Conceptual Modelling as an Instrument for Formal Specification of Statistical Information Systems

Bo Sundgren

R&D Report
Statistics Sweden
Research - Methods - Development
1989:18

Från trycket September 1989
Producent Statistiska centralbyrån, utvecklingsavdelningen
Ansvarig utgivare Åke Lönnqvist
Förfrågningar Bo Sundgren, tel. 08-783 41 48

ISSN 0283-8680
Printed in Sweden
Garnisonstryckeriet, Stockholm 1989

INLEDNING

TILL

R & D report : research, methods, development / Statistics Sweden. – Stockholm : Statistiska centralbyrån, 1988-2004. – Nr. 1988:1-2004:2.
Häri ingår Abstracts : sammanfattningar av metodrapporter från SCB med egen numrering.

Föregångare:

Metodinformation : preliminär rapport från Statistiska centralbyrån. – Stockholm : Statistiska centralbyrån. – 1984-1986. – Nr 1984:1-1986:8.

U/ADB / Statistics Sweden. – Stockholm : Statistiska centralbyrån, 1986-1987. – Nr E24-E26

R & D report : research, methods, development, U/STM / Statistics Sweden. – Stockholm : Statistiska centralbyrån, 1987. – Nr 29-41.

Efterföljare:

Research and development : methodology reports from Statistics Sweden. – Stockholm : Statistiska centralbyrån. – 2006-. – Nr 2006:1-.

R & D Report 1989:18. Conceptual modelling as an Instrument for formal specification of statistical information systems / Bo Sundgren.
Digitaliserad av Statistiska centralbyrån (SCB) 2016.

urn:nbn:se:scb-1989-X101OP8918

1989-09-15

Conceptual Modelling as an Instrument for Formal Specification of Statistical Information Systems

Bo Sundgren
Statistics Sweden
S-11581 STOCKHOLM

Paper presented at ISI 47th Session
Paris, August 29 - September 6, 1989

CONCEPTUAL MODELLING AS AN INSTRUMENT FOR FORMAL SPECIFICATION OF STATISTICAL INFORMATION SYSTEMS

Bo Sundgren
Statistics Sweden
S-11581 Stockholm

Abstract. Starting from a general, database oriented definition of 'information system' and 'statistical information system' this paper discusses the role and meaning of conceptual modelling in the design and formal specification of such systems. The paper also illustrates some graphical and other techniques for modelling a statistical information system, the statical and dynamical aspects of its object system (a piece of reality), and its four major subsystems for input, storage, transformation, and output. An algebra of conceptual operators is proposed as a precise high-level tool for formal description and manipulation of a statistical information system and its parts. The conceptual algebra could also be used as the basis for a statistical query language.

1 Statistical information systems: some basic concepts

1.1 Information systems

An **information system** may be thought of as consisting of four major parts, or **subsystems**:

1. a database, containing information (represented by data) about the status and development in a selected subsystem of the real world, the **object system**;
2. an **input-oriented subsystem**, making sure that the database properly reflects the status and development in the object system;
3. an **output-oriented subsystem**, making the contents of the database available to the users of the information system;
4. a **transformation subsystem**, transforming information in the database in accordance with requests from the other subsystems.

1.2 Statistical information systems

A statistical information system is an information system that produces aggregated, statistical information, **macroinformation** (represented by **macrodata**), as opposed to unaggregated information about individual objects, **microinformation** (represented by **microdata**).

Concepts like 'information' and 'information system' are abstract, contents-oriented entities. The corresponding concrete entities are 'data' and 'data processing system'. Thus data is the physical representation of information, and a data processing system is the technical realization of an information system.

In a statistical information system the four major subsystems have the following functions:

1. the database stores data about the object system; both microdata and macrodata may appear in the physical database;

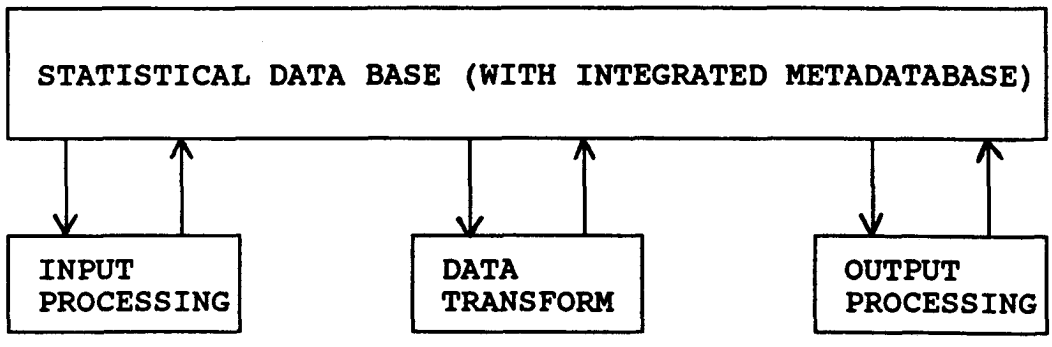
2. the input-oriented subsystem contains functions for updating the database, and for capturing, coding, and editing the input data;

3. the output-oriented subsystem contains functions for retrieving and presenting statistical data to end-users, and for initiating and presenting results of statistical analyses;

4. the transformation subsystem contains functions for transforming data in the database between different states, for example, between microdata and macrodata (aggregation), and between input-oriented data structures and output-oriented data structures; the transformation subsystem also contains functions for carrying out the algorithms of statistical analyses.

1.3 Database orientation

The above-mentioned definition of a statistical information system may be visualized in the following way:



This conceptualization of statistical information systems and statistical data processing systems may be described as database oriented, since the database subsystem is central in several respects:

- the database reflects the object system;
- all operations in all subsystems take all non-initial inputs from the database, and deliver all non-terminal outputs to the database;
- the database is also the central holding of meta-information (represented by metadata) in the information system; this "database within the database" is called the metadatabase.

However, this conceptual database orientation does not necessarily imply the use of software known as 'data base management systems'. Technically, the database could consist of a set of traditionally organized files, and the programs processing these files could be traditional statistical software.

2 Formal specifications of information systems

2.1 The needs for formal specifications and metainformation

In the previous section we defined a general structure for information systems. When we design, operate, and maintain a particular information system, there are several needs that motivate the development and maintenance of a formal specification of that particular information system, the application system, and its parts. Among other things the formal specification should serve as a basis for

- efficient communication between different categories of people, who are involved in the design, construction, operation, maintenance, and use of the information system;
- reliable documentation of the information system;
- efficient communication between the users and the information system during the operation of the information system.

Thus the formal specification should serve as a common frame of reference for different categories of people and as an interface between the users and the information system. Furthermore, the formal specification should also serve important internal needs for metadata of the information system itself. For example, most software components of the system will need formal descriptions of the files and records in the database, as well as descriptions of the mappings between the internal data representations and the external views of the data, as seen by the users.

2.2 Conceptual modelling and formal specifications of information systems

There are different methodologies for formally specifying information systems. Especially in connection with directive information systems [12] such as statistical information systems, is seems suitable to start from a conceptual model of the object system, to be reflected by the database part of the information system. This approach is often called 'conceptual modelling' or 'data modelling'.

A conceptual model can be used to fullfil many of the tasks defined above for a formal specification of an information system. It can serve as the basis for communication between users and designers of information systems. It can also serve as an interface between the contents-oriented and the technically oriented parts of an information system design and construction process. And it can be the model in terms of which the user communicates with the computerized information system.

What then is a conceptual model? The term emanates from database theory. For example, in the ANSI/SPARC three-level schema architecure for databases, the conceptual schema has the function of being a relatively invariant specification of (the contents of) the database, existing as an intermediary level in the mappings between the external schemas, reflecting dynamically changing user needs, and the internal schema, reflecting the technology-dependent, dynamically optimized hardware/software implementation of the database. One way of attaining the desirable invariance in the conceptual schema is to base it on a model of the real world, rather than on a specification of the ever changing information needs and/or data representations.

Other terms that have approximately the same meaning as 'conceptual model' are 'infological model' and 'semantical data model'. The term 'data model' is sometimes used as a synonym to 'conceptual model', but a data model is often assumed to be an abstraction of the physically existing database, more than a model of the real world represented by the database.

At Statistics Sweden conceptual modelling was introduced in the early 1970's [2]. This work was to a great extent inspired by seminal papers and books by Langefors, like [1]. On the international arena similar approaches have been presented in [3], [5], [6], and [9]. So far there is no international standard in this area, but different approaches seem to be converging. In the area of data modelling, the relational data model [7] now seems to have established itself as a de facto standard.

3 The OPR approach to conceptual modelling

3.1 Basic concepts

The Object-Property-Relationship (OPR) approach is an example of a methodology for conceptual modelling that is based on three concepts: objects, properties, and relations. With a slightly different terminology, this type of approach is often called the Entity-Attribute-Relationship (EAR) approach, or the Entity-Relationship (ER) approach.

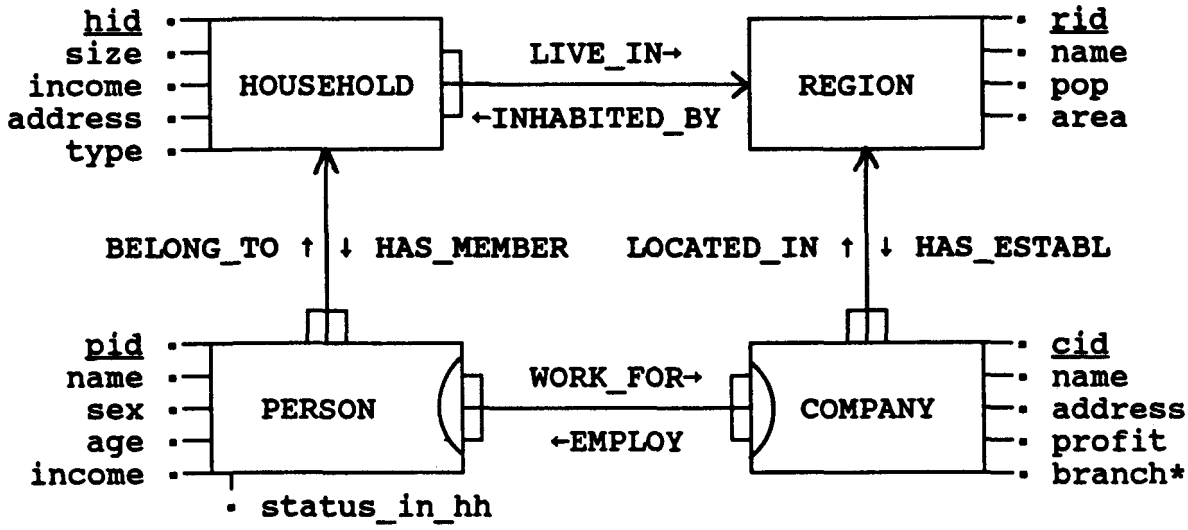
An **object** is any concrete or abstract entity (physical object, living creature, organization, event etc) that the users of the information system may be interested to have information about. Objects will always have **properties**, quantitative or qualitative. For example a person (as modelled for a certain purpose) may have an age, a home address, an income, etc. A company may have a certain number of employees, a certain legal form (like 'incorporated'), a certain economical result, etc. A traffic accident may be characterized by the time when it happened, the number of persons and vehicles involved, the properties of the road, where it took place, weather conditions, etc. For both objects and properties it is important to distinguish between **types** and **instances** (occurrences). Property types are usually called **variables** or **attributes**. A variable may also be thought of as a function between a set of object instances belonging to a certain object type and a set of values belonging to a certain **domain**. Being a function a variable should be **single-valued**, but sometimes modelling situations occur, where a variable needs to be specified as **multi-valued**. For example, one and the same company may be active in several branches of industry at one and the same time.

The logical link between an object and a property is sometimes called an **association**. Objects may also be linked to other objects. Such links are called **object relations**, or simply **relations** (relationships). Relations are often binary, that is, they link two objects to each other, but there are sometimes relations of a higher **degree**, or **dimension**. For example, a country (object 1) may export a certain commodity (object 2) to another country (object 3); thus this trade relation is an example of an object relation of degree 3.

3.2 The object graph

An object system can be visualized by means of a so-called **object graph**. An example is shown below.

In the object graph, the objects are represented by rectangles, the (object) relations are represented by lines between the object rectangles, and the variables of the objects are represented by "dots" linked to the object rectangles. An asterisk (*) after the name of a variable indicates that the variable is multi-valued. All concepts are represented on the **type** level in the object graph.



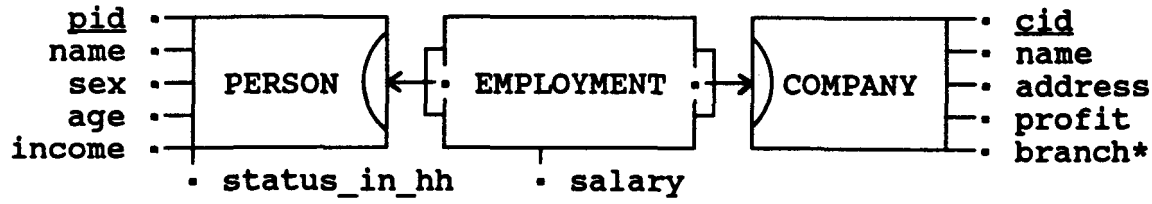
3.3 Relations: functionality, cardinality, and partiality

The object graph indicates the functionality for relations. A binary relation may be one-to-one (<--->), one-to-many (<---<), or many-to-many (>---<). Cardinality is a more general concept, which can be used also in the specification of relations of higher degree than 2.

The "half-circles" inside object rectangles in the object graph indicate partiality (as opposed to totality) of the relation visavi the object type, that is, only some (but not all) of the object instances belonging to the object type participate in the relation. For example, only some persons work for companies, and only some companies have persons employed.

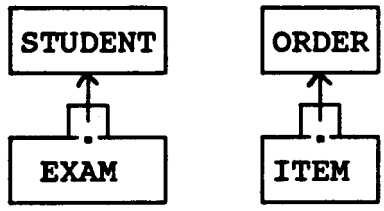
3.4 Relational objects and dependent objects

Sometimes we need to associate a property with a relation rather than with an object. For example, in the object graph above, we might like to indicate the salary that a person obtains by being employed by a certain company. This can be solved by objectifying the relation WORK_FOR/EMPLOY into a so-called relational object (an entity which is at the same time a relation and an object) EMPLOYMENT:



Relational objects are an example of dependent (or weak) objects in the sense that all instances of the object are, for their own existence, completely and essentially dependent on the existence of a particular instance of another object. Thus a particular EMPLOYMENT instance in the example above is for its existence essentially dependent on the existence of a particular PERSON instance, as well as on the existence

of a particular COMPANY instance. Other examples of dependent objects could be the EXAMs passed by a STUDENT, or the ORDER_ITEMs of an ORDER:



Note the usage of the dot symbol (•) to indicate the dependent object.

4 An algebra of conceptual operators for forming new concepts and views

It is often necessary to distinguish between the base version of a conceptual model and different views of it, containing permanent or temporary extensions and modifications that adapts the conceptual model to the needs of a particular (group of) user(s) or application(s). For example, in a statistical information system it is usually important to be able to apply both a micro perspective and a macro perspective to the same object system, and the same set of data, regardless of how the data are actually stored.

All the objects, properties, and relations in a view must be defined in terms of the concepts specified in the base version of the conceptual model. The author of this paper once outlined a conceptual (or infological) query language [4]. Here we shall show, by means of examples, how a generalized version of this language can be used for the specification of new concepts and views, when modelling a particular object system. A formalization of most of the syntax used here can be found in [15].

The different definition schemas, examplified below, for forming new objects, variables, and relations may be thought of as a set of conceptual operators that produces, in the sense of an algebra, new concepts from those which have already been defined.

4.1 Defining new object types

Example 1. Definition of a new object type:

```
HEAD_OF_HOUSEHOLD == PERSON(with status_in_hh = 1);
```

The general definition scheme for deriving new objects is:

```
<new object type> == <object type>(with <property>);
```

The <property> may in turn be a derived concept, involving boolean expressions, derived variables (see examples below), etc. The '==' symbol should be read 'equals by definition'.

4.2 Defining new variables by means of grouping, arithmetical transformation, aggregation, and adjunction

Example 2. Definition of a new variable by means of grouping:

```
PERSON.agegroup ==  1 if PERSON.age<18,
                    2 if PERSON.age>=18 and PERSON.age<66,
                    3 if PERSON.age>=66;
```

The general definition scheme for grouping variables is:

<grouped variable> == [<constant> [if <property>]]*;

where [<element>]* is a list of elements, separated by commas, and [<element>] is an optional element.

Hierarchical variables

Hierarchical variables, like (standard) classifications and codes are common in statistical information systems. They may be thought of as the result of a (multi-level) grouping. Thus the domain of a hierarchical variable is hierarchically partitioned, or classified, into groups (and groups of groups etc) of values. When a variable V is aggregated (cf Example 5 below) over an object type which is partitioned (cf 5.2) by a hierarchical variable H, aggregated V-values are typically generated for group values of H on some or all levels in the classification hierarchy.

Example 3. Definition of a new variable by means of arithmetical transformation:

```
HOUSEHOLD.income_per_capita == HOUSEHOLD.(income/size);
```

The general definition scheme for arithmetical transformations of variables is:

<new variable> == [<arithm expr> [if <property>]]*;

Note that with this definition scheme, variable grouping becomes a special case of arithmetical transformation.

Example 4. Definition of a new variable by adjunction:

```
PERSON.address == PERSON.HOUSEHOLD(via BELONG_TO).address;
```

The general definition scheme for adjoined variables is:

<adjoined variable> == <path>.<variable>;

where <path> determines a unique chain of relations from the object of <adjoined variable> to the object of <variable>. Each link in the chain has the form

<object type>[(via <relation>)].

where the 'via' part is necessary if there is more than one relation between two objects in the chain. Further it should be noted that <object type> could be derived. In particular it could be restricted by means of a '(with <property>)' clause in order to ensure that the ultimately generated adjoined variable becomes single-valued; (cf the definition of 'PERSON.main_employer' in Example 5 below).

Example 5. Definition of a new variable by means of aggregation:

```
HOUSEHOLD.size == HOUSEHOLD.PERSON(via HAS_MEMBER).count;

HOUSEHOLD.income == HOUSEHOLD.PERSON(via
                    HAS_MEMBER).sum(income);

PERSON.main_employer ==  PERSON.EMPLOYMENT(with
                         max(salary)).COMPANY.cid;
```

The effect of inserting an aggregation operator with zero (like count), one (like sum, average, max, min), or more (like correlation) arguments into a derivation is to reduce a set of values into one single value. The aggregation operator also has the effect of adjoining the aggregated variable to an object, which is on the next higher level in an aggregation hierarchy. The following question arises: what will be the object of the derived variable in the left part of the definition scheme above, if the right part contains only a base object followed by an aggregation operator? For example, what would be the object of the aggregated variable 'PERSON.count'? The most satisfactory answer to this question seems to be to regard this type of variable as being associated with the object system as such, an object that we may call 'SYSTEM'. With this conceptualization there is an implicit one-to-many relation between SYSTEM and every object type (like PERSON) in the object system.

4.3 Defining new object relations

Example 6. Definition of a new object relation:

```
PERSON.REGION(via DOMICILE) == PERSON.HOUSEHOLD(via
                               BELONG_TO).REGION(via LIVE_IN);

PERSON.REGION(via PLACE_OF_WORK) == PERSON.EMPLOYMENT(with
                                    max(salary)).COMPANY.REGION;
```

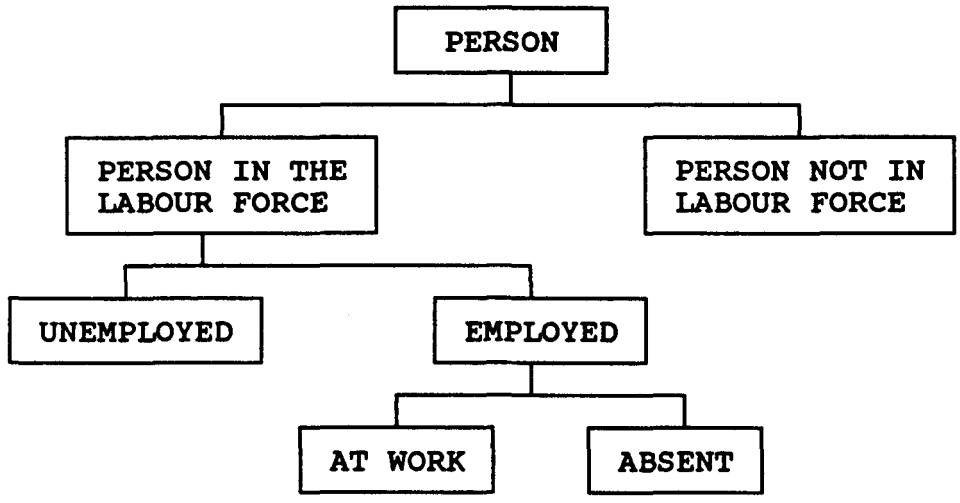
4.4 Conceptual redundance

In Example 5 above the definitions of 'HOUSEHOLD.size' and 'HOUSEHOLD.income' illustrate that even some of the concepts in the base version of a conceptual model may be derivable. Such conceptual (or infological) redundance can often be justified, but it should always be deliberately introduced and explicitly specified, since redundance may also cause problems in the design and operation of an information system.

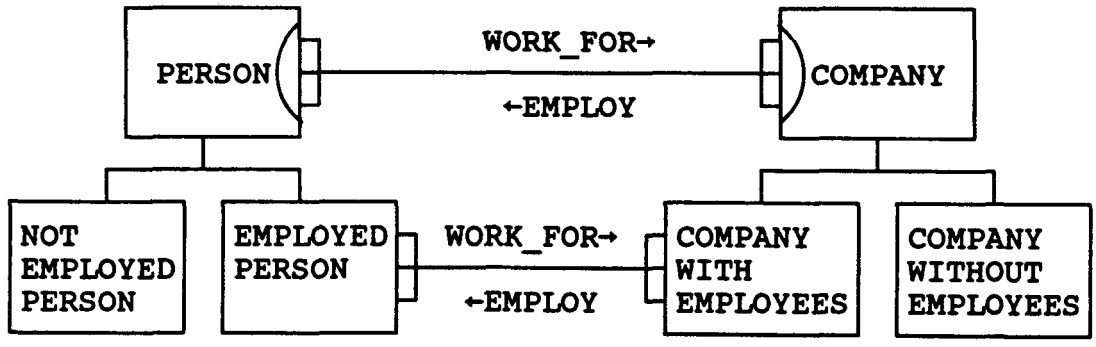
5 Further structuring of the object system

5.1 Generic hierarchies and subtyping

One important refinement of the OPR modelling approach is the introduction of generic hierarchies of object types ([8], [13]), where objects on lower levels represent subtypings of objects on higher levels, and where the lower level objects inherit the properties of the higher level objects. An example:



A related refinement is the distinction between total and partial relationships (cf 3.3 above and [13]); a relation is partial with respect to an object type, if only a proper subset of the objects belonging to the object type is involved in the relation. Partiality implies subtyping:



The example shows that by properly subtyping PERSON and COMPANY, we may change the status of the relation WORK_FOR/EMPLOY from 'partial' to 'total'.

5.2 Macroobjects: populations and domains of interest

Like many other concepts, statistical concepts like 'population' and 'domain of interest' are subject to the dualism of an extensional (set-oriented) and an intensional (type-oriented) interpretation. For example, a population may be thought of either in terms of the object instances that

belong to the population or in terms of the definition that determine which objects belong to the population, and which do not. Another dualism, typical for statistical concepts, is the micro/macro dualism when conceptualizing the object system. The parameters describing a population, or a domain of interest, may be thought of either purely as functions of the variables of micro-level objects, or they may be thought of as variables of the macro-level entities (populations, domains of interest etc), seen as objects, macro-objects, in their own right.

Statistical domains of interest are often structured in a typical way on the basis of a population (object type) and some variables of the objects belonging to the object type. A piece of aggregated statistical information like "number of households in Sweden and their average incomes by region and type of household" may be regarded as a piece of information about an object type, or a population, "household", which has been partitioned, or classified, into a structured set of domains of interest. With the macro-level approach both the population as a whole, and the domains of interest into which it has been partitioned, will be regarded as object instances (macroobjects) in their own right. The partitioned object type is often structured as an n-dimensional box, or cross-classification. In the example the box is two-dimensional, and it is spanned by the variables "region" and "type of household", both of which are properties of the micro-level objects "household". Up to dimension 3, boxes may be visualized by "real" boxes, but for boxes of higher dimensions, other modes of illustration must be chosen. An interesting graphical technique has been proposed by Shoshani [16].In its basic version this technique uses graphs with two types of nodes: **cross product nodes** (X-nodes) and **cluster nodes** (C-nodes).

We may also use the conceptual algebra, introduced earlier in this paper to specify macroobjects and information about macroobjects. With this syntax, the above-mentioned piece of information could be expressed as:

HOUSEHOLD(by region * type).(count, average(income))

The partitioning of a population into a set of domains of interest is not always a pure cross-classification. Other common structures of aggregated statistical information are concatenations and combinations of concatenations and cross-classifications. Examples:

HOUSEHOLD(by region|type).(count, average(income))

HOUSEHOLD(by region*(size|type).(count, average(income))

These examples may also be illustrated by graphs with X- and C-nodes [16].

An alternative approach would be to look upon the partitioning of a population into a multidimensional structure of

domains of interest as a generalization of the subtyping concept introduced in section 5.1.

6 Conceptual modelling of object system dynamics

As indicated by the object graph the basic version of an OPR model is essentially statical. It models the situation in the object system at a certain point of time. Later developments have introduced dynamical aspects into the OPR approach; see for example works by Jackson [14] and Malmborg [13]. Dynamical aspects are important in several ways:

1. In order to come to grips with the **definition** of a certain object type, it is often essential to penetrate the **criteria for birth and death** of objects belonging to the object type. For example: what is it that constitutes the birth (or the death) of a household, a company, etc; or similarly: what changes can an object undergo, and still remain in some sense **the same object**?

2. Births and deaths of objects in the object system, and (other) changes of states of (the variables of) the objects imply a need for **update transactions** visavi the information system and its database.

3. The users of the information system are often interested to study the **development over time** in the object system. This development can sometimes be described in terms of the situation in the object system at regular time intervals, but sometimes (for example in connection with longitudinal studies) it is necessary to be able to describe and request information about more complex chains of object-related events and changes of states.

The **consequence matrix** [12] and **life history diagrams** [14] are two examples of conceptual tools and graphical techniques that have been developed to systematize and illustrate the dynamical aspects of an object system. We shall use the following example to illustrate these tools:

6.1 Birth/death analysis

During birth/death analysis we look systematically for **birth/death events** of each one of the objects and each one of the relations in the object system. In the example above we may thus identify events like

↑ PERSON: E1 = "physical birth of person";
↓ PERSON: E2 = "physical death of person";

↑ HOUSEHOLD: E3 = "some person(s), but not all, leave a

household and form(s) a new one";

↓ HOUSEHOLD: E4 = "all persons (possibly one) leave a household and move into an existing household",

E5 = "all persons (possibly one) in a household die" (implies event E1*);

↑↓ REGION: E6 = "political/administrative decision to change the regional structure";

↑ BELONG_TO/MEMBER: E1, E3, E4,
E7 = "some person(s), but not all, leave a household, and move into another household";

↓ BELONG_TO/MEMBER: E2, E3, E4, E5, E7;

↑ LIVE/INHAB: E8 = "a household moves between regions";

↓ LIVE/INHAB: E8, E5.

6.2 Consequence analysis and the consequence matrix

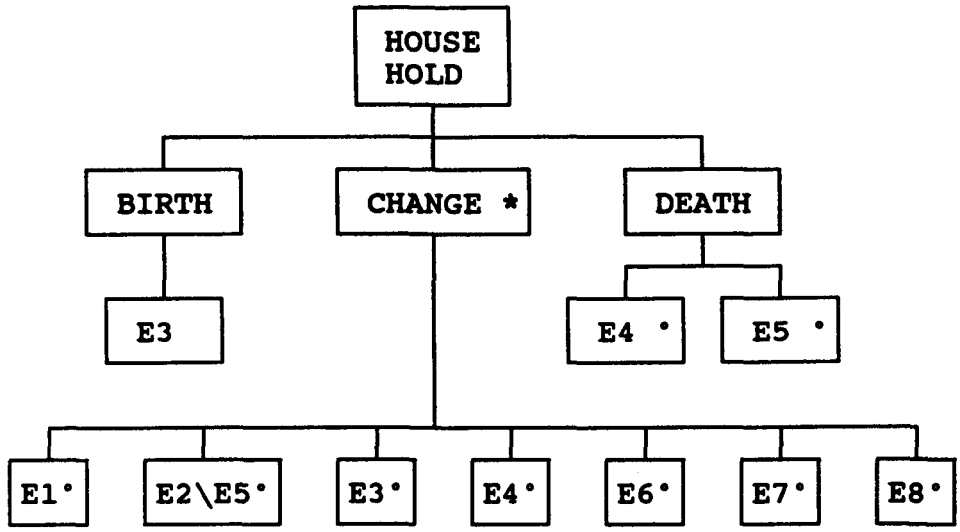
Birth/death analysis is a kind of precedence analysis [1]. We may turn it around into succedence analysis [1] by asking for all birth/death consequences of a certain event. The results of such a consequence analysis may be summarized in a consequence matrix. In the example we get:

Event	PERSON	BELONG_TO	HOUSE-HOLD	LIVE_IN	REGION
E1	↑	↑			
E2	↓	↓	↓°	↓°	
E3		↓*↑*	↑	↑	
E4		↓*↑*	↓	↓	
E5	↓*	↓*	↓	↓	
E6				↑*↓*	↑*↓*
E7		↓*↑*			
E8				↓ ↑	

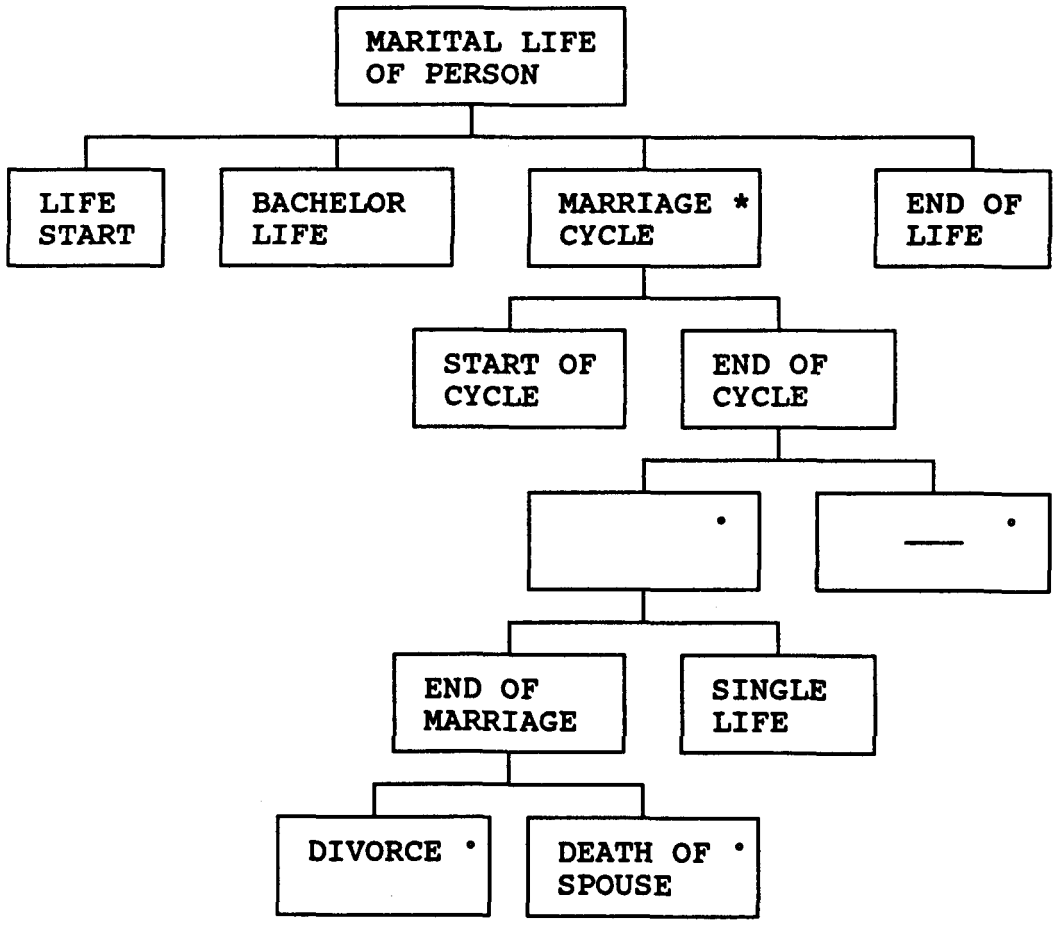
An upward directed arrow means "birth", a downward directed arrow means "death". Asterisk (*) means repetition (zero, one, or more), and a small circle (°) indicates a conditional effect. Every column should contain at least one birth event and (usually) at least one death event, otherwize we have missed something in the analysis.

6.3 Life histories and career analysis

By bringing together the information in the consequence matrix about events concerning a certain object and other objects' relations to this object, and by structuring this information properly, we can outline a typical life history of an instance of the object, as implied by the conceptual model. JSP diagrams [14] may be used to visualize life histories. In the example above we may model the life history of a household in the following way:

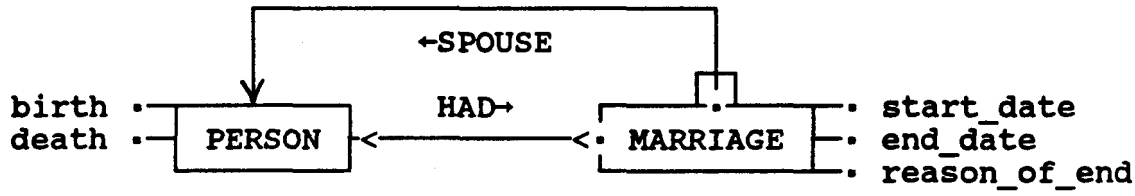


Similarly we may model so-called careers like the marital life, the education, or the criminal career of a person, the medical record of a patient, etc. Example:



In certain types of statistical information systems life histories of objects are of great interest, and as we have just seen, techniques like JSP diagrams may be used for modelling the life history of an object as a structured

flow of events associated with the object. The events that make up the life history of an object may also themselves be regarded as objects. By doing so we make it possible to incorporate the dynamical aspects of an object system in a traditional, state-oriented OPR model. This is done by defining one or more one-to-many relations between the object, whose life history we are interested in, and one or more categories of events. Time should be a mandatory property of the event objects, and a possibility to define a conceptual ordering of the instances of an object type on the basis of a property (in this case 'time') should be introduced among the conceptual tools of OPR, together with some operators (like 'first', 'last', 'next', 'before' and 'after') for referring to this ordering. For example we could model the marital history of a person like this:



And a derived variable like "a person's average time between marriages" could be expressed something like this (assuming proper handling of null values):

```
PERSON.avg_time_btw_marr == PERSON.MARRIAGE(via HAD).
                            avg(next(start_date)-end_date);
```

7 Modelling the four major subsystems of a statistical information system

We shall now discuss how conceptual modelling can be used for deriving the design of a statistical information system and its four major subsystems, and how we can do this both on a contents-oriented, infological level, and on a hardware/software-oriented, datalogical level.

7.1 Modelling the statistical database

On the infological level the specification of the database is identical with the conceptual model. If we assume that the database should be implemented under a relational database management system, or as a set of flat files managed by some suitable data manipulation language, like the Base Operator System [15] or SAS, the transformation of the infological model into a datalogical model becomes fairly straightforward, too. Basically, objects, many-to-many relations, and multi-valued variables become relational tables (flat files), and many-to-one relations become foreign key columns in the relational table corresponding to the object in the "many-end" of the relationship. This transformation into a default specification can easily be automated, but it may sometimes need to be optimized or "tuned" for efficiency reasons. For example, the designer may consider

splitting up a big relational table (by columns or by rows) into several smaller ones on the basis of an analysis of the expected transaction traffic against the database. Conversely the designer may contemplate the consolidation of several relational tables into one, in order to reduce the number of table accesses in order to respond to certain transaction types, even though this may lead to (datalogical) redundance, and thus a lower degree of normalization in the sense of the relational theory. Another important design decision on the datalogical level concerns the specification of indexes and other auxiliary structures that aim at speeding up the processing of the expected database traffic.

We have noted earlier that the conceptual model may contain (infological) redundance. For example, a certain variable may be derivable from other variables. A common situation in statistical databases is that variables of an object on a higher level of aggregation (macro-level) are derivable (by aggregation) from variables on a lower level of aggregation (micro-level). In such a situation the designer may choose to store only the microdata or both the microdata and the macrodata. Ideally a user of the database should never need to know which of the alternatives that the designer has chosen. Actually the user should have the freedom to think of the data in the database either as microdata or as macrodata, and to formulate queries accordingly, regardless of how the data are actually stored. Furthermore it is sometimes desirable that the user can think of the database in terms of (imaginary) microdata even when only macrodata are physically stored. Generally speaking, whenever there is redundance in the conceptual model, the designer has the option either to store all corresponding data, implying (datalogical) redundance, which can be used for speeding up retrieval operations, and possibly for consistency checks, or to store some corresponding data and derive others by software procedures corresponding to the definitions of redundant concepts.

Many of the datalogical design decisions mentioned in this section are dependent on a good specification of the expected transaction traffic between the database and its environment. This specification should be derivable from the conceptual model. In principle the model of the object system dynamics (cf chapter 6) implies the input-oriented traffic, and the specification of information needs (cf section 7.3) implies the output-oriented traffic, but when estimating the total transaction traffic that will hit the database, one must also take into account additional transactions generated inside the information system itself, for example in the data editing function of the input-oriented subsystem.

7.2 Modelling the input-oriented subsystem

The updating part of the input-oriented subsystem can by and large be derived from the dynamical aspects of the conceptual model, represented by the consequence matrix and/or life history diagrams.

The starting-point for modelling the input-oriented subsystem of a statistical information system could be a particular, input-oriented view of the base version of the conceptual model. This view should specify the data structures as they are conceptualized during data entry, coding, and editing. If the data is captured by questionnaires and forms, via paper and pencil, or directly through a computer terminal, the data structures are often hierarchical, corresponding to hierarchically related objects in the object system. The routing structure between the questions in a questionnaire can typically be described in the same way as a structured program [17], and thus modelling techniques known from the area of structured programming, like JSP diagrams could turn out to be useful. Different routings depending on the answer to a particular question typically correspond to different subtypes in the conceptual model (cf 5.1).

Coding can be regarded as a transformation from one domain of values to another domain for the same variable.

Theoretically the specification of editing rules becomes trivial if one has a complete conceptual model. In fact one editing rule is enough: the data should conform to the specified conceptual model. Thus checks for the validity and consistency of the input data should as far as possible be automatically derived from the conceptual model, including both its statical and its dynamical parts, rather than "invented" separately; see for example Graves [11]. In practice of course the editing problem is not quite so simple. Even if conceptual modelling is done conscientiously, it will not always be so complete that all desirable editing rules are automatically implied by the conceptual model. In particular, so-called macro-editing rules may still have to be added separately. Too many and too complex editing rules may also be generated, and the conceptual model in itself will not automatically prescribe what to do with data that do not conform with the conceptual model, for example whether it is the incoming data or some data already existing in the database that should be regarded as "wrong".

On the datalogical level it may be a fairly complex task to handle the mapping between a hierarchical, input-oriented view and the base version of the conceptual model, especially if editing should be done interactively with immediate rechecking of manually updated data. However, there are already some commercial software products like SAS and PARADOX that seem to be able to cope with this problem in an acceptable way.

7.3 Modelling the output-oriented subsystem

The output-oriented subsystem should be derived from a specification of the information needs of the users. Such a specification could be regarded as a complement to the conceptual model, and it could also be used as a tool for checking the completeness of the statical and dynamical parts of the conceptual model. Languages for specifying

information needs visavi conceptual models have been proposed, for example INFOL based on alfa-beta- and alfa-beta-gamma-analysis [2], [15].

The output-oriented subsystem of a statistical information system should be able to retrieve data from the statistical database, execute different kinds of statistical analyses, and produce tables, graphs, and other forms of presentations of statistical results. On a conceptual level many of these functions may be modelled in terms of statistical queries, or so-called alfa-beta-gamma-delta queries [15]:

α: for <object type> with <property>
β: give [<statistical operator>(<list of variables>)]*
γ: by <combination of variables>
δ: where <list of definitions>;

Example. "Number of households with more than one person, and their average income, by region and type of household":

```
for HOUSEHOLD with size > 1
give count, average(income)
by region*type
where region = REGION(via LIVE_IN).name;
```

Alternatively, we may think of the statistical query as an expression in the conceptual algebra:

<obj type>(with <prop>)(by <partitioning>).[<op>([<var>]*)]*

The same example as above could then be phrased:

```
HOUSEHOLD(with size>1)(by region*type).(count,
average(income))
```

It should be noted that all properties and variables appearing in the schemes above could themselves be expressions in the conceptual algebra. In the alfa-beta-gamma-delta scheme the delta clause offers a convenient alternative for introducing such derived concepts, and to give them, at the same time, their own names.

The graphical technique used by Shoshani [16] and others is another way of modelling statistical queries. This technique is sometimes not purely conceptual, but takes also into account table layout aspects of the query, like which variables in a cross-classification that should occur along the vertical axis, and which should occur horizontally.

By analyzing a representative set of statistical queries that are expected to hit the database, the designer could make rational decisions concerning access paths to be supported on the datalogical level. For example, variables appearing in the alfa part of an alfa-beta-gamma-delta query are candidates for being indexed.

7.4 Modelling the transformation subsystem

The transformation subsystem should be derivable from the combined specifications of the three other subsystems: different transformations are necessary to transform within and between

- data structures representing the base version of the conceptual model
- data structures representing input-oriented views
- data structures representing output-oriented views

To some extent these transformations can be described precisely, and yet on a relatively high level by means of languages such as (on the infological level) the conceptual algebra illustrated in this paper, and (on the datalogical level) the relational algebra [7], SQL [7], and the base operator algebra [15].

8 References

[1] Langefors, B: Theoretical Analysis of Information Systems; Studentlitteratur, Lund, 1966, 1973

[2] Sundgren, B: An Infological Approach to Data Bases; Urval nr 7, Statistics Sweden, Stockholm 1973

[3] Klimbie, J W and Koffeman, K I (editors): Data Base Management; Proceedings of the IFIP Working Conference on Data Base Management, North-Holland 1974

[4] Sundgren, B: Theory of Data Bases; Mason/Charter, New York, 1975

[5] Chen, P P-S: The Entity-Relationship Model - Toward a Unified View of Data; ACM TODS 1, No 1 (March 1976)

[6] Brodie, M L, Mylopoulos J, and Schmidt J W (editors): On Conceptual Modelling - Perspectives from Artificial Intelligence, Databases, and Programming Languages; Springer Verlag, New York, 1984

[7] Date, C J: An Introduction to Database Systems; Volume I, 4th Edition, Addison-Wesley 1986, and Volume II, Addison-Wesley 1983

[8] Smith, J M and Smith D C P, Database Abstractions: Aggregation and Generalization; ACM TODS 2, No 2 (June 1977)

[9] Kent, W: Data and Reality; North-Holland 1978

[10] ANSI/X3/SPARC Study Group on Data Base Management Systems: Interim Report; FDT (ACM SIGMOD Bulltetin) 7, No 2 (1975)

[11] Graves, R B: Data Base Modelling as an Aid to Data Editing; Statistics Canada and UN ECE/CES ISIS'79 Seminar, Bratislava 1979

[12] Sundgren, B: Conceptual Design of Data Bases and Information Systems; R&D Reports E19, Statistics Sweden 1984

[13] Malmborg, E: The OPREM Approach - An Extension of an OPR Approach to Include Dynamics and Classification; R&D Reports E12, Statistics Sweden 1982

[14] Jackson, M: System Development; Prentice-Hall 1983

[15] Sundgren, B: Coordination Proposals to Improve the Impact of SCP Software Development; Report (and appendices) to the UN/ECE Statistical Computing Project, Geneva 1988

[16] Shoshani, A: Statistical Databases: Characteristics, Problems, and Some Solutions; Proceedings of the 8th International Conference on Very Large Data Bases 1982

[17] Bethlehem, J G, Denteneer, D, Hundepool, A J, and Keller, W J: The BLAISE System for Computer-Assisted Survey Processing; Netherlands Central Bureau of Statistics 1987

R & D Reports är en för U/ADB och U/STM gemensam publikationsserie som fr o m 1988-01-01 ersätter de tidigare "gula" och "gröna" serierna. I serien ingår även **Abstracts** (sammanfattning av metodrapporter från SCB).

R & D Reports Statistics Sweden, are published by the Department of Research & Development within Statistics Sweden. Reports dealing with statistical methods have green (grön) covers. Reports dealing with EDP methods have yellow (gul) covers. In addition, abstracts are published three times a year (light brown /beige/ covers).

Reports published earlier during 1989 are:

1989:1 (grön) Går det att mäta produktivitetsutvecklingen för SCB? (Rune Sandström)

1989:2 (grön) Slutrapporter från U-avdelningens översyn av HINK och KPI (flera författare)

1989:3 (grön) A Cohort Model for Analyzing and Projecting Fertility by Birth Order (Sten Martinelle)

1989:4 Abstracts I - Sammanfattningar av metodrapporter från SCB

1989:5 (gul) On the use of Semantic Models for specifying Information Needs (Erik Malmborg)

1989:6 (grön) On Testing for Symmetry in Business Cycles (Anders Westlund and Sven Öhlén)

1989:7 (grön) Design and quality of the Swedish Family Expenditure Survey (Håkan L Lindström, Hans Lindkvist and Hans Näsholm)

1989:8 (grön) Om utnyttjande av urvalsdesignen vid regressionsanalys av surveydata (Lennart Nordberg)

1989:9 (grön) Variations in the Age-Pattern of Fertility in Sweden Around 1986 (Michael Hartmann)

1989:10 (grön) An Application of Generalized Precision Functions in the 1985 Swedish Family Expenditure Survey (Håkan L Lindström and Peter Lundquist)

1989:11 (gul) Statistics production in the 90's - decentralization without chaos (Bo Sundgren)

1989:12 (grön) Översyn av urvalen i de objektiva skördeuppskattningarna (Erling Andersson)

1989:13 (gul) ADBs roll i statistiken (Bo Sundgren)

1989:14 (grön) Krisgruppsarbetet och räknarexperimentet i HUT 88 **(Håkan L. Lindström)**

1989:15 (grön) On evaluation of surveys with samples from the revised Zimbabwe master sample frame **(Bengt Rosén)**

1989:16 (grön) Bortfallsbarometern nr 4 **(Sonia Ekman, Tomas Garås, Hans Pettersson, Monica Rennermalm)**

1989:17 (beige) **Abstracts II** - sammanfattning av metodrapporter från SCB

Kvarvarande **beige** och **gröna** exemplar av ovanstående promemorior kan rekvireras från Elisabet Klingberg, U/STM, SCB, 115 81 STOCKHOLM, eller per telefon 08-783 41 78.

Kvarvarande **gula** exemplar kan rekvireras från Ingvar Andersson, U/ADB, SCB, 115 81 STOCKHOLM, eller per telefon 08-783 41 47.