Joint Use of Data Banks*

(A paper by Christer Arvas, Sweden, consultant to the U.N. Economic Commission for Europe Secretariat)

The development of time-sharing systems and data communication technique has led to the concept of computer utility: the same data base and the processing facilities of large electronic computers are shared by many users for many different purposes. The next step will be the storage and sharing of data in central data banks.

1. INTRODUCTION

1.1. The national information system

The national information system could be illustrated in the extremely simplified way shown in Fig. 1.

Individuals ask for information about specific individuals and about different groups [1, 3]. Groups are interested in information about themselves, other groups and individuals [2, 4]. The decision-making institutions (planning and administrating) request information about individuals as well as about groups [5, 6].

In order to compile information about groups [3, 4, 6] — e.g. official statistics — we must utilize the individual information to a greater extent. It seems reasonable to assume — and it is actually a fact — that much of the data which build up flows [3, 4, 6] are contained also in flows [1, 2, 5]. The technical development, hand in hand with the rapidly increasing extent to which the results of this development are used by, e.g., administrative routines, makes it possible to use these primary individual data jointly.

The fact that individual information is available in machine-readable shape in many administrative routines means that data-banks actually are emerging, whether we choose to use them for compilation of statistics or not.

---

* Excerpts

It implies that the dissemination will take place on a lower level that earlier; elementary information can be supplied instead of approximations computed from aggregates.

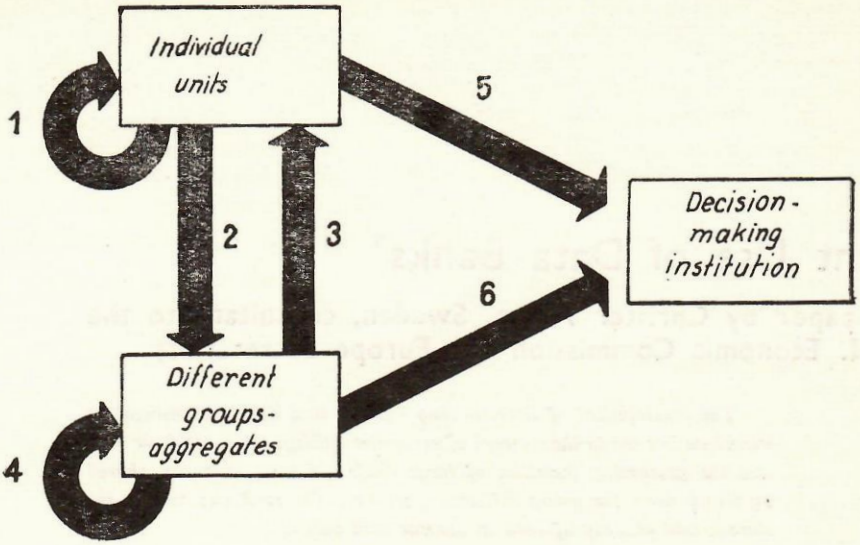


Fig. 1

This means, however, also that a great many problems associated to joint use of data have to be satisfactorily solved. The question, raised e.g. in [4], whether central "data-banks" will increase the availability of official statistics or not, will be largely dependent on how well these problems are solved.

1.2. **Data utility**

Within the field of data processing and automation the concept of "*computer-utility*" is being discussed more and more seriously. The term indicates that the processing facility of a computer could be regarded in much the same way as the "ordinary" utilities such as water, electricity, etc. The development of the so-called time-sharing systems and equipment for teleprocessing has made it possible for several users to share processor-time and still get the impression that they operate "their" machine. So far the computer utility has mainly meant joint usage of processor, core-memory and input/output equipment. The systems have, at least in some cases, been thoroughly analyzed, and, for example, the optimal number of users as a function of desired response-time have been computed [17].

Lately attention has been focussed on the possibility of sharing also the data on which the processor operates. This of course is of great interest for the statistician who has always to some extent been using data produced primarily for non-statistical purposes [13, 15]. This emerging possibility of joint

usage of the "*data-bases*" in computer systems could in analogy with computer-utility be thought of as data-utility. We are already familiar with many of the problems that will appear through experience with computer software. The collection of language translators, operating systems and other standard procedures which in one word is called software is of course nothing else but a somewhat specific example of data for joint usage. A great amount of work is required for standardizing and describing these data — standardization and description will be major problems also in the development of data-utility for "*real*" data. Needed is a sound theoretical base for the design of data-utility systems. Much work has been done (e.g. [7, 8, 9, 12, 13, 14]) and much remains to be done. Such a theoretical structure ought to cover evaluation criteria for the whole system, requirements necessary to ascertain communication and accessing possibilities, organization of the data files and management of the total system.

1.3. Data-banks

The term data-bank, frequently used for the data base which is to be jointly used, is not quite clearly defined. An information system consists of four sub-systems: one for inputting data, one for storing data, one for processing data and one for outputting data [9]. The mere fact that data is stored in a system obviously does not mean that the information system is a data-bank (DB).

We require that the data stored are to be used in several applications and by several users to call it a DB. This joint usage — integration — can take place in essentially two different ways. Either the same user can access the data at different points in time (e.g. the data from last year's survey used as reference in the editing of this year's survey) or different users can access the same data for different purposes (e.g. statisticians using administrative data). We shall in this paper require the latter type of joint usage to characterize the system as a DB. In order to make a meaningful distinction from ordinary information systems, we finally require that the requests from the users are not completely known at the time the system is designed. On the other hand, we do not give any restrictions concerning centralization, either physical or administrative. Rather we recognize the questions of how and when the systems should be centralized as two of the main questions in the design of a DB. The file system within a DB, often called the data-base, interacts with the environment in different modes. Figure 2, copied from [21], illustrates five modes: updating, interrogation, data base building, data base management and storage allocation and protection. The updating, interrogation, and protection modes are the fields where the major principal problems arise. The data base building and management are, however, of great economic importance in DB's for statistical use where the files become extremely large.

1.4. Technical changes

The development towards DB systems has been evoked by the very rapid technical evolution. There is a considerable gap between technical possibilities and our ability to make efficient use of these posibilities.

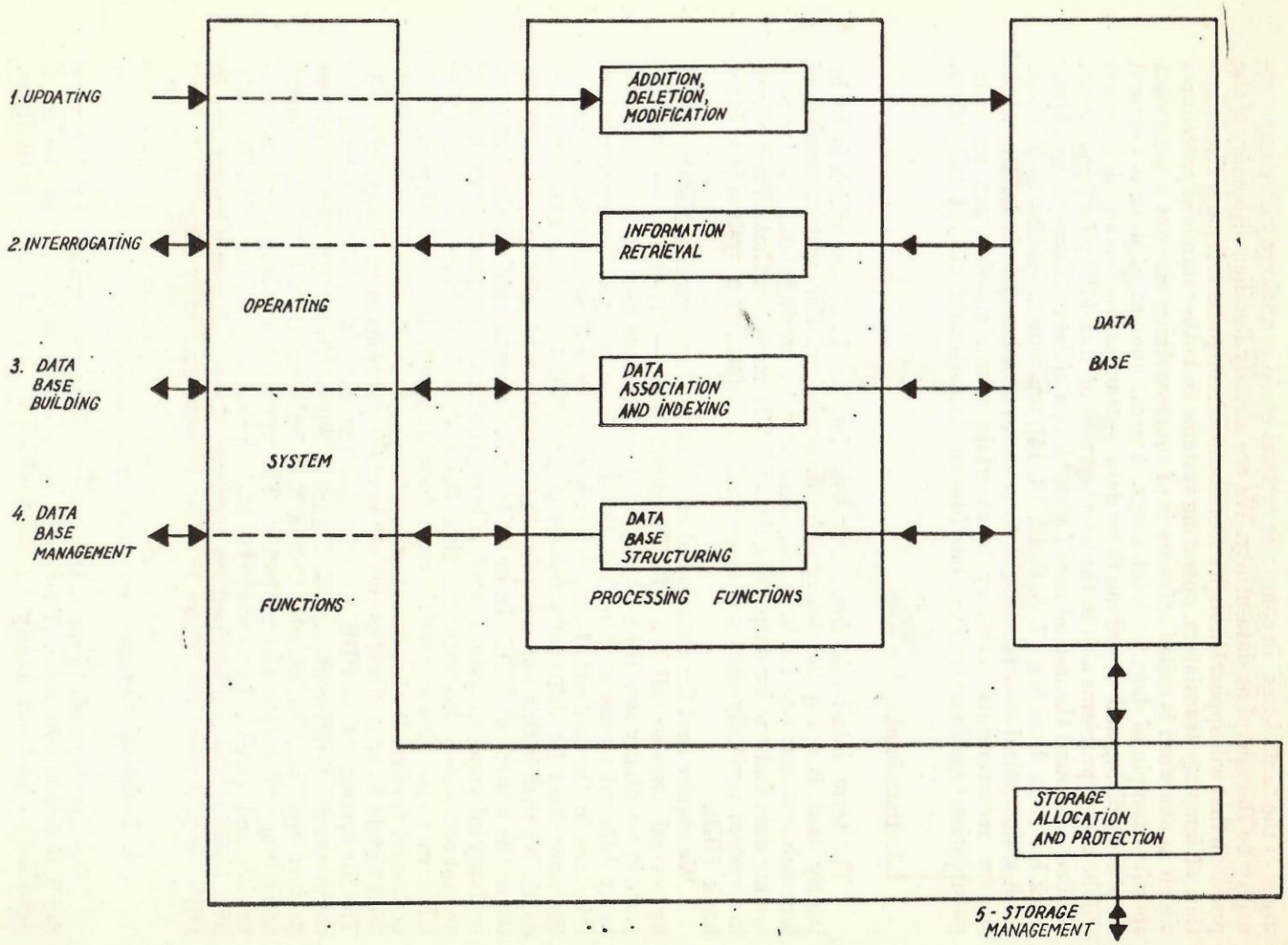


Fig. 2: Five modes of interaction with data base

On the one hand, we must try to overcome the gap by deliberately trying to be more technically conscious. It is not possible to construct large data processing systems without a profound knowledge of the technical tools available.

On the other hand, the evolution is still so fast that the relationships which hold today between costs and different types of system-designs may be obsolete tomorrow. One exemple pertains to memory costs. Central memories, today very expensive, may well soon become a comparatively less expensive [11] storage medium. The present DA memories, now relatively slow and costly because of the mechanical access-mechanisms, could possibly very soon develop to an economically much better alternative.

It seems as if we should try to concentrate much of our abilities on developing models, which are as independent as possible of the characteristics of the equipment. This means then that we must try to find basic technical relationships which we believe will hold true for relatively long periods.

In this paper the following questions will be briefly discussed:

- decisions about what data should be included in the DB;
- definition of elementary units and description of data;
- design of the physical file system;
- storage media and different ways of processing;
- linking and record matching;
- communications machine to machine and man to machine;
- information protection;

Finally, there will be given examples from the Swedish Central Bureau of Statistics, the register system and the merging of different data-files.

2. INCLUSION OF DATA

2.1. A general model

In the DB systems discussed by statisticians, the so-called registers are often considered to be the backbone of the system. The DB's are said to be built on and around the basic register system. In this section we shall try to present some formal aspects of the general question of inclusion of data in DB's.

We are actually looking for a set of criteria which could help us to make the decision whether a given file of data should be included in the DB system or not. Obviously we face difficulties from the very beginning. What exactly — or at least approximately — do we mean by "included in the DB system?" We have earlier stated that the files in the DB do not have to reside in the same geographical place. What then is the hall-mark by which one can tell that a certain file belongs to the DB system? I believe that as a temporary definition we could say that the DB manager "arbitrarily" decides which files are to be considered as members of the DB. By doing this he assumes responsibility for the actions which must be taken to make it possible for the users to process the file. A file in a DB can just as well be maintained by the DB as by some other institution. We thus have to try to outline a decision function for the DB manager to follow when he decides whether to include a specific material or not.

Joint usage of data is evidently only justified if the cost associated to joint usage is less the sum of the different users' costs for maintaining the material themselves. This fact is so evident that it may sometimes be overlooked. The decision to be made is to keep either one file F which is to be accessed by the users $u_1, u_2, \ldots, u_n$ or to keep n files: $F_{u_1}, F_{u_2}, \ldots, F_{u_n}$.

The DB alternative implies cost for maintenance of F : $M(F)$, accessing costs: $A_1(u_1)$, $A_1(u_2)$, ..., $A_1(u_n)$ and the cost of the users' losses due to worse quality than they would have if they designed their maintenance systems themselves: $L(u_1), L(u_2), \ldots, L(u_n)$. (These costs could in some cases be negative.)

The unco-ordinated alternative, on the other hand, implies maintenance and accessing costs for the n users: $M(F_{u_1})$, $M(F_{u_2}), \ldots, M(F_{u_n})$ and $A_2(u_1), \ldots$ $\ldots, A_2(u_2)$, $A_2(u_n)$. It should be noted that the important burden on respondents in statistical systems is to be included in the maintenance costs.

Although nearly every quantity mentioned, including the number of users: n, is unknown in a genuine DB, it seems obvious that the cost for the DB alternative

$C_1 = M(F) + \sum_{v=1}^{n} A_1(u_v) + \sum_{v=1}^{n} L(u_v)$ often exceeds the cost for the other alternative.

$$C_2 = \sum_{v=1}^{n} M(F_{u_v}) + \sum_{v=1}^{n} A_2(u_v).$$

However, this overlooks the main point of the DB concept. In the DB alternative there is, by the possibility of linking, great potential for the users to access data-combinations other than the originally requested $F_{u_j} (j = 1, 2 \ldots n)$. We should thus associate positive values $V_1(F), V_2(F), \ldots, V_n(F)$ to the DB alternative. Setting $V = \sum_{v=1}^{n} V_v(F)$ we can state that DB is preferred if: $V + C_2 > C_1$.

2.2. Registers in the model

In order to assess the desirability of including registers in the DB, we have to consider how different types of files F_{u_j} are apt to affect the components V, C_1 and C_2. V is directly dependent on the possibilities of linking different records. The efficiency of the linkage rules is of course greatly increased if the individual units are equipped with fixed identification numbers. A register is not only equipped with such numbers but it also serves as a base for maintaining those numbers in other files. Furthermore it is necessary that the data involved can be described in a "global" way. Register data usually consist of comparatively easily described variables such as name, address, size, location, occupation, etc.

This also indicates that the components $L(u_1), \ldots, L(u_n)$ in C_1 are relatively small: "a central register may be as useful as ou rown". An important part of those loss-components results from the fact that a commonly used large data-base may have a longer updating cycle than the local users would

have. This means that the kind of variables which seldom change their values give small losses, which is significant for register data.

The difference between the maintenance costs in C_1 and C_2, $M(F)$ and $\sum_{j=1}^{n} M(F_{u_j})$ respectively, of course depends directly on the degree of overlapping between the different F_{u_j}. Assume that:
$m(F_{u_j})$ = number of records in file F_{u_j}.
The greater is the difference:

$$\sum_{v=1}^{n} m(F_{u_j}) - m(F),$$

the greater the difference $C_2 - C_1$ will be and so the preference for a DB solution will increase. The accessing-costs $A_1(u_1), \ldots, A_1(u_n)$ and $A_2(u_1), \ldots, A_2(u_n)$ probably depend not so much on whether the data in the iles are of a register type or not.

It seems to follow from the discussion above that register data should have a good chance of being included in the DB systems at an early stage.

2.3. Ranking of data to be included

The interdependence between the different kinds of data and data combinations in the system can be described by a precedence operator P[9]. P operates on an arbitrary set of data, say M, and its value is a set of data; namely the data which directly precede M. $P(M)$ in the system shown in Fig. 3 would be: $P(g, f) = (c, d, e \text{ and } f)$. If we now let P operate on the set $P(M)$ we get $P(P(M)) = P^2(M) = P(c, d, e, f) = a, b, c, d$. Clearly $P^3(M) = \varnothing$. We can now see that the data necessary to produce a set M is given by

$$R(M) \equiv P(M) \cup P^2(M) \cup P^3(M) \ldots \cup P^k(M)$$

where $P^{k+1}(M) = \varnothing$.

We can also define an "inverse" operator S, so that $S(M)$ is the set of data directly succeeding M.

$S(M) = (g, h)$ in the systems above. Equivalent to $R(M)$ we can construct $D(M)$ to be the set of all data depending on M for their existence.

$D(M) = S(M) \cup S^2(M) \cup S^3(M), \ldots, S^k(M)$ where $S^{k+1}(M) = \varnothing$

$D(a, b) = e, g$ in the system above.

It is not possible to give a straightforward answer to the question: "What is a certain set of input data (a file) worth in a DB system?" The same data are used for many different purposes and certain output data — which may have prices — are constructed from many different data.

In our simplified "model" of the DB system we have said that the users evaluate the output possibilities with their function V_i. We assume that

these V_i can be used to assign a value to every possible output data. We can then state that the inclusion of file F_{u_j} into the system increases the value of the system with:

$$\Delta V = \Sigma v[D(F_{u_j})]$$

i.e. the sum of the values of the data dependent on the data in F_{u_j}. The quantity ΔV which may well be quite independent of V_j, could possibly be used

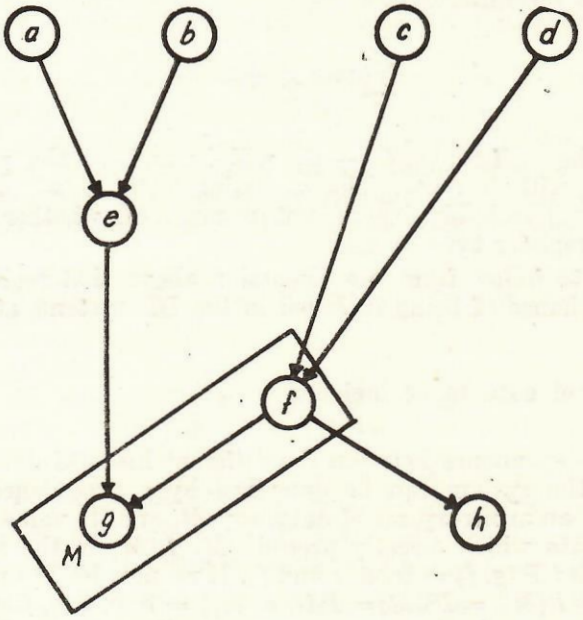


Fig. 3

for ranking potential members of the DB. The complicated way in which this very simple fact is expressed is due to the possibilities of constructing algorithms for systematic work on these lines. The operators P and S can be implemented as matrices (which will be further discussed in connection with the design of file systems.)

3. UNITS AND AGGREGATES IN THE SYSTEM

The objective of the DB system is to provide information about different parts of a social and economic system. This means that we must be able to compile data pertaining to different partitions of the system. Moreover, we must be ready to accept data about partitions on different levels. The system must work with data about individual units as well as different aggregates both as input and output. Let S_1^k, S_2^k, ... , be the different system-parts under the partition k. (e.g. S_1^k = { All men }, S_2^k = { all women } under the partition of all inhabitants by sex.)

If for every partitioning S^k there always exists another "higher" partitioning S^j so that for every m: $S^k_m \subset S^j_1$ for some 1, then the partitioning is hierarchical.

In this case data about any part can always be assembled from parts on a lower level. There is a "lowest" partitioning, the parts of which we call individual units. (Many of the classification systems used in the statistical field are hierarchical systems of this kind).

The DB system must, however, also be ready to work with partitioning systems where elementary units exist but where the "inclusion criterion" mentioned above does not hold true. Well-known examples of this can be found in the demographic area. The individual unit is well defined but we work at the same time with aggregates such as "family" and "household". These non-hierarchical partitionings of course raise difficult technical problems, but they are, at least theoretically, capable of solution. This follows from the existence of elementary units out of which every part in any partitioning can be constructed. (We can either mark the individual units according to which groups they belong to, or we can make lists for every group over the individual units included in the group).

The most difficult case is when we work with non-hierarchical partitionings in systems where we lack obvious individual units. This is the great problem in the field of economic statistics. The "elementary" economic activities are grouped in different ways by different users and for different purposes. The administrative institutions will often be interested in, and provide information about partitions on a judicial base, the local authorities in regional partitions, the economists in some kind of "span of control", etc. It is then necessary to construct cross-references and distribution rules to be able to carry information between parts in different partitioning systems. To this end the statisticians have constructed e.g. the artificial concept "establishment" which is meant to serve as an individual unit in different applications. The problems are, however, far from being solved and they will probably turn out to be one of the major sources of difficulty when the DB systems for joint usage of data in the field of economic activities start to develop.

These problems have of course always been the central questions in the field of official statistics. Much international discussion has already taken place with results in the form of different recommendations for standards. The main purpose hitherto has, however, been to secure comparability between aggregates on certain levels — both horizontally between surveys at different places and vertically between surveys at different points in time. With joint use of elementary data, the requirements are further sharpened.

4. DESIGN OF THE FILE SYSTEM

4.1. System matrices

The establishment of a DB system will result in a large volume of files. Different methods of describing and analyzing such file systems have been suggested [9, 13]. The methods described [9] are built in the concept of "elementary files". A brief review of this method will be given in the present

section. An elementary record consists of unit identification, a code for the characteristic or change which is registered, a point in time and the value measured. With the help of some simple algebraic tools the file system is then described and analyzed in terms of elementary files (consisting of elementary records).

The network in Fig. 4 can be used as an illustration of an information system.

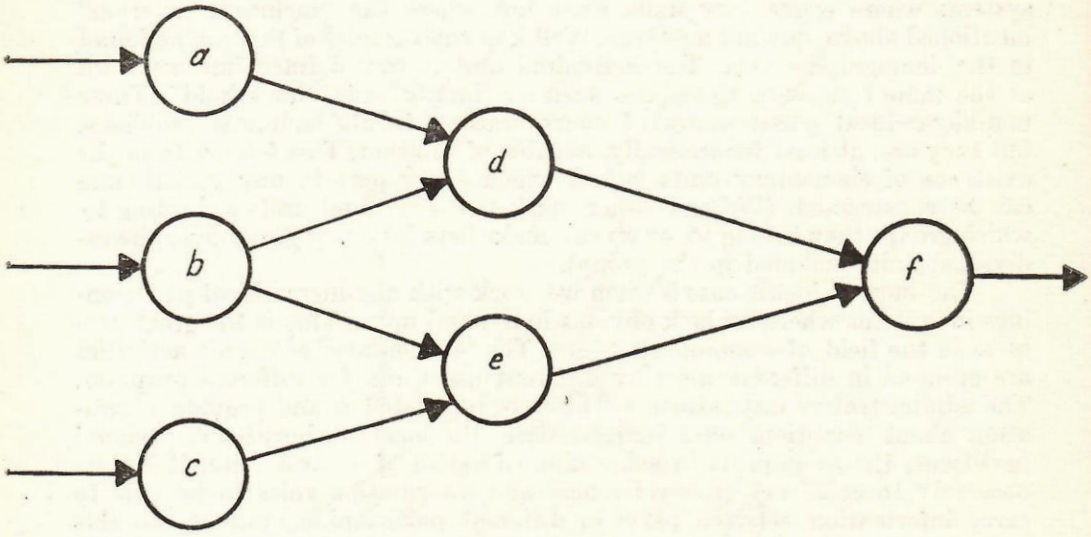


Fig. 4

The network can be completely described by a matrix P, with one row and one column for every node (circle in the figure).

$P =$

	a	b	c	d	e	f
a				1		
b				1	1	
c					1	
d						1
e						1
f						

The element P_{ij} (row i, column j) is set to 1 if node i is directly preceding node j.

The nodes can be interpreted as elementary information files. With this view of the network, we can draw it in a different way where we also account for the processes, taking for example files a and b and making file d (Fig. 5):

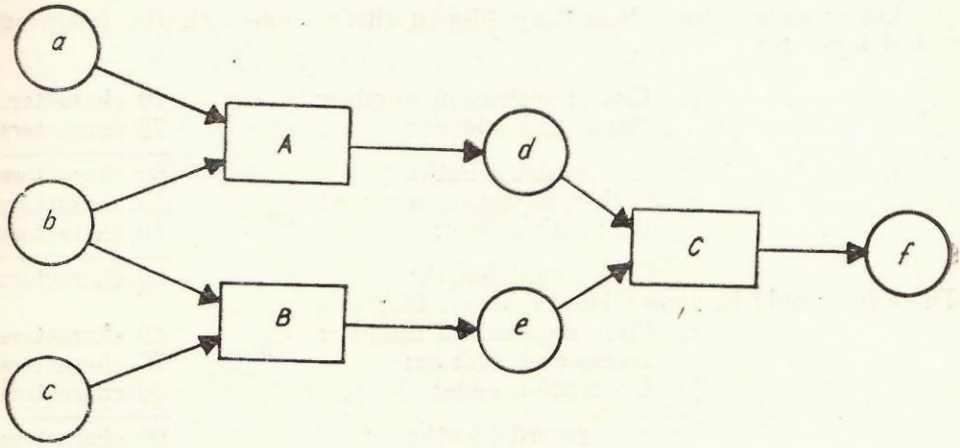


Fig. 5

This system is more conveniently described by another matrix E which has one row for every process and one column for every file.

$E =$

	a	b	c	d	e	f
A	1	1		−1		
B			1		−1	
C				1	1	1

The element E_{ij} (row i and column j) is set to 1 if file j is input to process i and to −1 if it is output.

The matrices P and E obviously describe the same thing and either one of them could thus be transformed to the other with the help of an algorithm.

4.2. Process grouping and file consolidation

The realization of a file system, when the elementary matrix E is given, usually calls for grouping of processes and consolidation of files.

By file consolidation, we mean here that two or more files are merged together to form one new file. In systems with serial files (magnetic tape),

consolidations are required by the limited capacity of input/output equipment. We also save on storing and transporting of identification fields. In general the file consolidation will, however, introduce "dead-weight" transports in the system. (Unnecessary information is transported to processes which do not make use of all fields in the records). The following simplified example illustrates the file consolidation.

Let us assume two elementary files in the system with the following record lay-outs:

a.	Civic registration number:	10 characters
	Name and address:	70 characters
	record length:	80 characters
b.	Civic registration number:	10 characters
	Occupation code:	10 characters
	record length:	20 characters

These files could be consolidated into one file:

c.	Civic registration number:	10 characters
	Name and address:	70 characters
	Occupation code:	10 characters
	record length:	90 characters

There can be three types of processes:

A: Name and address as well as occupation code required.
B: Only name and address required.
C: Only occupation code required.

If we have N units (persons) in the register, the data transports would be:

	Elementary files	Consolidated files
A	100 N	90 N
B	80 N	90 N
C	20 N	90 N

After a file consolidation, matrix E will be changed in the following way: All processes which in the old system had any of the files to be consolidated as input will in the new system have the consolidated file as input.

The grouping of processes of course saves file transport. In the case of sequential processing, one scan is saved for every file which is to be used. In the case of direct access, one transport is saved for every record which is to be used in both processes. The possibility of grouping will however be restricted by limited memory capacity and logical relations in the system. Matrix E will be changed according to the following rules: a file which was input to one or more processes to be grouped should be input to the new grouped process. If a file was output of one the processes, it should be output of the grouped processes — with one exception. This is when the file is only

used as input to another process to be grouped — in this case it need not be included at all in the new system. An example:

$E =$

	a	b	c	d	e
A	1	1	−1		
B		1	1	−1	
C	1			1	−1

will, if A and B are grouped, be transformed to:

$E_1 =$

	a	b	d	e
AB	1	1	−1	
C	1		1	−1

which then, if the files a and b are consolidated, will be transformed to:

$E_2 =$

	ab	d	e
AB	1	−1	
C	1	1	−1

The graphs of these systems are shown in figures 6, 7 and 8.

4.3. Use of the matrices

Matrix E can of course be stored in a computer and the transformations corresponding to grouping and consolidating can be done in a formalized way. After having stored the elementary matrix of an information system, we can thus compute, e.g. total transport sizes for different system alternatives (i.e. different alternative consolidations and groupings).

If the DB systems are to be analyzed on these lines, we have to introduce stochastic elements in the matrices as the elementary matrix E is not known at the time the system is designed.

The system design can be evaluated in accordance with different criteria. Design flexibility, simplicity, low establishment cost, low running cost, high level or "readiness" could be mentioned as examples of possible objectives. If low running cost is an objective (which of course it will always be, together with other objectives) then the data transports in the system will be of great interest. The method for systems analysis just discussed can be

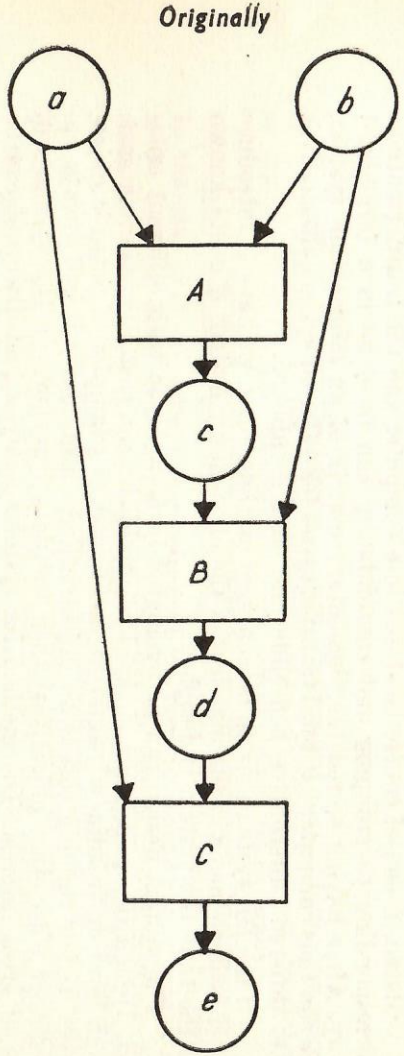


Fig. 6

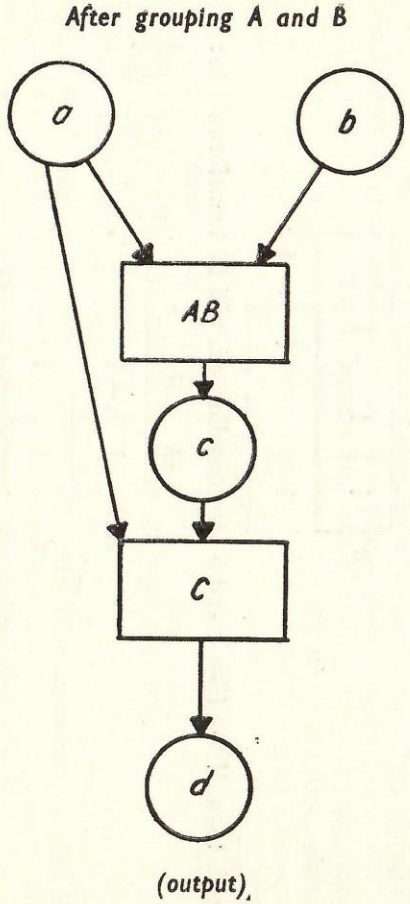


Fig. 7

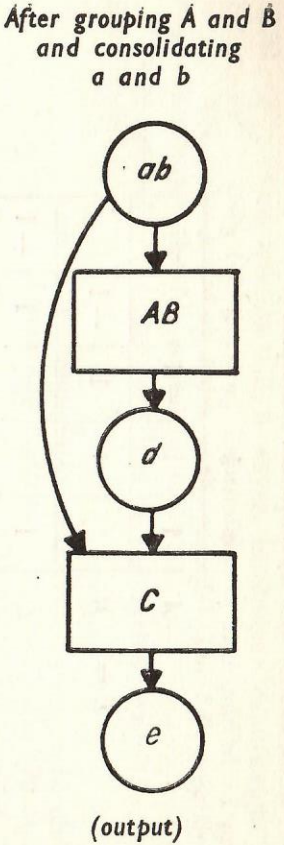


Fig. 8

used for an iterative search for system alternatives with low data transport volumes. It does not furnish any general method of minimizing the transports. This can, however, be done for certain classes of file systems [18].

5. STORAGE AND PROCESSING

5.1. Basic concepts

In order to discuss file organization adequately we must keep in mind that the file always resides on a device of some kind, (printed forms, paper tape, cards, magnetic tape, micro-film, drum, disc or something else). The organization of a file, which we can define as a collection of records, will only be discussed in connection with a device. The file proper can have a potential for being ordered in agreement with some field in the record, the key. If the value of a key field is unique for each record in the file, it can be used as an identification of the record.

The device may be thought of as an ordered number of blocks; in each block we may place one or more records. (The block is then the smallest addressable unit on the device.) The block size on a device most often varies according to the application. It is sometimes convenient to look upon the collection of a number of adjacent blocks as a "bucket". The file may then be regarded as an ordered number of buckets — the adress of each bucket being for example the address of the first block in the bucket. The devices are of essentially two different types: serial or direct access.

The serial devices form that subset of the set of all devices whose members are characterized by the inability to access a single block selectively; the blocks must be accessed one after another in the machine address order. [1]) The remaining types of devices are direct access devices.

Files are used in two essentially different ways: matching with another file or "computing" from the file. Matching with another file is the most common way, e.g. regular file maintenance is a matching between a transaction file and a master file where records in the master file corresponding to transaction records are deleted, inserted or changed.

Computing from a file takes place when we compute some characteristic from the file as a whole by applying formal rules, e.g. counting the number of records, computing mean and standard deviation for the contents of a field in records with certain characteristics, displaying the identifications of records with specified characteristics. (Displaying records with given identifications, on the other hand, is a matching of files).

There are two types of file organization, namely sequential and non-sequential. If the records are to be accessed in order (ascending or descending after some key), the organization is sequential. In other cases the file is non-sequentially organized on the device. [2])

---

[1]) Our definition of a file as a collection of records implies that a file can always be regarded as consisting of "sub-files". A file can well be non-sequentially organized with respect to its sub-files while each sub-file is sequentially organized.

[2]) The serial devices usually have a few other distinguishing characteristics which do not necessarily follow from the definition. For example, one cannot change the contents of a single block without re-writing the whole file. It is also usually faster to access a whole file when it is placed on a serial device than when it is placed on a direct-access device [3].

5.2. Different ways of matching

Two files can be matched in sixteen different ways, depending on the device type and the organization of the two files.

We shall here restrict ourselves to the eight cases which are possible when one of the files — let us call it the transaction file — resides on a serial device. In the first four cases we assume that the organization of the transaction file is sequential, in the next four cases non-sequential.

(*a*) Sequentially organized transaction file.

(*i*) Master file on serial device — non-sequential organization.

This is a hopeless and unrealistic case. The master file must be searched from the beginning for every record on the transaction file and this is very time-consuming.

(*ii*) Master file on serial device — sequential organization.

The classical case of file maintenance. Only one scan of each file is necessary.

(*iii*) Master file on DA device — non-sequential organization.

This is the most interesting case. The sequential ordering of the transaction file cannot be utilized, but each transaction record must independently be matched with the corresponding master record. Different methods such as using index tables or "randomizing" are used to locate the master record in question on the DA device. The files and the expected type of process must be thoroughly analyzed in order to design efficient accessing methods in this case.

(*iv*) Master file on DA device — sequential organization

This can often speed up the processing as compared with case (*iii*). The sequential organization can, as in case (*ii*), be obtained by having the master records ordered after ascending or descending machine-addresses. In this case, though, other ways of obtaining the sequential organization also exist. By chaining, list processing or ordered index tables the master records can be eligible for sequential processing also when they are physically spread in a "random" way over the device.

It should be noted that the matching methods mentioned in case (*iii*) can be used also in this case, whether the master records are in machine address order or not.

(*b*) Non-sequentially organized transaction file.

(*v*) The case with a non-sequential master file on a serial device is equivalent to case (*i*).

(*vi*) Sequentially organized master file on a serial device deserves no more attention. Both files should, if possible, be ordered when they reside on serial devices.

(*vii*) Non-sequential master file on DA device is exactly equivalent to case (*iii*).

(*viii*) Sequentially organized master file on DA device is the final case. The sequential organization is of no use in this case. Either we have to design methods for direct addressing as in case (*iii*), or we have to search the master file from the beginning for each transaction record. The search can follow the machine-address order, or the logical sequence if they are not the same.

5.3. Choice of device type

One can thus distinguish two main types of external storage media. The sequential access type — mainly magnetic tape — is characterized by the necessity to scan the complete file in order to access a subset of records. On the other hand, on a direct-access type (DA) it is possible to access any record selectively. Much can be said about organizing data on the different types of storage media. These questions are naturally of great importance in a data-bank system and must be given serious consideration. The huge masses of data will require much analysis in order to be handled in a sufficiently economic way. The joint use also means that the files have to be organized in such a way that many different persons and institutions can learn how to use them for processing.

We shall here give some brief consideration to the question: serial or DA memories? Let us make some rough assumptions:

n = number of runs in the reference period;
m = number of records requested in one run;
c_t = cost of one tape scan;
c_d = cost of accessing one record on the DA device.

(We assume it to be constant for all records.)
The following decision-rules can be stated:

(a) *n and m deterministic:*
Choose tape if:

$$nc_t < nmc_d;$$

$$c_t < mc_d; \tag{1}$$

(b) *n deterministic and m stochastic:*
Choose tape if:

$$nc_t < nE(m)\, c_d;$$

$$c_t < E(m)c_d;$$

$$\text{Let } E(m) = \bar{m} \text{ and } E(n) = \bar{n}$$

$$c_t < \bar{m}c_d \tag{2}$$

(Essentially the same rule as (1))

(c) *both n and m stochastic*
This is typical of the type of system which we call a data-bank in this paper. We have many users with unknown requirements on the system. In order to make any sensible design at all, we assign probabilities to different users' behaviour. (These probability figures are not necessarily quantified and used in computations, but we at least have them in the back of our minds).
Choose tape if:

$$E(n)c_t < E(mn)\, c_d$$

$$\bar{n}c_t < (\rho\sigma_n\sigma_m + \bar{n}\bar{m})\, c_d;$$

$$c_t < \frac{\rho\sigma_n\sigma_m c_d}{n} + \bar{m}c_d; \tag{3}$$

Both σ_n and σ_m can of course be relatively large; the question is how large is ρ and which sign it has.

We do not know anything about this, we can only speculate. If we have some vague notion of an approximately constant total access burden on the system (e.g. number of requested records), the correlation ρ coefficient will be close to -1.

We can then state that formula (3) means that direct-access devices (media) should be used to a greater extent in "stochastic" DB systems than in ordinary information systems.

The same rule is often stated in a more intuitive way: "In the DB system we are willing to accept the higher running costs of DA memories in order to be ready to meet unforeseen requests for small parts of large files".

NOTE: Rules (1) to (3) given above should be modified in an obvious way to take account of the fact that a DA device has a higher fixed cost than a tape device.

6. LINKING AND RECORD MATCHING

Joint use of data can be effected in two different ways: let us call them computation from a file and linking to a file. Computation from a file would then mean that the user "computes" measures from the file or some subset of it (see 5.2). Examples of this type of joint use are the computation of mean income for mean over 50 years of age from an income file or the listing of names and addresses of unmarried women under 25 living in Stockholm from a population register.

Linking, on the other hand, means the merging of two files and the matching of records (see 5.2). From a given list of records we have to find the matching records in the other file. This is the common situation in connection with joint use of data. We want to complete our own file of records with information from another file containing records for the same individuals. This linking procedure — which of course is the main point with the DB concept — has drawn attention of several writers. The papers [7] and [12] have references to theoretical work done in the field. It is necessary to construct linkage rules and computer systems for carrying out the rules. The linkage rules have to state with which records in the main file our list records are to be matched and when two records are to be considered as matched.

In the most general case every record in the list should be compared with every record in the file. A time-saving alternative to this is that we stop the comparisons when k records matching the list record are found. Different strategies could be designed so that we start the search for matching records in a certain part of the file.

The efficiency of this type of matching, where a great number of comparisons are to be made, is of course highly dependent on the design of the computer system. The main file will have to be run through once for every group of records from the list that can be taken into the memory simultaneously. If the main file may be grouped so that for a certain list record there is a good possibility that a matching record may be found in a specific group, then the merger may be much faster if direct-access devices are used.

In many cases, however, only one field (key) is used for the comparison between records. This of course greatly simplifies the procedure; the list and the main file can now both be ordered after the key field and a pure linear file process [9] can be attained. If more than one field is used in the comparison, but one field is given extra high weight in the matching rule, [7], this type of linear process may be used in a first stage. The remaining unlinked records in the list can then be considered as a new list which is treated in the more general way described above.

The matching rule operates on a comparison vector $C = (C_1, C_2, ..., C_k)$ where C_j is the result of the comparison between the j^{-th} components in the list record and the main file record. For every value of C the linkage rule L assumes one of the values: "matched", "unmatched" or "undecided". If sufficiently many of the fields agree well enough, the rule states "matched"; if many fields disagree to a considerable extent, it states "unmatched". In the case of "undecided" the rue L requires more information before a decision is made. Different fields will as a rule be given different weights in the comparison process and some types of disagreements between fields will be considered more serious than others.

Two types of errors can be made; the rule L can state "matched" for a couple of records which actually do not match or it can state "unmatched" for a couple of records which actually do match. The criteria for good linkage rules suggested in [7] say that with fixed probabilities for the two types of errors the probability of stating "undecided" should be minimized. (Because this decision can be considered an expensive one; more information must be gathered or the quality of the merger is lowered.)

One type of linkage process which is discussed in [12] is the regular updating of a file. The error of matching two records which actually do not match can here be divided into the case where a new record in the list is erroneously matched with an existing record in the main file and the case where a record in the list is matched with the wrong record in the main file. The error of missing an actual match would lead to an erroneous belief that the list contained a new record. We shall therefore consider three types of errors. The consideration of three types of errors instead of two seems relevant also in the case of regular merging.

A way of constructing a linkage rule would then be to assign weights to the three types of errors and minimize the sum of the expected values of these weighted errors [12]. The weights associated to the different types of errors would depend on the type of processing. In some cases the records in the list which are not represented in the main file, "new" records may be marked with a special code and sometimes we may even know that there cannot be any new records. Erroneous matching will probably most often be considered as the most serious error.

A very simple form of linkage rule is when two records are considered to be matching if an identification number has the same value in the list and in the main file. When the identification numbers have check digits, the probability of error is considerably reduced.

In the Swedish civic registration number one check digit is added. This means that if one digit is incorrect, the number will always be signalled as an error. If two or more digits are wrong it may or may not be signalled as an error

depending on the kind of error. If the errors are random, a number with two or more erroneous digits will, be signalled as an error in a little less than 90 per cent of all cases. If, on the other hand, the two digits for month of birth have been transposed with the two digits for day of birth, the error can never be found by use of the check digit. The probability that a record with an erroneous identification number which has not been signalled as an error may be (wrongly) matched with a record in the main file depends on the density of the identification number sequence, among other things.

7. DATA COMMUNICATION

7.1. General views

It is clear that one of the most essential problems in connection with the development of data-banks is concerned with data communication. In the first place we must accept vast amounts of data as input and they must be communicated to the system in an economic and safe way. We also recognize the possibility of decentralized DB systems where the communications within the system will constitute a major function. The first requirement, of course, is that the data must be described and registred so that they may be communicated in a meaningful way. Some standardization must be required. In the discussions concerning data description much emphasis has been laid on the necessity of flexible systems. A special data-description language has been suggested [13]. It could be noted here that the system for data description must include not only means for an exact definition of the data but also possibilities of providing information about their quality, (what

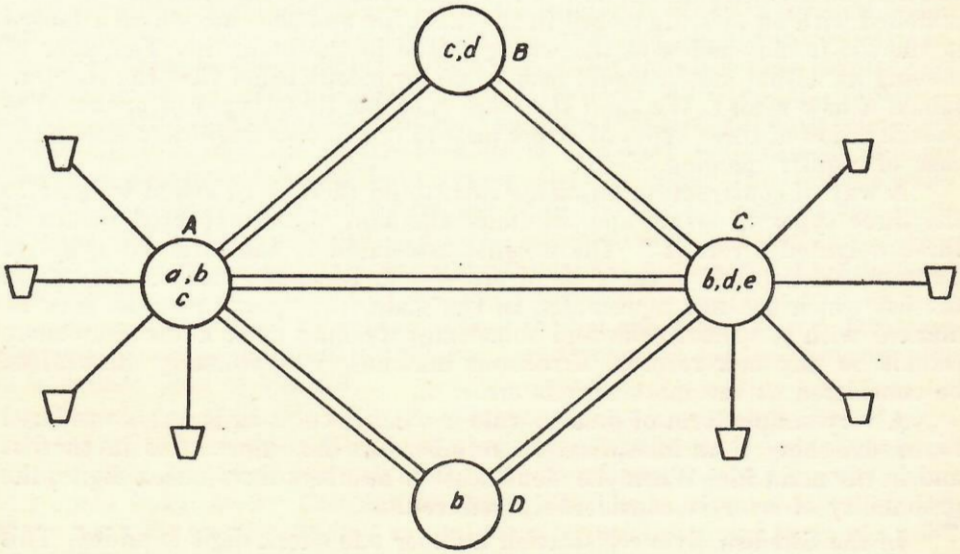


Fig. 9

kind of quality control and editing has been applied, when were the data edited, etc.).

Requirements of a more direct technical nature also have to be met.

A DB system could be visualized as in figure 9.

The circles represent computer systems, some of which are directly connected with users or data suppliers with terminal equipment (A, C). The five logically independent information systems may be operating in one computer system only (a, e), or they may use several computer systems (b, c, d). The computer systems may, on the other hand, be dedicated to one information system (D) or they may be shared by several user systems (A, B, C). Each of the computer systems consists of the processing means and the data base which to a greater or lesser extent is shared by the users. The communication links between computer systems is obviously of great interest in an organization of this kind. (It can further be clearly seen that here will be many problems of co-ordination if every computer system and every information system is managed "supremely").

7.2. Data transmission

The technique for data transmission within the DB system, both between terminals and computers and between the computer systems, is of great importance. The transmission may be over regular telephone lines or special transmission lines. The type of transmission may be simplex (only one direction), half-duplex (both directions but only one at a time) or duplex (both directions at the same time). The transmission speed (baud = bit/sec) may vary quite considerably. The relationships between these different alternatives and the costs must, of course, be thoroughly analyzed before the system is designed.

The methods of error detection and correction are important [10]. On a regular telephone line which is half-duplex, the time for switching the direction of transmission is fairly long. If data are transmitted in blocks of length b, the transmission speed is v, the switching time is t and the probability of erroneous transmission of one byte is p, we get the following approximate result:

Probability of error in a block: $1 - (1 - p)^b$.

Expected time for transmission of one byte will be:

$$t_b = 1/b\,[b/v + 2t + (1 - (1 - p)^b)\,(b/v + 2t)] = (1/v + 2t/b)\,(2 - (1 - p)^b)$$

The time is minimized if the derivative with respect to b is set equal to zero:

$$(1 - p)^b\,[2\,tv - (b + 2\,tv)\,b \log (1 - p)] - 4\,tv = 0$$

Standardization of both hardware and software equipment for data communication will of course be of great help in decentralized systems for joint use of data. Efforts are being made in this field and conventions are being proposed [6].

7.3. Duplication of files

The general question of file construction in an information system has been treated earlier. We shall now discuss the specific question of whether a file which is to be used in several geographically separated places should be kept as one central file or be copied to the different users. Obviously we have to compare the storing and communication costs of the two alternatives.

Let us assume the following:

(*i*) The file is to be used in n different computer systems. The transactions occur at one place — the system where the file is kept in the case where there is no duplication. (Only a minor modification has to be made if we assume that the transactions occur at random in all the different systems.)

(*ii*) The file size is N data units. The set of transactions during a given reference period is aN data units. (The "activity level" of the file is a.)

(*iii*) In the case in which there is only one physical file, f is the expected number of times that the file is requested from the other systems during the reference period.

(*iv*) The cost of storing one data unit during the reference period is s. The cost of transporting one data unit between two computer systems is c. This transportation can be done either by means of teleprocessing or by copying and distributing by, say, mail. In the latter case the unit cost c should be regarded as a penalty cost for delaying the processing or updating.

(*a*) *Duplicated files*
The cost for transport of transactions will be $(n-1)\ aNc$ and the cost for file storage will be nNs. The total cost will be $(n-1)\ aNc + nNs$.

(*b*) *One central file*
The cost for transport of files will be fNc and cost for file storage will be Ns. The total cost will be $fNc + Ns$.

Duplicated files are to be preferred when $(n-1)\ aNc + nNs\ fNc + Ns$. Evaluated for f we get:

$$f > \frac{(n-1)\,(ac+s)}{c}$$

If we drop the storage cost, which probably is comparatively small, we get:

$$f > (n-1)a$$

If we assume transactions to occur in all the systems we get:

$$f > \frac{(n-1)^2}{n}\,a$$

A cost for transport of transactions, $\frac{n-1}{n} aNc$ will then appear in the "one central file" alternative). As soon as we can make the assumption that unit costs for storing are small compared to "transport" costs, it thus turns out to be an easily operated rule.

7.4. Man-machine communications

In connection with the increasing use of time-sharing computer systems a new programming philosophy is beginning to materialize [19, 22]. The main characteristic of the new interpretative systems for on-line use is that they are used in a conversational mode and that they require very little training and computing knowledge to be used. The subject matter specialist can, sitting by this terminal, build up his programs piece by piece. He is allowed to make mistakes, some of which will be corrected immediately, and he is not required to have any knowledge whatsoever about the operations that actually take place. In the data-sharing systems we shall in a similar way have to allow the user access to the data he may want to work with, without knowing anything about the complicated structures in which his data are included. This requirement, which will become an absolute necessity when our DB systems are operated from remote places by the users themselves, ought to be fulfilled already at the present time. The structuring of the huge masses of data in the data banks is already getting so complicated that heavy costs are incurred when all the programmers have to learn exactly how to access different data. Such requirements can be met in different ways.

Some attempts have been made with standardized record formats for all files in a system [16]. In such a system the programmer must know on what file his data reside and he must be acquainted with the general characteristics of that file. But he must not know anything about the data fields he is looking for except their codes. There is no need for record layouts, knowledge of whether the fields are of variable length, etc. With a well-designed set of macro-instructions, even such a comparatively simple arrangement as that must be considered as a good step in the direction towards systems for joint of data.

Another approach, suggested in [9] and [14] assumes a computer-stored system description. The user provides the names of the elementary data he wants to access or some formal description of them. By consulting, e.g. the incidence-matrix of the system (see 4.1) the system-procedures find the relevant physical files to work with. Steps have been taken in this direction (stored "data-catalogues"), but the idea is still far from being fully materialized.

The number of existing programming systems is tremendous. Special-purpose systems for different machine systems grow like mushrooms. The statistical field is no exception [23]. This fact is of course in many ways unfortunate. Great problems arise when different machine systems are to be co-ordinated in a DB system. It is not possible for the DB managers to provide the software support needed to satisfy the different wishes of all users. Lately the development of so-called self-extending languages has started. In this type of language, the user himself can construct a language that suits

him, starting from a basic language "nucleus", In this way special purpose systems — e.g. for DB handling — can be built up from the general-purpose programming systems. (See, for example, Algol-68 and Control Data's "General Programming Language").

8. PROTECTION OF INFORMATION

Problems relating to disclosure in connection with data-banks have recently drawn the attention of many writers. (1, 8, 13, 14, 19). It is not necessary to regard the basic conflict only as an ethical problem for the managers of the data-bank; the conflict can instead be stated in cold decision-theoretical terms. There is a constant pressure on the DB to produce all kinds of information; there is a positive pay-off associated to "disclosures". On the other hand, the suppliers of the elementary data to the DB are extremely touchy about certain uses of the data. If they suspect that the information will be used in ways that are unfavourable to them, they will not be willing to submit the data. The DB manager has to find the optimal balance between a restrictive and an unduly unrestrictive policy which is quite complicated since it is hard to know when the suppliers have reached the point at which they refuse to submit data. For various reasons they will not stop returning the forms but rather start to adjust the figures they supply. This problem is one of the main obstacles to the construction of large data-banks for statistical purposes. As the DB is, by definition, characterized by uncertainty concerning the requests for output which will be made, it will be hard to convince the data-supplying individuals about the safety of the data they provide.

One of the main rules of thumb in statistical systems has earlier been: dentifiable individual data may not be disclosed, information about groups may be published. The weakness in this philosophy has been pointed out recently [13]. The complicated truth is, of course, that individual data may be quite harmless while certain information about groups may be of great damage to the group.

When the appropriate policies have been formulated, there still remains quite a complicated technical problem. In the ideal case the set of all possible combinations of the elementary data in the DB should be split into an open and a closed part, and this should be done in a different way for each category of users. Clearly this is not possible; a technically feasible solution must be found which can be accepted by the data suppliers, on the one hand, but which, on the other hand, does not hamper the flexibility of the data-bank too seriously. It should be noted that it could be perfectly legitimate, for example, for a user to work "in the machine" with confidential individual data in order to compile statistics for some group of individuals. It may be assumed that at least in the present childhood of data-banks the DB managers themselves must perform much of the processing in order to be sure that "closed" data or combinations of data do not leak from the system.

In the field of computer-utility the problem has been recognized. Systems for memory protection are implemented not only in order to prevent data from being erased but also to protect confidential information [8].

REFERENCES

1. ALS, G., *Statistical Secrecy and Related Judicial Principles in Statistical Planning*, "Statistisk Tidskrift", Central Bureau of Statistics, Stockholm, No. 6, 1967.
2. BARON, R., *Integration of Census and Register Data*, Conf. Eur. Stats/WG. 9/59 (and WG. 9/76).
3. BUCHHOLZ, W., *File Organization and Addressing*, "IBM Systems Journal", June 1964.
4. DALENIUS, T. E., *Official Statistics and Their Uses*, 36th Session of the International Statistical Institute.
5. DENNIS, J. B., *A Position Paper on Computing and Communications*, "Communications of the ACM", Vol. 11, No. 5, 1968.
6. EISENBIES, J.L., *Conventions for Digital Data Communication Link Design*, "IBM Systems Journal", Vol. 6, No. 4, 1967.
7. FELLEGI, I. P. and SUNTER, A. B., *An Optimal Theory of Record Linkage*, International Symposium on Automation of Population Register Systems, Jerusalem, September, 1967.
8. GRAHAM, R. M., *Protection in an Information Processing Utility*, "Communications of the ACM", Vol. 11, No. 5, 1968.
9. LANGEFORS, B., *Theoretical Analysis of Information Systems*, "Studentlitteratur", Lund, Sweden, 1966.
10. LYNCH, W. C., *Reliable Full-Duplex File Transmission over Half-Duplex Telephone Lines*, "Communications of the ACM", Vol. 11, No. 6, 1968.
11. MAYNE, D. *What's Next in Memories?*, "Datamation", Vol. 14, No. 2, 1968.
12. NATHAN, G., *Outcome Probabilities for a Record Matching Process with Complete Invariant Information*, "Journal of the American Statistical Association", Vol. 62, No. 318, 1967.
13. NORDBOTTEN, S., *Automatic Files in Statistical Systems*, Conf. Eur. Stats/WG. 9/52.
14. NORDBOTTEN, S., *Confidential Treatment of Data and the Use and Classification of Data*, paper to be published in "Statistisk Tidskrift", Central Bureau of Statistics, Stockholm.
15. OHLSSON, I., *Merging of Data for Statistical Use*, 36th Session of the International Statistical Institute.
16. RANTAPUU, M., *Standardiserad struktur av datarecord med varierande innehall*, "Nord SAM", 1966.
17. SCHERR, A. L., *An Analysis of Time-Shared Computer Systems*, "M.I.T. Research Monograph", No. 36, 1967.
18. SEPPÄLÄ, Y., *Definition of Extraction Files and Their Optimization by Zero-One Programming*", "BIT", Vol. 7, No. 3, 1967.
19. WEIZENBAUM, J., *On-Line User Languages*, "BIT", Vol. 6, No. 1, 1966.
20. ZWICK, C.J., *On Potential Problems of Invasion of Individual Privacy as They Relate to Proposals to Organize a Federal Statistical Data Center*, Bureau of the Budget, USA, 1967.
21. *Cobol Extensions to Handle Data Bases*, "ACM SIGPLAN Notices", Vol. 3, No. 4, 1968.
22. *Programming in Basic*, Central Electric Computer Time-Sharing Service, 1966.
23. *Statistical Programming*, "Applied Statistics', Vol. 16, No. 2, 1967.

UTILISATION COLLECTIVE DES BANQUES DE DONNEES

L'ouvrage porte sur des questions concernant l'utilisation collective des données. L'évolution des systèmes "time-sharing" et des techniques de transmission des données a determiné l'apparition de la notion d'utilisation de l'ordinateur par plusieurs bénéficiaires. L'étape suivante sera la classification des données dans des banques de données (BD). Un système de BD sera alors caracterisé par le fait que plusieurs utilisateurs emploient à des fins différentes la même base de données.

Un modèle général de banque de données doit tenir compte des dépenses d'entretien (y comprises les dépenses liées à la collection des données), des dépenses d'accès, des pertes (dues

à la qualité des données) et, enfin des avantages qui découlent de la possibilité de connecter les données à différents fichiers.

Une attention particulière sera accordée à la mise en mémoire et au traitement dans les systèmes comprenant un grand volume de données. La notion de banques de données dépend entièrement des possibilités d'interconnexion.

КОЛЛЕКТИВНОЕ ИСПОЛЬЗОВАНИЕ ЦЕНТРАЛЬНОГО АРХИВА ИНФОРМАЦИЙ

Работа касается вопросов, связанных с использованием информаций сообща. Эволюция систем « time-sharing » (распределения времени) и техники передачи информаций привели к появлению понятия использования счетно-вычислительной машины неслокькими потребителями. Следующим этапом будет классификация информации в центральных архивах информаций. Тогда система центральных архивов будет характеризоваться тем, что несколько потребителей будут использовать ту же информационную основу для достижения различных целей.

В общей модели центрального архива данных нужно иметь в виду расходы по его содержанию (включая задания по сбору информации, расходы по доступу, потери, исходящие из возможностей разноски информаций по различным карточкам.

Будет уделяться особое внимание вопросам, связанным с запоминанием и обработкой в системах, содержащих большое количество данных. Понятие центрального архива информаций целиком зависит от возможностей согласования в работе.