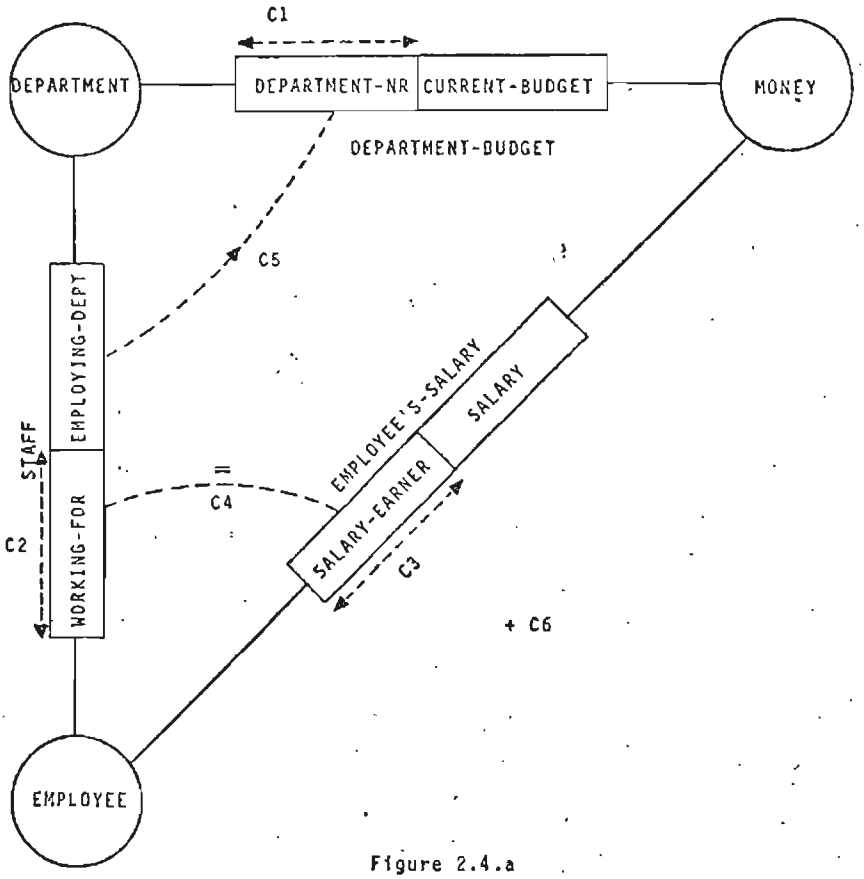


Figure 2.4.a

Note that the information type diagram contains only symbols for the constraints C1, C2, C3, C4 and C5. The constraint C6 is not denoted by a symbol in the information type diagram. In general, the information type diagram only contains symbols for a limited number of kinds of constraints.

```
CONCEPTUAL SCHEMA NAME IS EXAMPLE-TWO

OBJECT DIVISION

    OBJECT TYPE NAME IS DEPARTMENT
                NAME LENGTH IS 2 CHARACTERS.
    OBJECT TYPE NAME IS EMPLOYEE
                NAME LENGTH IS 3 CHARACTERS.
    OBJECT TYPE NAME IS MONEY
                NAME LENGTH IS 5 DIGITS.

ELEMENTARY SENTENCE DIVISION

    SENTENCE TYPE NAME IS DEPARTMENT-BUDGET.
        REFERENCED OBJECT TYPE NAME IS DEPARTMENT.
            ROLE IS DEPARTMENT-NR
        REFERENCED OBJECT TYPE NAME IS MONEY,
            ROLE IS CURRENT-BUDGET.

    SENTENCE TYPE NAME IS STAFF.
        REFERENCED OBJECT TYPE NAME IS DEPARTMENT,
            ROLE IS EMPLOYING-DEPT.
        REFERENCED OBJECT TYPE NAME IS EMPLOYEE,
            ROLE IS WORKING-FOR.

    SENTENCE TYPE NAME IS EMPLOYEE'S-SALARY.

        REFERENCED OBJECT TYPE NAME IS EMPLOYEE,
            ROLE IS SALARY-EARNER.
        REFERENCED OBJECT TYPE NAME IS MONEY,
            ROLE IS SALARY.
```

```
CONSTRAINTS DIVISION.

    CONSTRAINT NAME IS ONE-BUDGET-PER-DEPARTMENT.
               CODE IS C1.
        ROLE DEPARTMENT-NR IN SENTENCE DEPARTMENT-BUDGET IS UNIQUE.

    CONSTRAINT NAME IS ONE-EMPLOYING-DEPARTMENT-PER-EMPLOYEE.
               CODE IS C2.
        ROLE WORKING-FOR IN SENTENCE STAFF IS UNIQUE.

    CONSTRAINT NAME IS ONE-SALARY-PER-EMPLOYEE.
               CODE IS C3.
        ROLE SALARY-EARNER OF SENTENCE EMPLOYEE'S-SALARY IS UNIQUE.

    CONSTRAINT NAME IS SALARY-AND-EMPLOYMENT-ALWAYS-TOGETHER
               CODE IS C4.
        ROLE SALARY-EARNER OF SENTENCE EMPLOYEE'S-SALARY
             IS EQUAL TO
        ROLE WORKING-FOR OF SENTENCE STAFF

    CONSTRAINT NAME IS EMPLOYERS-MUST-HAVE-A-BUDGET
               CODE IS C5
        ROLE EMPLOYING-DEPT IN SENTENCE STAFF
             IS SUBSET OF
        ROLE DEPARTMENT-NR IN SENTENCE DEPARTMENT-BUDGET

    CONSTRAINT NAME IS DEPARTMENTS-SALARY-BETWEEN-30-AND-70-PERCENT-
                                                        OF-BUDGET
               CODE IS C6
        LET DEPARTMENT-SALARY BE DEPT-NR (REFERRING TO DEPARTMENT),
                                 DEPT-SAL (REFERRING TO MONEY).
        FOR EACH EMPLOYING-DEPT
        LET DEPT-SAL OF DEPARTMENT-SALARY
            BE SUM SALARY OF EMPLOYEE OF DEPARTMENT
        LET DEPT-SAL OF DEPARTMENT-SALARY BE MORE THAN 30 .
            AND LESS THAN SEVENTY PERCENT OF BUDGET
            OF CORRESPONDING DEPARTMENT-BUDGET.

END CONCEPTUAL SCHEMA.
```

If we would adopt the philosophy that an interaction would consist of just one ADD, one DELETE or one REPLACE on an elementary sentence instance, then let us analyse if we can populate an information base satisfying the conceptual schema named EXAMPLE-TWO, or informally described in the previous paragraph.

Before we start this it is good to consider figure 2.4.b in which is represented an information type-population diagram; please verify that this population satisfies all the rules of the conceptual schema called EXAMPLE-TWO. We will now try to build up this permitted population.

When the information base is empty, there are in the one-sentence-instance-at-a-time update interaction only three possibilities to start the filling of the information base, namely ADD an instance of

a. DEPARTMENT-BUDGET
b. STAFF
c. EMPLOYEE'S-SALARY

a. Suppose the information base is empty, and we want to start with adding a sentence instance of the kind DEPARTMENT-BUDGET, say 60 000 D1. If we would add this sentence instance to the empty information base then we would get an information base as represented in figure 2.4.c. Let us now check whether this information base satisfies all the pertinent constraints of conceptual schema EXAMPLE-TWO. Constraint C1, which prescribes that a department can have at most one current-budget is satisfied. The constraints C2, C3, C4 and C5 are not involved. However, constraint C6 is not satisfied; hence this population is not acceptable Let us analyse why not. The sum of the salaries of the employees working for D1 is zero, and zero is not between 18.000 (30% of 60.000) and 42.000 (70% of 60.000). Hence we can not start filling the database with a sentence instance of the kind DEPARTMENT-BUDGET.

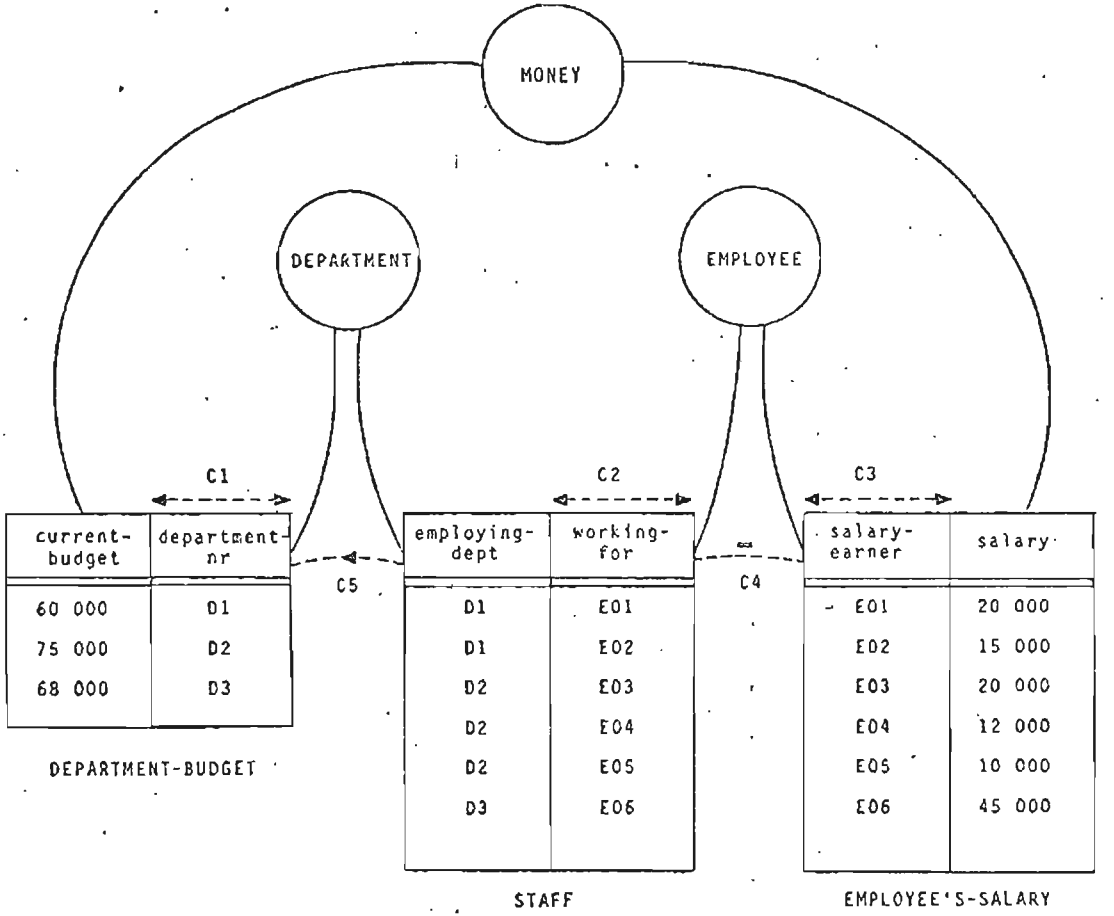

MONEY
DEPARTMENT
EMPLOYEE
C1
C2
C3
C4
C5
current-budget
department-nr
60 000
75 000
68 000
D1
D2
D3
DEPARTMENT-BUDGET
employing-dept
working-for
D1
D1
D2
D2
D2
D3
E01
E02
E03
E04
E05
E06
STAFF
salary-earner
salary
E01
E02
E03
E04
E05
E06
20 000
15 000
20 000
12 000
10 000
45 000
EMPLOYEE'S-SALARY

Figure 2.4.b

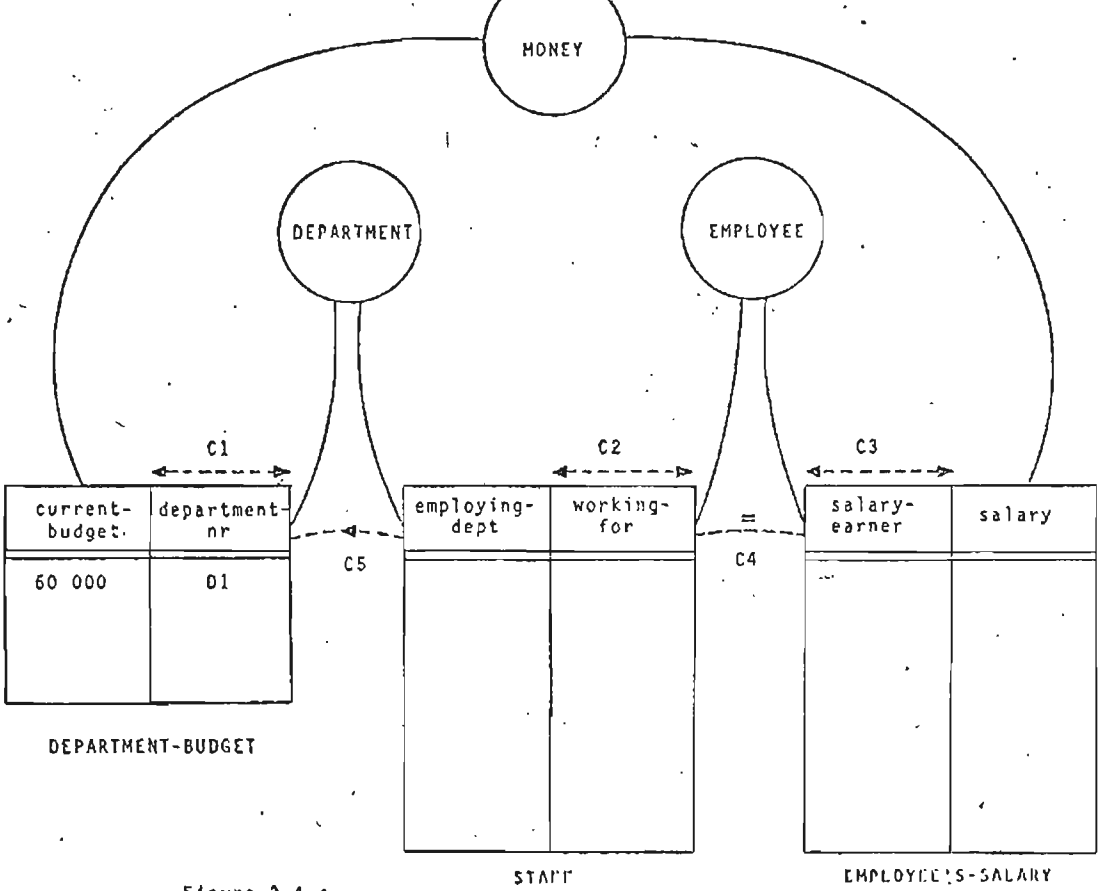

MONEY
DEPARTMENT
EMPLOYEE
C1
C2
C3
current-budget
department-nr
60 000
01
C5
employing-dept
working-for
=
C4
salary-earner
salary
DEPARTMENT-BUDGET
STAFF
EMPLOYEE'S-SALARY

Figure 2.4.c

b. Suppose the informationbase is empty and we want to start with adding a sentence instance of the kind STAFF, say D1 E01. If we would add this sentence instance to the empty information base, then we would get an information base as represented in figure 2.4.d. Let us now check whether this information base satisfies all the pertinent constraints. The constraint C2, which prescribes that an employee can at most work for one department, is satisfied. The constraints C1, C3 and C6 are not involved. The constraint C5, which specifies that we only accept a STAFF elementary sentence instance, if the employing department referred to in that elementary sentence of STAFF has a current budget (that is an elementary sentence instance of the type DEPARTMENT-BUDGET must exist in which department-nr is the same as employing-dept), is not satisfied. Furthermore, the constraint C4, which specifies that employing-dept and salary both must be known is not satisfied. Hence this population is not acceptable.

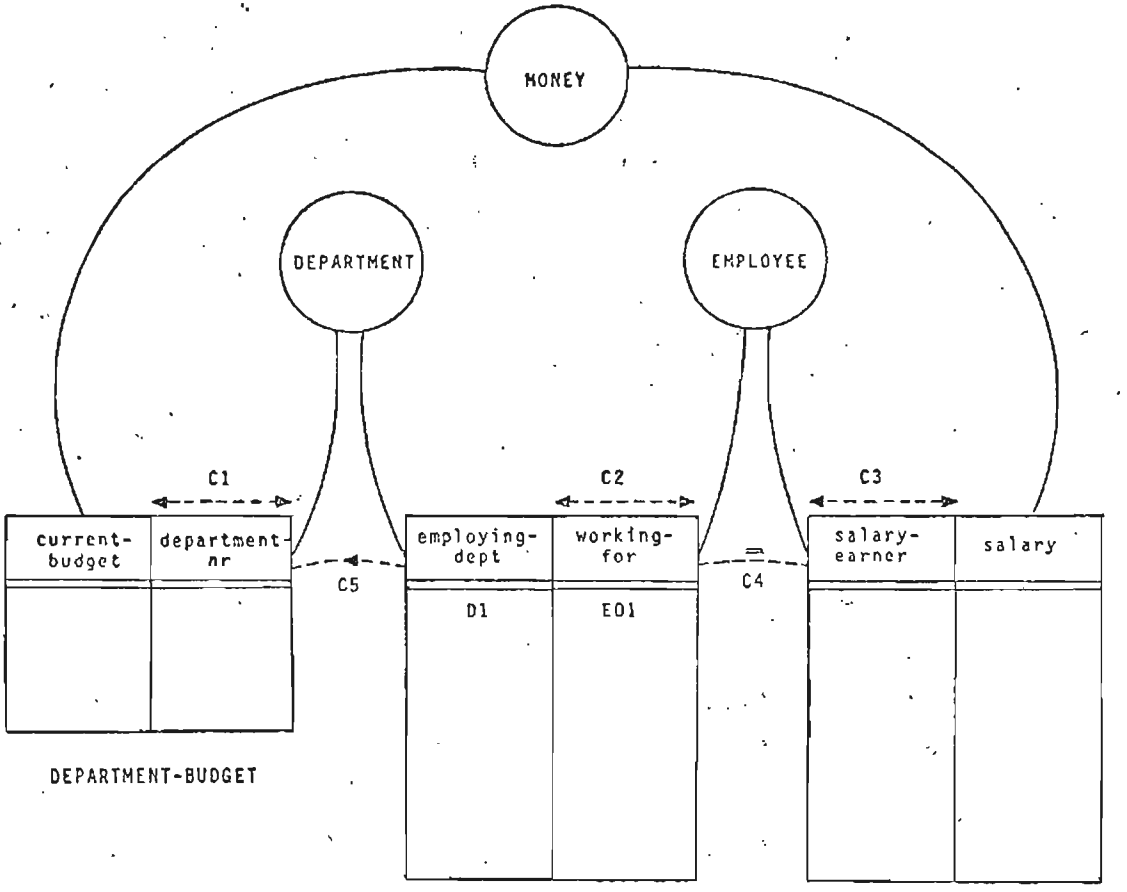

MONEY
DEPARTMENT
EMPLOYEE
C1
C2
C3
current-budget
department-nr
employing-dept
working-for
salary-earner
salary
C5
C4
D1
E01
DEPARTMENT-BUDGET
STAFF
EMPLOYEE'S-SALARY

Figure 2.4.d.

c. Suppose the informationbase is empty, and we want to start with adding a sentence instance of the kind EMPLOYEE'S-SALARY, say E01 20.000. If we would add this sentence instance to the empty informationbase, then we would get an informationbase as represented in figure 2.4.e. Let us check whether this informationbase satisfies all the pertinent constraints. C3 is satisfied, C1, C2 and C5 are not involved, but both C4 and C6 are not satisfied. Hence this population is not acceptable. Why not? There is no department with a budget, hence 20.000 cannot be between 30 percent and 70 percent of nothing.

d. Let us now perform an update which consists of the following unit:

```
BEGIN
    ADD EMPLOYEE'S-SALARY     E01   20.000
    ADD STAFF                 D1    E01
END
```

(This is the unit of update capability which is available in the current generation of database management systems, and this unit of update is sometimes referred to as one-record-occurrence-at-a-time-update-logic However, in the current generation database systems this is not described by a BEGIN and END but, because of the fixed number of atomic sentences involved, by a record. In this case it would be something like EMPLOYEE (E01, 20.000, D1)). If we would bring into the empty informationbase these two elementary sentence instances in one step, then we would get an informationbase as represented in figure 2.4.f. Does this informationbase satisfy all the pertinent constraints? C2, C3 and C4 are satisfied, C1 is not involved but C5 and C6 are not satisfied. C5 is not satisfied because department D1 has no budget, and C6 is not satisfied because 20.000 cannot be between 30 percent and 70 percent of nothing. Hence this population is not acceptable.

Remark. Please note that with the update capabilities of the current generation of database management systems, one cannot populate an informationbase which has to satisfy

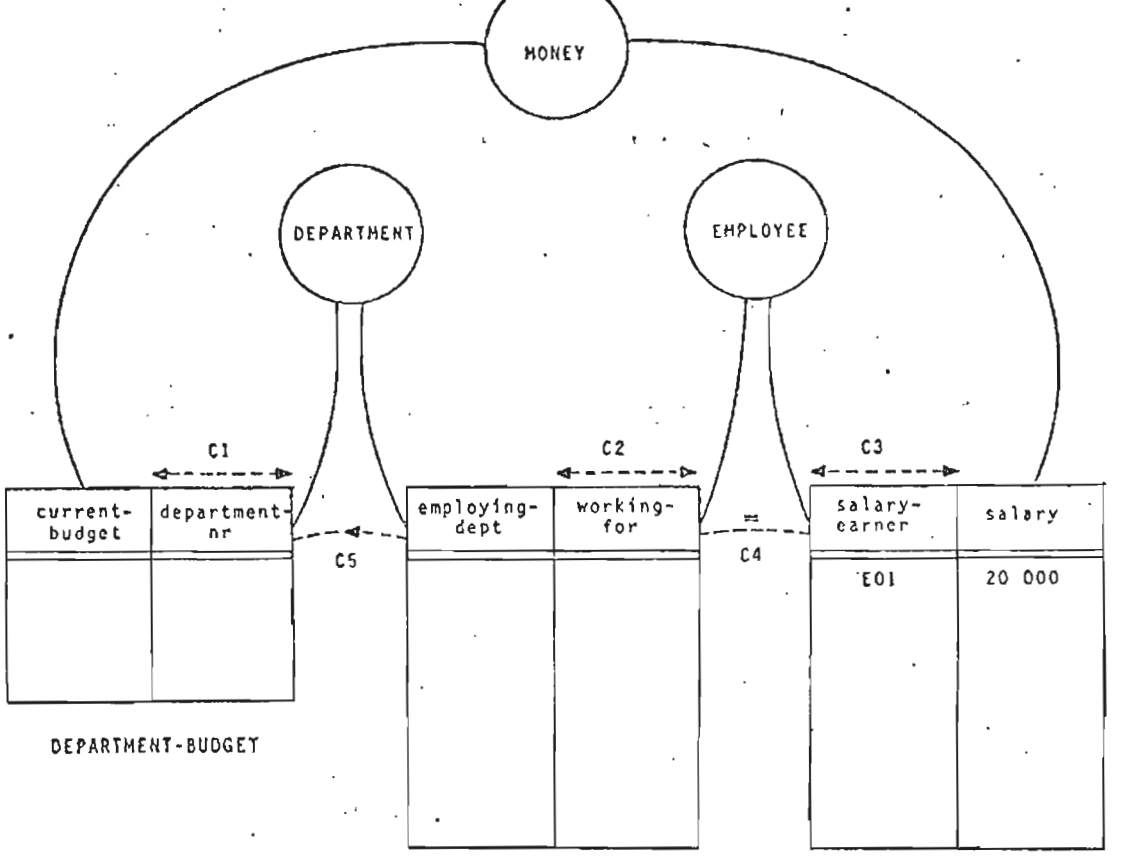

MONEY
DEPARTMENT
EMPLOYEE
C1
C2
C3
current-budget
department-nr
employing-dept
working-for
salary-earner
salary
C5
C4
=
E01
20 000
DEPARTMENT-BUDGET
STAFF
EMPLOYEE'S-SALARY

Figure 2.4.e

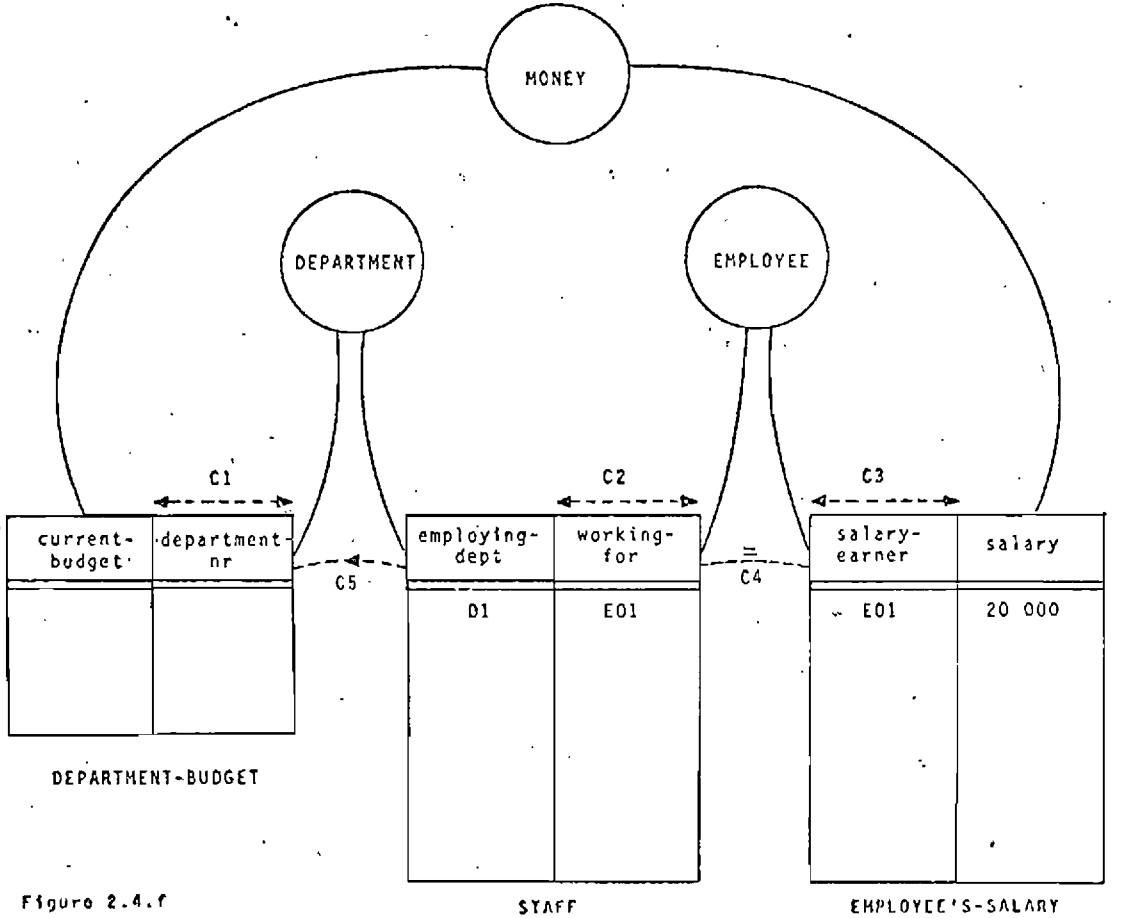

MONEY
DEPARTMENT
EMPLOYEE
C1
C2
C3
current-budget
department-nr
employing-dept
working-for
salary-earner
salary
C5
C4
D1
E01
E01
20 000
DEPARTMENT-BUDGET
STAFF
EMPLOYEE'S-SALARY

Figure 2.4.f

the conceptual schema of EXAMPLE-TWO. The question may arise: but problems with a kind of conceptual schema like EXAMPLE-TWO do exist in many practical information systems of today. How does one solve this then today? Well, the answer is that the constraint C6 is removed from the conceptual schema and is incorporated in the update program. This means that an update program is partly responsible for the correctness of the information base, and the conceptual schema is responsible for the remaining part. This is the biggest problem to enforcing correctness with the current generation of database management systems.

Let us repeat that the information base represented in figure 2.4.b satisfies the conceptual schema called EXAMPLE-TWO, on the other side, we have seen that we could not fill the information base with the one-sentence-instance-at-a-time update logic, nor with one-record-instance-at-a-time update logic.

To build realistic information systems with all constraints checked in the conceptual schema and none in the application program or interaction, a unit of interaction is needed which may contain an arbitrary number of ADD or DELETE of elementary sentence instances even of different sentence kinds. Let us call such an interaction a conceptual transaction. We may define a conceptual transaction as follows:

A conceptual transaction is a set of one or more pairs, where each pair consists of an ADD or DELETE of an elementary sentence instance.

We now require that the information base is in accordance with the conceptual schema at the start (begin) and at the finish (end) of the conceptual transaction, and in between the information base need not be in accordance with the conceptual schema. Please do not forget that a user may only contact an information base if it is in accordance with the conceptual schema. It is clear that this will bring

a heavy burden on the people who have to make an implementation of a database management system which can concurrently serve many users either in update or retrieve interactions.

e. Let us give one example of such a conceptual transaction:

```
CONCEPTUAL TRANSACTION.
    BEGIN.
    ADD  DEPARTMENT-BUDGET  60.000  D1
    ADD  STAFF              D1  E01
                            D1  E02
    ADD  EMPLOYEE'S-SALARY      E01  20.000
                                E02  15.000
    END.
```

Let us suppose we have an empty information base and we now add the just mentioned conceptual transaction to the empty information base. The result is as presented in figure 2.4.g and let us check all the constraints. Constraint C1 is satisfied because department D1 has only one budget. Constraint C2 is satisfied because employee E01 works for only one department and the same is true for employee E02. Constraint C3 is satisfied because both employee E01 and E02 have at most one salary. Constraint C4 is satisfied, because there is both the STAFF and EMPLOYEE'S-SALARY information of employee E01 and E02. Constraint C5 is satisfied because employee E01 and E02 work both for a department which has a budget. Constraint C6 requires that the sum of the salaries of the employees of a department is more than 30 and less than 70 percent of the budget of that department.

```
20.000 + 15.000 = 35.000
30% of 60.000   = 18.000
70% of 60.000   = 42.000
```

Hence constraint C6 is satisfied. This means that all constraints are satisfied and thus that the conceptual transaction is accepted.

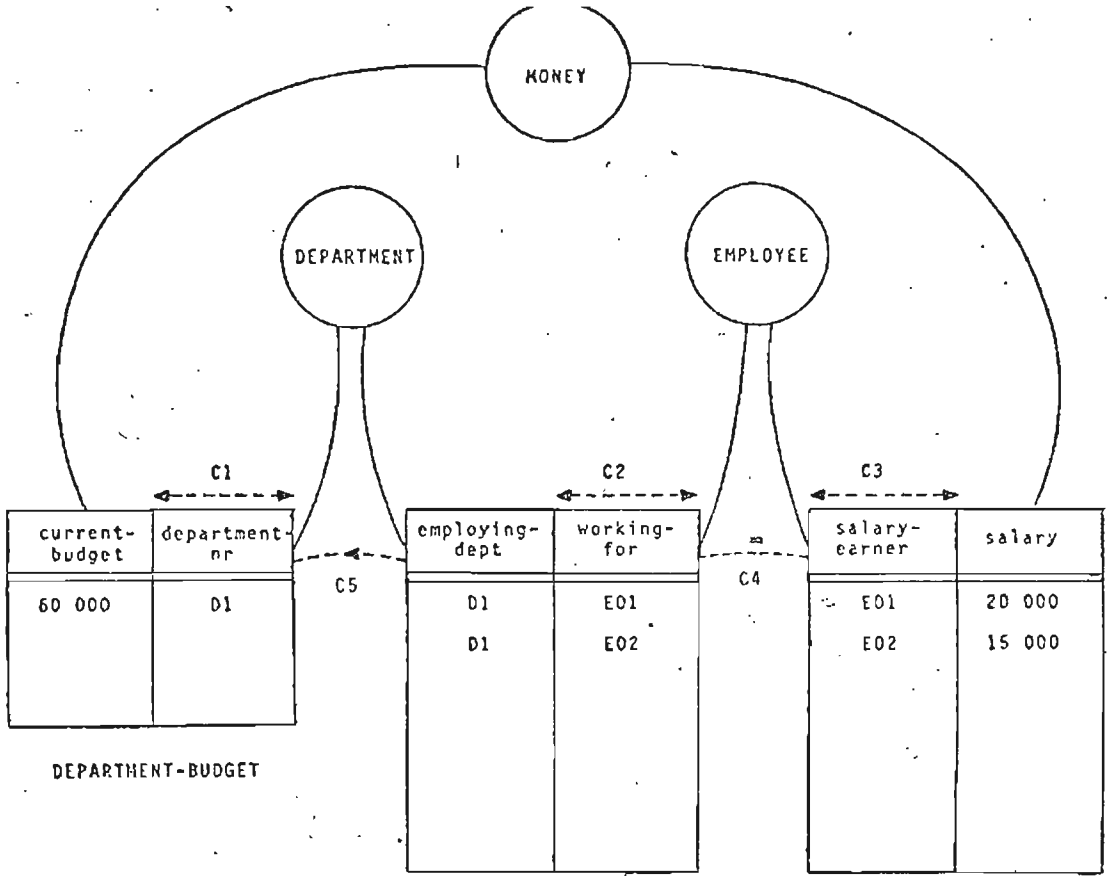

MONEY
DEPARTMENT
EMPLOYEE
C1
C2
C3
C4
C5
current-budget
department-nr
60 000
D1
DEPARTMENT-BUDGET
employing-dept
working-for
D1
E01
D1
E02
STAFF
salary-earner
salary
E01
20 000
E02
15 000
EMPLOYEE'S-SALARY

Figure 2.4.9

The concept of "conceptual transaction" is fundamental in our approach and is in my opinion essential if we put a high priority upon the total correctness (sometimes called integrity in the current database management jargon) of the informationbase.

If we denote an informationbase instance by the symbol IB, and the conceptual transaction by CT, then we may describe the following equation:

$$IB\ (t+1) = IB\ (t) + CT\ (t)$$

where both informationbase instances IB (t+1) and IB (t) as well as the transition IB (t)→IB (t+1) are permitted by the conceptual schema CS.

To summarize our approach, we say that an information system consists of three major components, the conceptual schema, the informationbase and the interaction. In our approach the conceptual schema defines completely the possible behaviour of the informationbase. The conceptual schema in our approach is an INformation SYstems GRAMmer, and we therefore have given this approach the name INSYGRAM. One consequence of the INSYGRAM approach is central control over the entire (not just a part like in current database management systems) correctness of the informationbase.

Remark on text and diagram format of the conceptual schema

Furthermore, from this example we may learn that an information type diagram does sometimes contain less specifications than does the conceptual schema in text form. This is generally the case with text and diagram communication. The major advantage of a diagram is that it concentrates on some (often the most important) aspects of a text and gives a visible "picture". All object types and elementary sentence types have a corresponding symbol in the information type diagram,

some forms of constraints have a symbol in the diagram, but the more complex constraints are more easily expressed in text form.

Remark on implementation

At compile time it is possible to check that the type of the conceptual transaction is too small; that is, no matter which elementary sentence instances are populating the conceptual transaction, it is impossible with any conceptual transaction of this type to be able to go from one (permitted) information-base instance to another (permitted) informationbase instance.

The minimum size of the transaction type is determinable from the constraints in the conceptual schema.

2.5 STATE AND TRANSITION CONSTRAINTS

As we have seen before, a conceptual schema governs completely the behaviour of the informationbase over time, said otherwise the conceptual schema describes the set of all possible informationbase instances as well as the set of all couples ⟨informationbase instance, successor informationbase instance⟩.

We have furthermore seen that informationbase instances consist of populations of atomic sentence instances which are subject to the constraints of the conceptual schema.

For reasons of better understanding it is useful to distinguish the conceptual schema constraints into two categories, namely

- state
  and
- transition

constraints.

Examples of state constraints are all the constraints C1, C2, C3, C4, C5 and C6 of the conceptual schema called EXAMPLE-TWO of section 2.4.

We will now give an example of a conceptual schema in which there is a transition constraint. In order to focus the attention on the transition constraint, the example is kept very simple, namely we are interested in instances of one type of atomic sentence, named EMPLOYEE'S-MARITAL-STATUS, with two roles, one STATUS-HOLDER, referencing the object type EMPLOYEE and the other role STATUS-HELD, referencing the object type MARITAL-STATUS. There are 2 constraints of the category state, namely C11 and C12, while there are 4 constraints of the category transition. (See Conceptual Schema EXAMPLE-THREE at page 59).

Figure 2.5.a contains an information type diagram, while figure 2.5.b contains an illustration of the 4 transition constraints.

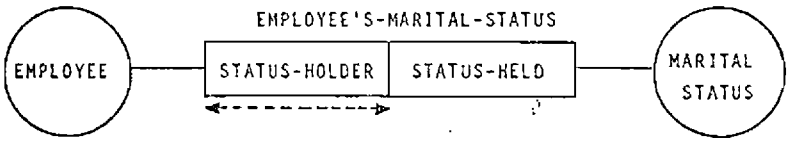
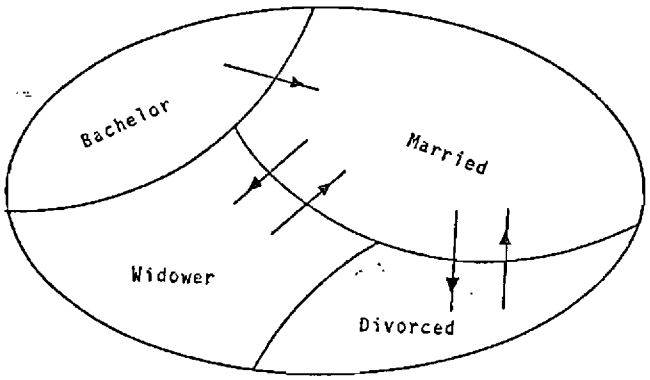


Figure 2.5.a



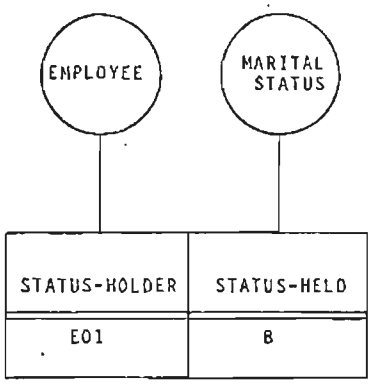
→ means permitted transition for two successive STATUS-HELD of the same STATUS-HOLDER

Figure 2.5.b

```
CONCEPTUAL SCHEMA NAME IS EXAMPLE-THREE

OBJECT DIVISION

    OBJECT TYPE NAME IS EMPLOYEE
        NAME LENGTH IS 3 CHARACTERS
    OBJECT TYPE NAME IS MARITAL-STATUS
        NAME LENGTH IS 1 CHARACTER.

ATOMIC SENTENCE DIVISION

    SENTENCE TYPE NAME IS EMPLOYEE'S-MARITAL-STATUS
        REFERENCED OBJECT TYPE NAME IS EMPLOYEE
            ROLE IS STATUS-HOLDER
        REFERENCED OBJECT TYPE NAME IS MARITAL-STATUS
            ROLE IS STATUS-HELD

CONSTRAINTS DIVISION

    CONSTRAINT NAME IS AT-MOST-ONE-MARITAL-STATUS-PER-EMPLOYEE
             CODE IS C11
        ROLE STATUS-HOLDER IN
             SENTENCE EMPLOYEE'S-MARITAL-STATUS
                 IS UNIQUE.

    CONSTRAINT NAME IS ONLY-ONE-OF-THESE-FOUR
               CODE IS C12
        ROLE STATUS-HELD IN
             SENTENCE EMPLOYEE'S-MARITAL-STATUS
                 MUST BE B OR M OR W OR D.
        COMMENT.  B MEANS BATCHELOR
                  M MEANS MARRIED
                  W MEANS WIDOWER
                  D MEANS DIVORCED
        END COMMENT.
```

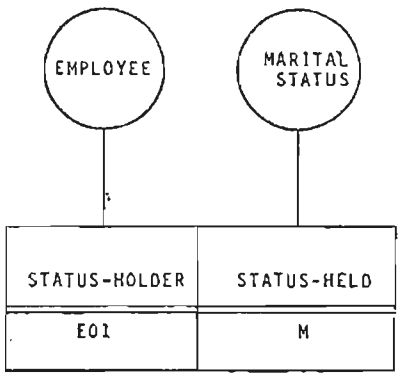
```
CONSTRAINT NAME IS FROM-BATCHELOR-TRANSITION.
           CODE IS C13
    ROLE STATUS-HELD IN
         SENTENCE EMPLOYEE'S-MARITAL-STATUS
            HAS AS PERMITTED TRANSITIONS
            FROM B TO M.

CONSTRAINT NAME IS FROM-WIDOWER-TRANSITION.
           CODE IS C14
    ROLE STATUS-HELD IN
         SENTENCE EMPLOYEE'S-MARITAL-STATUS
            HAS AS PERMITTED TRANSITIONS
            FROM W TO M.

CONSTRAINT NAME IS FROM-MARRIED-TRANSITION.
           CODE IS C15
    ROLE STATUS-HELD IN
         SENTENCE EMPLOYEE'S-MARITAL-STATUS
            HAS AS PERMITTED TRANSITIONS
            FROM M TO W, TO D.

CONSTRAINT NAME IS FROM-DIVORCED-TRANSITION
           CODE IS C16
    ROLE STATUS-HELD IN
         SENTENCE EMPLOYEE'S-MARITAL-STATUS
            HAS AS PERMITTED TRANSITIONS
            FROM D TO M.
```

We may show a few type-populations diagrams to illustrate the transition constraint.



population JOHN

Figure 2.5.c



population JANET

Figure 2.5.d

The informationbase instance called JOHN satisfies all the state constraints just as JANET does. However, only the transition from JOHN to JANET is permitted, not the one from JANET to JOHN.

It certainly would have been possible to describe in one constraint all the four constraints described in C13 through C17. This could have been done as follows:

```
CONSTRAINT NAME IS ALL-PERMITTED-MARITAL-STATUS-TRANSITIONS
           CODE IS C21
    ROLE STATUS-HELD IN
         SENTENCE EMPLOYEE'S-MARITAL-STATUS
            HAS AS PERMITTED TRANSITIONS
            FROM B TO M
            FROM M TO W
            FROM M TO D
            FROM W TO M
            FROM D TO M.
```

The result of this kind of combining constraints is that declaring them is faster, but, and this is often more important, the constraint code delivered to the application program has a lower "information value" or has less resolution as compared to providing of the codes C13 through C16.

Remark. In order to make it possible to "shorten" the writing of the transition declarations one could make the following conventions:

1. If all transition constraints pertaining to a role are of the PERMITTED kind, then these are the only permitted.

2. If all transition constraints pertaining to a role are of the FORBIDDEN kind, then these are the only forbidden.

3. If some transition constraints pertaining to a role are of the kind PERMITTED and some of the kind FORBIDDEN, then those not mentioned are forbidden.

2.6 THE CONCEPTS FILE, DATABASE AND INFORMATIONBASE

In this paper we have introduced the term informationbase and do not use the term database or file.
The definition of an informationbase is as follows:

An informationbase is a set of elementary sentences, satisfying all and only the rules of the conceptual schema.

This means that an application program has no influence on the correctness of the informationbase.

The concept database is used to denote a set of records and relationships among these records (CODASYL and some others) or a set of relations (relational approach) where the correctness is partly controlled by the schema and partly by the application program. In this sense a database is only a gradual step forward from files.

If we take a look at information systems using conventional file techniques, I think that some 95% of the correctness rules are embedded in the application program, and some 5% in the (file) schema. With current database management systems I think that still some 80% of the correctness rules are embedded in the application program and some 20% in the database schema. It is our aim to have 0% of the correctness rules in the application program and 100% in the conceptual schema, and for a collection of sentence instances satisfying such a conceptual schema we have introduced the name "informationbase".

name for collection of information \ % of correctness rules in	application program	schema
file	95	5
database	80	20
informationbase	0	100

3. ARCHITECTURE OF THE NEXT GENERATION DATABASE MANAGEMENT SYSTEMS

In chapter 2 we have introduced the major conceptual components of future information systems. Very essential in this architecture are the

- conceptual schema
- informationbase
- interaction (application program)

and the meaning we have given to the conceptual schema, namely a prescriptive grammar for an information system, that is, the conceptual schema defines completely the "behaviour" of the information base. The conceptual transaction may transform one information base instance into another, but only if the resulting information base and transformation from the previous to this information base is permitted by the conceptual schema.

3.1 THE COEXISTENCE ARCHITECTURE FOR DATABASE MANAGEMENT SYSTEMS

Nowhere in the derivation of chapter 2 we have introduced the concept of computer, which means that the model of an information system as described in the preceeding section is valid for all (formatted) information systems.

Let us now turn our attention to information systems in which a computer is used as a tool to improve service, or decrease costs or any other aim.

As we have seen, the conceptual schema is a set of rules describing *which* information may enter and reside in the information base, and the meaning of the elements of the information base.

So we now assume that we use a computer for the three major components (as described in section 2.2) of an information system which means

- to enforce the rules of the conceptual schema
- to store the information base
- to perform the interactions on the information base

If we concentrate for a minute on the second part of these three, the storage of the information base, it is then clear that in this era in history, the information base is stored on magnetic media. This means that we somehow need a set of rules which describe *how* the elements of the information base are physically stored on the computer's storage media.

Considerations like response time to retrieval interactions, or duration of update interactions, or safety regulations, etc. etc., require that the information of the information base is stored in a specific way on the computer's storage media. Constructions, like indexes to information, like an author and a subject index in a library, are added to the "basic" (or conceptual) information base. Such a physical information base contains exactly the same amount of knowledge as the conceptual information base, not a single bit more

However, it may contain constructions which e.g. are used to speed up retrieval (like an author and subject index in the library) or duplicate files to have a higher safety degree in case of failure of one of the physical media.

Corresponding to each conceptual informationbase is an informationbase as physically stored in the computers storage media; let us call this latter the internal informationbase or the physical informationbase.

As you remember from the previous section, the conceptual schema is a set of rules describing which conceptual informationbases are valid. As such one can consider a conceptual schema as a contract which specifies *what* is permitted in the informationbase.

For the internal informationbases one needs something similar, because one needs to specify *how* the elements of the physical informationbase are to be stored. This specification is referred to as internal schema. One may define this as follows:

An internal schema is a set of rules describing *how* information (of the conceptual informationbase) is physically represented on the storage media.

The model of figure 2.2.h containing the three major components of an information system can now be extended with the internal informationbase and internal schema as presented in figure 3.1.a.

We have seen that one can consider the conceptual schema as a grammar, which specifies which informationbase instantiations are permitted and which transitions between informationbase instantiations are permitted. Considering a conceptual schema this way, means putting some emphasis on the fact that an information system is a special subset of human communication, an area in which grammar plays an essential part.

In exactly the same way one may consider the internal schema as a grammar which specifies which internal informationbase instantiations are permitted.

If we look at figure 3.1.a, we may come to the question: how do we transform conceptual information into internal information? In figure 3.1.a there is a conceptual schema describing which conceptual informationbase instantiations are permitted and there is an internal schema describing which internal informationbase instantiations are permitted, but in figure 3.1.a is missing the set of rules describing how a conceptual construct is to be transformed into an internal construct and vice versa.

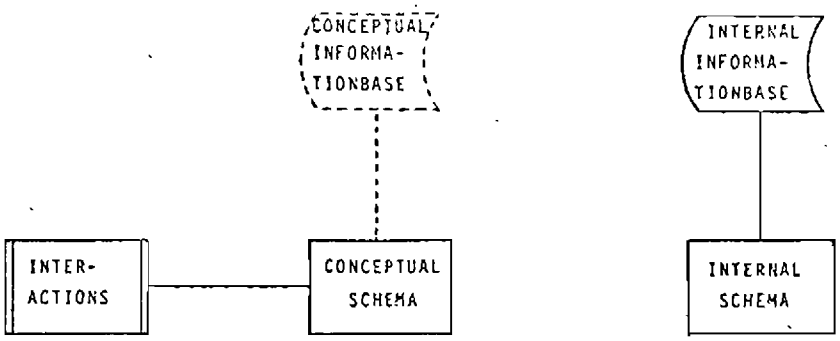


Figure 3.1.a

This set of rules need to be added to the components of figure 3.1.a as is done in figure 3.1.b. As the symbol to represent transformation rules we will use a hexagon, to represent a schema we use a rectangle, to represent an interaction we use a rectangle with double vertical lines and we use a mass storage symbol to denote an informationbase. Please note that the conceptual information base is a virtual one.

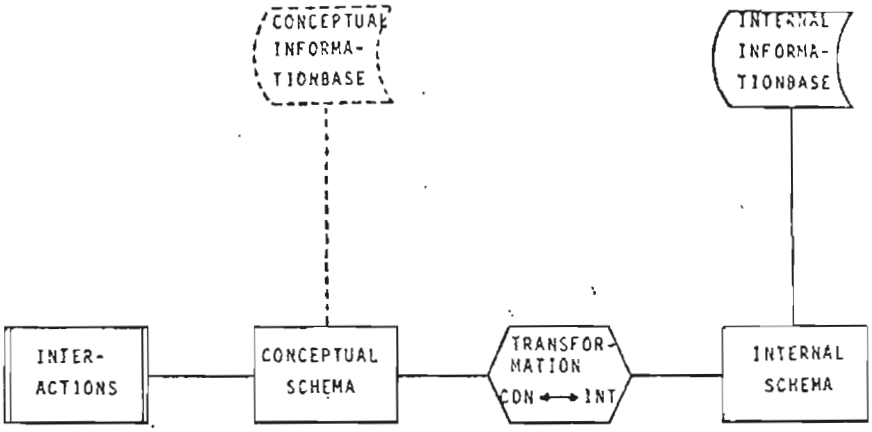


Figure 3.1.b

If we want to distribute the internal information base over more than one computer, and if we want to have an internal schema per computer, then we may extend figure 3.1.b with various internal schemata as in figure 3.1.c. However, figure 3.1.b and 3.1.c are basically the same.

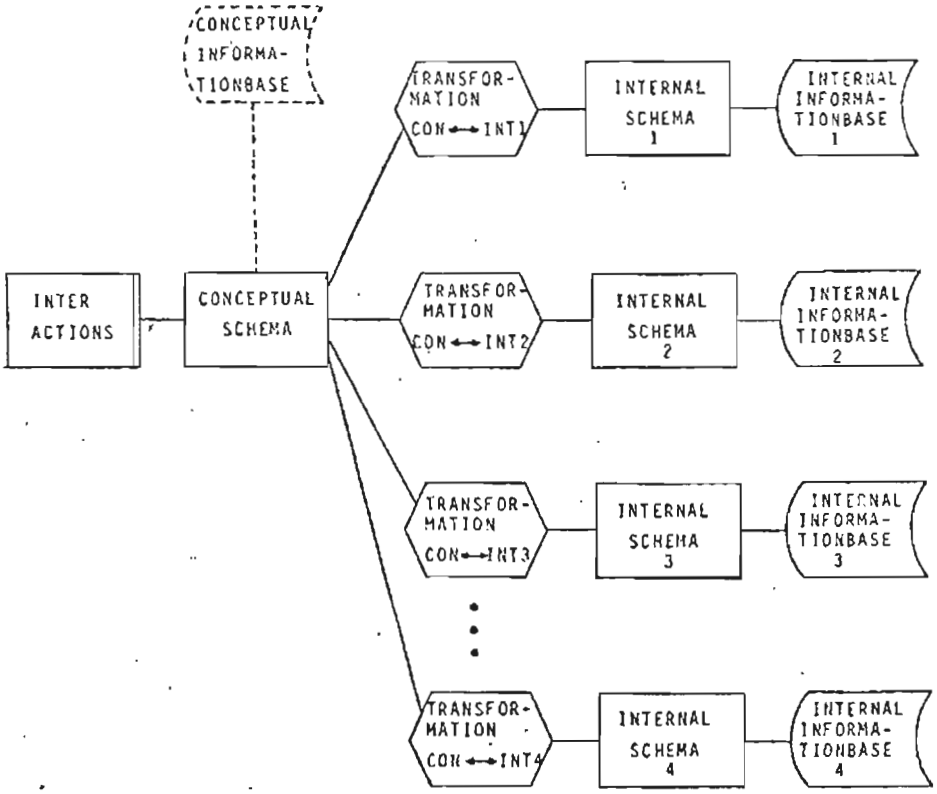

CONCEPTUAL INFORMA-TIONBASE
INTER ACTIONS
CONCEPTUAL SCHEMA
TRANSFOR-MATION CON←→INT1
INTERNAL SCHEMA 1
INTERNAL INFORMA-TIONBASE 1
TRANSFOR-MATION CON←→INT2
INTERNAL SCHEMA 2
INTERNAL INFORMA-TIONBASE 2
TRANSFOR-MATION CON←→INT3
INTERNAL SCHEMA 3
INTERNAL INFORMA-TIONBASE 3
TRANSFOR-MATION CON←→INT4
INTERNAL SCHEMA 4
INTERNAL INFORMA-TIONBASE 4

Figure 3.1.c

Let us now move our attention to the aspect of interactions. Interactions with the informationbase are described or specified in a programming language, in the broad sense; this means RPG, COBOL, FORTRAN, Relational Languages, etc. As of today, there are many different programming languages around, and each language has a certain area of use where it is selected to be the best (for whatever reasons).
Hence, it may be clear that the valuable asset, called information base, cannot be restricted to have only interactions expressed in one specific or standard language.

In other words, one has to permit that interactions may be specified in whatever programming language is selected.

Each programming language has associated with it a certain set of concepts which can be used to describe the structure (grammar) of the problem. This structure of the problem, as seen by a particular programming language, need not be the same as expressed in the conceptual schema.

The difference may be twofold. First the scope of the problem as seen by the program may be a subset of the scope as expressed in the conceptual schema. Second, the program may see that subset of the scope in different constructs, using another mental model. Such a program view is called external schema.

In analogy with the conceptual schema and the internal schema, one may consider the external schema as a grammar which - within the limits prescribed in the conceptual schema - describes which structure instances may be seen or generated by a program operating on this external schema.

One may also say that an external schema is a description which specifies *how* a user of an external schema can see a subset of each conceptual informationbase instantiation. This means that for each conceptual informationbase instantiation, the external schema defines the corresponding external informationbase instantiation.

In figure 3.1.d we have represented one program, one external schema, one conceptual schema, one transformation between conceptual and internal schema and vice versa, one internal schema and the internal informationbase. The question will arise: how are external constructs transformed into conceptual constructs? It is clear that we need a set of rules which describe how external constructs as specified in the external schema are transformed into conceptual constructs as specified in the conceptual schema. If this transformation is added to figure 3.1.d, we get figure 3.1.e.

We may conclude that there may exist many different program views, and on a given program view there may operate more than one set of interactions (or program). Figure 3.1.f is an illustration of the many external schemas and programs approach.

Figure 3.1.d

Figure 3.1.e

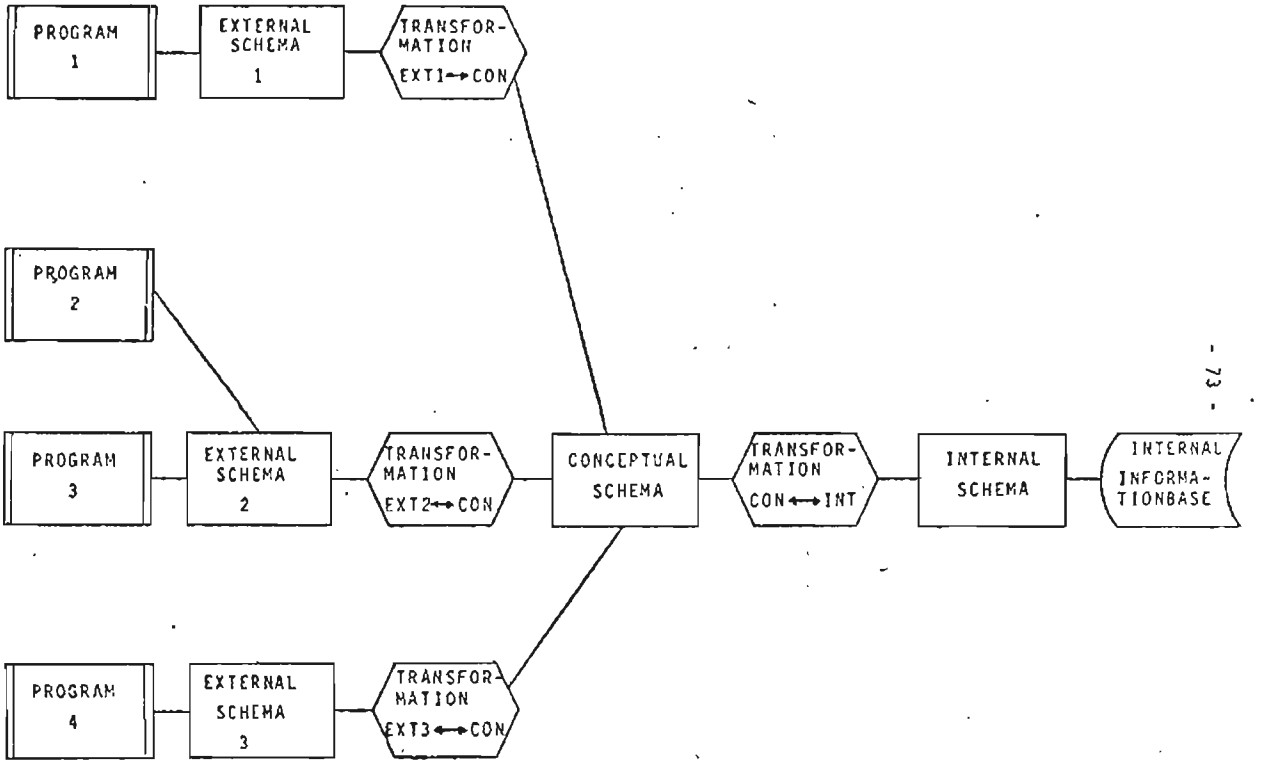

PROGRAM 1
EXTERNAL SCHEMA 1
TRANSFOR-MATION EXT1→CON
PROGRAM 2
PROGRAM 3
EXTERNAL SCHEMA 2
TRANSFOR-MATION EXT2↔CON
CONCEPTUAL SCHEMA
TRANSFOR-MATION CON↔INT
INTERNAL SCHEMA
INTERNAL INFORMA-TIONBASE
PROGRAM 4
EXTERNAL SCHEMA 3
TRANSFOR-MATION EXT3↔CON

Figure 3.1.f

The symmetry of the many external schemata and many internal schemata approach can be presented in a figure as is done in figure 3.1.g. This architecture of an information system support software (or would you prefer database management system) which is the result of applying common sense reasoning, starting with a model of an information system, understandable to all intelligent laymen, permits the coexistence of various interaction (or manipulation) languages, various external schemata and various internal schemata, plus powerful transformations between external and conceptual, and conceptual and internal such that a conceptual schema is only concerned with the problem description, independent of programming considerations or computer efficiency considerations; we have given this architectural model the name COEXISTENCE model (ref. 21). It may be useful to see how the most abstract model as presented in figure 2.1.a has been developed into the lesser abstract model of figure 2.2.h and finally in the model of figure 3.1.g, a gross architecture for future database management systems.

We want to stress that this COEXISTENCE architecture is in its result the same as the ANSI three schemata architecture (ref. 2, 21); however, *the justification for it is quite different.* In our approach it is essential that an information system is viewed as a special (simpler) case of communication among human beings. This brings us to the concept of a prescriptive grammar for such a communication situation. And such a prescriptive grammar we call a conceptual schema. The conceptual schema is in our approach a stand-alone entity and not a "level of indirection between the external and internal schema" as described by Date (in reference 10, page 15).

Remark. Another essential difference in our approach with various other approaches like ANSI, CODASYL, binary relational (Bracchi, Senko) and normalized relational (Codd) is the meaning we give to the name conceptual schema. In our approach the conceptual schema *completely* defines all possible information base populations. This is only possible if we have a conceptual schema in which constraints

may play an essential role. It is somewhat unfortunate to see that in the debates between the above mentioned four approaches, the constraints are barely mentioned. From this one could conclude that the concept conceptual schema is not set equal to grammar for an information in these approaches.

As intermediate recapitulation we may say that we started with a model of an information system understandable to all intelligent laymen (section 2.1). Based upon this solid and simple ground, we derived the three major components of any information system, namely the conceptual schema, the information base and the interaction (section 2.2).

If one wants to implement such an information system on a computer, we have seen a further need for differentiation, namely the internal schema which specifies *how* some of the information of the conceptual information base is to be stored on computer media, and the external schema, which specifies *how* some of the information of the conceptual information base is to be seen by a program.

It might now be understandable why I give the following advice: for each and every information system, always start with the description of the conceptual schema, or prescriptive grammar of the information system.

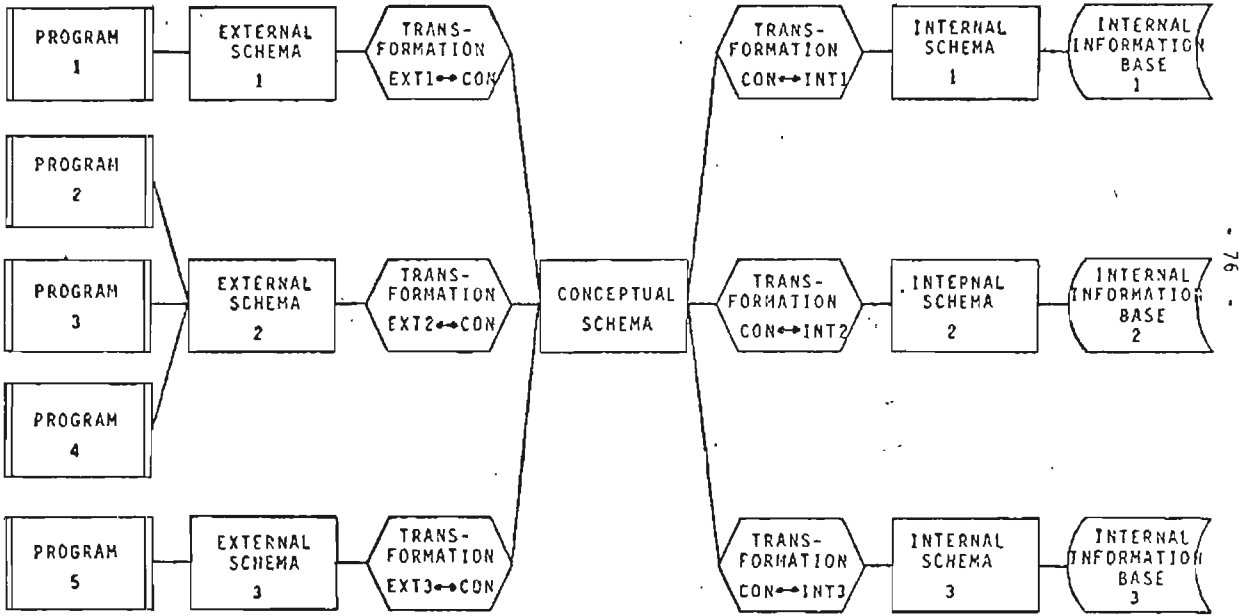

PROGRAM 1
EXTERNAL SCHEMA 1
TRANS-FORMATION EXT1↔CON
PROGRAM 2
PROGRAM 3
EXTERNAL SCHEMA 2
TRANS-FORMATION EXT2↔CON
PROGRAM 4
PROGRAM 5
EXTERNAL SCHEMA 3
TRANS-FORMATION EXT3↔CON
CONCEPTUAL SCHEMA
TRANS-FORMATION CON↔INT1
INTERNAL SCHEMA 1
INTERNAL INFORMATION BASE 1
TRANS-FORMATION CON↔INT2
INTERNAL SCHEMA 2
INTERNAL INFORMATION BASE 2
TRANS-FORMATION CON↔INT3
INTERNAL SCHEMA 3
INTERNAL INFORMATION BASE 3

Figure 3.1.g

3.2 COEXISTENCE ARCHITECTURE IN PRACTICE

How is the COEXISTENCE architecture used in practice? To get an answer to this question, we will describe the sequence of actions that take place to arrive at a running information system.

> During the remainder of this section we will often refer to figures 3.2.a, 3.2.b and 3.2.c. The elements in figure 3.2.a all have a name starting with a C, the elements in figure 3.2.b start with a D, and the elements in 3.2.c all start with an R.

First and formost, the conceptual schema is written (see figure 3.2.b, D1). The conceptual schema (C1; D1) is processed by the conceptual schema compiler (D11); the conceptual schema compiler checks the consistency and syntax of the various elements that make up the conceptual schema, and furthermore, it stores the conceptual schema in the metainformationbase (D17; R1). If errors are detected by the conceptual schema compiler, then the enterprise administrator (which has written the conceptual schema) will receive a description of these errors (see D81 in figure 3.2.d) which he then has to correct and submit again to the conceptual schema compiler.

Once there is a correct conceptual schema, one can submit the transformation (D2; C2) between conceptual schema and internal schema and vice versa to the CI-transformation compiler (D12). This compiler checks whether all transformations are expressed in elements existing in the validated conceptual schema (R1; part of D17) and whether the transformations are permitted. Errors are processed analogous to errors in the conceptual schema. The CI-transformation compiler (D12) will store the conceptual-internal transformation (D2) in the metainformationbase (D17; R2).

Then the internal schema (D3; C3) will be submitted to the internal schema compiler (D13). This compiler checks whether the internal constructs are expressed in elements described in the conceptual-internal transformation (C2; part of D17; R2) and whether these constructs are permitted.

Coexistence source language schematic

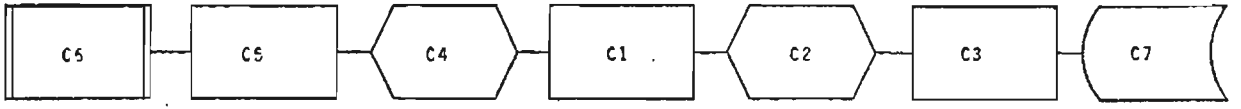
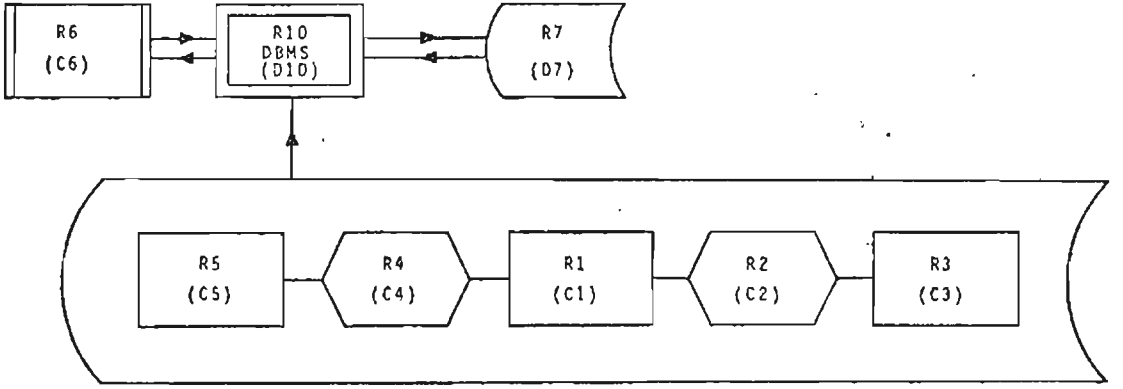
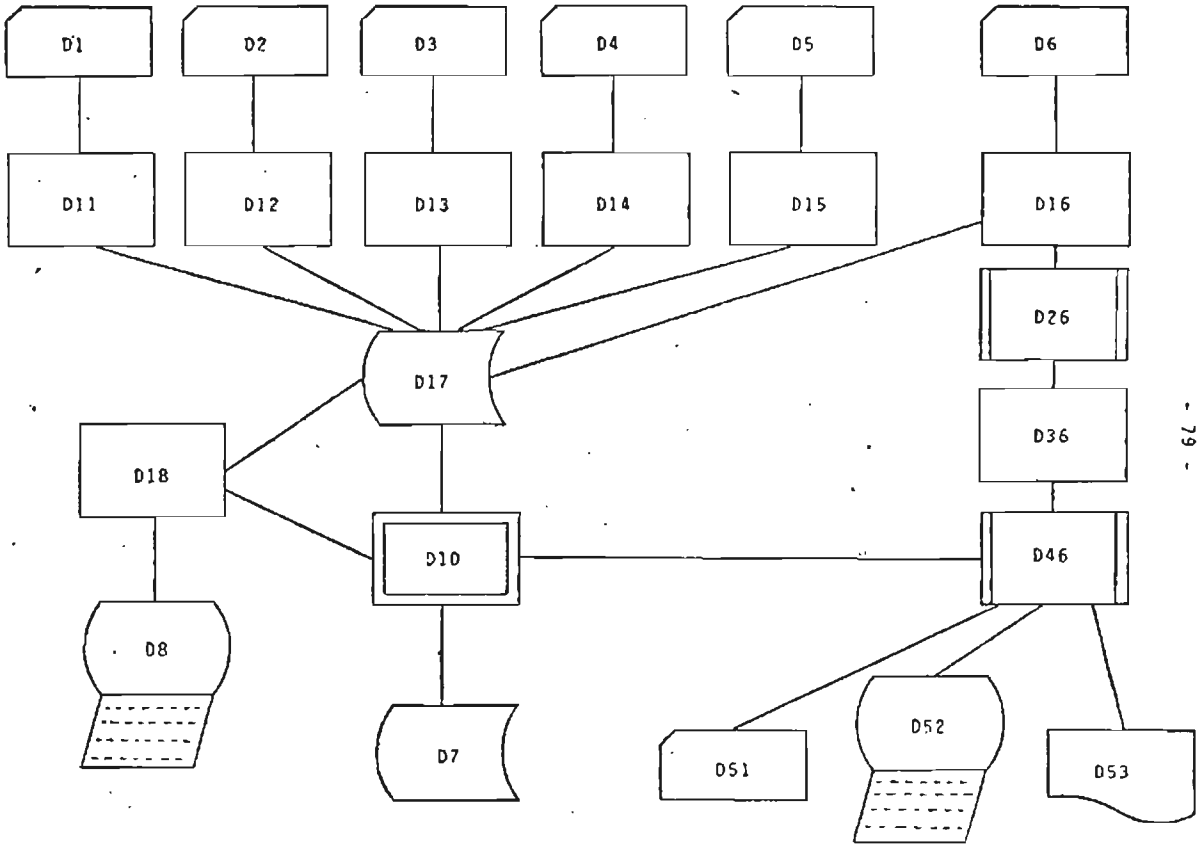


Figure 3.2.a



Coexistence run time schematic

Figure 3.2.c



Coexistence dynamics schemata Figure 3.2.b

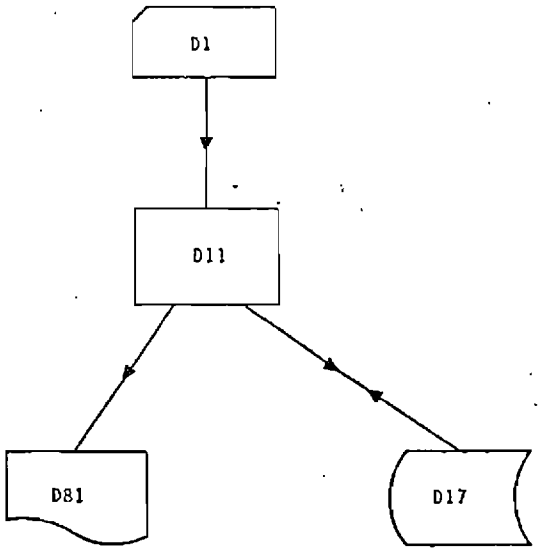


Figure 3.2.d

The compiler stores the internal schema (D3) in the metainformation-base (D17; R3). Errors are processed analogous to errors in the conceptual schema.

Then the transformation (D4; C4) between conceptual schema and external schema and vice versa is submitted to the CE-transformation compiler (D14). This compiler checks whether all transformations are expressed in elements existing in the validated conceptual schema (R1; part of D17) and whether the transformations are permitted. Errors are processed analogous to errors in the conceptual schema. The CE-transformation compiler (D14) will store the conceptual-external transformation (D4) in the metainformationbase (D17; R4).

Thereafter, the external schema (D5, C5) will be submitted to the external schema compiler (D15). This compiler checks whether the external constructs are expressed in elements described in the conceptual-external transformation (C4; part of D17; R4) and whether these constructs are permitted in this external view. (E.g. a binary external view does not permit n-ary relations with n larger than 2). Errors are processed analogous to errors in the conceptual schema. The compiler (D15) stores the external schema (D5) in the metainformationbase (D17; R5).

The previous five descriptions (C1, C2, C3, C4, C5 or D1, D2, D3, D4, D5 or R1, R2, R3, R4, R5) form a complete description of the various aspects of the information of interest. We now have to start with the manipulation aspects.

Once the previous five descriptions are validated, the application program (D6), containing a reference to the external view it wants to see and containing update and retrieve operators referring to constructs in this external view, is submitted to a precompiler (D16), specific to the language in which the application program is written.

This precompiler

- injects the external schema in the form of data descriptions into the application program in the native syntax of the latter.
- checks whether the operators in this application program refer to constructs existing in the external schema, referenced in the application program
- checks whether this program may use these operators
- converts the update and retrieve operators into a CALL or SUBROUTINE mechanism acceptable to the application language (e.g. COBOL, FORTRAN)
- stores the application program in the metainformationbase (D17).
- generates an application program (D26) acceptable to the standard compiler (D36).

The application program (D26) in this state can be considered as a standard application program (because its database update and retrieve operators are converted by the precompiler (D16) into standard CALLs) which will be submitted to the standard (COBOL, FORTRAN, PL1, ALGOL, etc.) compiler (D36). Like all these standard compilers, this compiler generates object code (D46; R6).

At execution time, the application program has normal contact with the input and output devices (card-reader, printer, visual display) and has contact with the informationbase (D7; R7) via the run time DBMS (D10; R10) via the CALL DBMS. The DBMS (D10; R10) will consult the descriptions R1, R2, R3, R4, and R5 (stored in D17) and take the necessary actions to respond to the CALL in the application program (D46; R6; C6).

Here we see that the application programmer has the illusion that he only operates on his external schema, and is isolated from other external schemas or internal schemas.

A different kind of manipulation of information is possible via an interpreter (D18). A user or programmer formulates his request in a certain language (D8), referring to a certain external schema (C5; R5; D17) previously validated. The interpreter will consult the descriptions R1, R2, R3, R4 and R5 to transform the user request into code understandable to the DBMS (D10; R10). The DBMS will consult D17 and provide an answer to the user by contacting the informationbase (D7).

4. A SET OF CONCEPTS FOR THE CONCEPTUAL SCHEMA

As we have seen before, the conceptual schema is, in our opinion, the most important of all schemata; more and more people tend to agree on this point. The conceptual schema is the formal description of a major part of the result of the information analysis work, and as such it will replace the more conventional documentation.

In our opinion, the conceptual schema should be the best documentation of the "what" of an information system; as such it is the common document of users and experts in information systems.

But now to the question: what is a best set of concepts for the conceptual schema?

In order to answer this question, we want to repeat that our basic assumption is that "a computerised information system can be considered as a special case of human communication using language expressions".

From linguistics we know that such communication consists of "sentences" or more precise of "sentence instantiations". If we furthermore remember that a conceptual schema is a contract which regulates a special kind of human communication, then it is clear that the central concept of a conceptual schema in our approach is a "sentence". In other words, the conceptual schema should describe which sentence instances may be of interest in a specific Universe of Discourse.

However, from a semantic point of view, many sentence instances may be broken down into a set of one or more elementary sentence instances. An elementary sentence instance is defined as the smallest unit of meaningful information exchange, in a certain selected Universe of Discourse.

It may now be clear that we want to take as central concept in

the conceptual schema the "elementary sentence type".

The elementary sentence type is a specification of the set of all possible sentence instances we are interested in.

An elementary sentence instance can be considered to convey knowledge of the object instances referred to in the elementary sentence instance. And object instances play a certain role in a sentence instance. Hence we may consider a sentence instance to consist of a set of one or more < object instance, role > pairs.

In the conceptual schema we describe the elementary sentence types, by specifying the roles which are played by the object types.

In our conceptual schema, we start with the declaration of the object types, about which we want to collect knowledge. This is done in the so-called OBJECT DIVISION. (see example page 21)

Thereafter we declare the elementary sentence types, which consist of one or more declarations of an object type and the role of this object type in the elementary sentence type. This is called the ELEMENTARY SENTENCE DIVISION.

And finally, we describe a set of rules which constrain the populations associated with the elementary sentence types. These rules are called CONSTRAINTS and all these rules are described in the CONSTRAINTS DIVISION.

Some of these constraints can be expressed in a descriptive form but some need to be expressed in a procedural form.

As a recapitulation, we may say that the conceptual schema consists of elementary sentence types, the object types involved and the applicable constraints, which jointly define all the possible information bases of interest for a specific Universe of Discourse.

5. THE NEED TO DISTINGUISH TWO CONCEPTUAL SCHEMAS

In the previous sections we have seen that the three schemata approach of the Coexistence Architecture is an overall conceptual framework, which, for instance can be used to explain the strong and weak points of the CODASYL database approach or the Normalized Relational database approach , or any other database approach.

One of the results of the Coexistence Architecture is the conclusion that neither the CODASYL network data model, nor the Normalized Relational data model provide the adequate answers to the database management problems.

Furthermore, it is our opinion that the concept "data model" (which covers certain aspects of the internal, conceptual and external schema as well as the external manipulation language) is too imprecise to be useful and is therefore obsolete.

Another result is that both a good Normalized Relational view as well as a good CODASYL network view may be used in an External Schema. A result of the Coexistence Architecture is to show that the exclusive-or of the CODASYL and Normalized Relational debate is to be transformed into a modified inclusive-or.

But with the three schemata architecture we seem to enter into a second great debate in database management, namely

- which is the best set of concepts to be used in a conceptual schema?

There are in our opinion seven serious candidates for the central (manipulatable) construct in the conceptual schema, namely

1. elementary sentences (without nesting)
2. elementary sentences (with nesting)

3. binary associations (being binary relations without nesting)
4. binary associations (being binary relations with nesting)
5. binary associations (being binary functions without nesting).

6. irreducable relations (without nesting)
7. irreducable relations (with nesting)

and these seven candidates can be classified into three classes.

The proponents of the first two candidates have as basic assumption that the conceptual schema has to prescribe a set of sentence instances of interest.

The proponents of the candidates 3, 4 and 5 have as philosophy that binary relations or functions are "simple" constructs.

The proponents of the candidates 6 and 7 have as basic assumption that a database can best be considered as a set of time-varying relations.

Each class of users has a lot of respectable supporters, and are successfully used in certain practical environments.

The question now is: do we enter, a new (second) great debate in database management? Or could it be that the second great debate is irrelevant once we see an architecture in which more than one candidate has a role to play, just as the first great debate became obsolete once one had accepted a many-external-views philosophy like in the Coexistence architecture.

Indeed, one can adopt an architecture with two conceptual schemata, one in which are described the elementary sentence instances (in which objects are referred to by name) and one in which one abstracts from the way names are assigned to objects. The conceptual schema with names is called significational schema, the one in which one has abstracted from the names is called ontological schema. Other terms for these two concepts could be sentence schema and idea schema. There is a well-defined transformation between these two conceptual schemas.

Theory as well as experience with the two kinds of conceptual schemata applied in practical problems during the last year has convinced me that the elementary sentence is the best basic construct for the significational schema, while the binary relation is the best basic construct for the ontological schema.

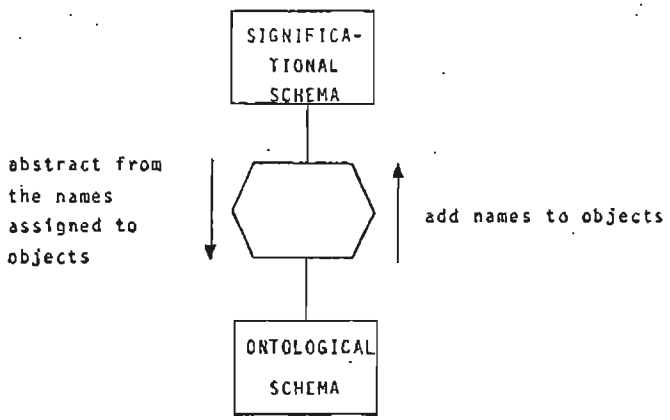


Figure 5.a

The second great debate in database management is, as far as this author is concerned, of limited value, because the framework for the debate used until now does not make the distinction between the significational schema and ontological schema. As soon as one accepts such a distinction, the debate will soon come to a finish.

We could give the following definitions:

A significational schema is a (prescriptive) grammar for an information system, taking into account how names are assigned to things (objects).

An ontological schema is a (prescriptive) grammar for an information system, in which one abstracts from the names assigned to things (objects).

The significational schema prescribes all the possible information bases consisting of elementary sentence instances, in which object names are referred to.

The ontological schema prescribes all the possible bases, consisting of elementary idea (association) instances, in which objects are referred to.

We now discuss in section 5.1 and 5.2 two examples in which the distinction between the significational and ontological schema is very clear.

5.1 EMPLOYEE-PROJECT-EXAMPLE

The first example we will discuss has to do with employees and projects. An example of an output report of this information system is given in figure 5.1.a. In discussions with the users, it turns out that the conceptual schema is as presented in the significational schema called EMPLOYEE-PROJECT-SIGNIFICATIONAL.

Remark. It might be useful to remark that a homonym is a name which refers to two or more object instances, while a synonym is a name which refers to one object instance, but there may be more than one synonym referring to the same object instance. One could say that a homonym is not a name in the sense of unique reference.

An information type diagram illustrating this schema is given in figure 5.1.b.

EMPLOYEE-NR	EMPLOYEE-CODE	EMPLOYEE-NAME	SOCIAL-SECURITY-NUMBER	SEX	WORKER-IN-PROJECT
E1	JACKY	BOUVIER	312	W	P1
E2	DICK	NIXON	608	M	P1 P2
E3	JERRY	FORD	471	M	P2
E4	JIMMY	CARTER	488	M	P1
E5	LBJ	JOHNSON	216	M	P2
E6	JFK JACK	KENNEDY	196	M	P2
E7	TED	KENNEDY	740	M	P1

PROJECT-NAME	PROJECT-NR	BUDGET	PROJECT-WORKER
COBOL	P1	120.000	E1
			E2
			E4
			E7
FORTRAN	P2	90.000	E3
			E2
			E5
			E6

Figure 5.1.a

```
SIGNIFICATIONAL SCHEMA NAME IS EMPLOYEE-PROJECT-SIGNIFICATIONAL.

OBJECT NAME TYPE DIVISION.

    OBJECT NAME TYPE NAME IS MONEY,
                     NAME LENGTH IS 6 DIGITS.
    OBJECT NAME TYPE NAME IS PROJ-NR,
                     NAME LENGTH IS 2 CHARACTERS.
    OBJECT NAME TYPE NAME IS PROJ-NAME,
                     NAME LENGTH IS 10 CHARACTERS.
    OBJECT NAME TYPE NAME IS EMPL-NAME
                     NAME LENGTH IS 10 CHARACTERS.
    OBJECT NAME TYPE NAME IS EMPL-NR,
                     NAME LENGTH IS 2 CHARACTERS.
    OBJECT NAME TYPE NAME IS EMPL-CODE,
                     NAME LENGTH IS 6 CHARACTERS.
    OBJECT NAME TYPE NAME IS SEX,
                     NAME LENGTH IS 1 CHARACTER,
                     PERMITTED VALUES ARE M, W.

ELEMENTARY SENTENCE DIVISION.

    SENTENCE TYPE NAME IS PROJECT-BUDGET
        REFERENCED OBJECT NAME TYPE NAME IS MONEY,
            ROLE IS BUDGET
        REFERENCED OBJECT NAME TYPE NAME IS PROJ-NR,
            ROLE IS PROJECT-NR.

    SENTENCE TYPE NAME IS PROJECT-REFERENCE
        REFERENCED OBJECT NAME TYPE NAME IS PROJ-NR,
            ROLE IS PROJECT-NR
        REFERENCED OBJECT NAME TYPE NAME IS PROJ-NAME
            ROLE IS PROJECT-NAME
```

```
SENTENCE TYPE NAME IS ASSIGNMENT
    REFERENCED OBJECT NAME TYPE NAME IS PROJ-NR
        ROLE IS PROJECT-NR
    REFERENCED OBJECT NAME TYPE NAME IS EMPL-NR
        ROLE IS PROJECT-WORKER.

SENTENCE TYPE NAME IS EMPLOYEE-HOMONYM
    REFERENCED OBJECT NAME TYPE NAME IS EMPL-NR
        ROLE IS EMPLOYEE-NR
    REFERENCED ONTN IS EMPL-NAME
        ROLE IS EMPLOYEE-NAME

SENTENCE TYPE NAME IS EMPLOYEE-SYNONYM
    REFERENCED ONTN IS EMPL-NR
        ROLE IS EMPLOYEE-NR
    REFERENCED ONTN IS EMPL-CODE
        ROLE IS EMPLOYEE-CODE

SENTENCE TYPE NAME IS EMPLOYEE'S-SEX
    REFERENCED ONTN IS EMPL-NR
        ROLE IS EMPLOYEE-NR
    REFERENCED ONTN IS SEX
        ROLE IS SEX

SENTENCE TYPE NAME IS EMPLOYEE-REFERENCE
    REFERENCED ONTN IS EMPL-NR
        ROLE IS EMPLOYEE-NR
    REFERENCED ONTN IS SSN
        ROLE IS SSN.
```

```
CONSTRAINT DIVISION.

    CONSTRAINT NAME IS AT-MOST-ONE-BUDGET-PER-PROJECT
               CODE IS C300
        ROLE PROJECT-NR IN SENTENCE PROJECT-BUDGET
             IS UNIQUE

    CONSTRAINT NAME IS AT-MOST-ONE-PROJECT-NAME-PER-PROJECT
               CODE IS C301
        ROLE PROJECT-NAME OF SENTENCE PROJECT-REFERENCE
             IS UNIQUE

    CONSTRAINT NAME IS AT-MOST-ONE-PROJECT-NR-PER-PROJECT
               CODE IS C302
        ROLE PROJECT-NR OF SENTENCE PROJECT-REFERENCE
             IS UNIQUE

    CONSTRAINT NAME IS ONLY-ONCE-THE-SAME
               CODE IS C303
        CONCATENATION OF ROLES PROJECT-NR AND PROJECT-WORKER
            OF SENTENCE ASSIGNMENT IS UNIQUE.

    CONSTRAINT NAME IS AT-MOST-ONE-EMPL-NAME-PER-EMPLOYEE
               CODE IS C304
        ROLE EMPLOYEE-NR OF SENTENCE EMPLOYEE-HOMONYM
             IS UNIQUE.

    CONSTRAINT NAME IS AT-MOST-ONE-EMPL-NR-PER-EMPL-CODE
               CODE IS C305
        ROLE EMPLOYEE-CODE OF SENTENCE EMPLOYEE-SYNONYM
             IS UNIQUE.

    CONSTRAINT NAME IS AT-MOST-ONE-SEX-PER-EMPLOYEE
               CODE IS C306
        ROLE EMPLOYEE-NR OF SENTENCE EMPLOYEE'S-SEX
             IS UNIQUE.
```

```
CONSTRAINT NAME IS AT-MOST-ONE-SSN-PER-EMPL-NR
           CODE IS C307
    ROLE EMPLOYEE-NR OF SENTENCE EMPLOYEE-REFERENCE
         IS UNIQUE

CONSTRAINT NAME IS AT-MOST-ONE-EMPL-NR-PER-SSN
           CODE IS C308
    ROLE SSN OF SENTENCE EMPLOYEE-REFERENCE
           IS UNIQUE

CONSTRAINT NAME IS ASSIGNMENTS-ONLY-IF-BUDGET-AVAILABLE
           CODE IS C309
    ROLE PROJECT-NR OF SENTENCE ASSIGNMENT
    ,    IS SUBSET OF
    ROLE PROJECT-NR OF SENTENCE PROJECT-BUDGET

CONSTRAINT NAME IS NO-ASSIGNMENT-WITHOUT-EMPL-NAME
           CODE IS C310
    ROLE PROJECT-WORKER OF SENTENCE ASSIGNMENT
         IS SUBSET OF
    ROLE EMPLOYEE-NR OF EMPLOYEE-HOMONYM

CONSTRAINT NAME IS EMPL-NAME-AND-SSN-TOGETHER
           CODE IS C311
    ROLE EMPLOYEE-NR OF EMPLOYEE-REFERENCE
         IS EQUAL TO
    ROLE EMPLOYEE-NR OF EMPLOYEE-HOMONYM

END  SIGNIFICATIONAL SCHEMA.
```

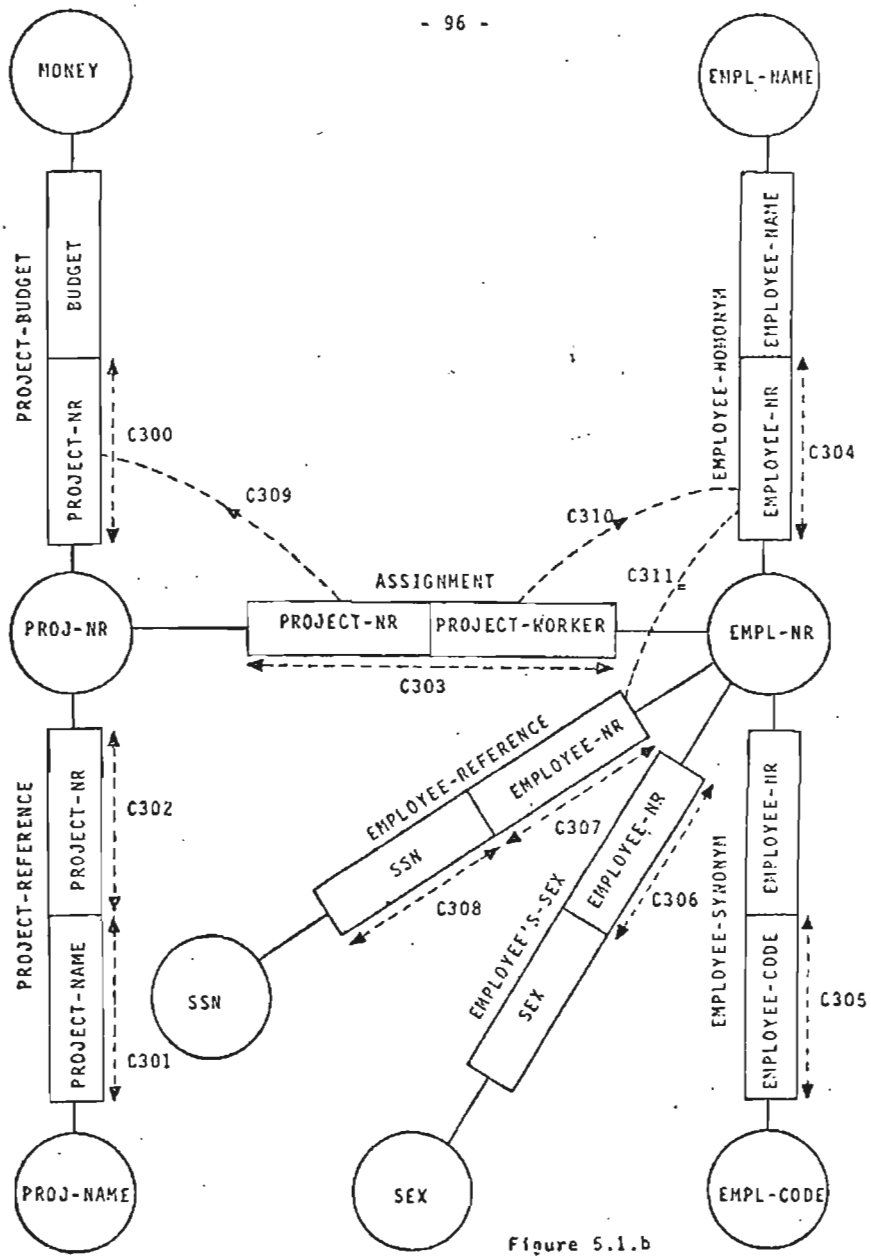

MONEY
EMPL-NAME
PROJECT-BUDGET
BUDGET
PROJECT-NR
EMPLOYEE-HOMONYM
EMPLOYEE-NAME
EMPLOYEE-NR
C300
C304
C309
C310
C311
ASSIGNMENT
PROJ-NR
PROJECT-NR
PROJECT-WORKER
EMPL-NR
C303
EMPLOYEE-REFERENCE
EMPLOYEE-NR
SSN
C307
C308
EMPLOYEE'S-SEX
EMPLOYEE-NR
SEX
C306
PROJECT-REFERENCE
PROJECT-NR
C302
PROJECT-NAME
C301
EMPLOYEE-SYNONYM
EMPLOYEE-NR
EMPLOYEE-CODE
C305
SSN
PROJ-NAME
SEX
EMPL-CODE

Figure 5.1.b

Thereafter, we start a discussion with the user to see what is the schema if we would abstract from the names assigned to an object. In this discussion it turns out that PROJ-NR as well as PROJ-NAME are identifying names for a project, and EMPL-NR and SSN are identifying names for an employee; furthermore it turns out that EMPL-CODE identifies an employee, but an employee may have more than one EMPL-CODE, hence this is a synonym name. And finally, EMPL-NAME does not identify an employee, because more than one employee may have the same EMPL-NAME.

After this explanation, we are able to write down the ontological schema, called EMPLOYEE-PROJECT-ONTOLOGICAL, and the idea type diagram of figure 5.1.c.

```
ONTOLOGICAL SCHEMA NAME IS EMPLOYEE-PROJECT-ONTOLOGICAL

OBJECT TYPE DIVISION.

    OBJECT TYPE NAME IS MONEY
    OBJECT TYPE NAME IS PROJECT
    OBJECT TYPE NAME IS EMPL-NAME
    OBJECT TYPE NAME IS EMPLOYEE
    OBJECT TYPE NAME IS SEX

ELEMENTARY IDEA DIVISION.

    IDEA NAME IS PROJECT-BUDGET
        INVOLVED OBJECT TYPE NAME IS MONEY
            ROLE IS BUDGET
        INVOLVED OBJECT TYPE NAME IS PROJECT
            ROLE IS PROJECT-REF

    IDEA NAME IS ASSIGNMENT
        INVOLVED OTN IS PROJECT
            ROLE IS PROJECT-REF
        INVOLVED OTN IS EMPLOYEE
            ROLE IS PROJECT-WORKER

    IDEA NAME IS EMPLOYEE-HOMONYM
        INVOLVED OTN IS EMPLOYEE
            ROLE IS EMPLOYEE-REF
        INVOLVED OTN IS EMPL-NAME
            ROLE IS EMPLOYEE-NAME

    IDEA NAME IS EMPLOYEE'S-SEX
        INVOLVED OTN IS EMPLOYEE
            ROLE IS EMPLOYEE-REF
        INVOLVED OTN IS SEX
            ROLE IS SEX
```

```
CONSTRAINTS DIVISION.

    CONSTRAINT NAME IS AT-MOST-ONE-BUDGET-PER-PROJECT
               CODE IS C300
        ROLE PROJECT-REF IN IDEA PROJECT-BUDGET
             IS UNIQUE.

    CONSTRAINT NAME IS ONLY-ONCE-THE-SAME
               CODE IS C303
        CONCATENATION OF ROLES PROJECT-REF AND PROJECT-WORKER
            OF IDEA ASSIGNMENT IS UNIQUE.

    CONSTRAINT NAME IS AT-MOST-ONE-EMPL-NAME-PER-EMPLOYEE
               CODE IS C304
        ROLE EMPLOYEE-REF OF IDEA EMPLOYEE-HOMONYM
             IS UNIQUE.

    CONSTRAINT NAME IS AT-MOST-ONE-SEX-PER-EMPLOYEE
               CODE IS C306
        ROLE EMPLOYEE-REF OF EMPLOYEE'S-SEX
             IS UNIQUE.

    CONSTRAINT NAME IS ASSIGNMENTS-ONLY-IF-BUDGET-AVAILABLE
               CODE IS C309
        ROLE PROJECT-REF OF IDEA ASSIGNMENT
             IS SUBSET OF
        ROLE PROJECT-REF OF IDEA PROJECT-BUDGET

    CONSTRAINT NAME IS NO-ASSIGNMENT-WITHOUT-EMPL-NAME
               CODE IS C310
        ROLE PROJECT-WORKER OF IDEA ASSIGNMENT
             IS SUBSET OF
        ROLE EMPLOYEE-REF OF IDEA EMPLOYEE-HOMONYM

END ONTOLOGICAL SCHEMA.
```

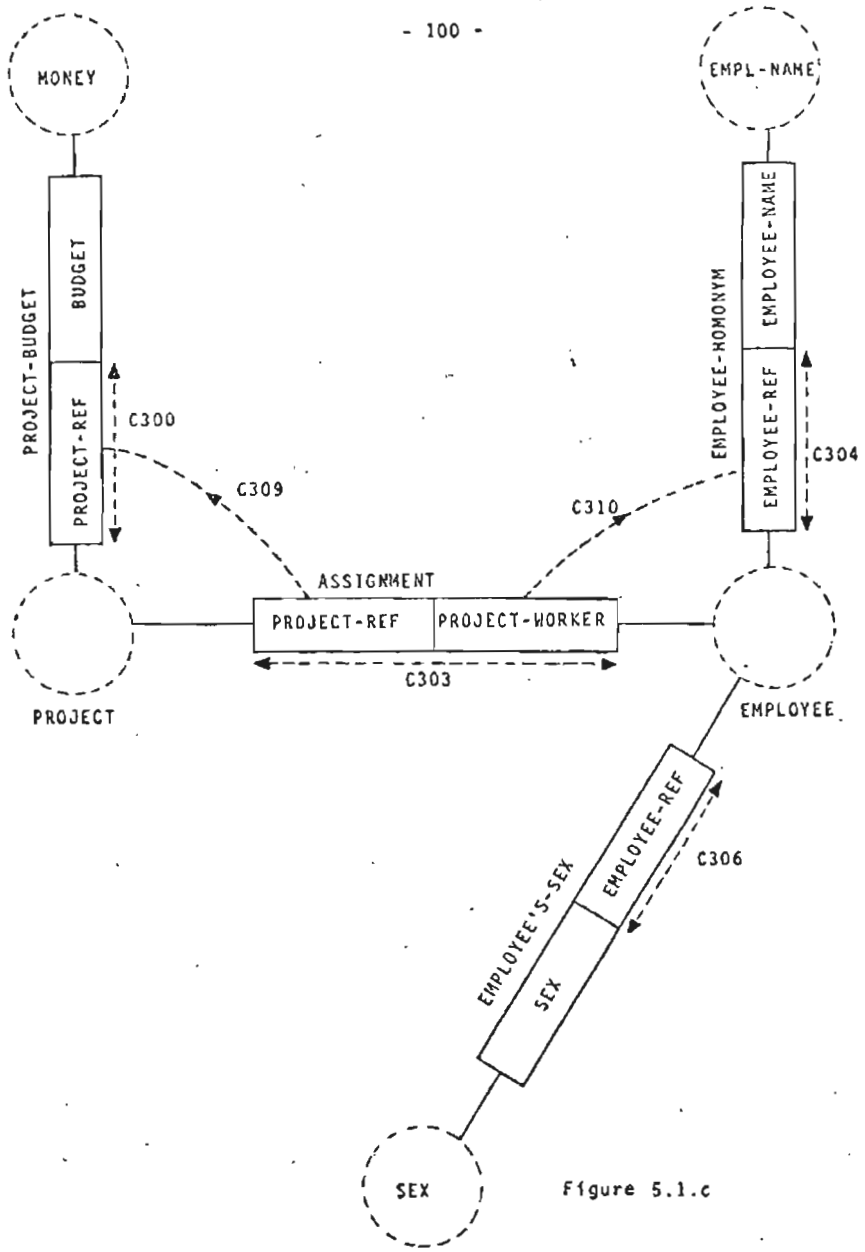

MONEY
EMPL-NAME
PROJECT-BUDGET
BUDGET
PROJECT-REF
C300
C309
C310
EMPLOYEE-HOMONYM
EMPLOYEE-NAME
EMPLOYEE-REF
C304
ASSIGNMENT
PROJECT-REF
PROJECT-WORKER
C303
PROJECT
EMPLOYEE
EMPLOYEE'S-SEX
EMPLOYEE-REF
SEX
C306
SEX

Figure 5.1.c

If one compares the significational and ontological schema, it is clear that the ontological schema is a "deeper" description than the significational schema.

My advice is to make both the significational and ontological schema, certainly for problems which are either abstract or complicated or both.

What could be a reason to maintain more than one name for a thing like two identifying names for each employee? It could be that communication with the social security agency can only be done using the SOCIAL SECURITY NUMBER, while communication inside the company uses (the already long established and shorter) EMPLOYEE NUMBER.

The result of a conceptualization, in which we may abstract from the ways that names are assigned to objects, applied to this example results in

- 5 object type names
  as opposed to
- 8 object name type names

and in 4 elementary idea types in stead of 7 elementary sentence types and 6 constraints in stead of 12.

We see that a deeper level of abstraction, namely abstracting from the name assignment, results in a conceptual schema which is "different" than the conceptual schema which takes naming into account. The conceptual schema, in which one abstracts from the naming assignment we call ontological conceptual schema, and the conceptual schema, in which one takes the naming into account we call significational conceptual schema. These terms are based upon terms introduced by Falkenberg. In a significational conceptual schema there may arise some unsolvable problems which do not come up in an ontological conceptual schema.

E.g. Why is the sex associated with the employee-nr, and not with the social security number?

5.2 PROFESSOR-HALL-HOUR-EXAMPLE

The second example to illustrate some differences between the ontological and significational schemata has to do with professors, lecture-hours and lecture-halls. Two output reports from this information system are presented in figure 5.2.a and 5.2.b. The first deals with the requests that professors forward to the board of the faculty to lecture at a certain hour in a certain hall. As additional information it is said by the users that the professors may forward at most one request for a given hour; furthermore, during a given hour a lecture is presented by one professor, and no other professor is involved. After a certain time the board of the faculty reviews all the requests put forward by the professors, and decides to assign a certain hour and lecture-hall to a certain professor. With respect to the assignment it is said that a given professor will lecture during a given hour in only one lecture hall, and he lectures alone.

We are now able to write down the significational schema as is done in the schema called PROFESSOR-HOUR-HALL-SIGNIFICATIONAL and the information type diagram is presented in figure 5.2.c. Please note that the sentences called REQUEST and ASSIGNMENT are elementary but contain three object-role declarations; furthermore, it is of interest to note that there applies one uniqueness constraint to the elementary sentence REQUEST, while there apply two uniqueness constraints to the elementary sentence ASSIGNMENT.

REQUESTS made by the professors

requestor	requested-hour	requested-lecture-hall
MILLER	H14	L51
MILLER	H15	L51
STEEL	H14	L51
STEEL	H16	L51
FORD	H12	L40
FORD	H14	L46
FORD	H15	L51

Figure 5.2.a

ASSIGNMENTS made by the management

lecturer	assigned-hour	assigned-lecture-hall
MILLER	H14	L51
MILLER	H15	L40
STEEL	H14	L40
STEEL	H16	L51
FORD	H12	L40
FORD	H14	L46
FORD	H15	L51

Figure 5.2.b

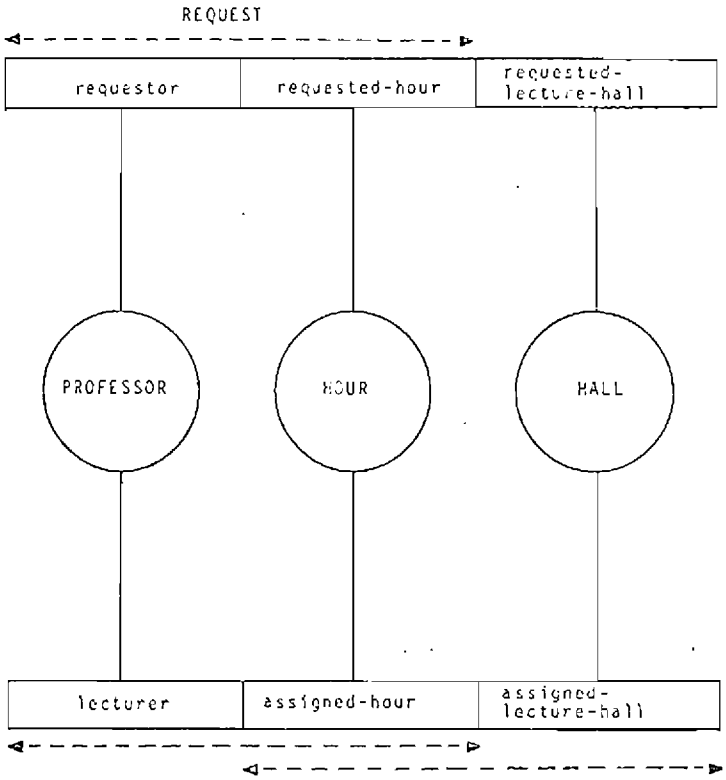

REQUEST
requestor
requested-hour
requested-lecture-hall
PROFESSOR
HOUR
HALL
lecturer
assigned-hour
assigned-lecture-hall
ASSIGNMENT

Figure 5.2.c

```
SIGNIFICATIONAL SCHEMA NAME IS PROFESSOR-HOUR-HALL-SIGNIFICATIONAL.

OBJECT NAME TYPE DIVISION.

    OBJECT NAME TYPE NAME IS PROFESSOR
           NAME LENGTH IS 14 CHARACTERS
    OBJECT NAME TYPE NAME IS HOUR
           NAME LENGTH IS 3 CHARACTERS
    OBJECT NAME TYPE NAME IS HALL
           NAME LENGTH IS 3 CHARACTERS.

ELEMENTARY SENTENCE DIVISION.

    SENTENCE TYPE NAME IS REQUEST
        REFERENCED OBJECT NAME TYPE NAME IS PROFESSOR
            ROLE IS REQUESTOR
        REFERENCED ONTN IS HOUR
            ROLE IS REQUESTED-HOUR
        REFERENCED ONTN IS HALL
            ROLE IS REQUESTED-LECTURE-HALL.

    SENTENCE TYPE NAME IS ASSIGNMENT
        REFERENCED ONTN IS PROFESSOR
            ROLE IS LECTURER
        REFERENCED ONTN IS HOUR
            ROLE IS ASSIGNED-HOUR
        REFERENCED ONTN IS HALL
            ROLE IS ASSIGNED-LECTURE-HALL
```

```
CONSTRAINTS DIVISION.

    CONSTRAINT NAME IS AT-MOST-ONCE-A-GIVEN-HOUR-PER-REQUESTOR
               CODE IS C201
        CONCATENATION OF ROLES REQUESTOR AND REQUESTED-HOUR
            OF SENTENCE REQUEST IS UNIQUE

        COMMENT.  A PROFESSOR MAY REQUEST ANY HOUR AND
                  ANY LECTURE-HALL WITH THE RESTRICTION
                  THAT HE CAN REQUEST ONLY ONCE A CERTAIN HOUR.

        END COMMENT.

    CONSTRAINT NAME IS TEACHING-ALONE-AN-ENTIRE-HOUR
               CODE IS C202
        CONCATENATION OF ROLES LECTURER AND ASSIGNED-HOUR
            OF SENTENCE ASSIGNMENT IS UNIQUE

        COMMENT.  A GIVEN LECTURE HOUR IS ENTIRELY TAUGHT BY
                  ONE PROFESSOR.

        END COMMENT.

    CONSTRAINT NAME IS TEACHING- ONE-LECTURE-AT-MOST-IN-ONE-PLACE
               CODE IS C203
        CONCATENATION OF ROLES ASSIGNED-HOUR AND ASSIGNED-LECTURE-HALL
            OF SENTENCE ASSIGNMENT IS UNIQUE.

        COMMENT.  IN CONSTRAINT C202 IT WAS SAID THAT A GIVEN LECTURE
                  IS ENTIRELY TAUGHT BY ONE PROFESSOR.  HERE WE ADD
                  THAT A GIVEN LECTURE IS ENTIRELY TAUGHT IN ONE AND
                  THE SAME LECTURING-HALL.  THIS MEANS THAT EVERY
                  COMBINATION OF ASSIGNED-HOUR AND ASSIGNED-LECTURE-
                  HALL CAN OCCUR AT MOST ONCE IN A POPULATION OF THE
                  SENTENCE TYPE ASSIGNMENT.

        END COMMENT.

END SIGNIFICATIONAL SCHEMA.
```

Thereafter we start a discussion with the user to find out what the conceptual schema would be if we would abstract from the names assigned to an object. In this discussion it turns out that we not only have to do with the object types PROFESSOR, HOUR and HALL, but also with the object types REQUEST and ASSIGNMENT. However, the board could not get the professors to the point where they would fill in request forms, having the REQUEST-NR preprinted, and to stay in line with this, the board decided not to use ASSIGNMENT-NR to refer to an assignment. After this explanation, we see that we have to do with 5 object types (not object name types) and 6 elementary ideas, and some constraints. The ontological schema describing this formally is presented hereafter and is called PROFESSOR-HOUR-HALL-ONTOLOGICAL, and the corresponding idea type diagram is presented in figure 5.2.d.

```
ONTOLOGICAL SCHEMA NAME IS PROFESSOR-HOUR-HALL-ONTOLOGICAL

OBJECT TYPE DIVISION.

    OBJECT TYPE NAME IS PROFESSOR.
    OBJECT TYPE NAME IS HOUR.
    OBJECT TYPE NAME IS HALL.
    OBJECT TYPE NAME IS REQUEST.
    OBJECT TYPE NAME IS ASSIGNMENT.

ELEMENTARY IDEA DIVISION.

    IDEA NAME IS REQUESTING-PROFESSOR
         INVOLVED OBJECT TYPE NAME IS PROFESSOR
             ROLE IS REQUESTOR
         INVOLVED OBJECT TYPE NAME IS REQUEST
             ROLE IS REQUEST-NR.
    IDEA NAME IS HOUR-REQUESTED
         INVOLVED OTN IS HOUR
             ROLE IS REQUESTED-HOUR
         INVOLVED OTN IS REQUEST
             ROLE IS REQUEST-NR
    IDEA NAME IS HALL-REQUESTED
         INVOLVED OTN IS HALL
             ROLE IS REQUESTED-LECTURE-HALL
         INVOLVED OTN IS REQUEST
             ROLE IS REQUEST-NR
    IDEA NAME IS PROFESSOR-ASSIGNED
         INVOLVED OTN IS PROFESSOR
             ROLE IS LECTURER
         INVOLVED OTN IS ASSIGNMENT
             ROLE IS ASSIGNMENT-NR
    IDEA NAME IS HOUR-ASSIGNED
         INVOLVED OTN IS HOUR
             ROLE IS ASSIGNED-HOUR
         INVOLVED OTN IS ASSIGNMENT
             ROLE IS ASSIGNMENT-NR
    IDEA NAME IS HALL-ASSIGNED
         INVOLVED OTN IS HALL
             ROLE IS ASSIGNED-LECTURE-HALL
         INVOLVED OTN IS ASSIGNMENT
             ROLE IS ASSIGNMENT-NR.
```

```
CONSTRAINTS DIVISION.

    CONSTRAINT NAME IS AT-MOST-ONE-PROFESSOR-PER-REQUEST
               CODE IS C210
        ROLE REQUEST-NR IN IDEA REQUESTING-PROFESSOR
             IS UNIQUE.

    CONSTRAINT NAME IS AT-MOST-ONE-HOUR-PER-REQUEST
               CODE IS C211
        ROLE REQUEST-NR IN IDEA HOUR-REQUESTED
             IS UNIQUE.

    CONSTRAINT NAME IS AT-MOST-ONE-HALL-PER-REQUEST
               CODE IS C212
        ROLE REQUEST-NR IN IDEA HALL-REQUESTED
             IS UNIQUE.

    CONSTRAINT NAME IS AT-MOST-ONE-PROFESSOR-PER-ASSIGNMENT
               CODE IS C213
        ROLE ASSIGNMENT-NR OF IDEA PROFESSOR-ASSIGNED
             IS UNIQUE.

    CONSTRAINT NAME IS AT-MOST-ONE-HOUR-PER-ASSIGNMENT
               CODE IS C214
        ROLE ASSIGNMENT-NR OF IDEA HOUR-ASSIGNED
             IS UNIQUE.

    CONSTRAINT NAME IS AT-MOST-ONE-HALL-PER-ASSIGNMENT
               CODE IS C215
        ROLE ASSIGNMENT-NR OF IDEA HALL-ASSIGNED
             IS UNIQUE.
```

```
CONSTRAINT NAME IS PROFESSOR-AND-HOUR-TOGETHER-REQUESTED
           CODE IS C221
    ROLE REQUEST-NR OF IDEA REQUESTING-PROFESSOR
         IS EQUAL TO
    ROLE REQUEST-NR OF IDEA HOUR-REQUESTED

CONSTRAINT NAME IS PROFESSOR-AND-HALL-TOGETHER-REQUESTED
           CODE IS C222
    ROLE REQUEST-NR OF IDEA REQUESTING-PROFESSOR
         IS EQUAL TO
    ROLE REQUEST-NR OF IDEA HALL-REQUESTED

CONSTRAINT NAME IS PROFESSOR-AND-HOUR-TOGETHER-ASSIGNED
           CODE IS C224
    ROLE ASSIGNMENT-NR OF IDEA PROFESSOR-ASSIGNED
         IS EQUAL TO
    ROLE ASSIGNMENT-NR OF IDEA HOUR-ASSIGNED

CONSTRAINT NAME IS PROFESSOR-AND-HALL-TOGETHER-ASSIGNED
           CODE IS C225
    ROLE ASSIGNMENT-NR OF IDEA PROFESSOR-ASSIGNED
         IS EQUAL TO
    ROLE ASSIGNMENT-NR OF IDEA HALL-ASSIGNED.
```

```
CONSTRAINT NAME IS AT-MOST-ONCE-A-GIVEN-HOUR-PER-REQUESTOR
           CODE IS C201
    CONCATENATION OF ROLE REQUESTOR OF IDEA REQUESTING-PROFESSOR
                 AND ROLE REQUESTED-HOUR OF IDEA HOUR-REQUESTED
                 IS UNIQUE.

CONSTRAINT NAME IS TEACHING-ALONE-AN-ENTIRE-HOUR
           CODE IS C202
    CONCATENATION OF ROLE LECTURER OF IDEA PROFESSOR-ASSIGNED
                 AND ROLE ASSIGNED-HOUR OF IDEA HOUR-ASSIGNED
                 IS UNIQUE.

CONSTRAINT NAME IS TEACHING-ONE-LECTURE-AT-MOST-IN-ONE-PLACE
           CODE IS C203
    CONCATENATION OF ROLE ASSIGNED-HOUR OF IDEA HOUR-ASSIGNED
                 AND ROLE ASSIGNED-LECTURE-HALL OF IDEA HALL-ASSIGNED
                 IS UNIQUE.

END ONTOLOGICAL SCHEMA.
```

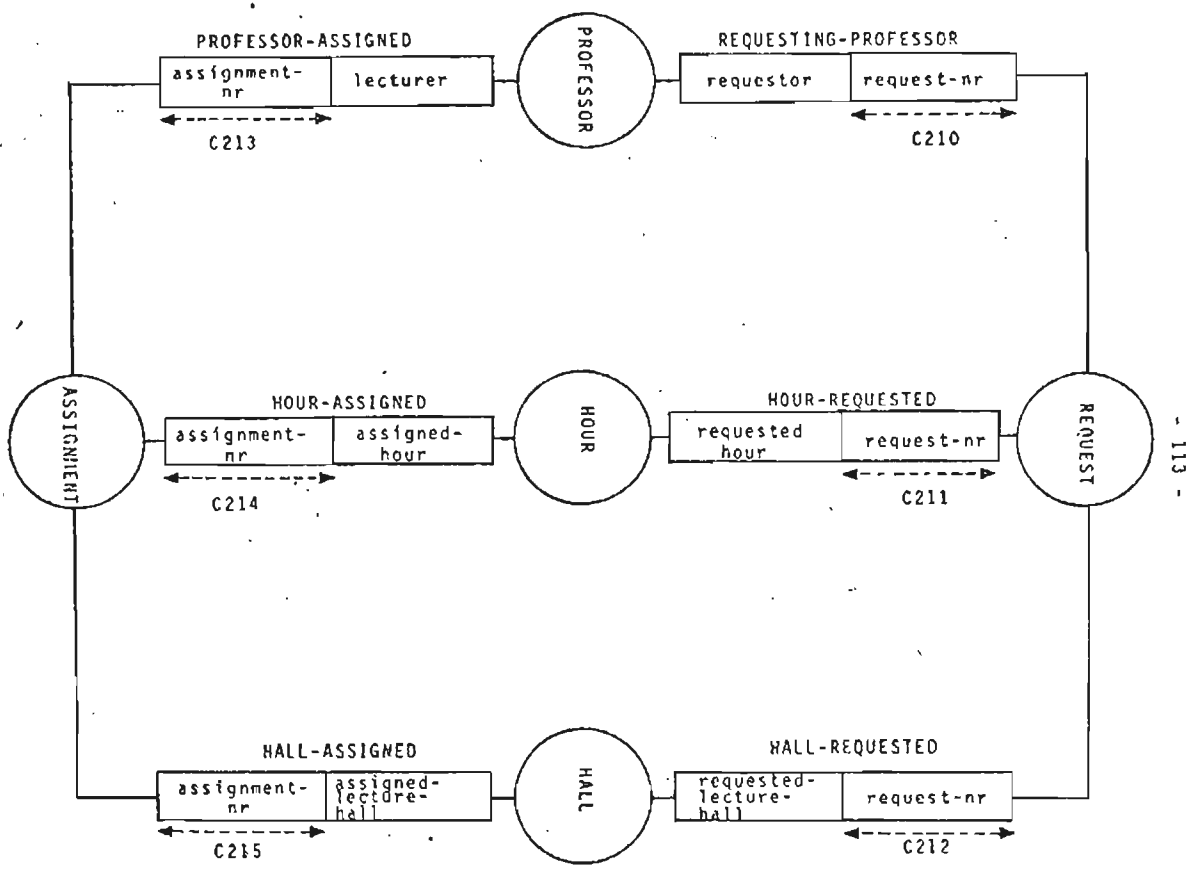

PROFESSOR-ASSIGNED
assignment-nr
lecturer
C213
PROFESSOR
REQUESTING-PROFESSOR
requestor
request-nr
C210
ASSIGNMENT
HOUR-ASSIGNED
assignment-nr
assigned-hour
C214
HOUR
HOUR-REQUESTED
requested hour
request-nr
C211
REQUEST
HALL-ASSIGNED
assignment-nr
assigned-lecture-hall
C215
HALL
HALL-REQUESTED
requested-lecture-hall
request-nr
C212

Figure 5.2.d

The result of a conceptualization, in which we may abstract from the ways names are assigned to objects, applied to the professor-hour-hall example results in:

- 5 object type names
as opposed to
- 3 object type names

and in 6 elementary idea types in stead of 2 elementary sentence types and 13 constraints in stead of 3.

Please note that in the professor-hour-hall example we have to do with classifying names, one for the object REQUEST, but two for the object ASSIGNMENT.

5.3 SOME OBSERVATIONS ON ONTOLOGICAL AND CONCEPTUAL SCHEMATA

Some people tend to assume that the higher the level of abstraction, the smaller the number of concepts. If we however take a look at figure 5.3.a, we see that for the example of 5.1 there are 27 things in the significational schema and 15 things in the ontological schema, however for the example of 5.2 there are 8 things in the significational and 24 things in the ontological schema. These two examples show that the law "the higher level of abstraction, the smaller the number of concepts" is not a general law.

But why has example 5.1 a smaller number of things in the ontological schema as compared to the significational schema, and why the reverse for example 5.2.?

This difference is directly related to the kinds of names which are assigned to objects.

In example 5.1 there are objects which have more than one identifying name (e.g. PROJECT and EMPLOYEE) and one object (EMPLOYEE) has an identifying-synonym name. In example 5.2 there are objects which have only a classifying name (e.g. REQUEST and ASSIGNMENT).

In general one may say that the significational and ontological schema have the same number of things in the case where all objects have exactly one identifying name.

If objects have more than one identifying or identifying-synonym name, then the corresponding ontological schema contains a smaller number of things.

If objects have one or more classifying names, then the corresponding ontological schema contains a larger number of things.

		significational	ontological
example 5.1	object name types	8	
	object types		5
	elementary sentence types	7	
	idea types		4
	constraints	12	6
example 5.2	object name type	3	
	object types		5
	elementary sentence types	2	
	idea types		6
	constraints	3	13

Figure 5.3.a

6. A DBMS ARCHITECTURE WITH FOUR SCHEMATA AND THREE TRANSFORMATIONS

In section three we have seen that it is useful to distinguish three areas of effectiveness namely information analysis, machine resources and programming. And a three schema architecture with a conceptual, internal and external schema is in line with this distinction. Within information analysis it turns out to be useful, as described in section 5, to distinguish two different levels of abstraction, one in which we take the names of things into account and one in which we abstract from the names of the things. A DBMS architecture which takes this last distinction into account, is presented in figure 6.a. It is important to note that in the transformations between the ontological and significational schema, one adds the names to the things; the other two transformations remain the same as was discussed in section 3.1.

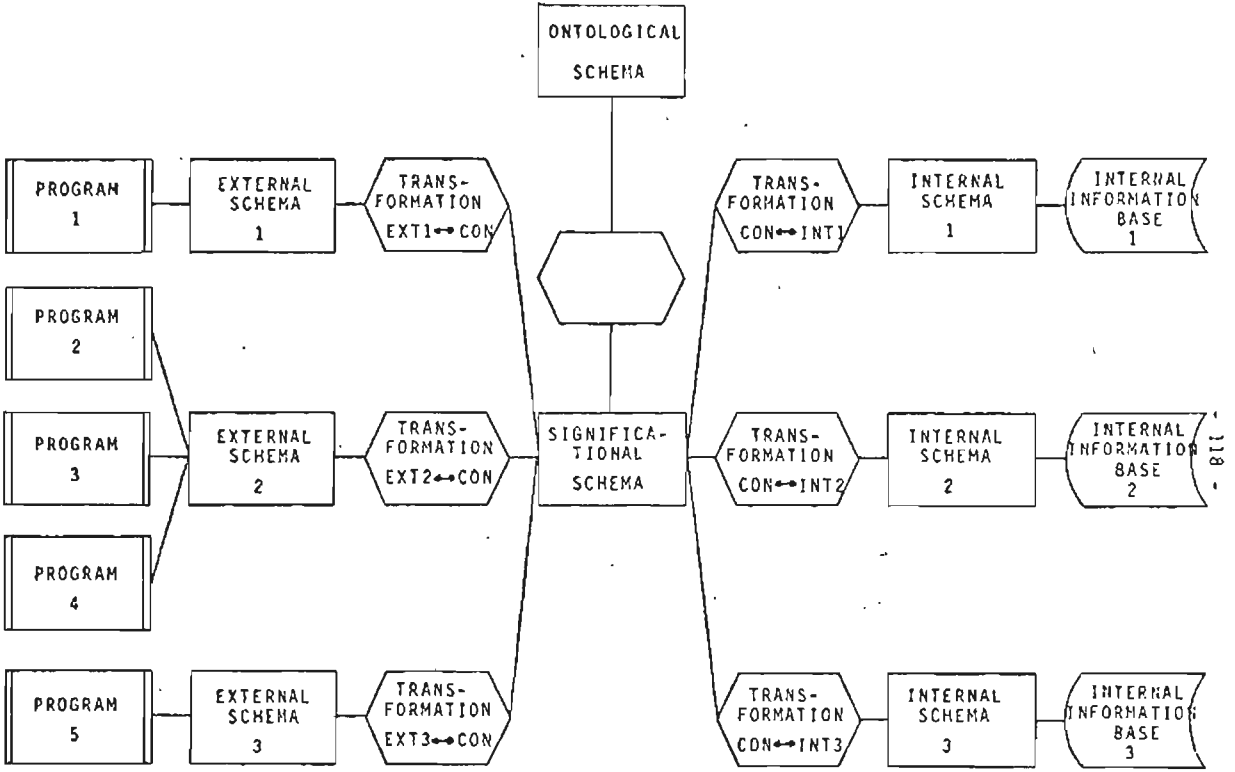

ONTOLOGICAL SCHEMA
PROGRAM 1
EXTERNAL SCHEMA 1
TRANS-FORMATION EXT1↔CON
PROGRAM 2
PROGRAM 3
EXTERNAL SCHEMA 2
TRANS-FORMATION EXT2↔CON
PROGRAM 4
PROGRAM 5
EXTERNAL SCHEMA 3
TRANS-FORMATION EXT3↔CON
SIGNIFICA-TIONAL SCHEMA
TRANS-FORMATION CON↔INT1
INTERNAL SCHEMA 1
INTERNAL INFORMATION BASE 1
TRANS-FORMATION CON↔INT2
INTERNAL SCHEMA 2
INTERNAL INFORMATION BASE 2
TRANS-FORMATION CON↔INT3
INTERNAL SCHEMA 3
INTERNAL INFORMATION BASE 3

Figure 6.a

7. Comments on "Concepts for the Conceptual Schema" edited by G.C.M. Sharman (BSI/DPSI3/WG1 DBMS; draft 1.3)

I have read with great pleasure the report listed in the title above (from here on referred to as BSIDBMS Report). It is indeed a pleasure to see that the BSIDBMS Report differs considerably from the ANSI, BCS DDSWP, CODASYL and GUIDE-SHARE reports on databases with respect to focussing on the real issues. If this kind of report is still possible as a result of a standards committee, there is still hope that one may get a good standard in the area of database management.

7.1 Introduction

In the preface to the paper it is said that the issue of the BSIDBMS paper is:

> "WHAT OBJECTS MAY BE USED IN A CONCEPTUAL SCHEMA TO DESCRIBE AN ENTERPRISE OR OTHER REAL-WORLD SYSTEM?"

ontological

and then we read: "This issue is considered to be fundamental to the future use of all existing and future database management systems."

I conclude from these two quotations that the BSI/DPSI3/WG1 (in contrast with several other groups) has clearly understood what the most pressing problem is in existing and future database systems. On this basis it must be possible to make rapid progress.

Further on in the preface we read: "It is now clear that a conceptual schema should be a product of the systems analysis and design process which precedes the implementation of a database management system. Successful formulation of an appropriate conceptual schema will in future be considered as essential prerequisite for the implementation of a computerised information system."

I fully agree with these statements; I could rephrase it by saying that a conceptual schema is made by or with the essential help of the users and each computerised information system needs one.

Near the end of the introduction of the BSIDBMS report we read: "... these concepts (conceptual schema concepts) will ultimately prove to be simple ...". Indeed, this is necessary if users have to contribute to the definition of conceptual schemata.

And at the end we read: "... this (mathematical formalism) is not a primary consideration: practical utility must be the overriding objective". I agree with this, and would like to add that a proposed set of conceptual schema concepts needs to be tested on several "benchmarks", an idea also found in the BSIDBMS report at the end of section 1.4.

Somewhat further in the preface we read: "Currently, there are a number of approaches to the construction of a conceptual schema. These have been described in various ways using differing terminology, formalism, degree of definitional rigour, and more significantly differing fundamental concepts."
I appreciate very much that the BSI, once again in contrast with some other groups, clearly states that fundamental concepts are more significant than terminology and formalism.

I furthermore welcome the idea to have experiments with respect to the practical usability of conceptual schema design approaches as suggested at the end of section 1.4. Some approaches, some of which are even widely advocated as future standards, will fail in a honest practical test.

7.2 The Basic Philosophy

Although the BSIDBMS Report contains many positive points with respect to progress in database technology, it is a pity that I could not detect a basic philosophy in the report.

In the field of database, one may encounter such philosophies as:

a. A database is a set of records and functions among records (CODASYL approach)
b. A database is a set of time varying relations (Normalized Relational Approach)
c. An informationbase is a set of sentence instances. (ENALIM)

In my paper I have clearly stated that the basic philosophy is: "An information system is a special kind of human communication."

It is my opinion that the lack of basic philosophy is the major source of my few negative comments on the BSIDBMS Report. Once there is a clear basic philosophy in the BSIDBMS Report, it is possible to answer the open questions as well as correcting some errors.

7.3 The Information Systems Framework

In my paper I have described an information systems framework which as a first approximation contains three major components namely

- the conceptual schema
- the informationbase
- the application program

where the conceptual schema is defined as the only and complete grammar, and as a consequence the application program has no influence on the correctness of the informationbase instances. The application program may request to the DBMS to perform a conceptual transaction (or a retrieval), and it is the conceptual schema and the current informationbase instance which make the DBMS to determine whether or not the conceptual transaction will be accepted or rejected.

It is unfortunate to say that the BSIDBMS Report misses - to some degree - these three concepts and their relationships.

7.4 The meaning of Conceptual Schema

What is meant by the term "Conceptual Schema" in the BSIDBMS Report? Well, chapter 1.1, page 1 gives the answer:

> "When an analyst or systems designer sets out to build an information system, he first constructs a "mental picture" of the information that that system should contain and the function it should perform. At a later stage, the analyst will wish to write down his mental picture in order to communicate it to others. We call this written version a Conceptual Schema, regardless of what notation is used."

My question here is: Is this conceptual schema the only and entire grammar? Or, as is usual today, do the application programs contain some (usually the majority) of the correctness rules.

7.4.1 Conceptual Schema as only Grammar

In my paper I have specified that a conceptual schema consists of:

- Elementary sentence types (manipulatable construct)
- Object types (non-manipulatable)
- Constraints (non-manipulatable)

Furthermore, I have said that the constraints in the conceptual schema are all the constraints applying to the informationbase.

I know from various practical projects in which such a conceptual schema was made that the constraints are usually the major portion of the conceptual schema.

It is a pity to observe that nearly all constraints are not yet included in the discussion in the BSIDBMS Report as can be concluded from chapter 4.

In this chapter it is said that only "a diagrammatic notation for the conceptual schema should be developed" and no other.

However, as I have said in my paper, it is in general impossible to represent each constraint in a diagram.
Therefore, one needs one format of a conceptual schema which is a highly formulated and formalized text, in which the real problem is the description of the constraints; one needs a diagram, in which one can concentrate on some aspects of a conceptual schema.

7.4.2 One or Two Conceptual Schemata

As we have seen in chapter 5 of my paper, it is useful (and necessary) to distinguish two kinds of conceptual schemata, namely the ontological and significational schema and the significational-ontological transformation.

Is this distinction available in the BSIDBMS Report?

In my opinion, one can find this in the Report because one could (see chapter 1.1 and 1.2 of the BSIDBMS Report) consider the ontological schema as the object system grammar and the significational schema as the information systems grammar.

It would be useful if the BSIDBMS Report did describe the transformation between object systems and information systems.

This distinction would be useful in chapters 2.1, 2.1.1, 2.1.2 and 2.1.3.

In section 1.2 we read: "The conceptual schema is ... a document which is equivalent to the meta-model". If we introduce the distinction between the ontological and significational schema, I propose to rephrase this as follows:
"The ontological schema is ... a document which is equivalent to the meta-model".

Section 1.4 of the BSIDBMS report contains a few names for concepts which are differently named in my paper. The table below gives the correspondance.

BSIDBMS	NIJSSEN
model	informationbase instance
meta model	conceptual schema (informationbase schema)
meta meta model	conceptual metaschema or meta conceptual schema

7.4.3 How Many Basic Constructs?

One of the most often misunderstood problems has to do with the basic constructs in the conceptual schema; or, to use a better term, the manipulatable construct. A manipulatable construct has the property that one can add construct instances to the informationbase, that we can delete construct instances from the informationbase, and that one can select an arbitrarily defined subset of construct instances from the informationbase.

It is here where we need a clear basic philosophy. Namely, in case an informationbase contains a set of one or more sentence instances, it is clear that the only manipulatable construct is a sentence.

Because we do not want to operate on whatever compound sentence is selected, we consider only atomic sentences to be the only manipulatable construct at the conceptual level. This is no restriction, because all forms of compound sentences can be constructed from atomic sentences.

In other words, because our basic philosophy is that an information system is a special case of human communication, the traffic between the human beings and the information system is in SENTENCE instances. And hence there is no reason to have more than one manipulatable construct.

Proposals which want as basic constructs entities and relationships, or entities, relationships and attributes are not based on the assumption that a conceptual schema is a grammar for an INFORMATION system.

Because of the foregoing, it is clear that I strongly disagree with the first paragraph of chapter 1.4 of the BSIDBMS Report.

Remark. I think it is good to remember that each construct has to be accompanied with two update operators and one or more retrieval operator.

7.4.4 Conceptual Transaction

In section 2.4 of my paper, I have defined the concept of "Conceptual Transaction". The answer to the question in the BSIDBMS Report in chapter 1.3 (at the end), namely "which is the smallest step between two states" can now be answered as follows: the smallest step between information-base instances (states) is a conceptual transaction.

Furthermore, in the BSIDBMS report one distinguishes at the end of chapter 1.3 "two basic approaches

- constraint oriented
  the meta-model contains assertions which must be true of the model at all times

- function oriented
  the meta-model defines those permitted functions which take the model from one state to another.

Discussion point: are there approaches mutually exclusive?"

My answer is clearly no. A conceptual schema defines both all the permitted states as well as all the permitted state transitions as I have said at page 21 and 22 of my paper.

7.4.5 Type, Population and Instance

Contrary to several other reports, it is encouraging to read in chapter 2.2 of the BSIDBMS Report that one clearly distinguishes between the construct type and the construct population.

However, not everywhere in the report one has carefully used the distinction. E.g. in the introduction we read: "... a (conceptual) schema comprises a unique central description

of the information contents of a user system". I would propose to state this as follows: a conceptual schema comprises a unique central description of every possible informationbase population of a user system. This is to emphasize that a conceptual schema is the same as the informationbase type, which permits a set of informationbase populations.

7.5 Conceptual schema concepts

It is in this area where I have major disagreement with the BSIDBMS report, and more specifically chapter 2 of the BSIDBMS report. In this chapter we read that there are three basic types in the meta model, namely

- entity type
- relationship type
- value type

The first problem is that probably the most important thing of any conceptual schema is not mentioned, namely the constraint.

Secondly, because there is no distinction between ontological and significational schemata, there is confusion over entities and values as illustrated by the question at the end of section 1.1.

And thirdly, it is said in the BSIDBMS report that one can manipulate entities and relationships. But where do we find the justification for such an approach?

My proposal is to concentrate the discussion on manipulatable constructs and other constructs in the conceptual schemata. And, I propose to have only one manipulatable construct in the significational realm, namely the atomic sentence.

7.6 Conceptual Schema and Data Dictionary

On page 2, chapter 1.2, we read in the BSIDBMS Report:
"The conceptual schema is ... a document which is equivalent to the meta-model. Of course, the meta-model may itself be implemented as an information system (e.g. a data dictionary system) either separate from or integrated with the information system which implements the model."

Here we encounter a very essential aspect, namely the conceptual schema can be the object of an information system, say the meta-information system.

I think that it is essential for ensuring correctness that the information system and meta-information system are integrated; what holds for user information holds also for the information called conceptual schema. I, therefore, propose not to accept the "separate" option.

I think that chapter 1.2 is essential because it shows the relationship between two related levels. In my concepts and terminology I prefer to represent it as in figure 7.6.a.

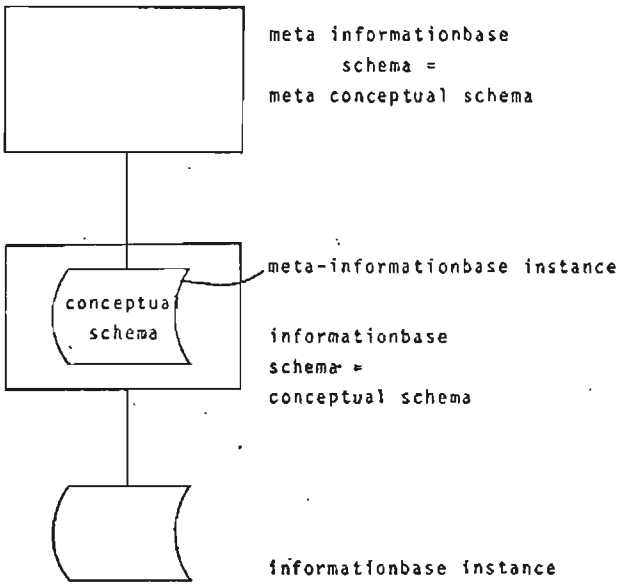


Figure 7.6.a

7.7 Rejection of the Record and Relation Concepts as Conceptual Schema Concepts

In chapter 1.3 of the BSIDBMS Report the serious deficiencies of the record approach of CODASYL as well as the Normalized Relational approach are very briefly touched. I think it is worthwhile to make this more clear because to many people, involved in the standardization of database languages, this is not at all clear.

8. Answers to the questions of 78.05/03

In this chapter we will answer the discussion points of working paper 78.05/03, however in a different sequence than in 78.05/03.

8.1 Purposes and roles of conceptual schema (3)

The purpose of the conceptual schema is to be the complete and only grammar of an information system. As such it is the only documentation which is used by users and information system specialists to define the "what" of an information system.

The conceptual schema is part of the result of the information analysis phase, as well as major input document to the implementation phase.

8.2 Should we consider two kinds of conceptual schema, with perhaps one being enterprise oriented and the other more data oriented? (2)

My answer is that we should distinguish two kinds of conceptual schemata. One is the ontological schema, which is the object system grammar; the other is the significational schema, which is the information systems grammar. One gets from the significational schema to the ontological schema by abstracting from the way names are assigned to object instances.

8.3 The conceptual schema for a given enterprise is not uniquely determined (22)

I think that it is meant to say "the conceptual schema for a given information system is not uniquely determined".

My answer is that the surface structure of two conceptual schemata, defining the same information system, may be different, but their deep structure, or their generating power is the same.

8.4 Entity types. May they overlap? Is "type" different from other attributes? What distinguishes types from other sets? (1)

There are several questions?

a. May entity types overlap?

In the model we have described it is possible that one role refers to a subset of an object type and another role refers to another subset of the same object type, but the two subsets may have an overlap.

b. Is "type" different from other attributes?

I do not understand the question.

c. What distinguishes types from other sets?

I think one could say that a type is the union of all possible populations, where a population is a set.

8.5 Are relationships also entities? Can there be multiple occurrences of the same relationship type between the same two entity instances? (4)

Here again two questions

a. Are relationships also entities?

My answer is that there are objectified sentences in the significational schema. The reason for this is the phenomenon of classifying names for object instances.

b. Can there be multiple occurrences of the same relationship type between the same two entity instances?

My answer is no. Why? Because there is no way to distinguish between the multiple occurrences. $\{a,a,b\} = \{a,b\}$.

8.6 Relationship types. Are they disjoint? (5)

In our significational as well as ontological schema, the manipulatable construct types are disjoint.

8.7 Clarify the notions of "value" and "value system". How is a value distinct from a symbol? Are values atomic? (6)

All three questions may be answered by saying that it is necessary to distinguish very carefully between things and names of things.

Thereafter one may distinguish between different kinds of names such as:

- identifying-only
- identifying-only-with-synonyms
- classifying
- classifying-with-synonyms

8.8 Identifiers (7)

Is identifier another name for the concept name?

8.9 Is "attribute" a distinct concept from relationships and entities? (8)

As said before, the only manipulatable construct at the significational (conceptual) level is an atomic sentence. Objects play a certain role in such an atomic sentence but objects are not manipulatable because it are only sentence instances which are exchanged between human beings and the informationbase.

Attributes and relationships are surface (external view) structures of atomic sentences.

And, atomic sentences are clearly distinct from objects.

8.10 Define what it means for an entity to be "in" the model. Are they necessarily finite? When is explicit creation required? (9)

This ternary question will be broken down into three questions.

a. Define what it means for an entity to be "in" the model.

We can redefine this by asking: "define what it means for an object to be "in" the informationbase?

An object is in the informationbase if at least one sentence instance is in the informationbase which refers to the object.

b. Are they necessarily finite?

Question is not clear.

c. When is explicit creation required?

In the approach with atomic sentences an object is never explicitly created, because it are only sentence instances which are created in the informationbase, or in BSIOBMS terms, created in the model.

8.11 The same for relationships (10)

An instance of an atomic sentence is in the model if it is in the informationbase. An atomic sentence instance can enter into the informationbase by the execution of an ADD atomic-sentence-instance command.

8.12 Re-examine the separation between model and meta-model (11)

We may rephrase this question into: re-examine the separation between informationbase instance and informationbase schema.

The answer is that an informationbase schema defines a set of informationbase instances.

8.13 Binary vs. n-ary relationships (12)

In my opinion this is an irrelevant question for the conceptual level. In the significational schema one describes semantically atomic sentences; some of these can be described with the mathematical model of binary, some with n-ary (with n>2) relations.

8.14 One-to-many vs. many-to-many relationships (13)

This again is an empty question. For some atomic sentences one wants a constraint of the kind one-to-many, for others one wants the constraint many-to-many. In other words, there is no vs.

Remark. The restriction of some well-known database approach to one-to-many seems to have its influence.

8.15 Multiple entity types may occur in one domain of a relationship (14)

In my opinion this is the same problem as discussed in 8.4.

8.16 How is time reflected in the conceptual schema? (15)

My answer is that time is reflected in the conceptual schema just like it is reflected in atomic sentence instances in which one role refers to time.

On the other hand, it might be useful to introduce special time-oriented retrieval operators.

8.17 How are events reflected in the conceptual schema? (16)

An event can be described by one or more related atomic sentence instances and are thus of no special nature.

8.18 Model dynamics (17)

In our approach we have defined a conceptual schema as a set of rules describing which informationbase instances are permitted as well as which informationbase instances may follow a given informationbase instance. Hence the model dynamics are completely covered.

8.19 Constraints and implications. (propagations) (18)

We have stated that the conceptual schema contains all the constraints. It is not clear to me what is meant by implications. Is meant that some constraints imply another stated constraint? This is certainly an area for future research.

8.20 Does a pictorial form in itself impose certain constraints on the concepts for the conceptual schema? (20)

The answer is clearly yes, if the pictorial form is the only form. It is namely impossible to have a pictorial form for every kind of constraint. And because one needs all kinds of constraints in the conceptual schema, it is clear that one needs at least two kinds of presentations of a conceptual schema, namely a text which is a complete and a pictorial form which only represents certain aspects of the conceptual schema.

9. CONCLUSIONS

In this paper we have to make an effort to show that the Coexistence architecture with three schemata, in which the conceptual schema defines completely the dynamic behaviour of the information base, needs to undergo a modification. It is my opinion that the conceptual schema needs to be replaced by two other schemas, namely a significational schema (= conceptual schema without abstraction from names) and an ontological schema (= conceptual schema with abstraction from names). It is furthermore our opinion that the basic construct in the significational schema is the elementary sentence, and the basic construct in the ontological schema is the binary association or binary idea.

I hope that this enlarged conceptual framework will contribute to the timely finish of the second great debate such that the experts involved in that debate can spend their attention to more useful aspects of database management.

REFERENCES

1. ANSI

Interim Report Study Group on Data Base Management Systems

American National Standards Institute, ANSI/X3/SPARC DBMS Study Group, February 1975.

2. ANSI

The ANSI/X3/SPARC DBMS FRAMEWORK

Report of the Study Group on Data Base Management Systems, edited by D. Tsichritzis and A. Klug; AFIPS Press, 1977.

3. BACHMAN C.W.

Data Structure Diagrams

Data Base, Volume 1, Summer 1969, nr. 2

4. BENCI E. et al.

Concepts for the design of a conceptual schema

In the same proceedings as 25.

5. BILLER H. and NEUHOLD E.J.

Concepts for the conceptual schema

In the same proceedings as 26.

6. BRACCHI G., PAOLINI P. and PELAGATTI G.

Binary logical associations in data modelling

In the same proceedings as 25.

7. BUBENKO J.A.

The temporal discussion in information modelling

In the same proceedings as 26.

8. CODASYL

Data Base Task Group Report 1969, ACM, New York.

9. CODASYL

Data Base Task Group Report 1971, ACM, New York.

10. CODASYL

DDL Journal of Development, June 1973 Report.

11. CODASYL

COBOL Journal of Development

Published by the Technical Services Branch, Department of Supply and Services, Ottawa, Ontario, Canada, 1975.

12. CODASYL

DDL Journal of Development, 1978.

13. CODD E.F.

Recent Investigations in Relational Data Base Systems

In: Information Processing, 1974, Proceedings of IFIP Congress. August 5-10, 1974, Sweden, North-Holland, Amsterdam 1974.

14. DATE C.J.

An Introduction to Database Systems

Addison-Wesley Publishing Company, U.S.A., 1977 (second edition).

15. DE JONG W.

Experience with a New Database Approach

Paper presented at the IFIP IAG Conference, "Managers, Databases and Information Systems", held November 28-30, 1977, Amsterdam, Proceedings by North-Holland.

16. DURCHHOLZ R. and RICHTER G.

Information Management Concepts (IMC) for Use with DBMS Interfaces

In the same proceedings as 25.

17. FALKENBERG E.

Concepts for modelling information

In the same proceedings as 25.

18. FALKENBERG E.

Significations: The Key to Unify Data Base Management

Information Systems, vol. 2, no. 1, 1976, 19-28.

19. FALKENBERG E.

Data Models: the next five years

Paper presented at INFOTECH Conference "Data Base - The Next Five Years", December 12-14, 1977, London.

20. FILLMORE

The Case for Case

In Universals in Linguistic Theory, Bach and Harms (eds), 1-90.

21. HALL P., OWLETT J., TODD S.

Relations and entities

In the same proceedings as 25.

22. KENT W.

Entities and relationships in information

In the same proceedings as 26.

23. MERCZ L.

How to increase Data Base Programming Productivity

Paper presented at INFOTECH Conference "Data Base - The Next Five Years", December 12-14, 1977, London.

24. HOULIN P. et al.

Conceptual model as a database design tool

In the same proceedings as 25.

25. NIJSSEN G.M.

A gross architecture for the next generation Database Management Systems

In: Modelling in data base management systems, Proceedings of the IFIP Working Conference held in Freudenstadt, Germany, 5-8 January 1976, North-Holland Publishing Company, Amsterdam 1976.

26. NIJSSEN G.M.

Current Issues in Conceptual Schema Concepts

In: Architecture and models in data base management systems, Proceedings of the IFIP Working Conference, held in Nice, France, 3-7 January 1977, North-Holland Publishing Company, Amsterdam 1977.

27. NIJSSEN G.M.

On the Gross Architecture for the Next Generation Database Management Systems

Invited paper, presented at the IFIP Congress, August 1977, Toronto, Canada; Proceedings by North-Holland Publishing Company, 1977.

28. NIJHEIJER E.

Higher Programming Productivity through better Database Concepts

Paper presented at Infotech Conference, October 1977 Copenhagen.

29. O'CONNELL M.

The CODASYL DDLC 1978 Proposals

Paper presented at Infotech Conference "Data Base - The Next Five Years", December 12-14, 1977, London.

30. SENKO M.E.

DIAM as a detailed example of the ANSI SPARC architecture

In the same proceedings as 25.

31. SCHMID H.A.

An analysis of some constructs for conceptual models

In the same proceedings as 26.

32. STAMPER R.

Physical objects, human discourse and formal systems

In the same proceedings as 26.

33. STEEL T.B. Jr.

The 1977 ANSI/SPARC DBMS Proposals

Paper presented at Infotech Conference "Data Base - The Next Five Years", December 12-14, 1977, London.

34. TOZER E.E.

The CODASYL DSDL 1978 Proposals

Paper presented at Infotech Conference "Data Base - The Next Five Years", December 12-14, 1977, London.

35. VANDIJCK E.

A Survey of Relational Languages

Paper presented at Infotech Conference "Data Base - The Next Five Years", December 12-14, 1977, London.

36. WITTGENSTEIN L.

Tractatus Logico-Philosophicus

Routledge Kegan Paul, London.

ACKNOWLEDGEMENT

It is my pleasure to acknowledge many interesting discussions with the members of the EDMS group who have successfully implemented at Control Data Europe, the third version of a Database Management System providing as external interface an improved CODASYL network view and a normalized relational view, and various manipulation languages. The system is called CYBER-EDMS and is currently used in real production at a few sites. I am furthermore indebted to Jitze Couperus and Jim de la Beaujardiere, both of Control Data development California, and H. Sol of the University of Groningen, Holland, for valuable criticism and suggestions on an earlier draft of this paper.