

AN
ARCHITECTURE
FOR
KNOWLEDGE
BASE
SOFTWARE

G.M. NIJSSEN
CONTROL DATA
UNIVERSITY OF BRUSSELS

July, 1981

© G.M. NIJSSEN, 1981

Paper ~~presented~~ presented to the Australian Computer Society,
July 30, 1981, Melbourne.

Published in Proceedings SPOT-2 conference, Stockholm,
1981

Table of Contents

Summary

1. Reality and Information System
2. Architecture of a conceptual information system
3. Architecture of an information base management system
4. The META principle
5. Overall architecture of an integrated information systems development and support software
6. Summary and Conclusions

Foreword

The aim of this paper is to describe an overall architectural framework for database management software and closely related areas, such as information systems design and software engineering software; a significant part of the latter two is sometimes called data dictionary.

The format of this paper has been selected in order to satisfy two goals, namely being easily understandable to management, users and EDP experts, while at the same time being sufficiently formal and precise. These goals are equally important.

The architecture as described in this paper could serve the same purpose for database management as the ISO (International Standards Organization) 7 layer model performs for communications software.

In order to reach the above mentioned goals, namely understandability and precision, I have decided to make heavy use of pictures, which are tied together with some text. The symbols in the pictures are well defined and the semantics of the formal pictorial language will be explained as needed.

The structure of the paper is such that it starts with a very simple model. Thereafter this model is gradually extended by introducing a small number of principles and applying the intellectual tools of decomposition and composition and de-abstraction and abstraction.

It has been proven more than once, that those who understand this overall architectural framework for database and software engineering software should also understand:

- why Information Analysis is an effective common basis;
- why a three-schema architecture is essential for future dbms;
- what kind of tool ISDIS is;

- why we need deep structure languages like RIDL to serve as the basis for a family of new languages;
- why not only traditional data base problems (in commercial or administrative areas) can be solved with this approach, but also specialized problem areas such as compilers;
- why we need a database management system which permits an application program to communicate with more than one database;
- why both software specification software and the database management software of the next generation need to be an integrated package;
- what distributed databases are and how they can be implemented without introducing information pollution;

If you find the preceding questions of sufficient interest, please use some of your energy to understand the pages which follow.

1. REALITY and INFORMATION SYSTEM

The first very high level model as presented in figure 1-1 will be used to specify which part will be the object of further discussion and which will be excluded.

In figure 1-1 there are three components, the Universe of Discourse, the Information System and the Environment, and four flows b1, b2, b3 and b4. (The terms used here are those of the ISO report, Concepts and Terminology for the Conceptual Schema, ISO TC97/SC5/WG3, February 1981, reference 10).

The Universe of Discourse is that part of the real world which is of interest for a specific information system. It could be seats on a flight, the network of a power system, the marital status of employees, etc. We will sometimes use the abbreviation UoD in stead of Universe of Discourse.

The Environment (a user or a sensor) is a system which receives signals (via arrow b3; e.g. electric, electromagnetic, acoustic, etc.) from the Universe of Discourse and transforms these signals into linguistic messages which are sent via arrow b1 to the Information System, and which receives (e.g. via a screen, or printer) linguistic messages (via arrow b2) from the Information System, and usually transforms these into signals (actions) which are sent to the Universe of Discourse. The result of such signals (actions) may be a change in the Universe of Discourse.

The Information System is a system which maintains a store of linguistic messages describing the Universe of Discourse, and answers questions about its store of linguistic messages.

It may be useful to illustrate these three concepts with a practical example.

Let us suppose our Universe of Discourse encompasses the pulse rates at a certain moment in time of students attending a specific workshop on Conceptual Schema design. It is clear that the Universe of Discourse, for this information system, is in this illustration,

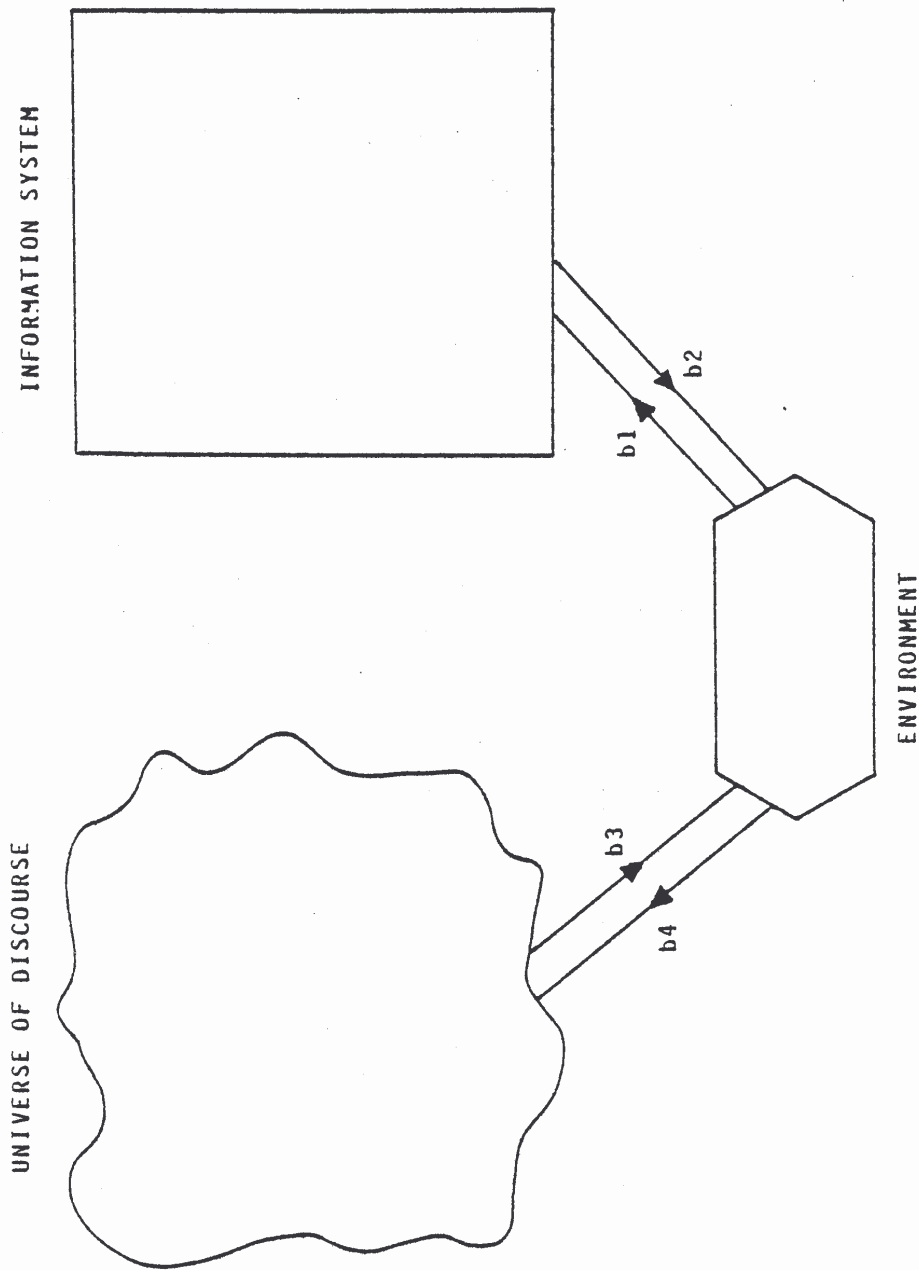


Figure 1-1

like in (nearly?) all others, a very small part of the real world.

We assume that the students have a unique reference, let us say, family name and initials; we furthermore assume that the pulse rate is meant per minute, and that "a certain moment in time" is referred to as hours and minutes (note that it could also have been just hours, or hours and quarters).

The Environment will measure the pulse rate of a specific student at a specific time, and will then forward linguistic messages to the Information System, of which the following are two examples:

The student with name Ford L.P. had at 9.35 A.M. a pulse rate of 72.

The student with name King M.L. had at 9.38 A.M. a pulse rate of 60.

These two linguistic messages are examples of the traffic among the communication line b1.

But what about the kind of traffic along b3?

In this case, we assume that the Environment is a human being, who will use the second hand of a watch and his or her fingers to count the number of pulses during 30 seconds, multiplies that number by 2, looks at the watch to know the hours and minutes, and looks at the name on a piece of carton in front of the student. These are enough signals along b3 to enable the Environment to transform these signals into linguistic messages like the 2 given above.

In the remainder of this paper we will concentrate our attention on the Information System, the Environment and the traffic between these two, represented as arrow b1 and b2.

We exclude from our discussion the Universe of Discourse and the traffic between the Universe of Discourse and the Environment, arrows b3 and b4 (see figure 1-2). This is a very interesting topic in philosophy.

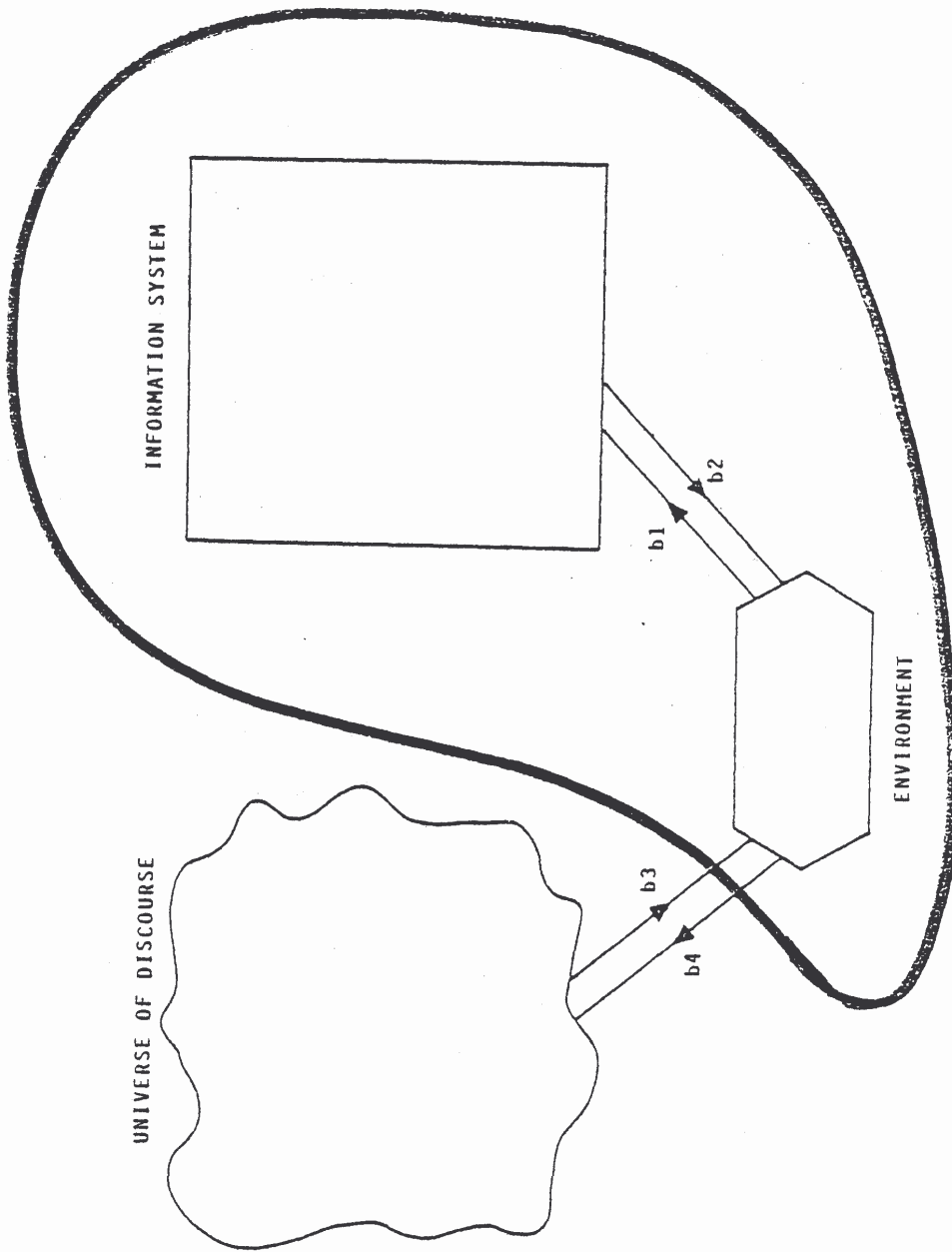


Figure 1-2

2. ARCHITECTURE OF A CONCEPTUAL INFORMATION SYSTEM

In this chapter we will start with a model of an information system which consists only of 3 components (see figure 2.1),

- the Environment
- the Information System
- the traffic between those two

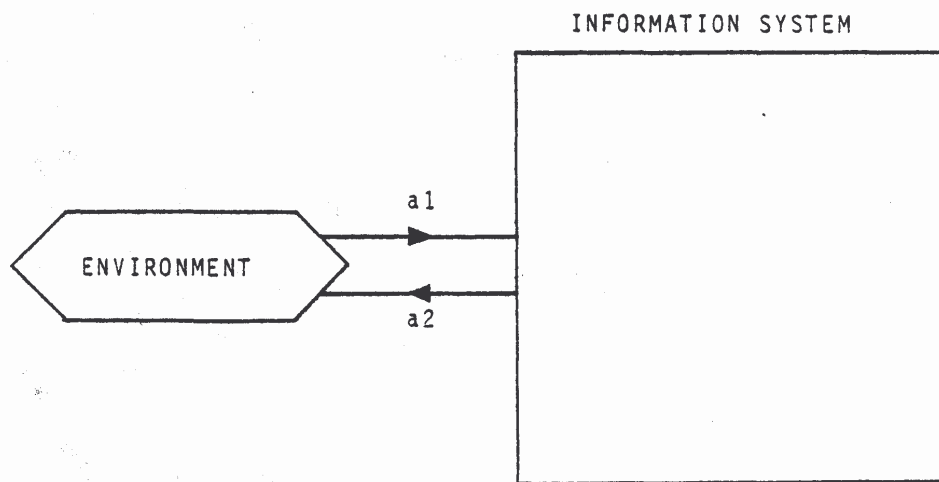


Figure 2.1

We will now formulate the first principle of our approach to information systems and databases by making a statement about the traffic between the Environment and the Information System. Later on in this chapter as well as in later sections we will concentrate on the contents of the black box, called information system.

It is essential to state explicitly that for this chapter, we are only interested in conceptual aspects; conceptual aspects can be defined as those aspects which deal exclusively with

the WHAT of a problem, not with any HOW. Later on in this paper we will introduce aspects of computer effectiveness and programming effectiveness.

ENALIM PRINCIPLE: ALL PERMITTED TRAFFIC BETWEEN THE ENVIRONMENT AND THE INFORMATION SYSTEM CONSISTS OF A SET OF DEEP STRUCTURE, ELEMENTARY SENTENCE INSTANCES.

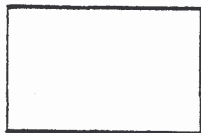
Instead of all permitted traffic, we could have said all permitted messages.

A definition of deep structure sentences and elementary sentences will be given later after we have introduced a few concepts which then permit us to give a more precise definition than is possible in this place. For now, a deep structure sentence means emphasis on the underlying meaning without the details of representation, often called surface structure in linguistics.

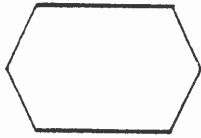
We want to stress that this principle is independent of the application area of the Information System; i.e. it is valid for a salary system, for an airline reservation system, for a compiling system of a programming language, for an accounting system, etc.

In other words, the ENALIM principle states that the traffic flowing along arrows a1 and a2 of figure 2-1 can always be considered to consist of a set of deep structure, elementary sentence instances.

It is now time to define the meaning of the symbols used in figure 2-1.



The rectangle is a symbol which represents a function or a process. We can precisely specify how this function or process reacts for every input.



The hexagon is a symbol which represents an environment (often a user). In general it is not possible to specify how an environment reacts to an input.



The arrow represents a flow of information.

*
* The following 21 pages until 2-25 can be skipped, although it *
* discusses in more detail deep structure sentences and the *
* distinction between things and their names, which is an essential *
* aspect of the ENALIM approach. *
*

We could now ask the question: is there a template, or formula, covering all the different sentences about all the different application areas? The answer is yes.

We will start with a form which is very useful in practice. Before doing this however, we will first give five examples of deep structure, elementary sentences which are communicated between the Environment and the Information System.

S1: The employee
with the employee number
E1
works for
the department
with department number
D1.

S2: The employee
with employee number
E2
works for
the department
with department number
D1.

S3: The employee
with employee number
E3
works for
the department
with department number
D2.

S4: The employee
with employee number
E9
supervises
the employee
with employee number
E1.

S5: The employee
with employee number
E8
supervises
the employee
with employee number
E3.

If we analyze these five different sentence instances, then we can draw a few conclusions:

- a. In these sentences we make a clear distinction between kinds, classes, categories or types of things (e.g. employee, department) sometimes called entities or objects, and kinds of names for things (e.g. employee number, department number).

We will call each kind of things, or each type of objects, or each category of similar things a "non-lexical object type", abbreviated as NOLOT. (E.g. Employee is a NOLOT, just as department). We use the word non-lexical to stress that we mean the things themselves, not any lexical symbols which refer to things.

We will call each kind of name for things (e.g. employee number, department number) a "lexical object type", abbreviated as LOT.

- b. In each of these five sentences there are two lexical objects (e.g. E1 and D1 in S1; E2 and D1 in S2), etc.
- c. Each of these sentences has a predicate (e.g. works for, supervises).

After these conclusions we present the first general formula for deep structure sentences:

$$S = P (\langle \text{NOLOT-1}, \text{LOT-1}, \text{LOI-1} \rangle , \langle \text{NOLOT-2}, \text{LOT-2}, \text{LOI-2} \rangle , \dots)$$

where

S = sentence instance

P = sentence predicate

NOLOT = non-lexical object type

LOT = lexical object type

LOI = lexical object instance

Very essential in this formula is the distinction between things (non-lexical objects) and the names of things (lexical objects). This same distinction is an essential part of our natural language and this was the overriding argument to include this distinction as a basic aspect in our conceptual approach.

To get more familiar with the above listed general formula for deep structure sentences, we present at the next page the sentences of page 2-4 extended with the corresponding constituents of the formula.

S1: The employee.....	non-lexical object type
with the employee number....	lexical object type
E1..	lexical object instance
works for.....	predicate
the department.....	non-lexical object-type
with department number....	lexical object type
D1..	lexical object instance
S2: The employee.....	non-lexical object type
with employee number.....	lexical object type
E2.....	lexical object instance
works for.....	predicate
the department.....	non-lexical object type
with department number....	lexical object type
D1..	lexical object instance
S3: The employee.....	non-lexical object type
with employee number.....	lexical object type
E3.....	lexical object instance
works for.....	predicate
the department.....	non-lexical object type
with department number....	lexical object type
D2..	lexical object instance
S4: The employee.....	non-lexical object type
with employee number.....	lexical object type
E9.....	lexical object instance
supervises.....	predicate
the employee.....	non-lexical object type
with employee number.....	lexical object type
E1.....	lexical object instance
S5: The employee.....	non-lexical object type
with employee number.....	lexical object type
E8.....	lexical object instance
supervises.....	predicate
the employee.....	non-lexical object type
with employee number.....	lexical object type
E3.....	lexical object instance

* The next two pages may be skipped by those not interested *
 * in the logic approach *

In predicate logic one has a formula for sentences which is as follows:

$$S = P (x, y, z, \dots)$$

where S = sentence

P = predicate

x, y, z, ... = term (other word for name)

If we take this formula, then which part of S1 is predicate and which ones are the terms?

The answer is usually as follows:

E1 and D1 are terms

works for is predicate.

But what about the rest?

Sometimes, it is stated that the predicate is the following:

The employees with employee number ... works for the department with department number ...

This is a valid interpretation, if one confines oneself to predicate logic.

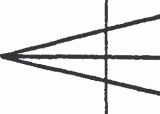
But for information systems it is useful to use a "microscope with more power" in order to detect more structural elements in the amorphous predicate. Applying an "electron microscope instead of an ordinary microscope" to deep structure, elementary sentences, we can distinguish in the predicate (as seen by the logicians) three structural components, namely

- NOLOTs (1 or more)
- LOTs (1 or more)
- Predicate (in a more narrow sense)

Therefore, it is very useful for information systems to extend the formula as used in logic with the introduction of two more structural components, namely the kind of thing (the NOLOT) and the kind of name for the things (the LOT) and the result is the formula of page 2-6.

In other words, it is claimed that it is useful to explicitly present the NOLOT (kind of thing) and LOT (kind of name of thing) which is either not considered, or at best, implicitly available in the predicate logic approach.

The following table gives a comparison of the structural elements of sentences in logic and the conceptual, deep structure information approach.

logic	conceptual, deep structure approach
predicate	 <p>predicate NOLOT(s) LOT(s)</p>
terms	lexical object instances
$Q (T_1, T_2, \dots)$	$R (NOLOT_1, LOT_1, LOI_1 ,$ $NOLOT_2, LOT_2, LOI_2 ,$ $\dots)$
$Q = \text{predicate}$	$R = \text{predicate}$
$T_1 = \text{term 1}$	$NOLOT_1 = \text{non-lexical object type 1}$ $LOT_1 = \text{lexical object type 1}$ $LOI_1 = \text{lexical object instance 1}$

We will use an extended tabular format as one of the additional presentation techniques for deep structure sentences. In this extended tabular format, we include various different sentence instances of the same sentence type, or sentence category. The extended table is divided by a double line to separate the type and instance information. In the first line we put the NOLOTs, the non-lexical object types, in the second one the LOTs, the lexical object types and in the third line, the predicates. (Note that there may be more than one predicate (WORKS-FOR or EMPLOYS; SUPERVISES or REPORTS-TO) which are semantically equivalent, but require a different sequence of the NOLOTs, LOTs and LOIs). Then we have the double line as a separator between the type and instance information. Below the double line we can have any number of lines containing the lexical object instances. (See figures 2-2 and 2-3)

	EMPLOYEE	DEPARTMENT	← NOLOT
	EMPLOYEE-NUMBER	DEPARTMENT-NUMBER	← LOT
	WORKS-FOR	EMPLOYS	← PREDICATE
S1	E1	D1	
S2	E2	D1	
S3	E3	D2	

Figure 2-2

	EMPLOYEE	EMPLOYEE
	EMPLOYEE-NUMBER	EMPLOYEE-NUMBER
	SUPERVISES	REPORTS-TO
S4	E9	E1
S5	E8	E3

Figure 2-3

If we look at figure 2-2 and compare the contents with the 3 upper sentences, S1, S2 and S3 of page 2-4, we could conclude that the extended tabular format is a handy format to show the similarities and differences of sentence instances belonging to the same sentence type.

Above the double line in the extended table format, we present those aspects of sentence instances which are the same for the various different sentence instances. This is the information which is characteristic for the sentence category, or sentence type or sentence class. One could say that this information (the one above the double line) is necessary for the interpretation.

Below the double line we present aspects of sentence instances which are different for the various sentence instances. This is the information which is characteristic for the sentence instance, or sentence occurrence.

Furthermore, this extended table format permits to read the sentence in any sequence, e.g. the employee with employee number E1 works for the department with department number E, or, the department with department number D employs the employee with employee number E1.

We now want to go into deeper analysis of the deep structure sentences in order to enhance our understanding of the fundamental aspects of deep structure sentences (the next 8 pages).

Let us look at sentence S1: The employee with employee number E1 works for the department with department number D1.

What we really want to express with this sentence is this:
a certain employee (which we somehow can identify) works for
a certain department (which we somehow can identify).

If we represent a certain non-lexical object instance (examples are: a specific employee, a specific department, etc.) as a dot (see figure 2-4).



Figure 2-4

and an association instance as a line connecting the object instances involved in the association instance (see figure 2-5).

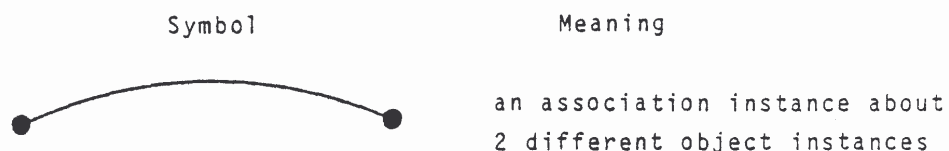


Figure 2-5

we could use the graphical representation of figure 2-5 to represent the knowledge that a certain employee works for a certain department. However, it is useful to express in the graphical symbolism to which type, class or category the various non-lexical objects belong, abbreviated as NOLOT.

This can be done by a circle enclosing the symbol for the non-lexical object instance, and attaching to the circle the name of the non-lexical object type. Figure 2-5 will then be extended to become figure 2-6.

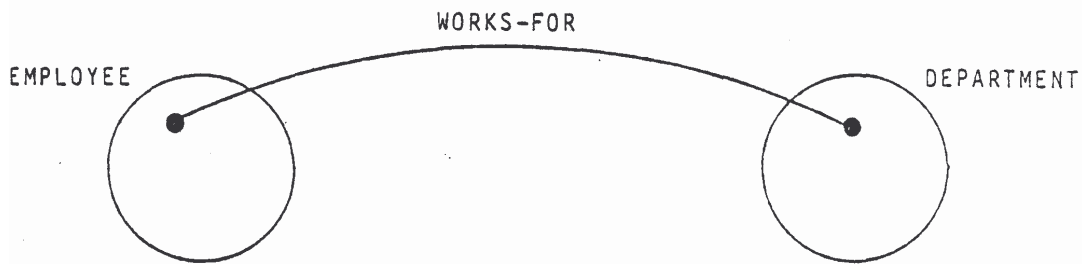


Figure 2-6

If we now want to represent with this graphical formalism that there is a certain employee (different from the one already in figure 2-6) who works for the same department (essentially S2), then this can be represented as in figure 2-7.

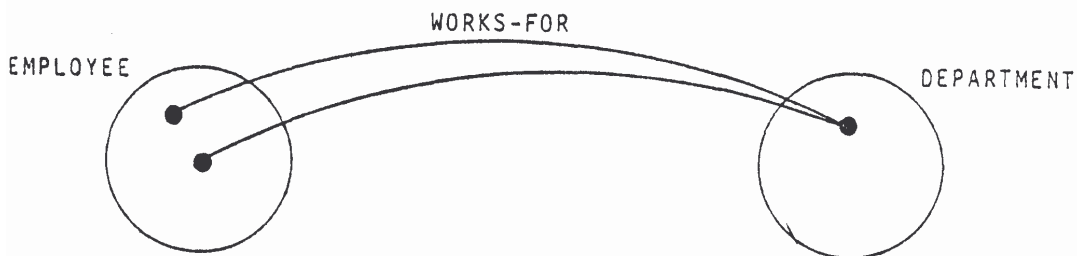


Figure 2-7

We may go on and add to the graphical knowledge representation of figure 2-7 that there is a certain employee (different from the two already in figure 2-7) who works for a certain department (different from the department already in figure 2-7). In doing so, we get figure 2-8.

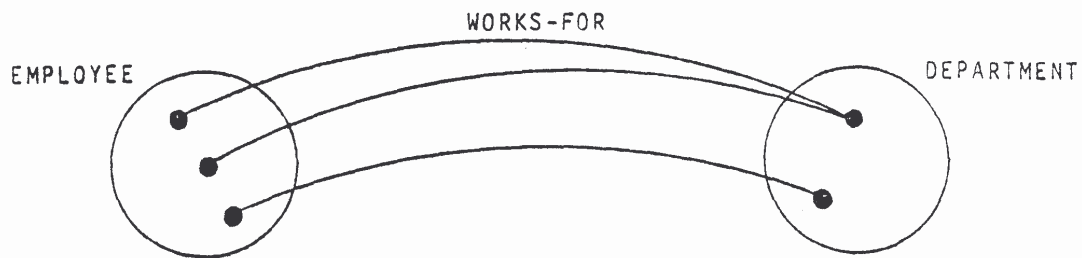


Figure 2-8

Using a similar symbolism, we could graphically represent the knowledge "There exists a certain employee with employee number E1" as is done in figure 2-9.

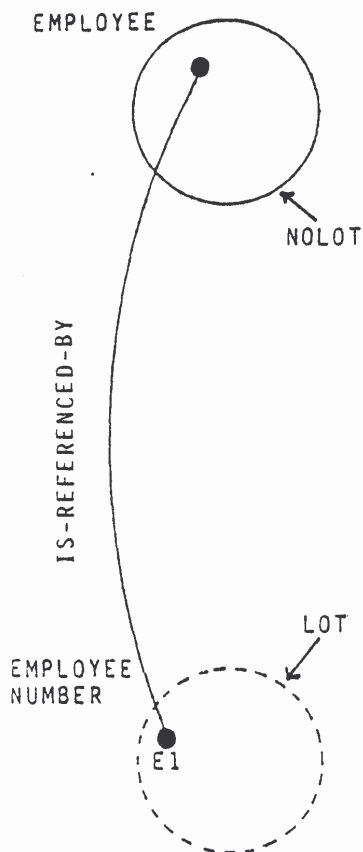


Figure 2-9

The knowledge "There exists a certain employee with employee number E1" can be regarded as two messages:

- a. "There exists a certain employee" to communicate the existence of an employee.
- b. There is a convention to refer to a specific employee with the help of the symbol (or employee number) E1.

Message a. communicates "existence", while message b. communicates a "naming convention". We take the position that message b. implies a. The reason for this is that it is our position, that an object comes only into existence if we know something about it; and knowing is only possible if there is a naming convention. (Naming conventions need not be as simple and direct as in this example as we will see later).

The knowledge "There is a certain employee with employee number E1" can be considered as a sentence instance between a non-lexical object instance (a certain employee) and a lexical object instance (employee number E1). Non-lexical object types have been expressed as circles, and lexical object types will be expressed as interrupted circles to emphasize the difference.

We can go on in this way and add to figure 2-9 that there is a certain employee (different from the one in figure 2-9) with employee number E2. In doing so, we get figure 2-10.

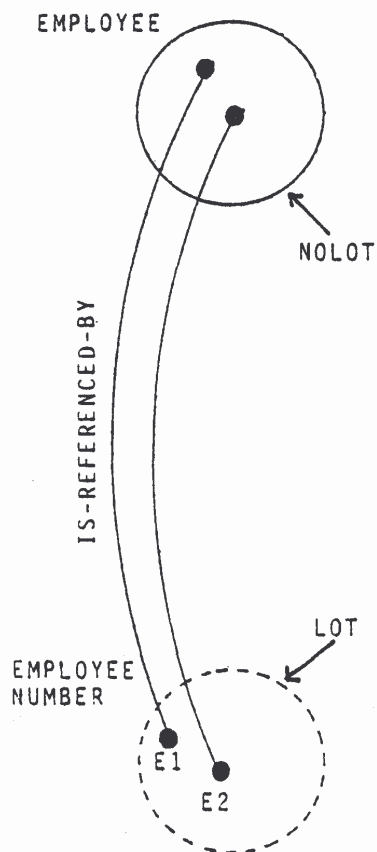


Figure 2-10

And to the knowledge represented in figure 2-10 we may add that there is a certain employee (different from the two employees already in figure 2-10) with employee number E3, and the result is figure 2-11.

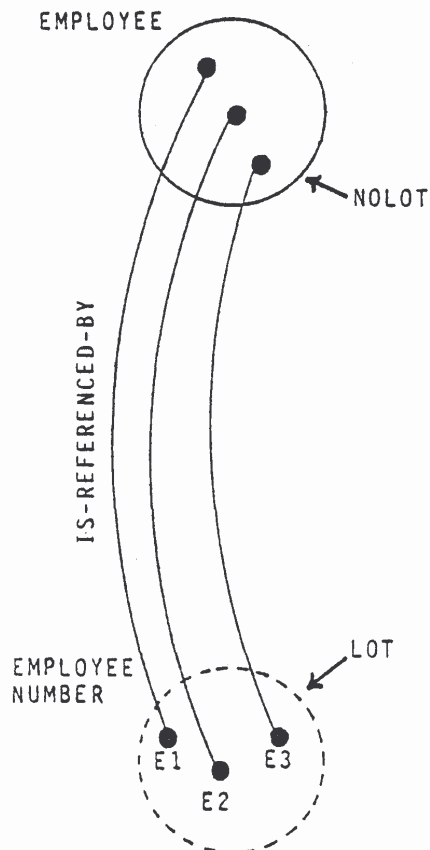


Figure 2-11

So far, we have represented in figure 2-11 the knowledge about naming conventions between three specific employees and three employee numbers.

We can do the same procedure for the two departments involved. We first represent: There is a department with department number D1.

2-17

The result is figure 2-12.

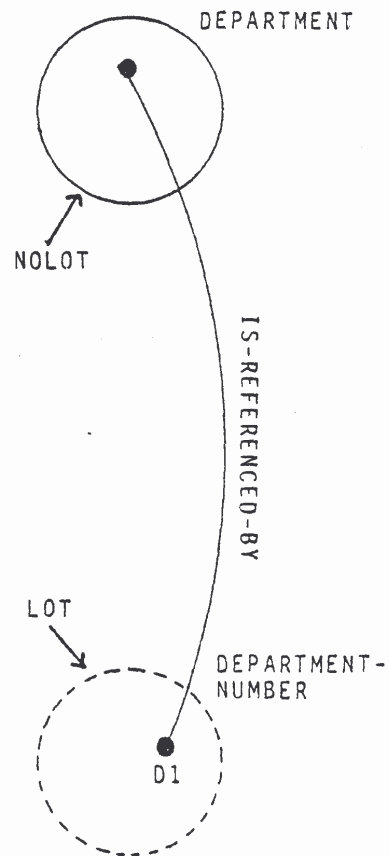


Figure 2-12

We may add to figure 2-12 that there is a department (different from the one in figure 2-12) with department number D2, and the result is represented in figure 2-13.

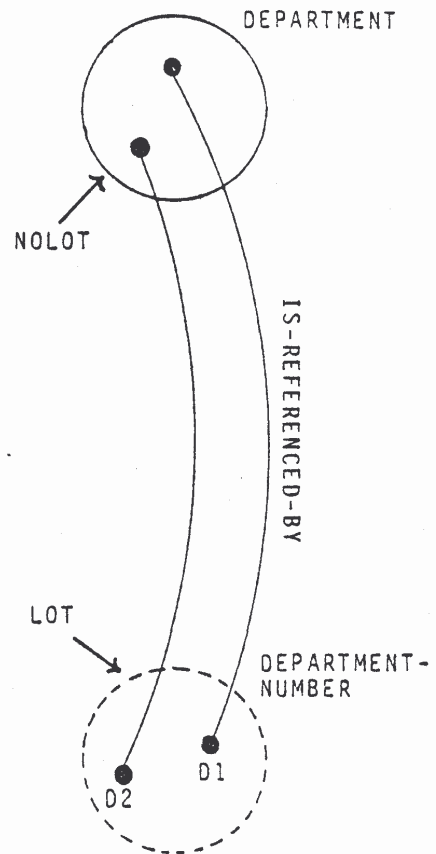


Figure 2-13

If we now combine figures 2-8, 2-11 and 2-13, we get figure 2-14.

A thorough comparison of the knowledge represented in figure 2-2 and figure 2-14 reveals that these two different graphical representations represent the same deep structure sentences, namely S1, S2 and S3 of page 2-4 (or 2-7).

If we carefully look at figure 2-14, we see that there are two kinds of object types involved, namely

- non-lexical object types (uninterrupted circles)
- lexical object types (interrupted circles)

Furthermore, there are association instances between two non-lexical object instances (the upper three) and association instances between

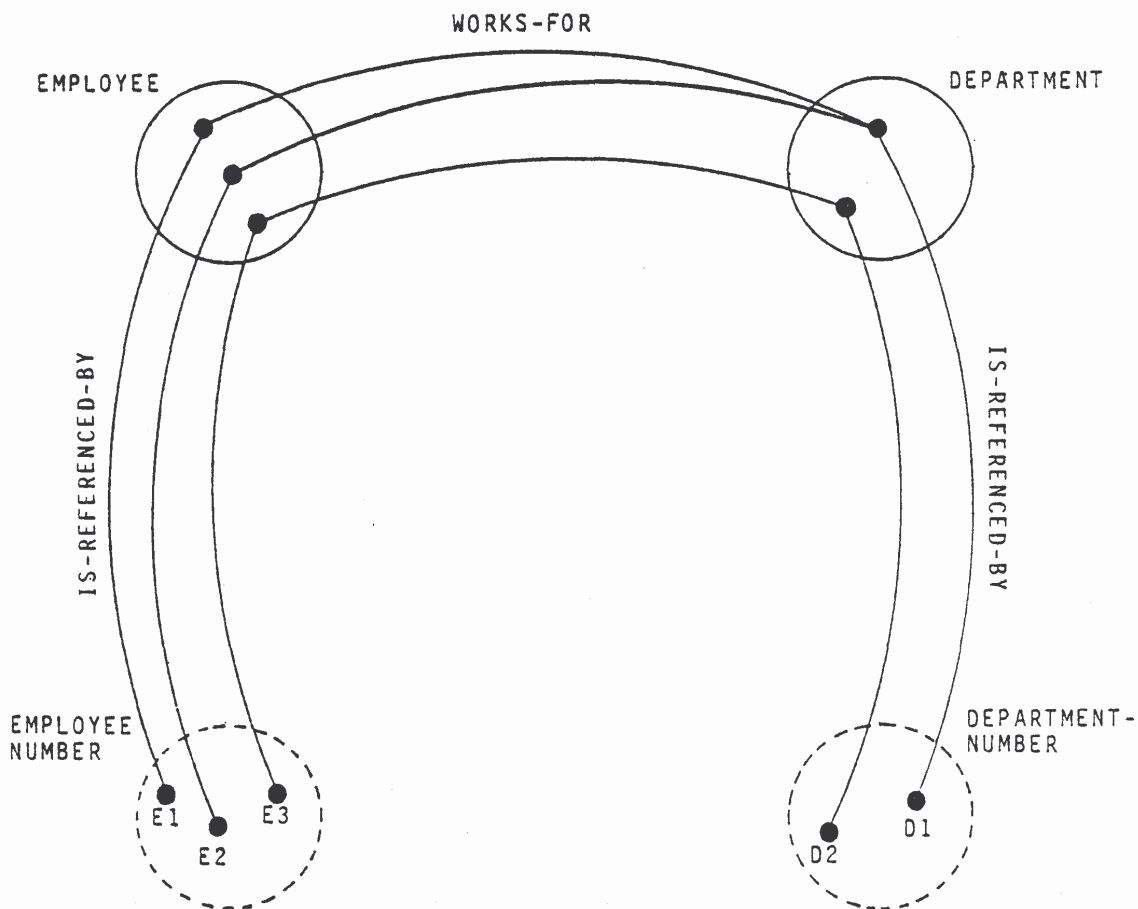


Figure 2-14

a non-lexical object instance and a lexical object instance (the left three and right two). The first category of associations we call "ideas", the second category "bridges".

To rephrase it, an idea is an association in which only non-lexical object instances are involved. A bridge is an association between a non-lexical object instance and a lexical object.

Therefore a deep structure sentence instance consists of bridge and idea instances.

Sometimes a deep structure sentence instance is just one bridge instance (e.g. there is an employee with employee number E1). Sometimes a deep structure sentence instance is an idea instance with one or more bridge instances (e.g. there is a certain employee who works for a certain department; the certain employee has employee number E1, the certain department has department number D1).

We can classify the associations into three classes depending upon the classes of objects involved, namely

- idea (association between non-lexical object instances)
- bridge (association between non-lexical object instance and lexical object instance)
- phrase (association between lexical object instances)

Question. What is the advantage of having both the extended tabular format and the graphical representation?

The answer to this question is easier to give if we first discuss the phenomenon that certain non-lexical objects have more than one lexical object associated with them. (Said in more common language, a thing can have more than one name).

Let us assume that each department has on top of its unique number a unique name. In our example we may say that the department with department number D1 is the same as the one with department name Purchasing; and D2 is the same as Sales. We could add this knowledge to the information base represented in figure 2-14 and the result is figure 2-15.

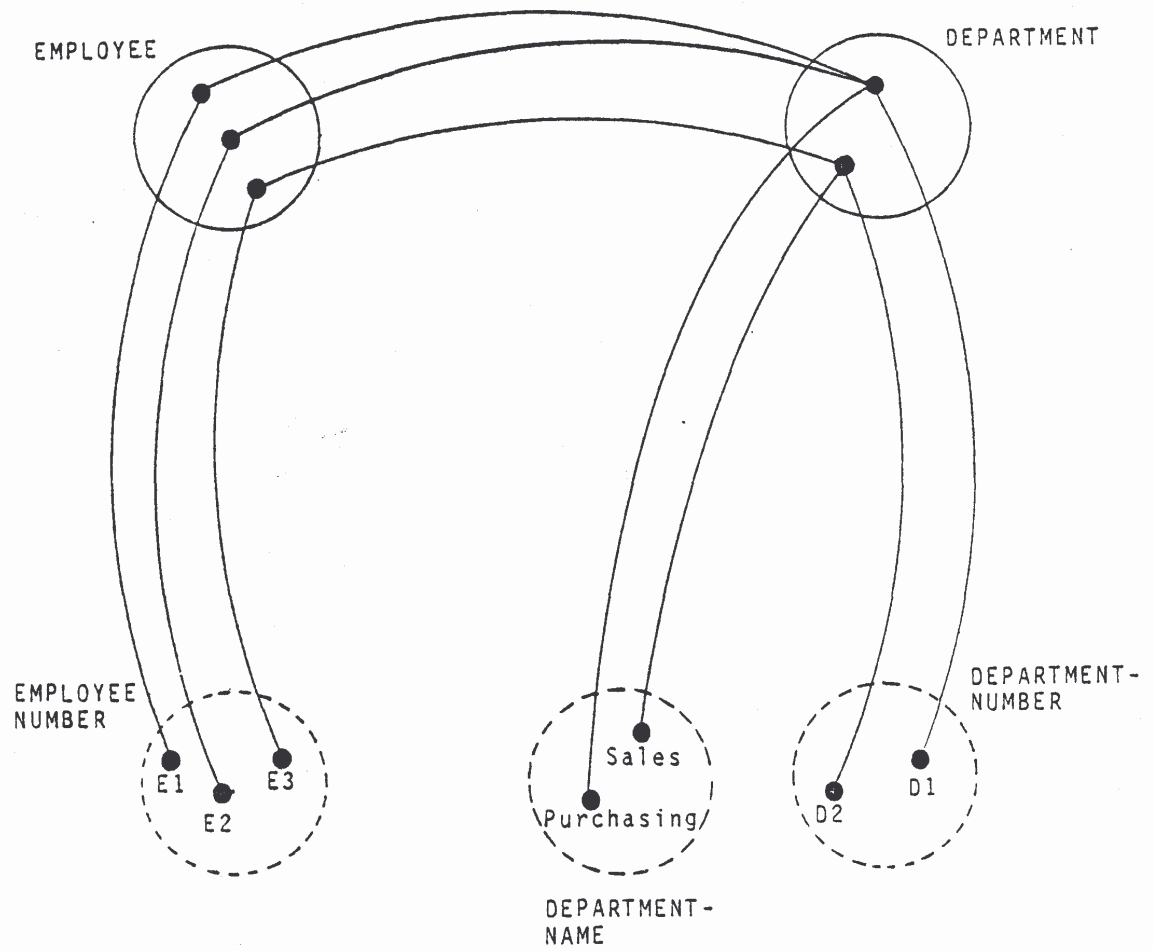


Figure 2-15

If we want to represent the knowledge of figure 2-15 in an extended table format form, we get figure 2-16. It is now necessary to establish a certain convention e.g. the appearance of a given predicate more than once indicates the presence of synonyms, i.e. the situation where a thing can have more than one name or lexical object to refer to it.

EMPLOYEE	DEPARTMENT	DEPARTMENT
EMPLOYEE-NUMBER	DEPARTMENT-NUMBER	DEPARTMENT-NAME
WORKS-FOR	EMPLOYS	EMPLOYS
E1	D1	PURCHASING
E2	D1	PURCHASING
E3	D2	SALES

Figure 2-16

Question: Which predicates show up in the extended tabular format, those associated with bridges or with ideas?

If we analyze figure 2-16, we see that only the predicates associated with the ideas are represented in the extended tabular format. Furthermore, the thing with more than one name (in case of the NOLOT DEPARTMENT) shows up more than once. In order to let the synonym aspect more easily show up, we prefer, in case of synonyms, to use an extended tabular format, where the predicate of the idea and the NOLOT of the thing with more than one name show up only once. With that convention, figure 2-16 becomes 2-17.

EMPLOYEE	DEPARTMENT	
EMPLOYEE-NUMBER	DEPARTMENT-NUMBER	DEPARTMENT-NAME
WORKS-FOR	EMPLOYS	
E1	D1	PURCHASING
E2	D1	PURCHASING
E3	D2	SALES

Figure 2-17

* *
* *
* Those persons who skipped the previous 21 pages, please *
* resume from here *
* *


From now on until the end of this chapter, we will apply the intellectual operation of decomposition to the rectangle called INFORMATION SYSTEM in order to arrive at the next level of "explosion" of INFORMATION SYSTEM. In other words, we want to find out the next level of detail of the box called INFORMATION SYSTEM.

So far in chapter 2 we have discussed the nature of the traffic (see figure 2-1) between the Environment and Information System. We will now start to discuss the contents of the black box called Information System.

If the Information System contains such a thing as a knowledge container, or information tank, or data base, or information base, then this container, tank or base consists of a set of deep structure, elementary sentence instances. Why is this so? Because the only traffic or flow into the rectangle Information System of figure 2-1 consists of arrow a1, which according to the ENALIM principle consists of a set of deep structure, elementary sentence instances.

From now on, we will use the word "Information Base" to refer to this knowledge container, information tank or data base.

We will use a special symbol to be able to represent an Information Base explicitly in our graphical language.

Symbol	Meaning
	The symbol on the left represents an Information Base. Its contents at every moment in time is a set of deep structure sentence instances, called an Information Base state. Usually different moments correspond to different Information Base states.

If we add the result of this reasoning to figure 2-1, we get figure 2-18.

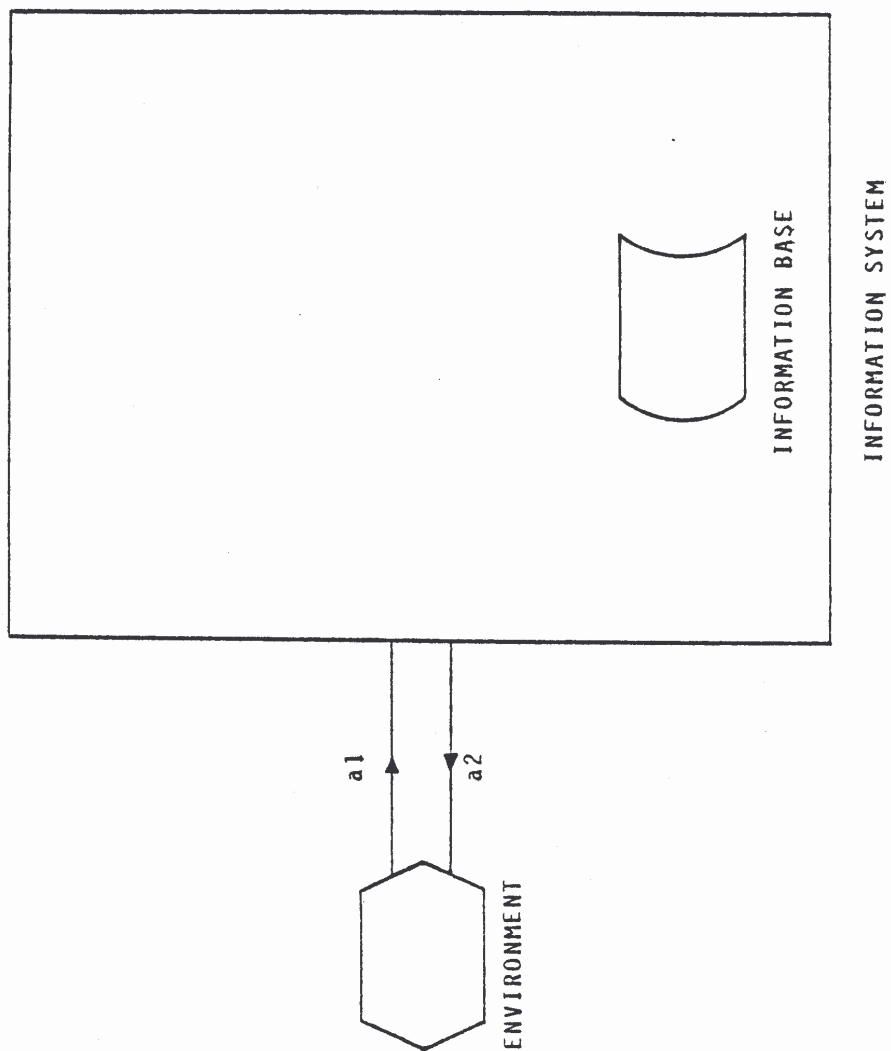


Figure 2-18

The next step in our process of decomposing the rectangle Information System is to introduce the second principle.

Before introducing the second principle, we want to note that the contents of the Information Base may be different at different moments in time, if the contents of the Information Base has to be a reflection of the Universe of Discourse. The contents of the Information Base at a given moment in time is called the Information Base state at that moment in time. Hence we have the series

$$IB_{t=0}, IB_{t=1}, IB_{t=2}, IB_{t=3}, \dots$$

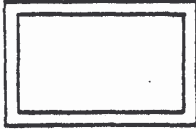
This means that we also have transitions (= the step from a given Information Base to the next Information Base state) from an Information Base state to the next.

The second or 100 PERCENT PRINCIPLE makes a statement about the permitted states of the Information Base and the permitted transitions.

100 PERCENT PRINCIPLE: THERE IS ONE GRAMMAR WHICH COMPLETELY AND EXCLUSIVELY PRESCRIBES (PREDEFINES) ALL THE PERMITTED INFORMATION BASE STATES AND ALL THE PERMITTED INFORMATION BASE TRANSITIONS. THIS GRAMMAR IS CALLED CONCEPTUAL GRAMMAR OR CONCEPTUAL SCHEMA.

Why this principle has been given the name 100 PERCENT will be explained in a later section, once we have introduced the notion of application program and a few others.

In order to represent the Conceptual Schema in the figures we need a symbol.



The double rectangle is a symbol which represents a grammar or schema.

In our process of decomposing the rectangle Information System, we may add the Conceptual Schema to figure 2-18 and get figure 2-19.

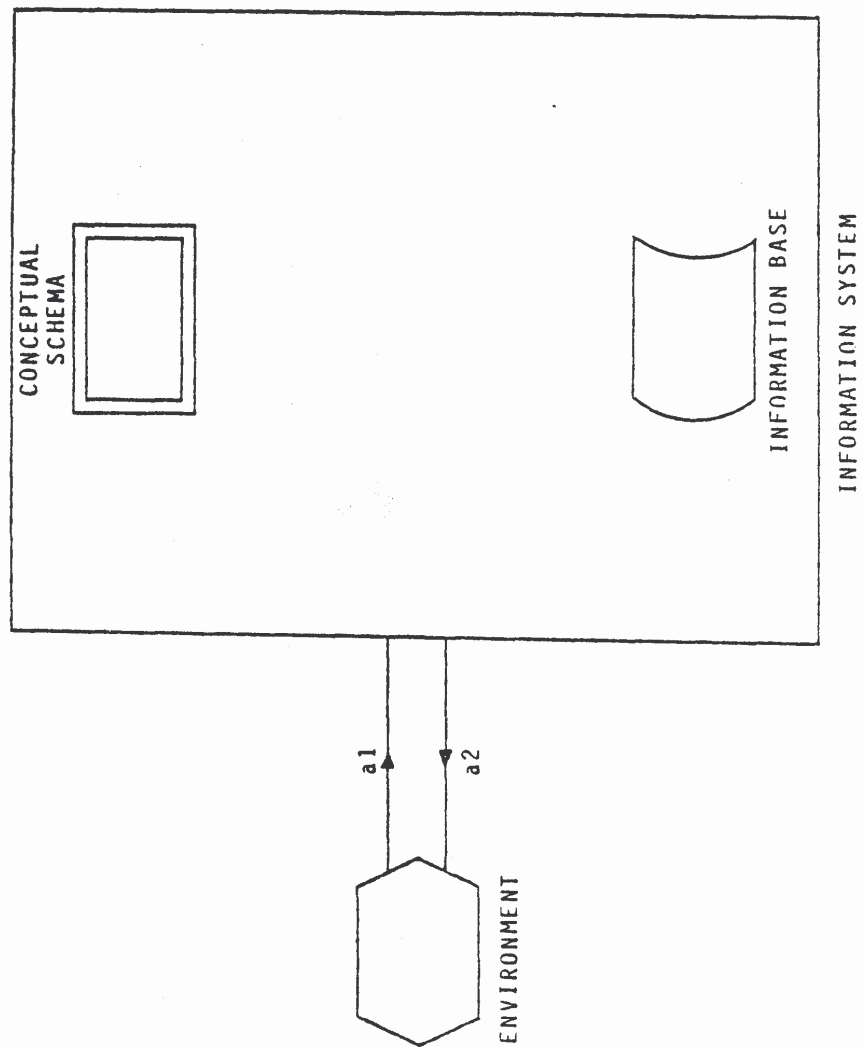


Figure 2-19

In the remainder of chapter 2, we will accept the Conceptual Schema as a black box, and continue the decomposition process of the black box Information System of figure 2-19.

A Conceptual Schema is like a contract or a law. It predefines (prescribes) for the information base which happenings are permitted and which are forbidden.

A Conceptual Schema prescribes which information base states and transitions are permitted, but it is just like a law, it can not "force" the information base to behave according to the rules of the conceptual schema.

Because we are extremely interested in having Information Base states and transitions which satisfy the Conceptual Schema, we will introduce an agency which has as one of its responsibilities to guarantee that only Information Base states and transitions can come into existence which satisfy the Conceptual Schema. This agency performs a process or function and is called Conceptual Information Processor.

The word Conceptual is used to stress that this processor is only concerned with the WHAT aspects of the information, not with any HOW.

The Conceptual Information Processor can, in our process of decomposing the Information System, be added to figure 2-19 and we get figure 2-20.

In order to be able to perform its function, the Conceptual Information Processor needs to be able to read the Conceptual Grammar (or law). We therefore introduce the arrow a5 from the Conceptual Grammar to the Conceptual Information Processor (see figure 2-21).

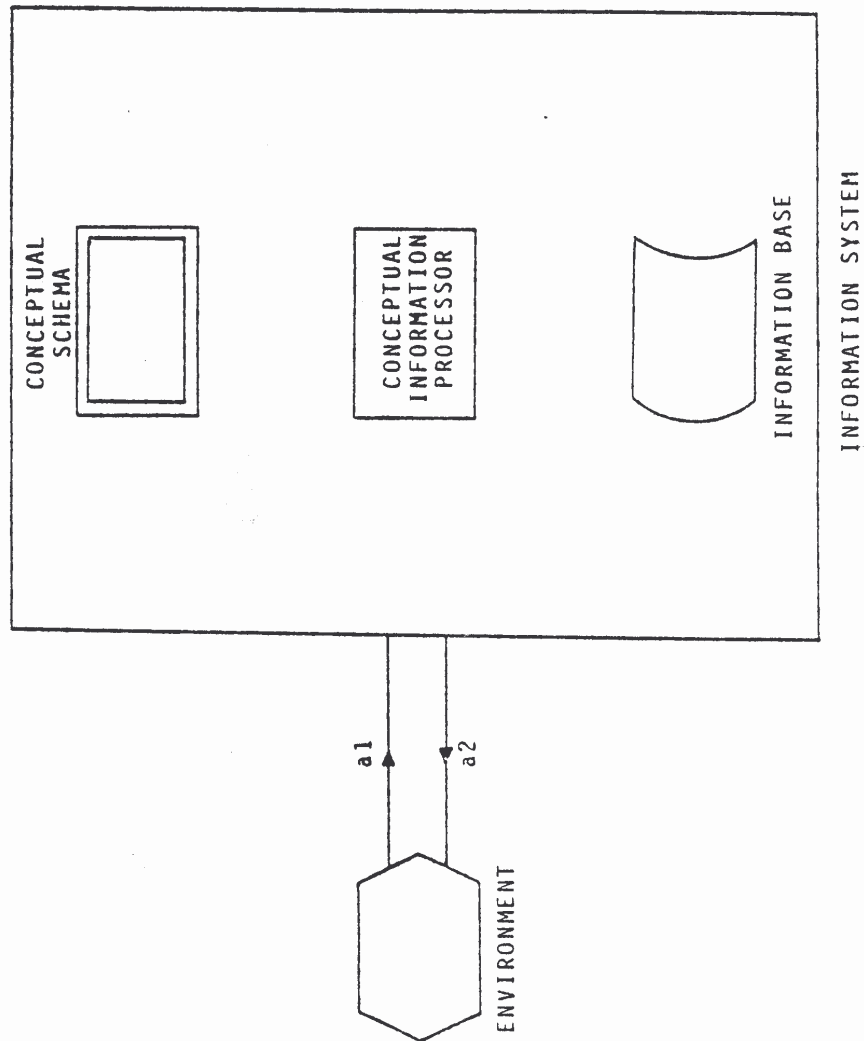


Figure 2-20

But the Conceptual Information Processor needs to have the capability to write in the Information Base. In fact it is the only unit which may write in the Information Base. Hence we extend figure 2-20 to include the writing capability of the Conceptual Information Processor into the Information Base, represented by arrow a6, and the result is figure 2-21.

Before new sentences can be added to the Information Base, the Conceptual Information Processor not only needs to read the Conceptual Schema (the law), it also needs to know what is already in the Information Base. Hence, there is a need for a reading capability between the Conceptual Information Processor and the Information Base, which is represented by arrow a7 in figure 2-21.

It is useful to remark that the things represented by



and



can read (access) and write (update). They are active components.

The things represented by



and



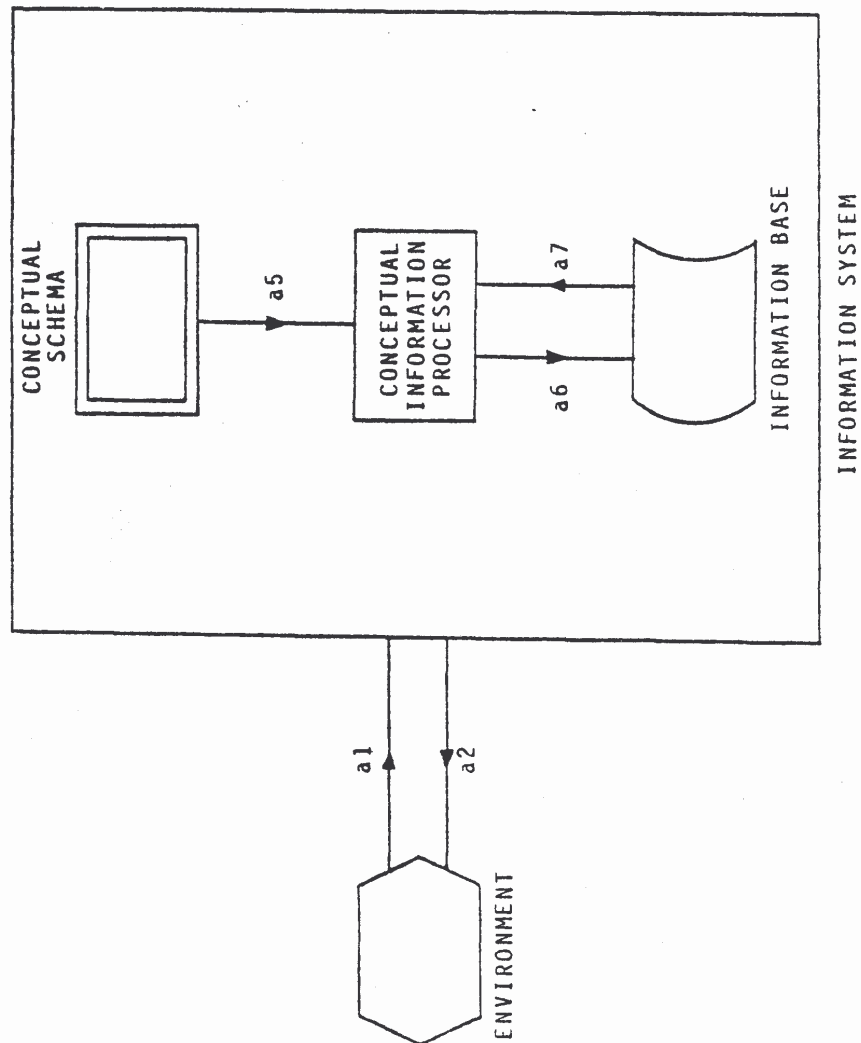


Figure 2-21

are passive components. They can neither read nor write.

If we now look carefully at figure 2-21 we see that we are not yet finished with the next level of decomposition of the rectangle called Information System, because the arrows a1 and a2 are still connected to the black box which we are decomposing.

Where should these arrows go? The answer is obvious, to the Conceptual Information Processor. This means that the active component of the Information System, called Conceptual Information Processor, is at this stage of the decomposition, also responsible for the communication with the Environment. The result is represented in figure 2-22. Please note that the construct Information System is represented by a dotted rectangle to illustrate that we have finished the first level of decomposition of it, and that we have arrived at a new stable model, however now with more detail. Please compare figure 2-1 and 2-22.

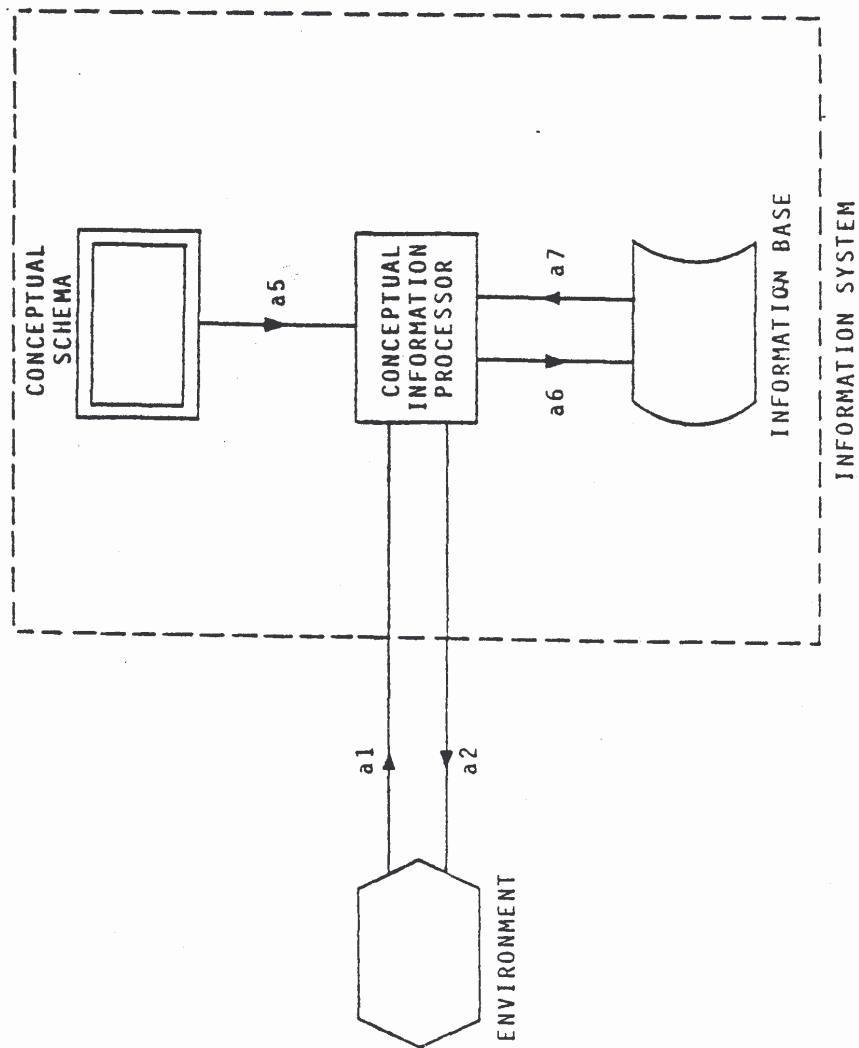


Figure 2-22

SUMMARY: ARCHITECTURE OF A CONCEPTUAL INFORMATION SYSTEM

The architecture as represented in figure 2-22 has been obtained by taking as starting point the simple, easy to understand and to accept model of figure 2-1. We then have introduced the ENALIM principle. Thereafter we started the process of decomposition of the rectangle called Information System. We applied decomposition to this rectangle to get at a next level of detail. The first component of the next level was the Information Base. It consists of a set of deep structure, elementary sentences, and this is a consequence of the ENALIM principle. Thereafter we introduced the 100 PERCENT principle, with the Conceptual Schema as the law prescribing all the Information Base states and Information Base transitions. The active component Conceptual Information Processor was introduced to guarantee that the contents of the Information Base was always in accordance with the Conceptual Schema and to perform the communication with the Environment. At this point, the decomposition has reached a new and stable level (see figure 2-22).

The architecture of figure 2-22 can be used to illustrate both to experts and to laymen, how the major conceptual components of an Information System work; or said otherwise, it can be used to illustrate the dynamics of an Information System.

Which component is made by whom in the model of the Conceptual Information System?

The two boxes that need to be made during the construction of a Conceptual Information System are the Conceptual Schema and the Conceptual Information Processor.

The user, possibly with the help of a professional information analyst, has to specify (design, define, describe, write down, etc.) the Conceptual Schema.

The software manufacturer will provide the Conceptual Information Processor.

Very often, a user performs the function of Environment. In that function he requests the Conceptual Information Processor to add, delete or retrieve sentences from the Information Base.

During the operation of a Conceptual Information System, the user (but sometimes another Conceptual Information System) requests the services from the Conceptual Information Processor, and as the result of these requests determines the contents of the Information Base.

In other words, the most important component for the user is the Conceptual Schema.

3. ARCHITECTURE OF AN INFORMATION BASE MANAGEMENT SYSTEM

In chapter 2 we have said explicitly that we were only interested in conceptual aspects (or WHAT aspects) and that we would disregard other aspects in chapter 2.

In chapter 3 we will enlarge our view and will consider the aspects of first computer effectiveness and then programming or user effectiveness. In other words, we will take the architecture as represented in figure 2-22 as starting point, and apply the process of de-abstraction, or include new aspects so far abstracted from.

COMPUTER EFFECTIVENESS

The first new aspect which we will include in our build-up of the architecture is computer effectiveness. In chapter 2 we had an information base but we were not interested how its contents (deep structure, elementary sentences) was represented on storage media. Now we will enlarge our view by introducing the INTERNAL principle.

INTERNAL PRINCIPLE: THERE IS ONE GRAMMAR WHICH COMPLETELY AND EXCLUSIVELY PRESCRIBES (PREDEFINES) THE PHYSICAL REPRESENTATION OF EACH PERMITTED INFORMATION BASE STATE. THIS GRAMMAR IS CALLED INTERNAL GRAMMAR OR INTERNAL SCHEMA. THE PHYSICAL REPRESENTATION OF AN INFORMATION BASE IS CALLED INTERNAL DATA BASE.

The Internal Schema can be added to the architecture of figure 22-2, and we get figure 3-1.

The Conceptual Schema is a grammar which prescribes information base states with deep structure, elementary sentences. The Internal Schema is a grammar which describes the internal data base states with records, access paths and other pointer structures.

In other words, the traffic into and from the internal database is now in terms of records, access paths and pointer structures. Just like we had to do with the enforcement of the conceptual schema where we introduced the conceptual information processor, we will now

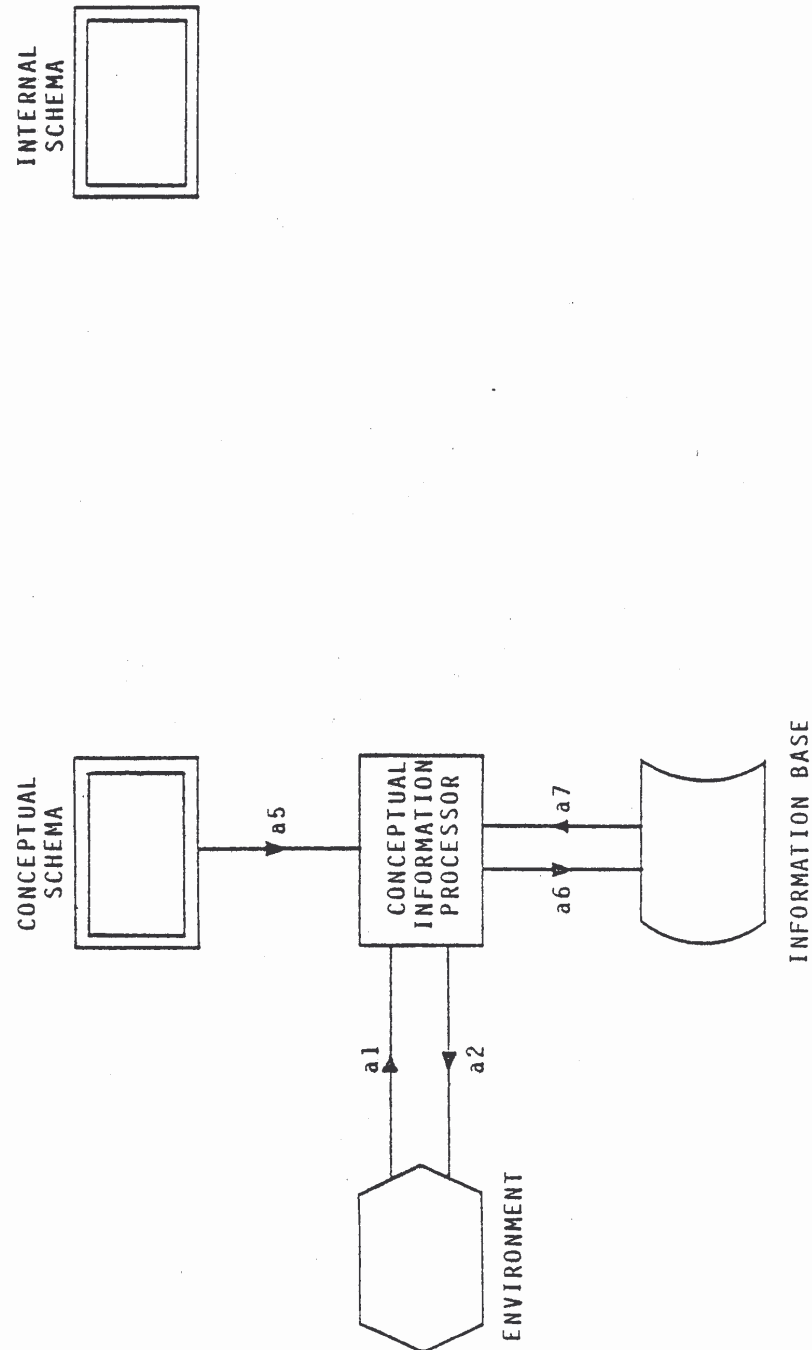


Figure 3-1.

introduce the internal information processor, (see figure 3-2). Please note that the traffic to and from the internal database is via arrows a10 and a11, which are different from a6 and a7 because we now consider a flow of records, access paths and other physical structures (although their deep structure still follows the ENALIM principle).

The Environment or user, who communicates via the arrows a1 and a2 with the Conceptual Information Processor, still thinks that the knowledge container is an Information Base; such a user does not think in terms of the Internal Data Base. Because the Information Base is a virtual view, we represent it with interrupted symbols.

But, if the conceptual schema prescribes information bases in terms of deep structure, elementary sentences, and the internal schema prescribes internal databases in terms of records, access paths and other physical structures, it is obvious that we need a description which tells us how deep structure, elementary sentences are to be translated into records, access paths and physical structures and vice versa. Said otherwise, we need a schema or grammar which prescribes how each permitted information base in deep structure sentences is to be encoded in an internal database with records, access paths and physical structures. For convenience reasons, we assume that this schema, which could be called Conceptual Internal Transformation Schema is included as a separate section in the Internal Schema.

And now we have to introduce a process of the same kind as we did to enforce the conceptual schema, namely the conceptual information processor; and also for the internal schema, namely the internal information processor. We now introduce the Conceptual Internal Transformation Processor. Its function is to transform deep structure sentences into records, access paths and other physical structures and vice versa and to do this according to the law specified in the Conceptual Internal Transformation Grammar or Schema. For convenience reasons, we assume that this transformation processor is a component of the Internal Information Processor.

As we see in figure 3-2, the Internal Information Processor can read the Internal Schema, but with which other things does it communicate?

The Conceptual Information Processor speaks in deep structure sentences, hence the arrows between the Conceptual Information Processor and the Internal Information Processor are essentially a6 and a7 (see figure 3-3).

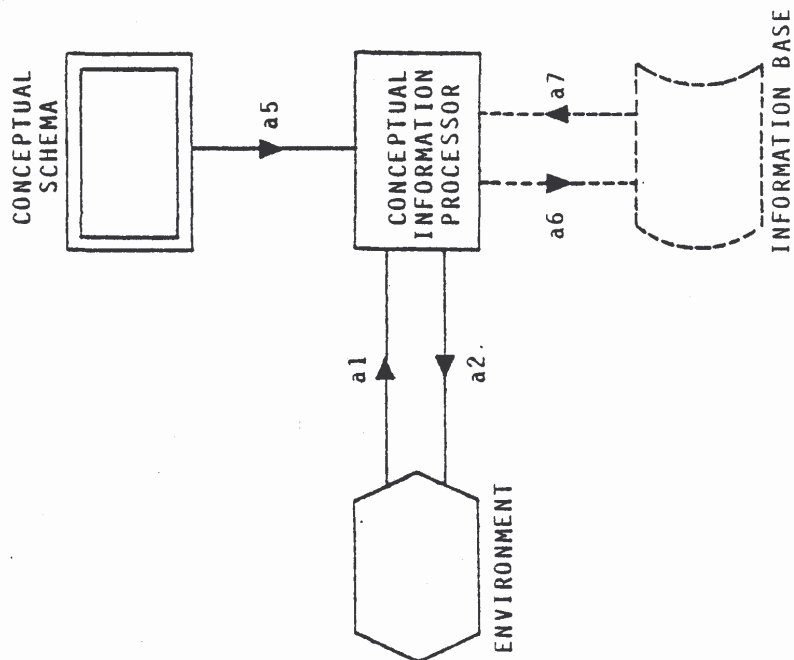
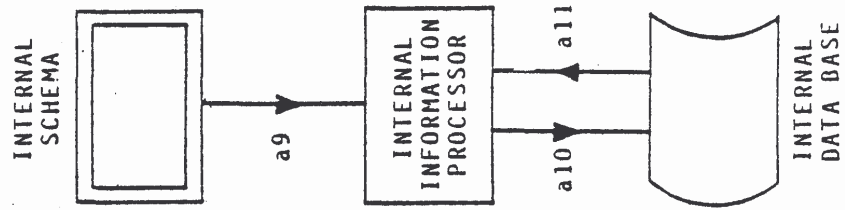


Figure 3-2

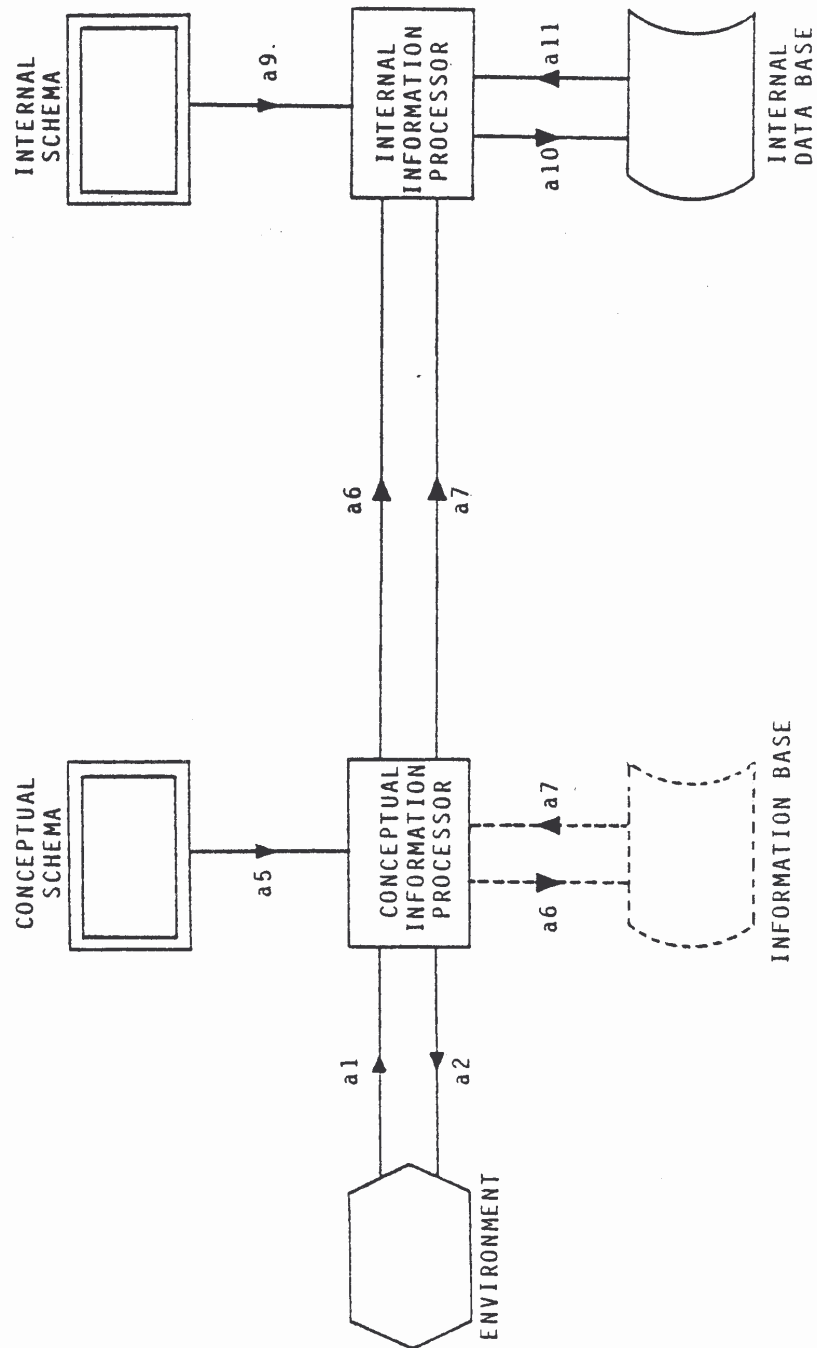


Figure 3-3

SUMMARY: INTERNAL PRINCIPLE

As an interim recapitulation we may say that we started with the model of figure 2-22. We then took the aspect of computer effectiveness into account (= de-abstraction) by introducing the INTERNAL principle and via analogous reasoning as we did for the 100 PERCENT principle (introducing the Conceptual Schema) we arrived at figure 3-3, which is again a stable state of the architectural framework. From figure 2-1 we arrived at figure 22-2 by applying decomposition, from figure 22-2 we arrived at figure 3-3 by applying de-abstraction.

PROGRAMMING AND USER INTERFACE EFFECTIVENESS

So far in chapter 3 we have included in the architecture the components associated with computer effectiveness. And the resulting architecture is presented in figure 3-3. If we analyze the architecture of figure 3-3, we repeat (what we have seen before) that the traffic along interface a6 and a7 is in terms of deep structure, elementary sentences. In chapter 2, where we only considered the conceptual aspects, i.e. deep structure, elementary sentences, the interfaces a1 and a2 were introduced. Hence the traffic along interface a1 and a2 is in terms of deep structure, elementary sentences. But not all users are willing to speak in deep structure, elementary sentences, nor do programming languages exist (except RIDL, reference 14) which can deal with deep structure aspects.

It is therefore understandable that some users want to use deep structure, elementary sentences, while others want to use (often for good reasons) other constructs such as forms, or Normalized Relations, or special trees or graphical constructs, etc. This means that such users perceive the information base as a set of forms, Normalized Relations, CODASYL records and sets, special trees or graphical constructs. In order to cover this new aspect (= de-abstraction) we introduce the EXTERNAL PRINCIPLE.

EXTERNAL PRINCIPLE: AN EXTERNAL GRAMMAR OR SCHEMA CAN PRESCRIBE A VIEW OF AN INFORMATION BASE WHICH MAY BE DIFFERENT FROM THE CONCEPTUAL VIEW (TRANSFIGURATION). THE KNOWLEDGE BASE ASSOCIATED WITH THIS VIEW IS CALLED EXTERNAL DATA BASE.

If we add the External Grammar or Schema to figure 3-3, we get figure 3-4. In analogy with the consequences of the INTERNAL PRINCIPLE, we have to introduce an EXTERNAL INFORMATION PROCESSOR. Adding this to figure 3-4, we get figure 3-5.

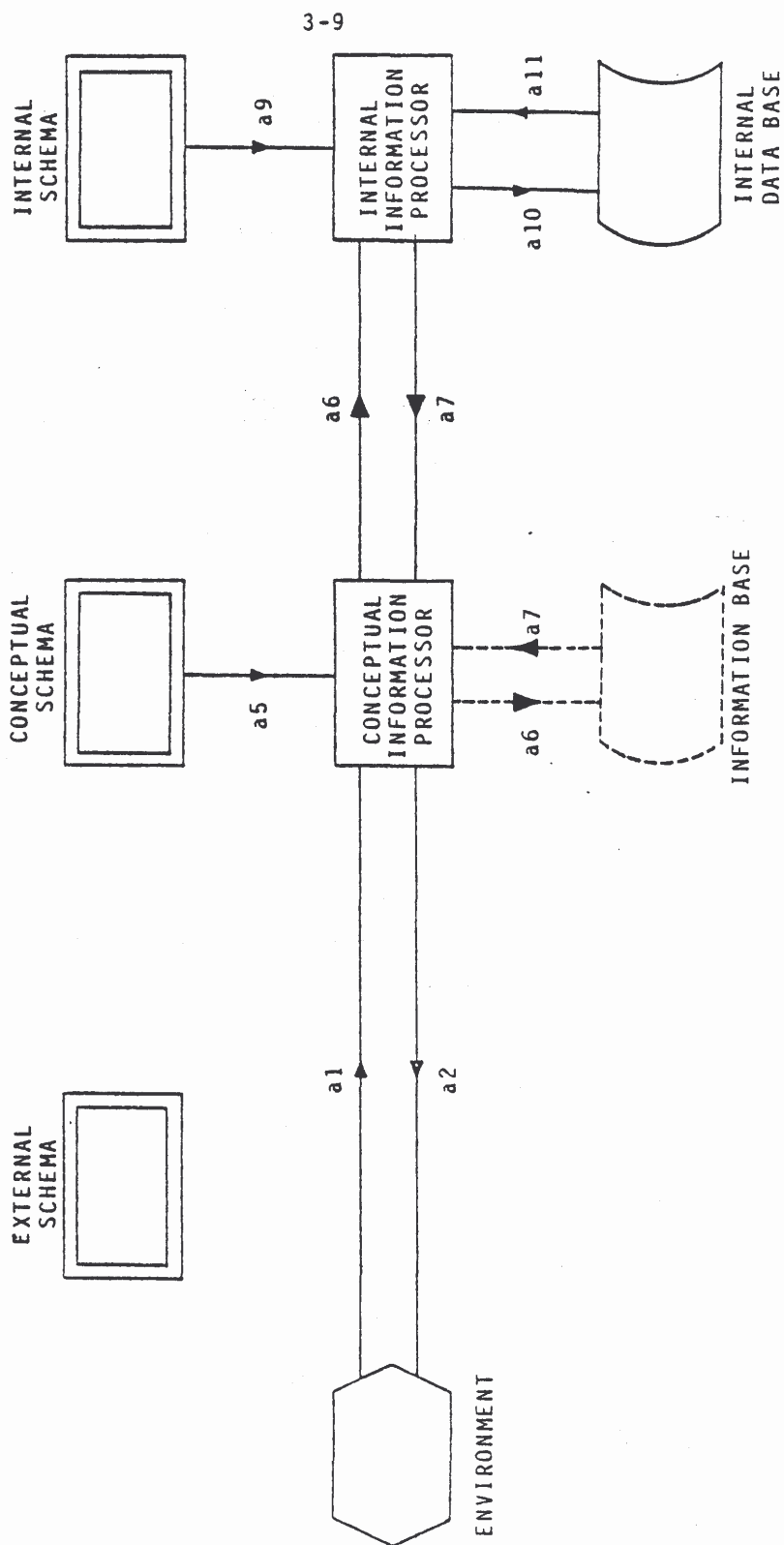


Figure 3-4

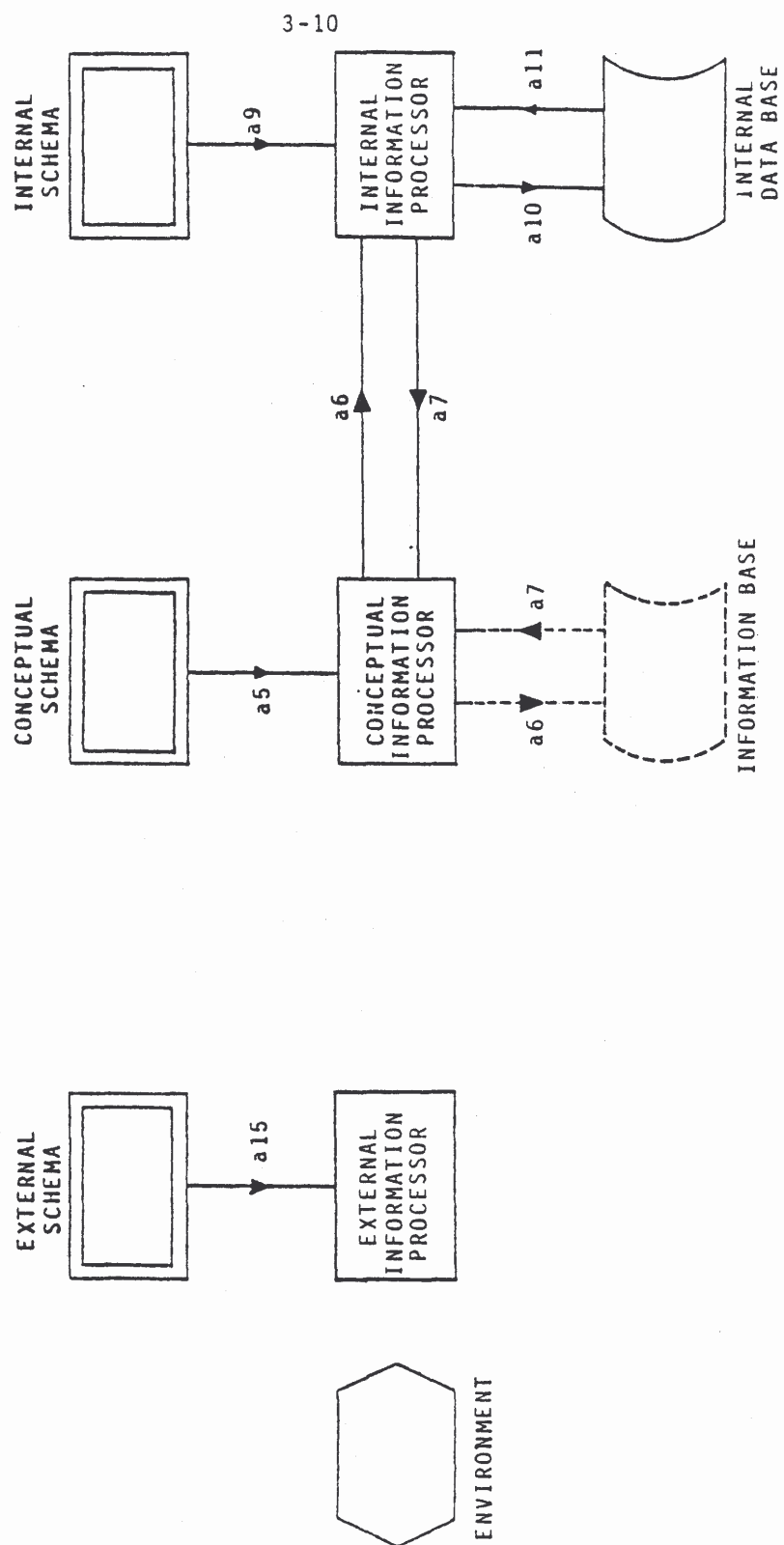


Figure 3-5

The External Schema prescribes a view of database in forms, Normalized Relations, trees, graphical structures, etc. The Conceptual Grammar deals with deep structure elementary sentences. Hence we need a grammar or schema which specifies how the external concepts, such as forms, Normalized Relations, trees, graphical structures, etc. have to be transformed into deep structure, elementary sentences and vice versa. This grammar is called CONCEPTUAL EXTERNAL TRANSFORMATION SCHEMA; for convenience reasons we assume that this transformation schema is included as a separate section in the External Schema.

Another function of this Conceptual External Transformation Schema is to hide a certain amount of the contents of the (common) database from an external user.

In analogous reasoning as used for the consequences of the INTERNAL PRINCIPLE, we have to introduce a Conceptual External Transformation Processor. Its function is to transform external constructs such as forms, Normalized Relations, trees, graphical constructs, etc. into deep structure, elementary sentences and vice versa, and to do this according to the law specified in the Conceptual External Transformation Grammar. For convenience reasons, we assume that this transformation processor is a component of the External Information Processor.

Figure 3-6 contains a stable decomposition, containing all the necessary interfaces.

The Environment or user, who communicates via the arrows a16 and a17 using external constructs, thinks that the knowledge container he deals with, consists of a set of external constructs such as records, or normalized relations or others. This knowledge container has been given the name External Data Base. Because of the fact that it is virtual, it is represented with interrupted symbols.

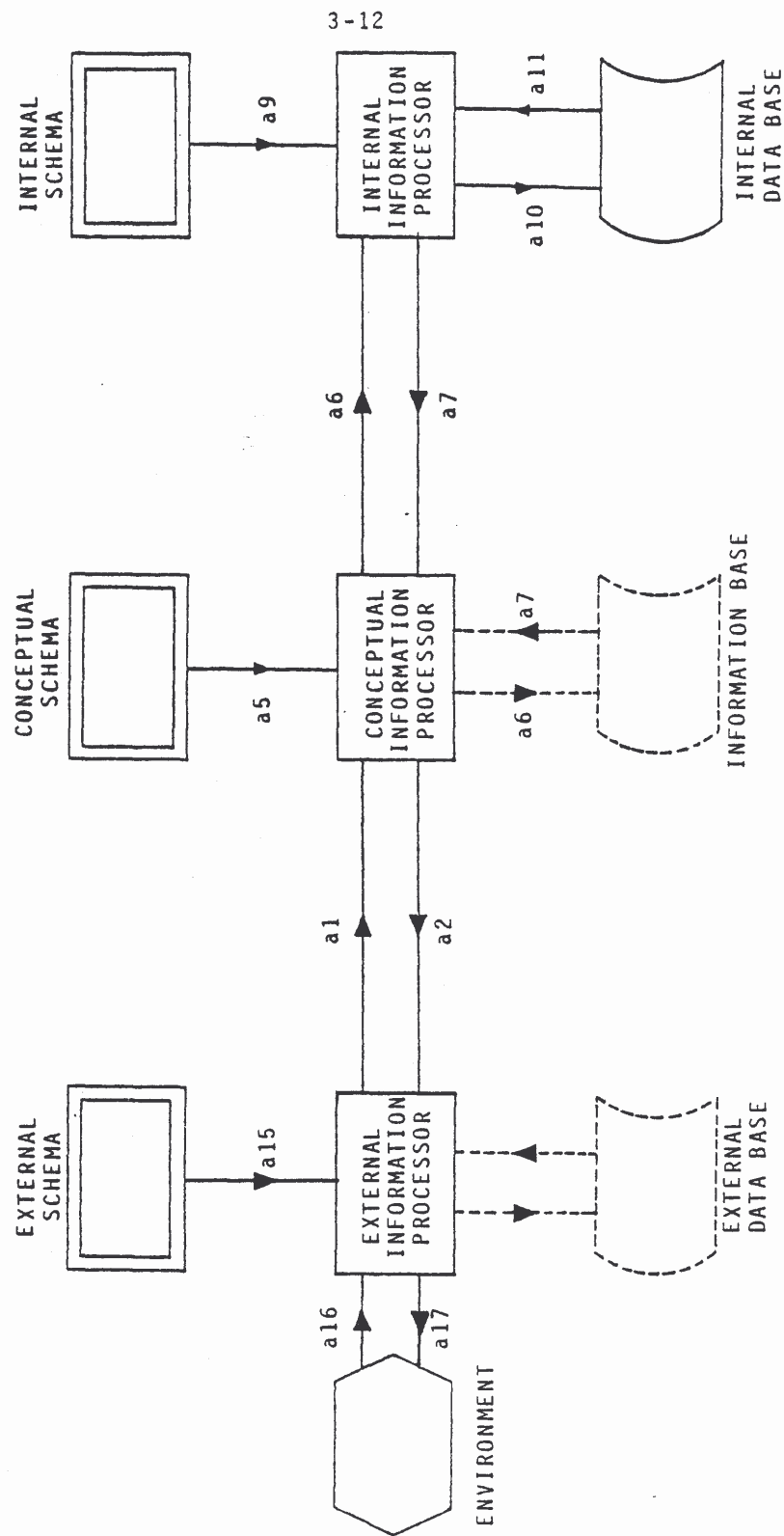


Figure 3-6

In order to be complete, we need to add one more processor, which deals with various aspects such as collection of messages and assemble these into a transaction to be transferred to the External Information Processor, or the editing of information to be presented to the user on a screen or other output medium. This processor is given the name APPLICATION PROGRAM. Figure 3-7 contains the APPLICATION program.

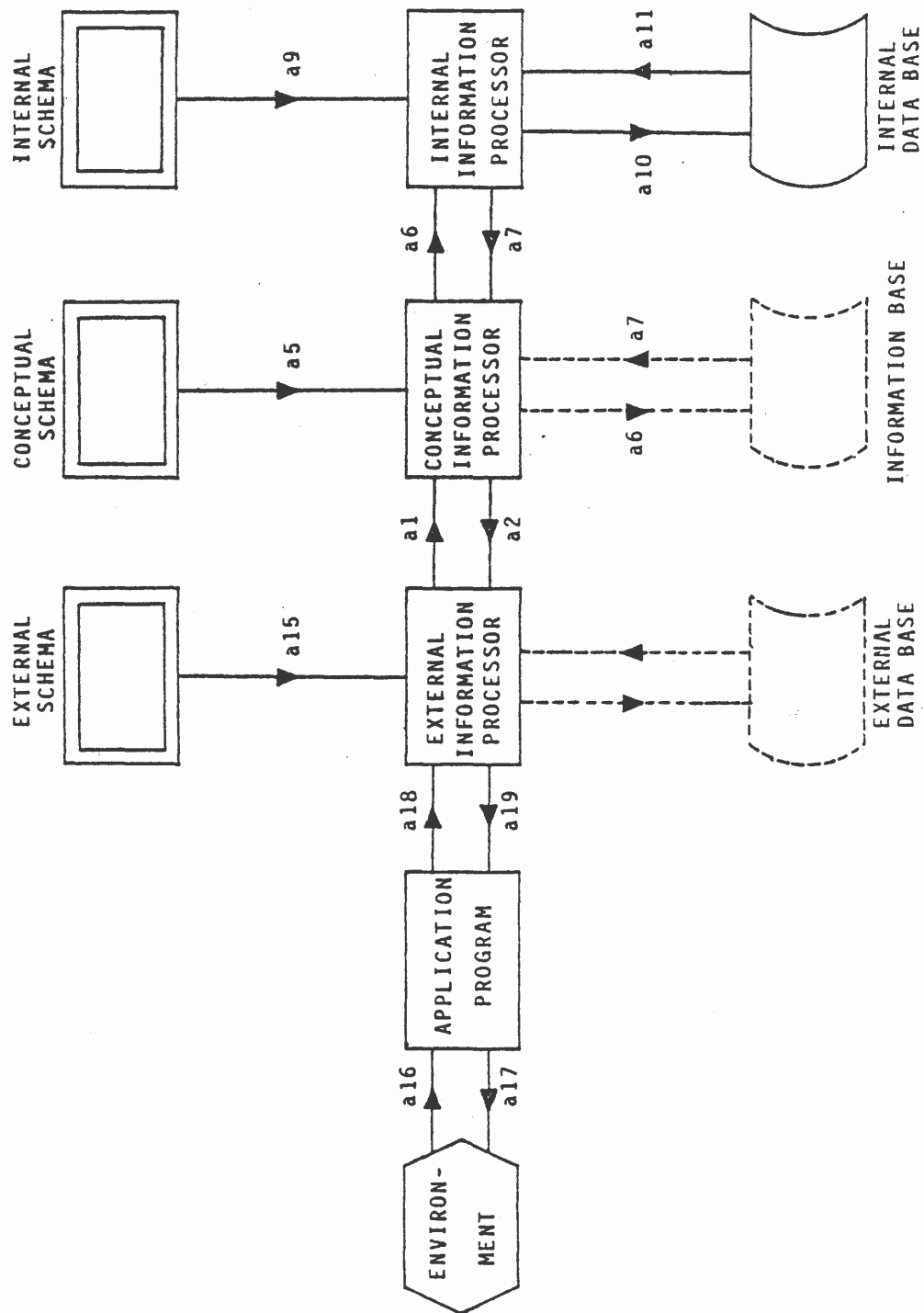


Figure 3-7

SUMMARY: ARCHITECTURE INFORMATION BASE MANAGEMENT SYSTEM

As an interim recap, we repeat that we started with the model of figure 2-1. We then applied decomposition to the rectangle called Information System and the result was the architecture of figure 2-22. We then applied de-abstraction to figure 2-22, by taking computer effectiveness into account, and the result was the architecture of figure 3-3. Then we again applied de-abstraction to figure 3-3 by taking programmer and user interface effectiveness into account, and the result is figure 3-6. Adding explicitly the processor concerned with message assembling and information presentation, the APPLICATION PROGRAM, we get figure 3-7.

Very often the architecture of figure 3-7 is referred to as the three schema architecture.

(In the past, I have used the name Coexistence Architecture to refer to the architecture of figure 3-7).

In order to facilitate reasoning, and to see the analogy with database software, it is useful to combine all the processors in figure 3-7, except the application program. This is applying the process of composition (see figure 3-8). The processor which combines the three processors is often called the DBMS. The end result of this composition is represented in figure 3-9.

What happens if we have two different user views on the same information base? The result is represented in figure 3-10. It is important to stress that only the application programs and the external schema are different. Here we see one of the real advantages of database software. The Conceptual Schema, the Internal Schema, are only to be made once, and serve for many environments or users.

Similarly the Internal Schema may have to be remade if the hardware changes, but in this case neither the conceptual schema nor the external schemas, nor applications programs will change.

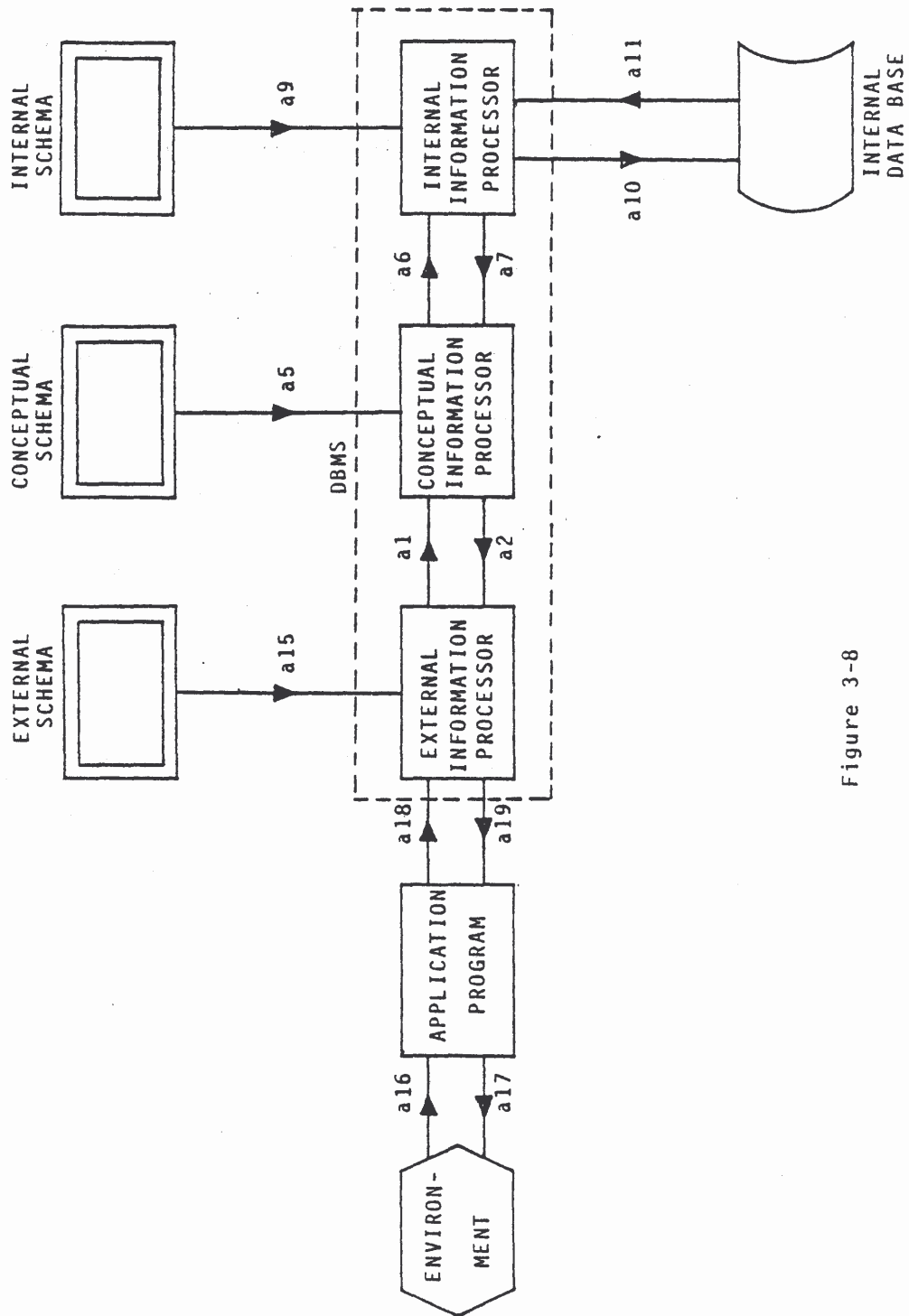


Figure 3-8

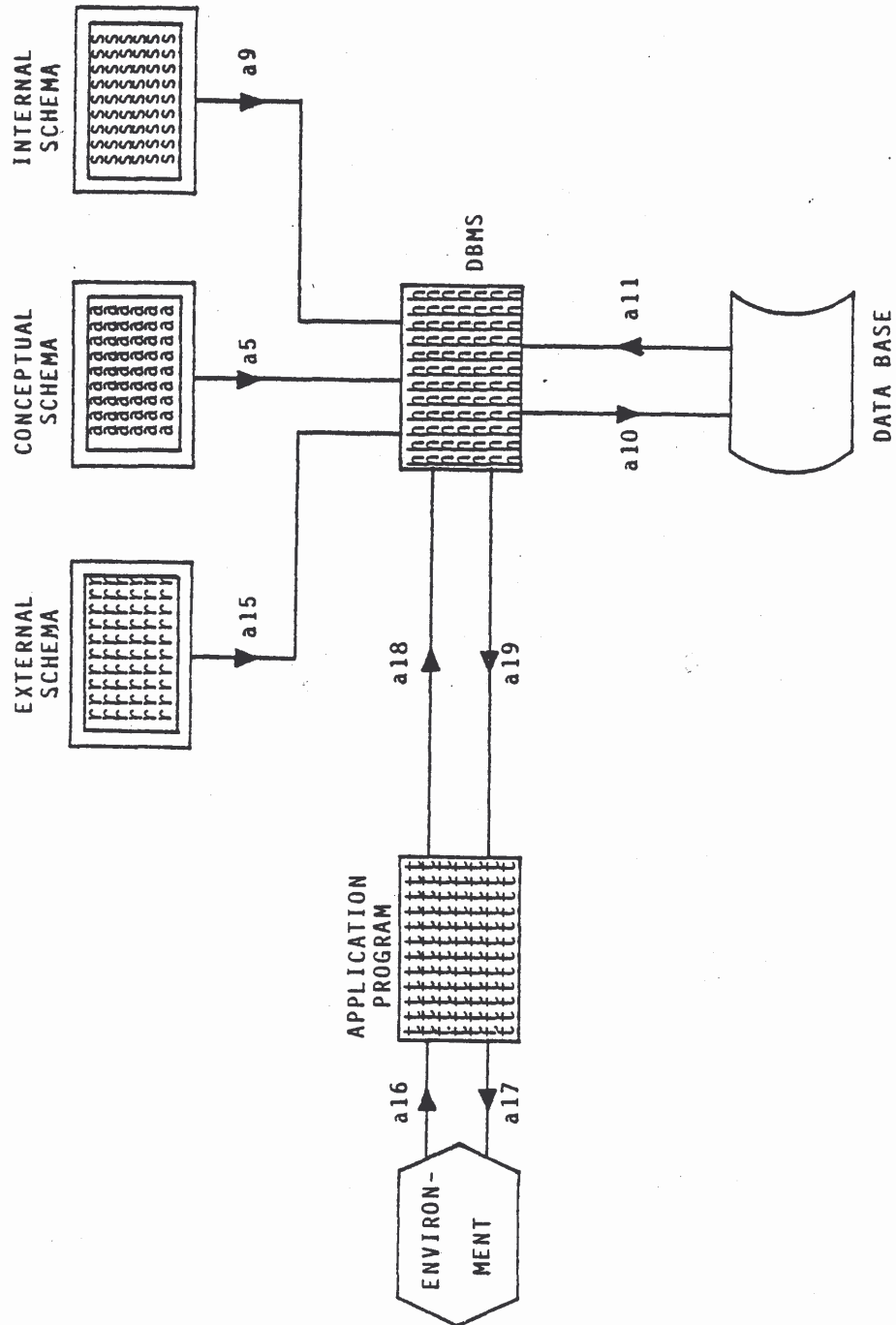


Figure 3-9

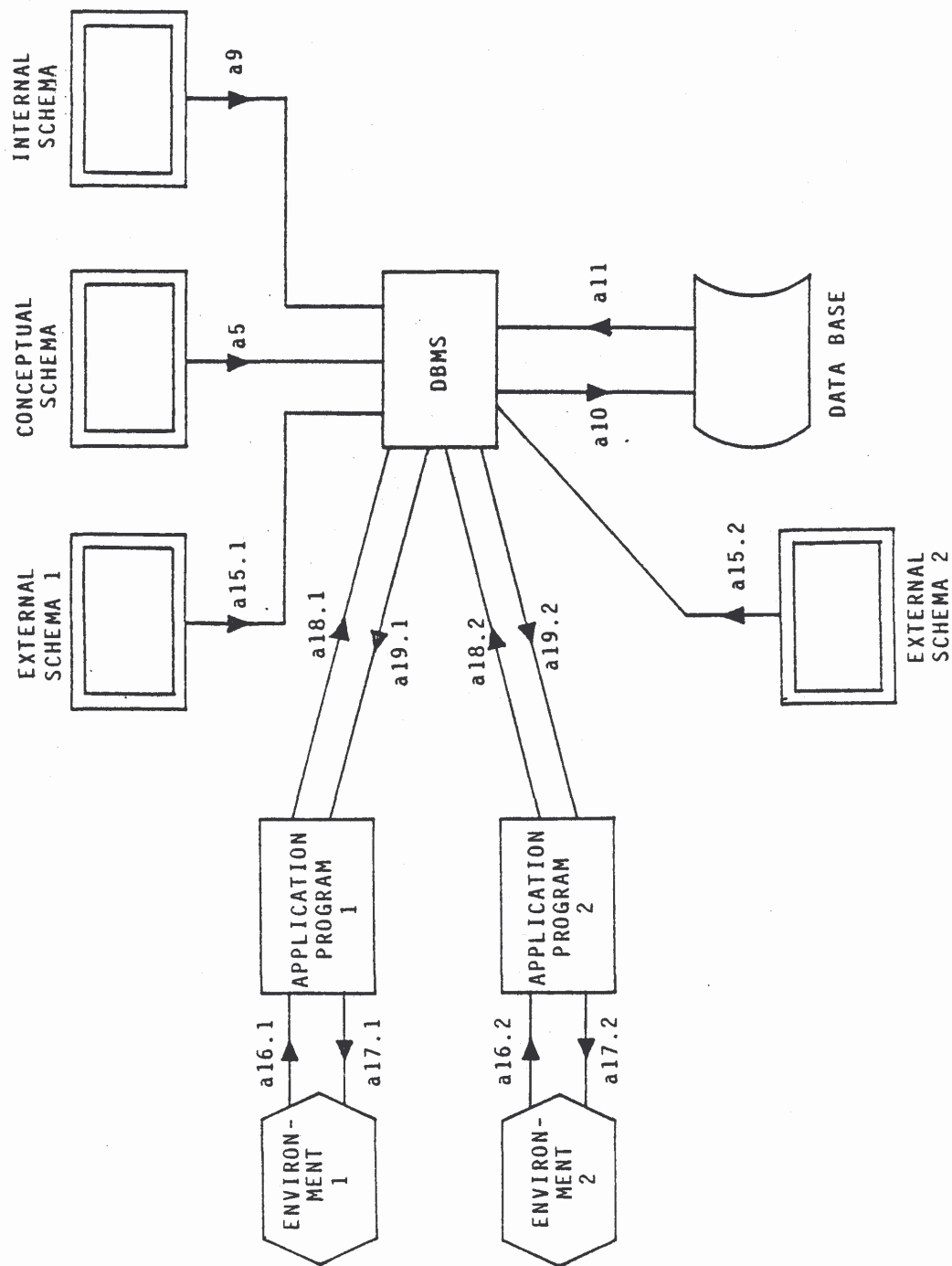


Figure 3-10

THREE SCHEMA ARCHITECTURE OF FIGURE 3-9 AND CURRENT DBMS

The essential characteristics of the architecture as represented in figure 3-9 are that the Conceptual Schema contains all the rules that define the permitted contents of the Information Base and no other rules, the Internal Schema contains all the rules which prescribe the physical aspects of the Information Base and no others, and the External Schema contains only rules which have to do with the external view (and certainly no internal aspects); furthermore, the dbms is solely responsible for guaranteeing the rules of the conceptual, internal and external schema and the transformations; and the application program is only dealing with application aspects, and certainly no internal aspects, or conceptual schema rules.

This can be represented in a figure by adding symbols to figure 3-9 and the result is 3-11.

The architecture of current dbms is different in several aspects. Let us discuss current dbms which have three schemata (in case of two, the result is even worse).

The first major difference is that not all rules of the conceptual grammar are described in the conceptual schema. In most cases, only record types, or relation types, and a few rules are described in the conceptual schema. The majority of the conceptual grammar rules are described in the application program. They are often called validation rules. Because these validation rules are not part of the Conceptual Schema, the DBMS cannot enforce these rules. In the case that there is more than one application update program, there is redundancy in description.

In the case where one of the application programs is directly made by a user with an end-user language, one cannot realistically expect that these casual users will program all the validation rules. Also some EDP "experts" take sometimes shortcuts and bypass validation rules.

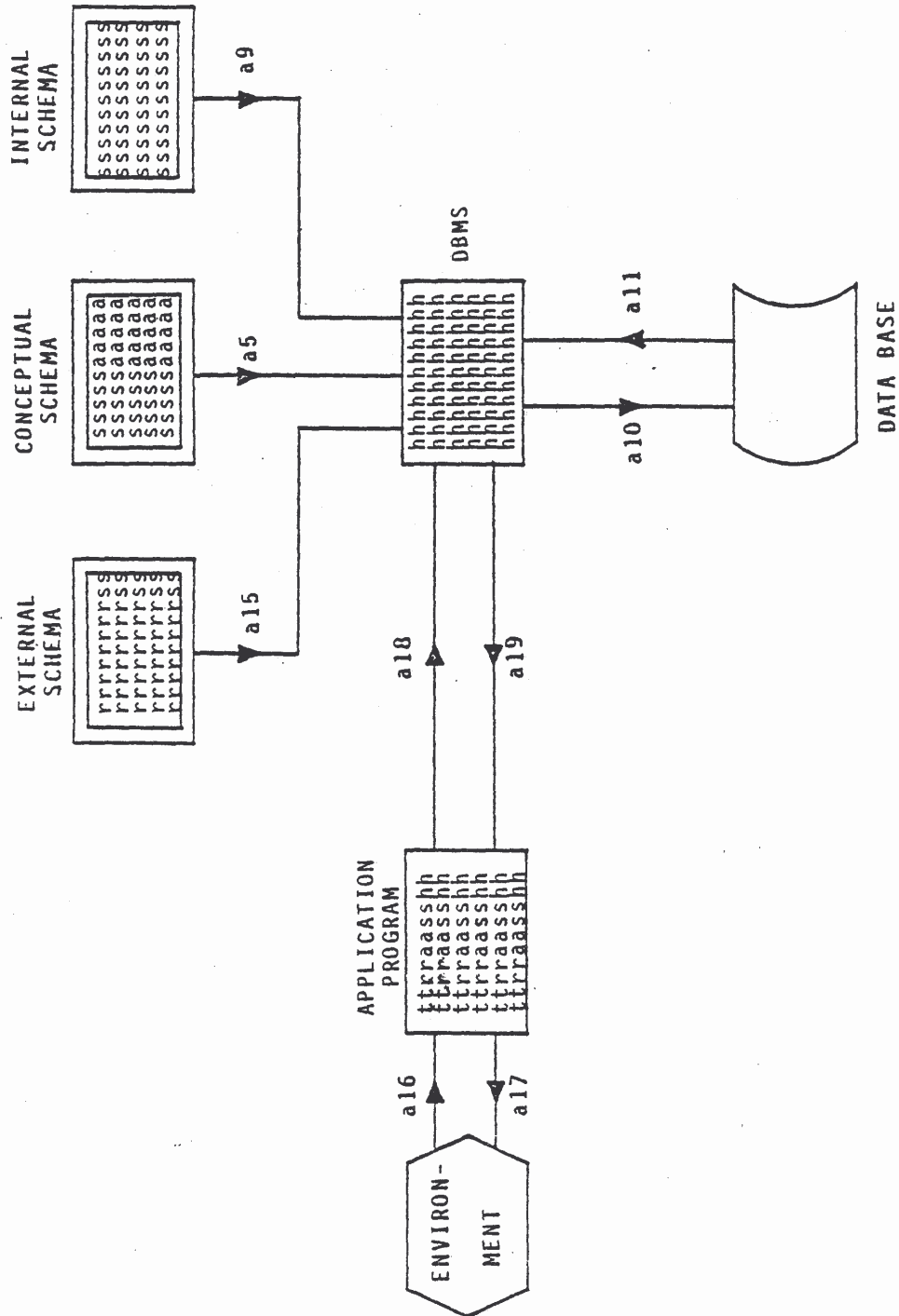


Figure 3-11

What is the result?

POLLUTION IN THE INFORMATION BASE.

The negative aspects of this kind of pollution need not be elaborated upon, I hope.

The second major difference is that with current DBMS the external schema contains some internal constructs. This means that in the code of the application program there are references to internal constructs. If for reasons of increasing machine efficiency (e.g. because of an increase in the number of transactions, or a change in the usage frequency of certain kinds of questions) the internal schema is modified, the consequence with current DBMS is fairly often reprogramming part of the application program. This reprogramming is a substantial part of the so-called maintenance of running applications.

4. The META principle

So far, we have built up an architecture which covers the WHAT (= conceptual) and the HOW (= external; = internal) aspects.

In this section we try to bring more homogeneity in the architecture of figure 3-9. We will use an example as an introduction.

Let us look at figure 4-1. There we deal with an architecture which can be derived from figure 3-9, by eliminating the internal and external schemas.

Let us now assume that the database-1 contains such sentences as represented at page 2-7. Application program 1 communicates with Environment 1, by accepting input consisting of such sentences or providing answers in terms of such sentences.

Let us now look at figure 4-2. There we deal with an application which is used during information systems design to store conceptual schemas. Let us give the associated application program 2 the name ISDIS (Information Systems Design and Implementation System). Let us suppose that we are using ISDIS to store a conceptual schema, specifying that we are interested to know which employees identified by their employee number are working for which department identified by their department number, with the additional restriction that each employee may work for at most one department.

Because the architecture of figure 4-1 is applicable to all kinds of different application areas, it is applicable to an ISDIS application.

Let us now look at figure 4-3. If we carefully look at this figure, we have to conclude that the conceptual schema-1 is the same as the database-2.

Why is this the case? Database-1 contains sentences such as "The employee with employee number E1 works for the department with

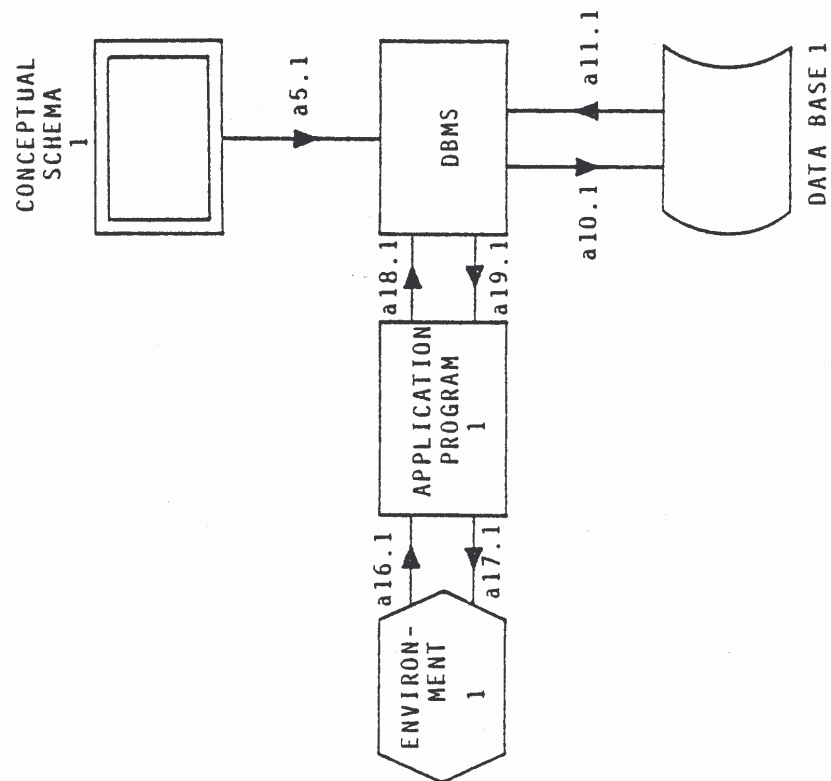


Figure 4-1

department number D1" etc. etc. Conceptual schema 1 is the law or the contract for database-1.

Therefore conceptual schema 1 consists of rules (just like a law or contract) specifying that the contents of database-1 may consist of sentences which follow the following pattern: the employee with employee number ... works for the department with department number ... with the additional restriction that each employee may at most work for one department.

Database-2 is an information base filled with the help of the application program called ISDIS. Database-2 contains the formal result of information analysis. This means that database-2 consists of rules (just like a law or contract) specifying that the contents of the database corresponding to this application may consist of sentences which follow the following pattern: the employee with employee number ... works for the department with department number ... with the additional restriction that each employee may at most work for one department.

But this is precisely the same as the contents of conceptual schema 1 as we have seen in the paragraph preceding the last.

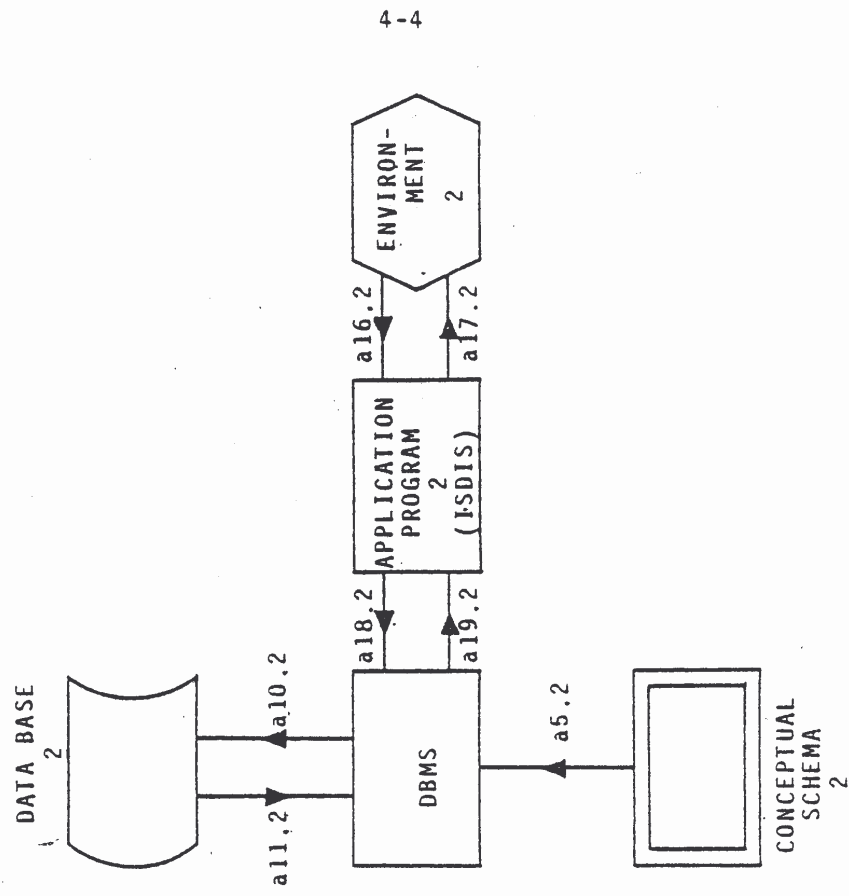


Figure 4-2

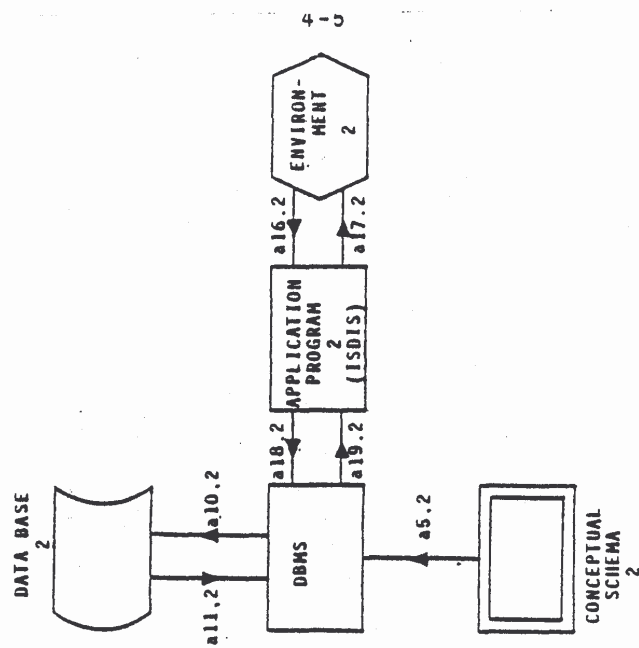
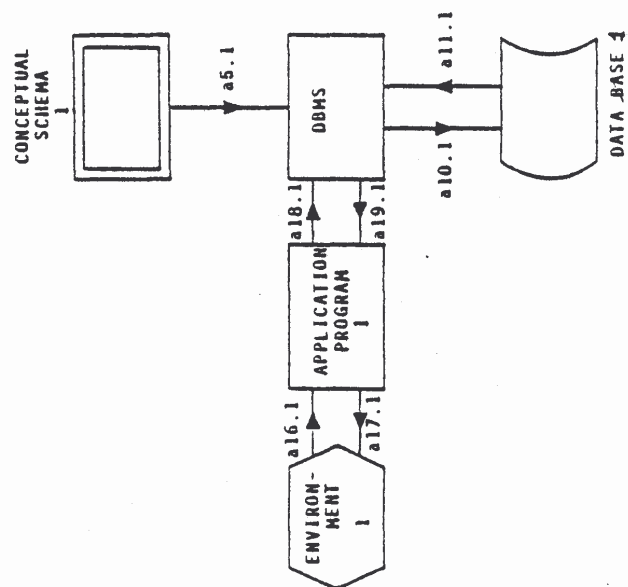
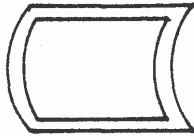


Figure 4-3

We will now introduce a new symbol to reflect the fact that one and the same thing can be an information base or database and schema.



This symbol is an information base or database, which is also a schema.

With the help of this new concept and associated symbol, we are able to present an architecture of the integrated system as is done in figure 4-4. This is a truly integrated architecture and is the first result of the META principle.

META PRINCIPLE: ALL SCHEMAS OF THE COEXISTENCE ARCHITECTURE (see figure 3-9) ARE AT THE SAME TIME TO BE CONSIDERED AS INFORMATION BASES.

If we apply this principle to all schemas of figure 3-9, we get figure 4-5.

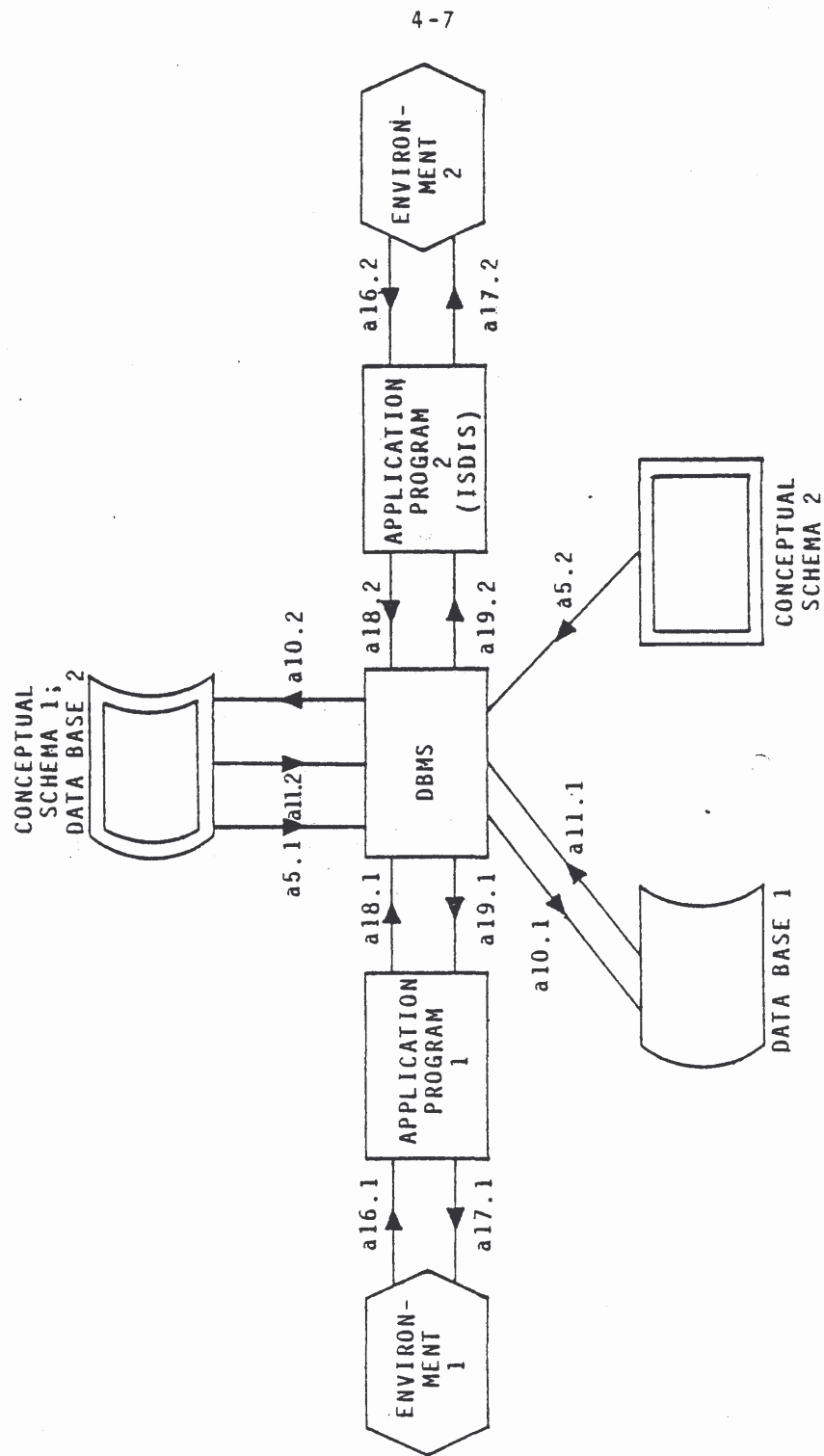


Figure 4-4

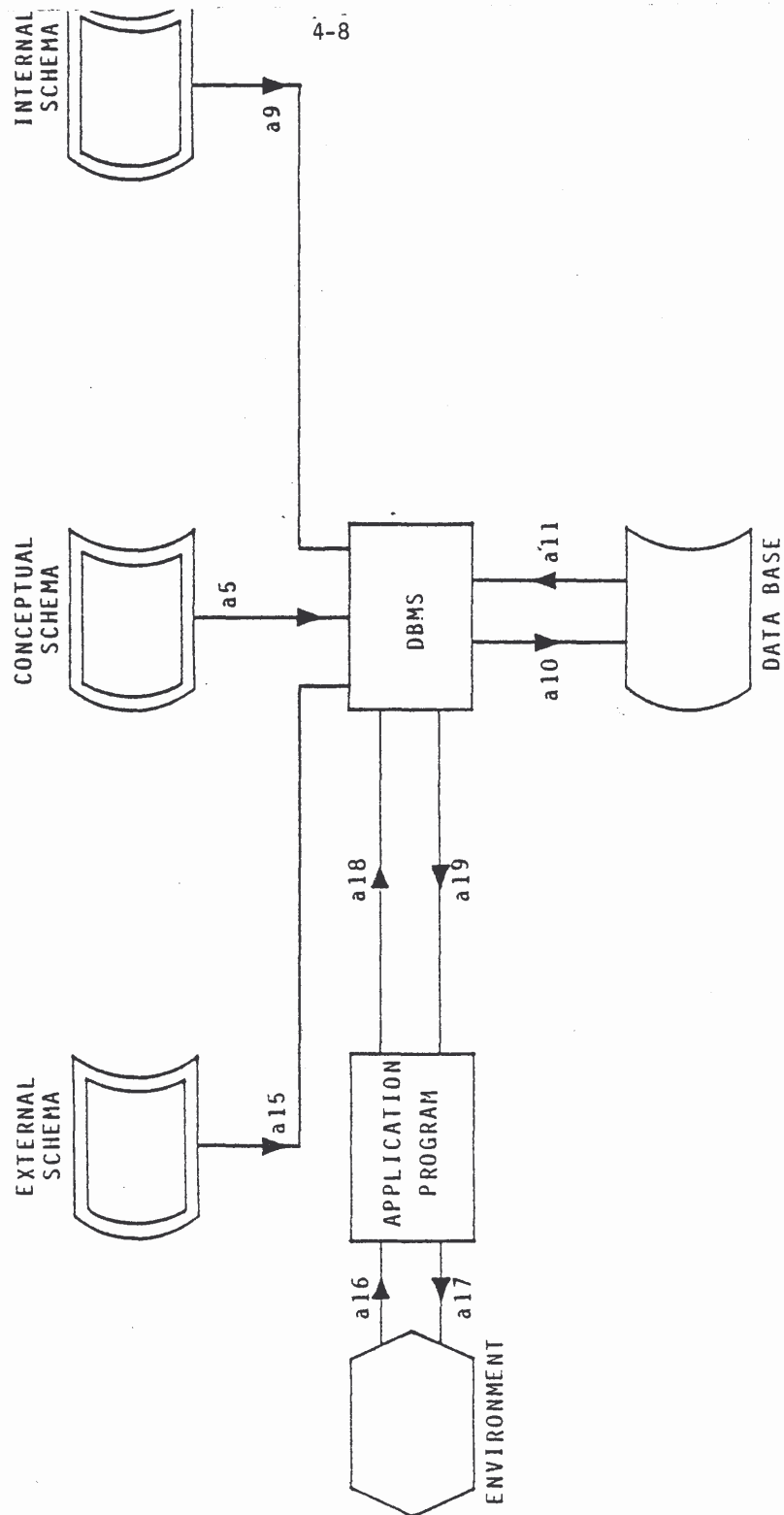


Figure 4-5

SUMMARY: META PRINCIPLE

As a recapitulation we may say that we started with the simple model of figure 2-1 in chapter 2.

First we introduced the ENALIM and 100 PERCENT principles, applied decomposition to the black box of figure 2-1, called Information System and the result was the Architecture of a Conceptual Information System of figure 2-22, where we have an Information Base, a Conceptual Schema, a Conceptual Information Processor, the Environment and several communication interfaces.

Thereafter we applied de-abstraction to the Conceptual Information System by introducing the INTERNAL and EXTERNAL axiom. The result was the three schema architecture. We then applied de-abstraction when introducing the application program. This architecture is general enough to describe the various current dbms as special parameterized implementations of this architecture.

Thereafter, we introduced the META principle which results in an architecture in which the software specification (Data Dictionary or DD) and execution of data base programs (Data Base Management System or DBMS) are completely integrated. This integrated architecture shows us the kind of software we may expect during the coming years in this area. It also can be fruitfully used as a general model to explain current DBMS and DD and their associations.

ARCHITECTURE OF INTEGRATED DBMS-DD AS IN FIGURE 4-4 AND CURRENT DBMS AND DD.

There are currently in use software packages which are called Data Dictionaries (DD for short). Current Data Dictionaries are used as a documentation tool in which record types, sometimes entity and relationship types, a few (if any) validation rules, some characteristics of programs and of operational aspects are described. The righthand part of figure 4-3 is an information system which will be a part of future data dictionaries. The righthand part of figure 4-3, however, describes the complete conceptual schema, whereas current DDs describe only a small part of it, and most often not in conceptual terms, but a mixture of external-internal terms.

The lefthand part of figure 4-3 is today often called DBMS.

Very characteristic for current DBMS and DD is the fact that the DBMS and DD are not integrated (as in figure 4-3). In contrast to this is the architecture we have described in figure 4-4.

One of the major arguments of persons who try to convince users of different departments to use a common database is the integration of the information with all the well known advantages. This apparently does not yet apply the two related "departments" DBMS and DD.

5. BUILD-UP OF THE OVERALL ARCHITECTURAL FRAMEWORK FOR DATABASE MANAGEMENT SOFTWARE

In this chapter we will apply the results of the previous chapters, until we have an overall architectural framework for database management software.

Let us start with figure 4-5. In this figure we have represented the results of all five previous principles. The model of figure 4-5 is a general model which presents clearly that one has separated the WHAT aspects (the conceptual schema) from the two HOW aspects (internal and external schema).

For many practical purposes it is more useful to use an abbreviated form of figure 4-5, one in which there is only one schema, namely the conceptual schema. The result is represented in figure 5-1. In figure 5-1 there are two descriptions which must be programmed by the user's organization, namely the Conceptual Schema and the Application Program. In nearly all cases the DBMS is built by the hardware vendor (or software supplier) whereas the Database is being filled and modified as a result of interaction between the Environment and the Application Program and the rest of the system.

For the build up of the overall architectural framework for database management software, we will repeatedly use the model of figure 5-1 to add aspects to figure 4-5.

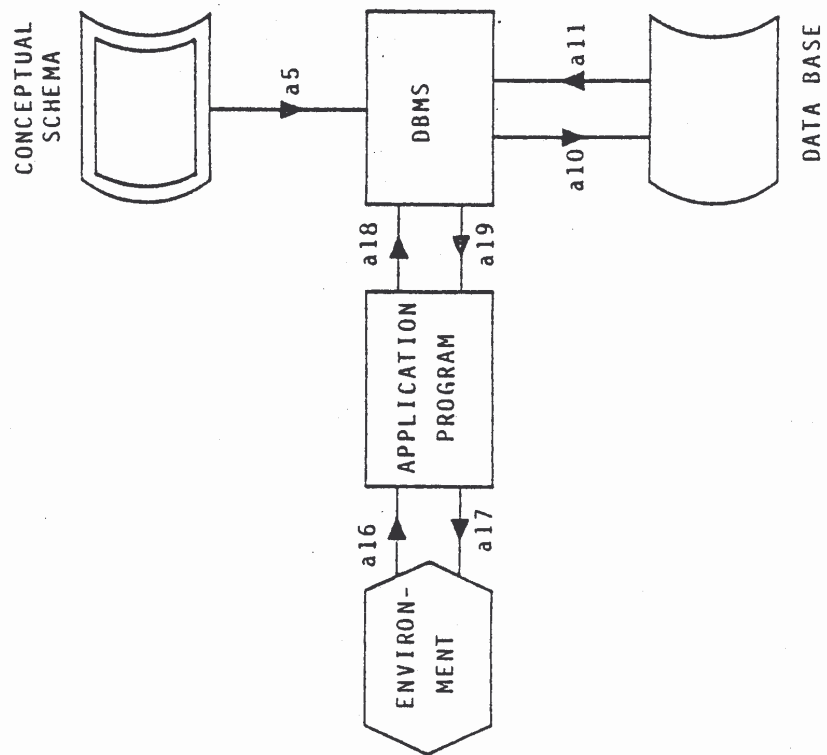


Figure 5-1

In figure 5-1 we have a Database which is filled, modified and interrogated by the Environment through interaction with an Application Program. In figure 4-5, there are 3 schemas which are of the nature of a database, and hence they are basically subject to the same treatment as the Database of figure 5-1.

Let us start with the conceptual schema of figure 4-5. This conceptual schema is at the same time a database, hence we can apply the model of figure 5-1 to treat this database (the conceptual schema) and the result is represented in figure 5-2.

The Environment is the Information Analyst who wants to store the results of Information Analysis (the user conceptual schema) in a database. He can do this by using the application program called ISDIS-IA (IA = Information Analysis), which in turn calls upon the services of the DBMS to store the user conceptual schema in a database. The various states and transitions of such a database are completely determined by one grammar or schema, hence the DBMS consults (arrow a5.2) the conceptual schema for conceptual schemas or conceptual metaschema.

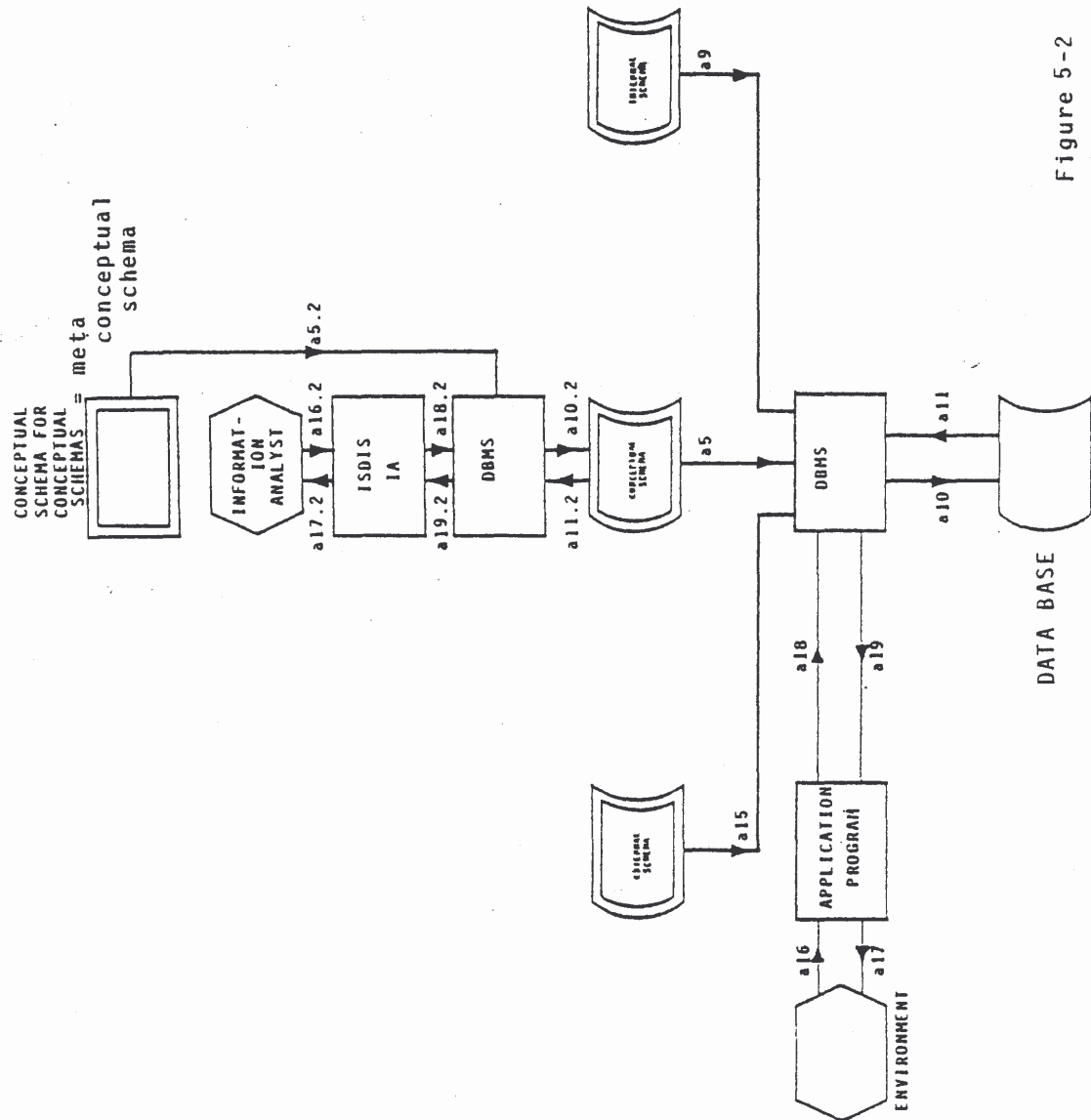


Figure 5-2

It is clear that we can apply the same kind of reasoning to the remaining two other schemas, the external and internal schema, which are at the same time databases. The results are represented in figures 5-2 and 5-3.

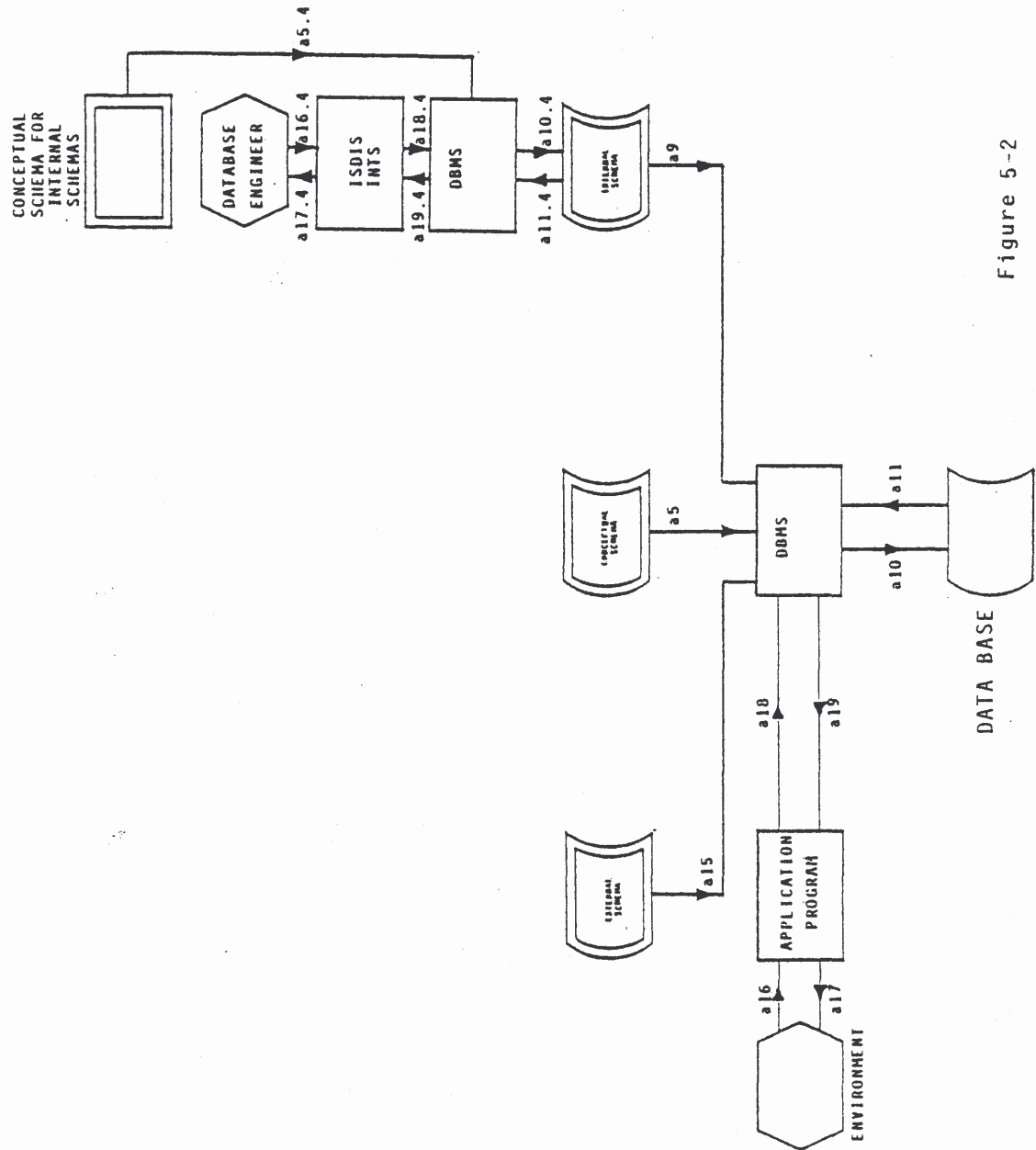


Figure 5-2

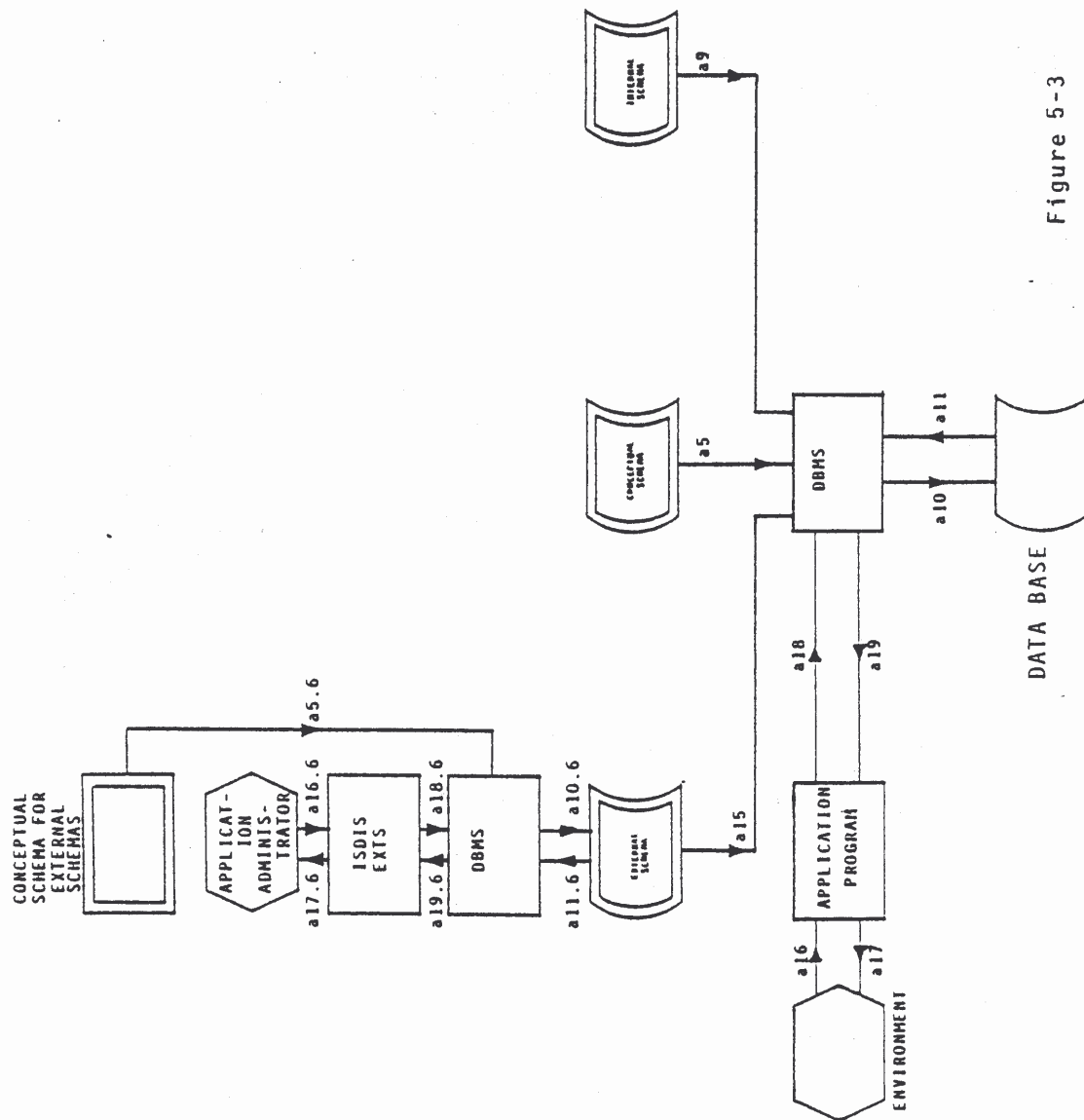


Figure 5-3

We can now combine the previous three figures and the result is represented in figure 5-4. The model of figure 5-4 could be taken as the overall architectural framework for database management software. It represents an architecture in which the meta information system (or ISDIS) is integrated with the information system (using the DBMS, or data base management system).

Once again, the point can hardly be overstressed that the overall architecture of figure 5-4 is obtained by applying the same simple reasoning, the model of figure 5-1, to all the databases involved.

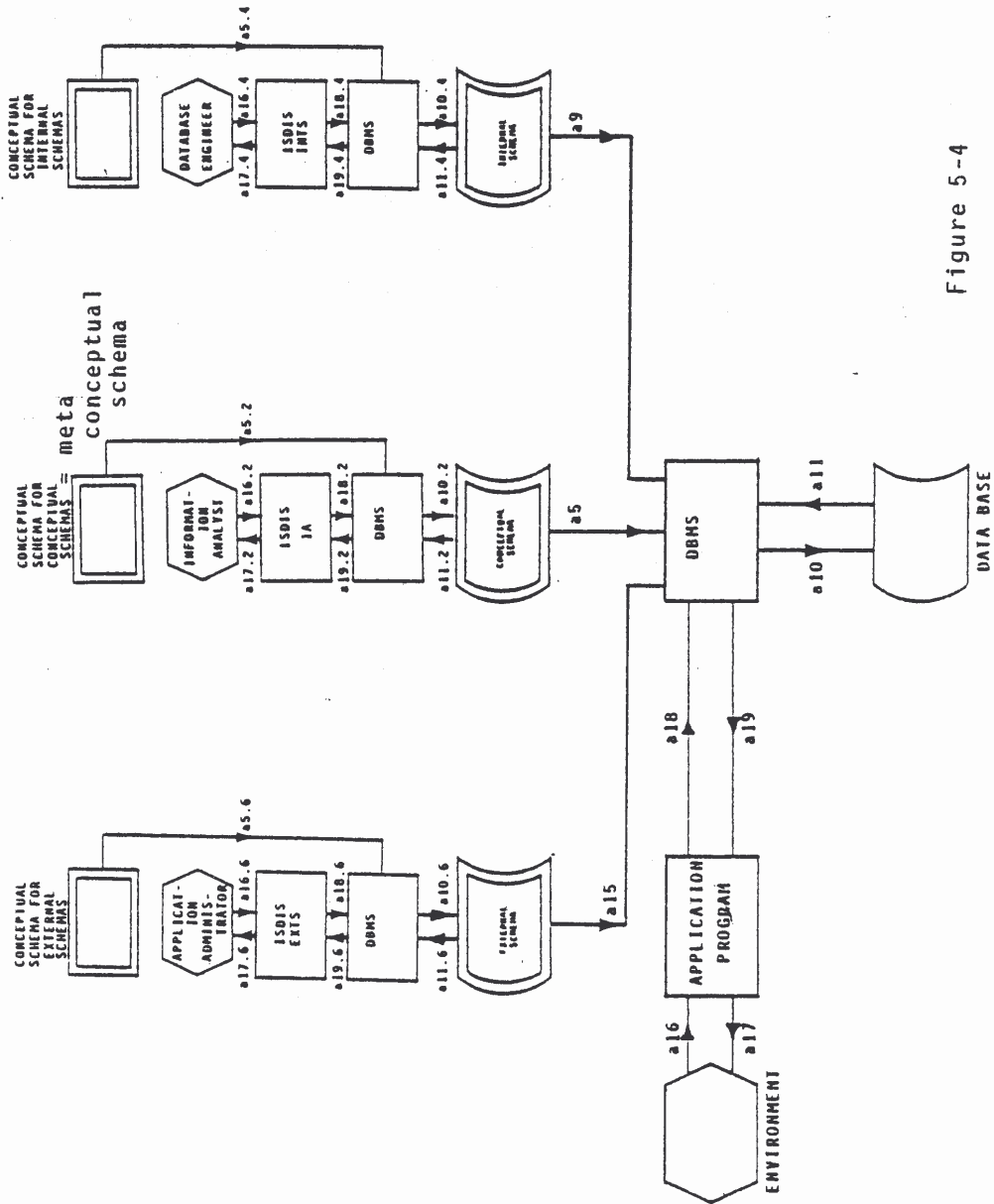


Figure 5-4

It is useful to transfer figure 5-4 by representing the 3 instances of the DBMS (which are all equivalent) by one instance of the DBMS, and by combining the 3 conceptual schemas into one conceptual schema, called conceptual schema for meta information. (To avoid a lot of crossing lines, there are still two instances of the DBMS in the picture which are completely equivalent). The result is represented in figure 5-5.

SUMMARY OVERALL ARCHITECTURAL FRAMEWORK FOR DATA BASE MANAGEMENT SOFTWARE

The architecture or model as represented in figure 5-5 is primarily of unifying nature. It shows that building an application of a user (say payroll) can be treated the same way as building a conceptual schema compiler, or external schema compiler etc.

It is this unifying nature, based on a small number of principles, that makes this architecture attractive for the author.

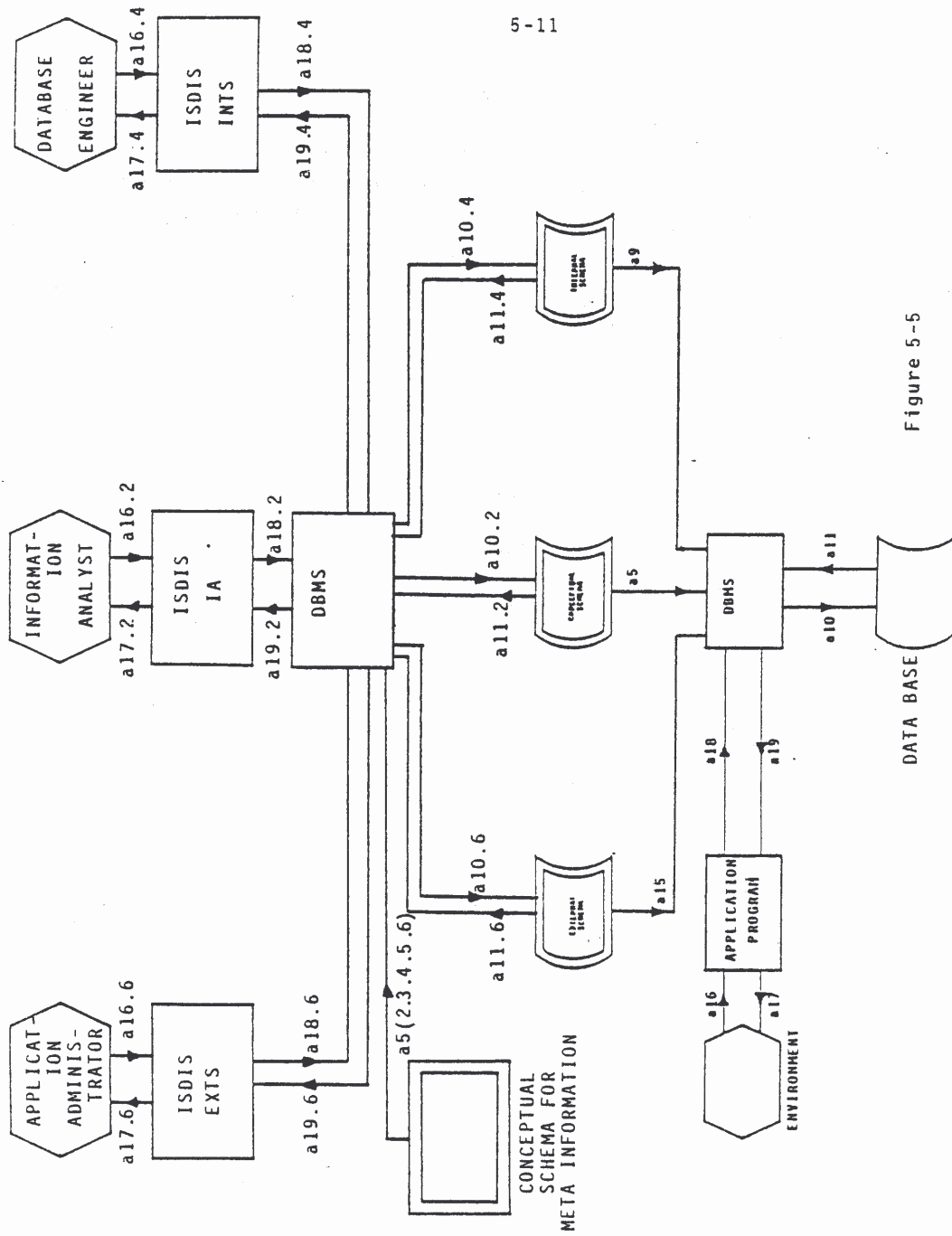


Figure 5-5

6. SUMMARY AND CONCLUSIONS

In this paper we have described an architecture for an integrated data dictionary (software engineering) and database management system.

We started with a model in the beginning of chapter 2 which is now also published in the ISO report of February 1981 on Conceptual Schemas.

We thereafter introduced five principles, each of which can be easily verified, and the operations of decomposition, composition, abstraction and de-abstraction to come to the final architecture of figure 5-5.

REFERENCES AND BIBLIOGRAPHY

1. 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).

2. BRACCHI G., CERI S., PELAGATTI G.

Structured Methodology for Designing Static and Dynamic Aspects of Data Base Applications.

In: Information Systems, volume 6, number 1, 1981, pp. 31-45.

3. BUBENKO J.A.

Time-handling in Data Base Management Systems.

In: Nijssen G.M. (ed.): Architecture and Models in Data Base Management Systems. North-Holland, Amsterdam, 1977.

4. CHEN P.P.

The Entity-Relationship Model - Toward a Unified View of Data.

ACM Transactions on Database Systems, March 1976, Vol. 1, Nr. 1, pp. 9-36.

5. CODD E.F.

A Relational Model of Data for Large Shared Data Banks.

Communications of the ACM 13, 6 (June 1970), pp. 377-387.

6. CODASYL

DDL Journal of Development, (1978).

7. FALKENBERG E.

On the Conceptual Approach to Data Bases.
Proceedings of the International Conference on Data Bases,
University of Aberdeen, U.K., (1980).

8. FALKENBERG E.

Foundation of the Conceptual Schema Approach to Information
Systems.
Lecture Notes, NATO Advanced Study Institute, Data Base Manage-
ment and Applications, June 1-13, 1981, Portugal.

9. ISDIS 2.2 USER MANUAL

Data Base Management Research Laboratory, Brussels, 1981.

10. ISO

Concepts and Terminology for the Conceptual Schema.
Preliminary Report, ISO TC97/SC5/WG5, edited by J.J. van
Griethuysen, February 1981.

11. KENT W.

Data and Reality.
North-Holland Publishing Company, Amsterdam, (1978).

12. NIJSSSEN G.M.

Modelling in Data Base Management Systems.
Euro-IFIP 1979, Proceedings North-Holland Publishing Company,
Amsterdam, (1979).

13. NIJSSSEN G.M.

A Framework for Advanced Mass Storage Applications.
IFIP MEDINFO 1980, Tokyo, Proceedings North-Holland Publishing
Company.

14. RIDL USER AND REFERENCE MANUAL V.1

Data Base Management Research Laboratory, Brussels, 1979.

15. SENKO M.E.

Conceptual Schemas Abstract Data Structures, Enterprise
Descriptions.

In: International Computing Symposium 1977, Proceedings of the
International Computing Symposium 1977, Liège, Belgium,
April 4-7, 1977, published by North-Holland Publishing
Company, Amsterdam, (1977).