Analysis and design of data warehouses

Han Schouten
Information Systems Dept.
Technical University - Delft, The Netherlands
han.schouten@is.its.tudelft.nl

Abstract

With the large-scale introduction of the data warehouse concept, a new phenomenon has appeared in the field of information systems development. Facts in a data warehouse – as opposed to those in an operational database – mainly represent immutable, aggregated or otherwise derived, historical information. The aggregation level and specific layout of management information reports often cannot be specified on beforehand. Therefore, a data warehouse must be designed in such a way, that it provides optimal support for aggregation on the fly and for navigation through aggregation hierarchies, that it allows easy access to time series and that it enables reporting in any desired layout.

This publication describes the outline of two complementary methods for the analysis of data warehouse relations; one simple and the other advanced. The simple method exploits the knowledge contained in an ordinary relational schema. The advanced method is based on the analysis of derivation rules. Subsequently, the design of data warehouses based on these methods is investigated. Special attention has been given to the actuality of data warehouses that contain historical information, to the transitivity of derivations, to the navigation through aggregation hierarchies via so-called drill paths and to the maintenance of various aggregation levels within a single data warehouse relation.

Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)

Heidelberg, Germany, 14. - 15. 6. 1999

(S. Gatziu, M. Jeusfeld, M. Staudt, Y. Vassiliou, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-19/

1 Introduction

At least two different types of computer applications can be distinguished in the information management of a company: on-line transaction processing (OLTP) and on-line analytical processing (OLAP). The first supports the primary processes of the company with ordinary *transaction processing systems*. The latter concerns management information about and control over primary processes by means of *management information systems*. [Inmon93] gives several valid reasons for the physical separation of the databases and the applications in these two areas.

A database filled with facts derived and aggregated from an ordinary operational database for the sole purpose of policy making is called a *data warehouse*. The facts in a data warehouse may represent key performance indicators to be used in a balanced score card approach to evaluate the operation of a business. There are sound methods for the analysis and design of ordinary transaction processing systems. A comparable method for the development of management information systems remains to be invented. Inmon deals with many phenomena related to data warehouse design, but leaves the 'how' of it completely untouched. This article is an attempt to devise a proper way of thinking and working to achieve this goal.

When developing a transaction processing system, normally a distinction is made between the analysis of the ideas of the people in and around an organisation and the design of the way in which the operation of these ideas is supported with information technology. This distinction minimises the complexity of information analysis and guarantees generally applicable and reusable definitions. The information analysis that always precedes the design of information systems results in definitions for elementary concepts, such as *facts*, *constraints*, *derivations*, *norms* and prescriptions for desirable *human behaviour*. These concepts represent the whole communicable experience of people in and around an organisation. These concepts form the very basis of any conceivable application in the form of an information system. The definition

of these concepts is totally independent of some specific application. The designer of an information system determines the eventual form in which the facts and the guidance for the behaviour of human workers will be presented. The prescriptive qualities of a method for the analysis and the design of information systems guarantee reproducibility, verifiability and accountability.

Management information – as opposed to operational information – is always aggregated and, therefore, derived information. Updating management information is rarely necessary. Redundantly storing facts is, therefore, admissible and sometimes – for better performance – even necessary. Redundancy often concerns the registration of management information at various aggregation levels. Well-specified information requirements will often translate into moderate levels of redundancy. On the other hand, the automated support for policy development in its broadest sense often results in considerable, yet necessary levels of redundancy and extraordinarily large databases. In this sense, the structure of a data warehouse more or less depends of its intended usage.

The information in a data warehouse can be presented in a countless variety of different reports. Facts of various types can be plotted one against the other. When such facts occur within some aggregation hierarchy, policy analysts can pass through this hierarchy from the highest to the lowest level (*drill-down*). At a given aggregation level, the analyst can investigate each aggregation class individually. The data warehouse design should, therefore, optimally facilitate drill-down.

The distinction between analysis and design equally applies to the design of management information systems. During analysis, more attention than with ordinary information systems will have to be devoted to derivation rules. In a data warehouse, each relation represents a collection of derived facts grouped in a particular way. The analysis consists in the detection of derivable facts and the applicable derivations. The design consists in grouping derivable facts into data warehouse relations. The theory of grouping has been documented in [Bakema94]. In the present article, the way of grouping derived facts into datawarehouse relations is necessarily somewhat less orthodox. Analysis and design according to the prescriptions described in this article yield an optimally structured data warehouse. What we consider optimal, largely depends of the intended usage: standard report or broad analysis.

Success in the analysis of management information systems is proportional to the analytical skills of the persons in charge. A precise and complete prescription for information analysis is given in [Nijssen94]. Full treatment of this subject exceeds the scope of this article. Apart from detailing the notions of *concept*, *identity* and *meaning*, we will not dwell on these matters. These notions are necessary and sufficient for the analysis and formulation of derivation rules.

The next chapter provides two theories and two methods for the analysis of data warehouse relations. The first is our workhorse that produces the bulk of our data warehouse definitions. The second is our fancy horse and concerns the analysis of derivation rules in conceptual terms, i.e. independent of the shape in which the eventual derivation will be presented. Armed with the result of a fully-fledged information analysis, a data warehouse can be designed. The design consists in the structure of the data warehouse. We will devote special attention to evolving objects in data warehouses, to the transitivity of derivation, to drill paths and to aggregation levels. In the chapter called 'Discussion' various potentials and impracticalities of the approach presented in this article will be considered.

2 The analysis of derivations

2.1 Concepts

Natural language oriented information analysis consists in a set of prescriptions for the decomposition of natural language sentences into phrases and the classification and qualification of these phrases. Analysing sentences such as:

(1) the person with identification number 12038 is resident of The Netherlands
(2) the person with identification number 12039 is resident of Belgium

produces the following fully classified and qualified result:

```
 ST1     Nationality
SFT1     <person: ST2-OFT2> is resident of
                                  <country: ST3-OFT3>
 (1)          12038                The Netherlands
 (2)          12039                     Belgium

 ST2     Person
SFT2     <identification number: NT1-NFT1>
                                  identifies a person
 (3)          12038
 (4)          12039
OFT2     the person with <identification number: NT1-
         NFT1>

 ST3     Country
SFT2     <country name: NT2-NFT2b>
                                  identifies a country
 (5)          The Netherlands
 (6)               Belgium
OFT3     <country name: NT2-NFT2a>
```

NFT1	identification number <…>
NT1	identification number
NT2	Country name
NFT2a	the country name <…>
NFT2b	<…>

The prescription that yields this result is documented in [Nijssen94] and [Schouten94]. Moreover, [Nijssen94] provides prescriptions for recognition and modelling of various types of constraints and cases of generalisation, specialisation and synonymy. Although the method may help to recognise and analyse prescriptions for desired human behaviour, it does not provide any prescription for it. In this article, modelling derivation rules will receive full attention. Only if all prescriptions are followed meticulously, the analysis result will be reproducible and may survive the severest scrutiny.

Apparently, a sentence of the type 'Person' can be expressed in two different ways: as a *sentence formulation* according to 'SFT2' and as an *object formulation* according to 'OFT2'. The same observation applies to sentences of the type 'Country' and the formulations 'SFT3' and 'OFT3'. Many people wonder if objects are facts and the other way around. Indeed, a sentence and the reference to an object embedded in a sentence are just different manifestations of a single concept in someone's mind.

A human being is capable of communicating part of what he is aware of. The unit of information in the human conscience is called a *concept*. A concept is indivisible, has an identity of its own and has at least one meaning. By necessity, information analysis restricts itself to communicable concepts and consists in the recognition of the identity, all meanings and all manifestations of concepts that are worth considering in a given context. Among the rational and communicable concepts actually we only find *facts* and other *tenets*. A fact can be communicated as an assertion. A tenet can be communicated as a general rule expressed in some natural language or otherwise. A tenet always concerns facts of types that have been recognised as relevant to a given context.

We can expand our experience with the notion that a person is either a man or a woman and with the notion that we can also identify every Dutch citizen with a so-called 'Social Fiscal Number': the SoFi-number. Moreover, we know the date of birth and possibly the date of death of a person. This produces the following result:

CT1	Nationality	
ST1	Nationality	
SFT1	<person: CT2-OFT2> is resident of	
		<country: CT3-OFT6>
(1/1)	12038	Belgium
(2/2)	12039	The Netherlands

CT2	Person	
ST2	Person	
SFT2	<identification number: NT1-NFT1> identifies a person	
(3/3)	12038	
(4/4)	12039	
OFT2	the person with <identification number: NT1-NFT1>	
ST3	Man	
SFT3	<identification number: NT1-NFT1> identifies a person that is a man	
(7/4)	12039	
OFT3	the man with <identification number: NT1-NFT1>	
ST4	Woman	
SFT4	<identification number: NT1-NFT1> identifies a person that is a woman	
(8/3)	12038	
OFT4	the woman with <identification number: NT1-NFT1>	
ST5	Person	
SFT5	<SoFi-number: NT4-NFT4> identifies a person	
(9/4)	1482.63.152	
OFT5	the person with <SoFi-number: NT4-NFT4>	
CT3	Country	
ST6	Country	
SFT6	<country name: NT2-NFT2b> identifies a country	
(10/5)	The Netherlands	
(11/6)	Belgium	
OFT6	<country name: NT2-NFT2a>	
CT4	Point in time	
ST7	Now	
SFT7	<time value: NT3-NFT3b> identifies the current point in time	
(12/7)	18-03-1999 14:12:35	
ST8	Reference date	
SFT8	<time value: NT3-NFT3b> identifies a point in time that is a reference date	
(13/8)	01-01-1999	
(14/9)	01-02-1999	
(15/10)	01-03-1999	
(16/11)	01-04-1999	
CT5	Birth	
ST9	Birth	
SFT9	<person: CT2-OFT2> is born on <time value: NT3-NFT3b>	
(17/12)	12038	10-12-1956
(18/13)	12039	03-01-1934
CT6	Death	
ST10	Death	

SFT10 (19/14)	<person: CT2-OFT2> has deceased on 12039	<time value: NT3-NFT3b> 18-03-1999
CT7	Senior population size	
ST11	Senior population size	
SFT11	the male Dutch senior population at <time value: NT3: -NFT3a> is	<number: NT5-NFT5>
(20/18)	01-01-1999	0
(21/19)	01-02-1999	1
(22/20)	01-03-1999	0
NT1	Identification number (Categorical)	
NFT1	identification number <…>	
NT2	Country name (Categorical)	
NFT2a	the country name <…>	
NFT2b	<…>	
NT3	Time value (Interval)	
NFT3a	the time value <…>	
NFT3b	<…>	
NT4	SoFi-number	
NFT4	the SoFi-number <…>	
NT5	Number	
NFT5	a number of <…>	

The broadest class of comparable concepts is called a *concept type*. In a role definition like <country: CT3-OFT6>, CT3 identifies the type of concepts fulfilling the role 'country'. OFT6 identifies the class of similarly formed objects as well as the class of concepts with the same meaning. In the sample sentences above, the number in front of the slash identifies the sentence; the number following the slash identifies the concept. The sentences 3, 7 and 9 are three different manifestations of concept 3. Likewise, the sentences 4 and 8 are manifestations of concept 4. Belgians don't have a SoFi-number. Consequently, such manifestations don't occur. The meanings of sentences 3 and 9 do not differ necessarily. The meaning of sentence 7 is more specific than that of sentences 3 and 9.

The conjunction of several concepts is conceivable if every individual concept in the conjunction is valid. Consequently, every conjunction of concepts is divisible and derives its validity from the validity of the concepts involved. Primarily, information analysis restricts itself to elementary assertive sentences. Composite sentences are dealt with only after all elementary sentences have been analysed. A compound sentence that is a simple conjunction is divided in its elementary components; each individual component is then treated as an ordinary elementary sentence. Whenever the nature of composition is more than a simple conjunction, such as an implication or a disjunction, we have recognised a tenet that must be treated as such. Every composite sentence that is not a simple conjunction represents a *constraint*, a *derivation*, a *norm*, or a prescription of desired *human behaviour*.

We may expand our consciousness, i.e. conceive new concepts, by personal observation, by reasoning or by communication with other persons. One person does not have any means of access to the consciousness of a second person other than the second's reaction to what the first has communicated. Only if the second person's behaviour is consistent with the first's expectation on basis of what he has communicated, the latter may conclude that the other person has understood the concept communicated as intended.

In this sense, information analysis is not more than a way to specify tenets and a set of sophisticated behavioural experiments to test tenets against the experiences of the domain experts involved. A composite concept defines a tenet correctly and completely if the composite concept – given any set of valid composing concepts – is valid in the opinion of every domain expert involved.

2.2 Derived Concepts

The expectation that some set of known, independent concepts invariably leads to one particular result is expressed with a *derivation rule*. Every conceivable derivation according to this rule yields – precisely and completely – the identity and the specific meaning of a single derived concept. This section provides a notation and a method to recognise, formulate and verify logical expressions that document our tenets with regard to derivations.

2.2.1 A Notation for Rules

Any data warehouse embodies information that has been derived from some operational information source. Given a conceptual information model of the source, it is often not particularly difficult to specify the derivation rule with some derived concept type. Any formalism for logical expressions can be used for this purpose: the Peano Russell notation, existential graphs, Prolog, or SQL. Personally, I prefer a notation derived from the existential graphs of Charles Sanders Peirce [Hartsh60]. [Creasy89] has been the first to link this formalism to natural language oriented information analysis under the acronym of ENIAM. My own contribution exists in a profound theoretical and methodological foundation of the analysis of rules as part of natural language oriented information analysis [Schouten99]. Existential graphs provide the minimally required basic concepts (identification, negation, conjunction), are theoretically sound and are practically indispensable.

The logical expression in figure 1 counts the number of persons that are male Dutch citizens that have reached the age of 65 years and are alive at a given reference date and creates a concept of the type 'Senior population size' accordingly. This expression is indivisible and hence repre-

sents an elementary concept. This is a tiny yet representative example of aggregation in a data warehouse. An explanation of this figure follows below

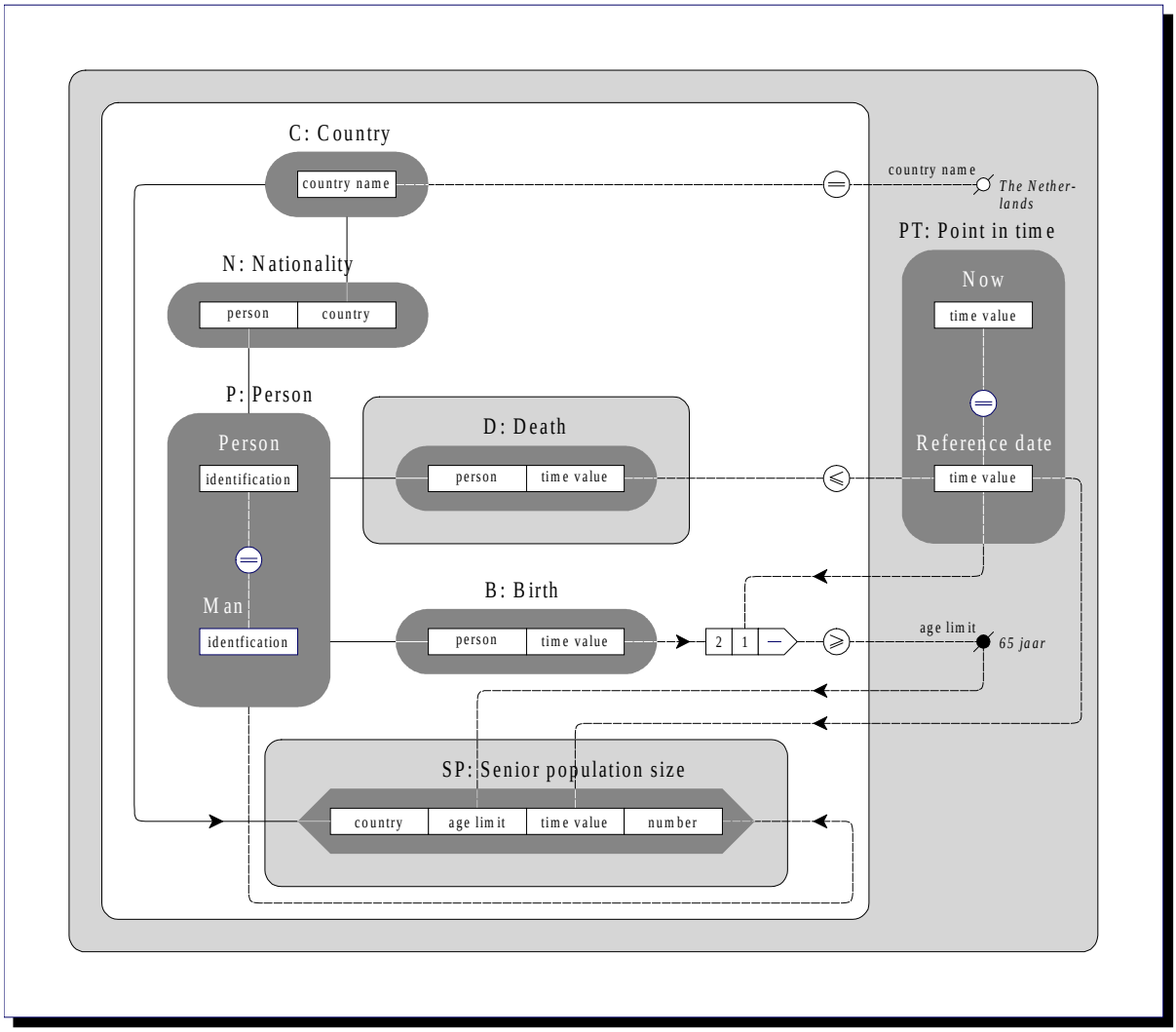

Figure 1: The derivation of the number of male Dutch citizens that have reached the age of 65 years and that are alive at a particular date

The white background with the shadow on which the graph is portrayed is the *sheet of assertion*. Every assertion placed on the sheet of assertion is true by definition. Every concept that is placed on it is valid by definition. Every combination of concepts that is placed on the sheet of assertion represents a valid conjunction.

The rectangles with rounded corners that are alternately coloured light grey and white, are called *cuts* and symbolise the negation of every expression that is placed within its boundary. If a cut is placed directly on the sheet of assertion, this means that it is true that anything placed within this cut is not true. Two cuts can never intersect.

In an existential graph, a dark grey ellipse symbolises all conceivable concepts of the given type that satisfy the conditions imposed. An alias and a concept type name accompany each concept type symbol. The alias symbolises any specific concept that can be instantiated under this concept type symbol. A white rectangle within the ellipse symbolises the role of a concept or the role of a name or a value. Several contiguous roles symbolise a sentence type with a meaning. An individual concept may have several meanings. In such a case, the concept type symbol shows as many sentence types with their names as there are meanings. A role can be lexical or non-lexical,

depending of its referent being a simple value without identity, or a concept having its own, inalienable identity.

A lexical role can be compared with some other role or some other value expression such as a computation, a user defined value or a literal constant. Whenever a concept is being created, its lexical roles receive their values from other roles and value expressions. A dashed line with an operator symbol on it symbolises a comparison. A dashed line with an arrowhead on it symbolises an assignment. The arrangement of two small squares and a pentagon with a minus sign symbolises a subtraction. The small squares symbolise the arguments. The number in each square defines the order of evaluation. The pentagon symbolises the operator as well as the result of the computation. In this case, the result of the subtraction is compared with the user-defined value of '65 years'.

There are two types of user-defined values: mandatory and optional. A mandatory user-defined value is depicted black. An optional user-defined value is depicted white. A mandatory user-defined value is only valid when it is provided with a significant value. Since a mandatory user-defined value is a prerequisite for the derivation, its symbol is always placed on an odd cut. An optional user-defined value is always true and does not influence the evaluation of a logical expression in any way. Hence, it may be placed anywhere in an existential graph.

The concept that fulfils a non-lexical role can be identified; the concept that is going to fulfil a non-lexical role can be instantiated. A solid line symbolises the identification of a concept. A solid line with an arrowhead symbolises the instantiation of a role with some concept.

The dark grey hexagon labelled 'Senior population size' symbolises the concept to be created as soon as some situation enables its creation.

We should interpret an existential graph from the outside inwards. The outer cut postulates that it is not true that a significant age limit has been specified and that there is a valid point in time 'Now' that is equal to some other valid point in time 'Reference date' and that the expression within the second cut would not be true. This is logically equivalent to saying: "If the age limit has been specified and the reference date arrives, then <some derivation possibly will take place>". The derivation itself is also a negation within a negation and can be read as an implication: "If <the conjunction of all concepts in the antecedent is true>, then create a concept of the type 'SP: Senior population size'. In this derivation, not the mere existence of some concept of the type 'P: Person' is tested, but the number of all concepts that obey the conditions imposed. A concept of the type 'Person' counts if it is a man, i.e. if it represents an element in the population of the sentence type 'Man'. Moreover, this person should be a Dutch citizen, i.e. a concept of the type 'Nationality' with this person fulfilling the role 'person' and a concept 'Country' representing 'The Netherlands' fulfilling the role 'country' must be valid. This person must also be 65 years old or older, i.e. there must be a concept of the type 'Birth' with this person fulfilling the role 'person' and the value in the role 'time value' differing at least 65 years with the reference date. Also, this person should be alive at the reference date, i.e. there should not be a valid concept of the type 'Death' with this person fulfilling the role 'person' that has a value in the role 'time value' smaller than or equal to the reference date. If all these conditions are met, nothing can stop the creation of a concept of the type 'Senior population size'. The role 'country' of this concept receives its fulfilling concept from the concept instantiated under the concept type 'Country'. The role age limit receives its value directly from the user-defined value. The role 'time value' receives its value from the reference date. The role 'number' receives its value from the number of persons counted. If the user-defined value 'country name' has not been specified, a concept of type 'SP: Senior population size' will be derived for each defined concept of the type 'C: Country'.

3 Analysing management information systems

We may become aware of a derivation in several ways. Apart from the organisation structure and prescriptions for desirable human behaviour, the implementation of organisational objectives produces a set of criteria for the degree in which the organisation fulfils its objectives. Often an organisation will produce reports at regular intervals that present management information. Whenever the organisation structure and the standard reports provide insufficient inspiration, we can survey the corporate information model for the presence of countable or measurable items.

In management science, *key performance indicators* (KPI) are popular for measuring the degree in which an organisation lives up to its *critical success factors* (CSF) [Kaplan96]. For a service company, the elapsed time between the notification of a defect and its repair is a KPI in the light of a CSC customer's satisfaction. Twenty four hour availability, is another. A data warehouse must provide an impression of the state of every applicable KPI at any given point in time. The interpretation of a set of KPI's in the light of some CSC, can be expressed as a *norm*.

Given a particular KPI, there is a simple way and a more robust, fundamental way to analyse its derivation. The simple way exists in defining elementary data warehouse relations merely on basis of weak functional dependencies. The fundamental approach defines the way in which some KPI can be derived with the help of existential graphs. One way does not exclude the other, but each is best suited to its own particular field of application. Be-

cause of their complementary nature both approaches will receive attention in this article.

3.1 The easy way

There is an easy method to capture the bulk of a data warehouse from readily available data in operational databases, provided, there exists a well-conceived relational data model. First, this model must be transformed into a graphical form that suits our purposes. When some foreign key in a table or relation references another, the relation that contains the reference is called *child* and the relation that is referred to is called *parent*. When some relation is a specialisation of another, the first is called *subtype* and the second *super type*. The subtype relation itself may be the super type of another set of subtypes. Preferably, we draw this type of diagram as follows:

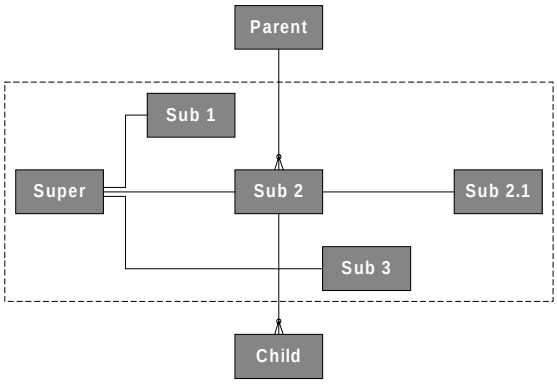

Figure 2. Search paths in a relational model.

In this diagram, the relation 'Sub 2' has one parent relation 'Parent' and one child relation 'Child'. The relation 'Sub 2' is one of the three subtypes of the relation 'Super' and has itself one subtype 'Sub 2.1'.

An attribute of a relation that represents a KPI is almost always some magnitude of the absolute, ratio or interval scale type. Every attribute of a relation is either part of the primary key to that relation, or functionally depends on it. If a relation has an alternate key, then every dependent non-key attribute also fully functionally depends on the alternate key. Apart from functional dependencies of primary and alternate keys, we may assume a weak functional dependency of some non-key attribute of every other attribute within the same relation. As far as the latter are worth considering, we call these *classifying attributes*. Very often, the strength of the dependency of some KPI on classifying attributes will be the subject of data analysis on a data warehouse.

Attributes that belong to the categorical or ordinal scale type lend themselves perfectly as classifying attributes. Attributes of the absolute, ratio or interval scale type must often be cast to some range of discrete intervals first. The latter necessarily represents some ordinal scale. If a data warehouse contains historical data, the cast of a timestamp to some ordinal scale of time periods often provides a most suitable classifying attribute.

The set of weak functional dependencies is not depleted with the non-key attributes within the same relation. If some KPI weakly depends on a foreign key, we may extend this dependency to the primary key that it references and to every attribute that functionally depends on it. Following foreign key references is – in the diagram shown above – literally a way of *look-up*. Look-up legitimately provides additional classifying attributes. Searching in the opposite direction is called *drill-down*. It is absolutely inconceivable, that some attribute would depend of attributes reached by drilling down.

Conceptually, every instance of a relation has its own inalienable identity. In the case of an instance of a super type, however, every applicable subtype relation provides attribute values that specifically belong to the meaning associated with that subtype and to that particular instance of the super type. Therefore, if some KPI weakly depends on the attributes in a particular super type or subtype relation, it weakly depends on all attributes of all applicable subtypes of the super-most relation and on all parents of these super- and subtypes.

The selection of classifying attributes determines the applicability of a data warehouse. Insufficient selectivity results in a bulky data warehouse that will be difficult to manage and hard to understand. If we restrict ourselves too much we may loose opportunities. Irrespective of how many classifying attributes we take into consideration, the combination of all classifying attributes with one dependent attribute precisely represents one elementary functional relation. The chapter on the design of data warehouses explains how such elementary relations can be grouped into useful data warehouse relations.

3.2 A more advanced approach

Some KPI's don't exist as such in an operational database and must be derived. Sometimes, the logical expression for such a derivation is simple. Often, however, its analysis and verification requires huge efforts. Then, a perfect way of working and modelling pays for itself.

An organisation invariably derives the criteria for its performance from facts obtained from its 'shop floor'. Domain experts can decide which concepts assist in achieving a particular derivation result and how. Analysts can help to find the proper logical expression to describe the derivation.

Existential graphs have proven to be a tremendous help in formulating and verifying the exact nature of our beliefs.

In an existential graph that expresses our belief concerning the derivability of some concept, the concepts that trigger the derivation always appear in conjunction in the first cut. The concepts that provide classifying variables and the derived concept always appear in conjunction in the second cut. Every classifying concept will fulfil an independent role in the concept to be derived. A simple conjunction may not produce the required result in the case of optional dependent concepts, such as 'Death'. Here, the non-existence of a concept of this type is required and a simple negation suffices. In other cases an embedded implication, equivalence or non-equivalence is called for.

4 Designing management information systems

4.1 Design considerations

A management information system provides for the derivation of management information obtained from the ordinary course of affairs in a company, its storage and its use. The logical expression that documents a derivation rule is the precipitation of a tenet concerning this course of affairs. A logical expression is itself a composite, yet irreducible concept. We may state the following:

> *A composite concept defines a belief correctly and completely if this is – without any reservation – conceivable itself in the experience of an arbitrary domain expert given an arbitrary set of composing elementary concepts that are conceivable themselves.*

We may expect the same quality of the implementation of any belief in a management information system. All operations performed on operational data during and after loading it into a data warehouse, should fully maintain the logical coherence that we have recognised during analysis. Deviation from this principle would damage the credibility of the system and could seriously endanger the management of a company.

A conceptual information model including derivation rules describes conceivable concepts in their most elementary form. In this form, no consideration whatsoever has been implied of the way in which concepts will be stored or presented, let alone a way that guarantees optimal performance.

The relational model provides an optimal strategy for storage and retrieval of information. This model clusters a variety of elementary concepts, in which another concept fulfils a unique role, around this concept and stores it as such. Such a cluster of concepts is called a *tuple* or *row*. A relational database only provides lexical references to concepts. The notion of concept type only exists in the name of the *relation*, *tuple type* or *table* that acts as a container for similar tuples. This way of storing information prevents information redundancy and that is an essential instrument for the maintenance of information integrity. Moreover, clustering improves query performance.

Data warehouse design leaves us greater freedom. A data warehouse derives its consistency mainly from the consistency of the source data. Customarily, data in a warehouse are never changed. Consequently, the fifth normal form is no longer crucial to information integrity. Better performance can be obtained by:

1. Cluster elementary concepts that represent dependencies of KPI's on similar ranges of classifying attributes;
2. Aggregating KPI's in a clustered relation over all values of a classifying attribute or combinations of these.

To enhance understanding and usage of a management information system, the correspondence between roles in the data warehouse and roles in the operational database must be known and documented.

4.2 Designing the data warehouse

The analysis of derivation rules may have resulted in a rather large number of logical data warehouse relations. Irrespective of the way in which these relations have come about, these mainly represent *functions*, i.e. dependencies of a single, possibly derived variable on a set of classifying variables. The purpose of data warehouse design is, to cluster as many as possible elementary, functional relationships into the smallest possible number of composite data warehouse relations.

Mainly, there are two reasons for clustering elementary data warehouse relation into one composite relation:

1. The sets of classifying attributes are equal;
2. The set of classifying attributes of one data warehouse relation is a subset of the other.

The first reason is the ordinary clustering of elementary relationships according to similarities in primary keys. Very often, a derivation rule is a *function*: the derivation applied to a particular combination of independent classifying variable values produces a single dependent variable value. By definition, the instantiation of a function is an elementary sentence. Elementary sentences sharing the same set of independent variables can be clustered into a composite relation. In a data warehouse, the set of independent classifying attributes represents its primary key and identifies each object residing in it.

The second reason can be understood by considering the semantics of existential graphs. If a role of a concept in an existential graph is unbound, its referent does not influence its evaluation in any way. If such a role is connected to an optional connector via a line of comparison and that

connector is not provided with some significant value, then this role does not influence the evaluation process either. If the dependent variable in a data warehouse relation represents an aggregation (sum, average, etc.) of some variable, an undefined classifying variable results in the aggregation over all values that it assumes in that relation.

Grouping is not allowed if two elementary relations represent derivations that logically exclude each other. This may be the case, if both derivations concern different specialisations of the same concept type. E.g., the relation that represents the result of counting the number of male persons like the one in figure 1 can not be grouped with a similar relation for women, just like that. We must establish, first of all, that either two different units of counting are concerned, or that we must add a classifying attribute 'Gender' to the composite data warehouse relation. With the first solution, the nature of the population is interlaced with the name of the derived attribute, e.g., 'Number of men' and 'Number of Women'. The second solution requires a semantically equivalent schema transformation. Originally, the gender that was implied in the specialisations 'Man' and 'Woman' of the concept type 'Person'. Now, the gender will be expressed explicitly in a binary concept 'Gender' with a role 'person' and a role 'expression of gender' with two possible values 'male' and 'female'. This solution is demonstrated in figure 3. The latter being more elegant, we will henceforth assume this solution. If the gender is not specified, all persons that meet the other conditions imposed will be counted.

In a data warehouse relation, an unspecified classifying variable means that each aggregation has been performed over all values of that variable. In this respect, a data warehouse relation differs noticeably from its relational counterpart in OLTP. The relational model according to [Codd90] prohibits undefined values for primary key at-

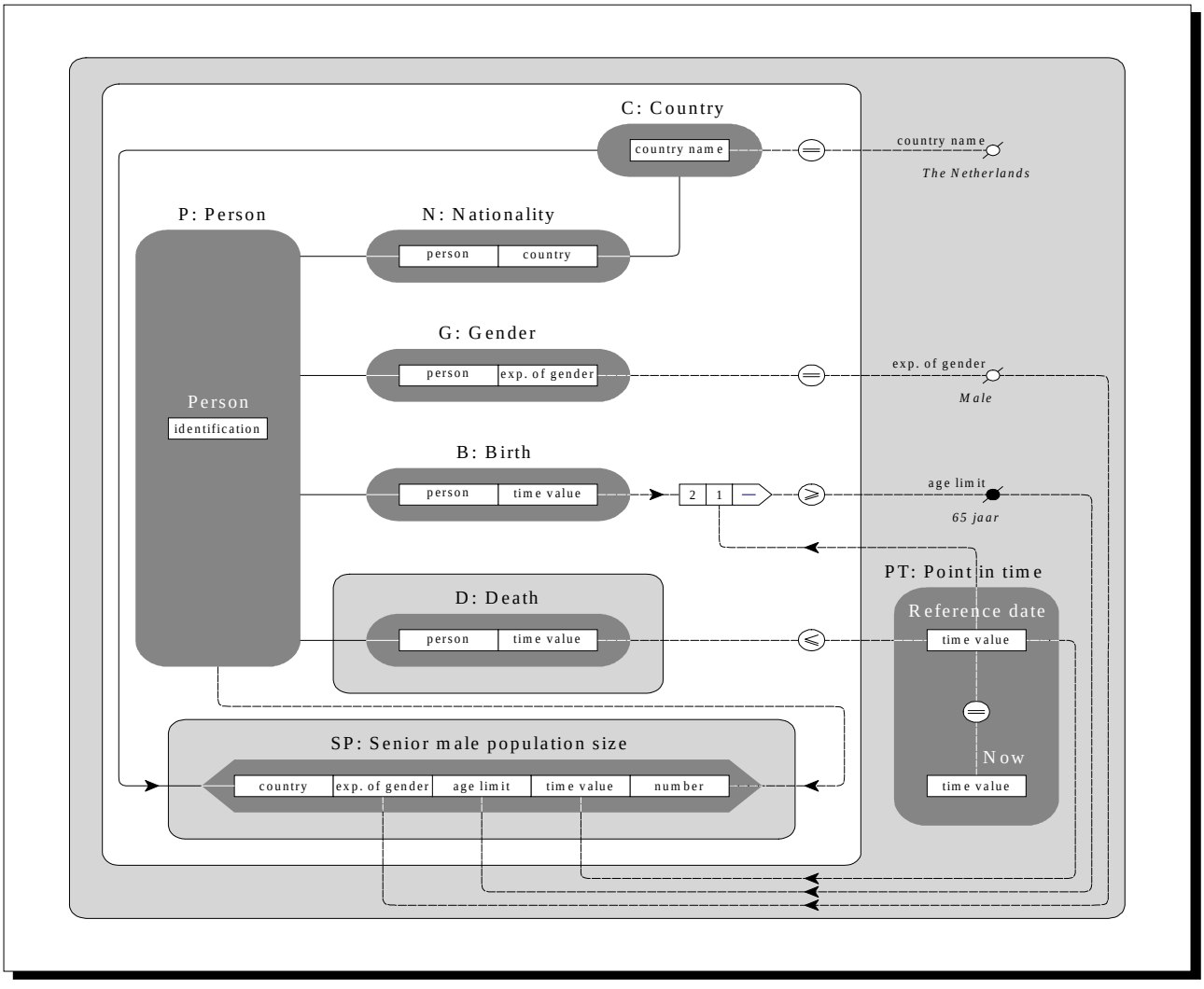

Figure 3. A semantic equivalent of the derivation in figure 1.

tributes. An undefined independent variable value in a data warehouse relation has an unambiguous, well-understood meaning and is entirely admissible.

So far, we have recognised two sources of elementary data warehouse relations: weak functional dependencies in operational databases and derived concepts produced by the invocation of a derivation rule. Irrespective of its source, we may visualise a data warehouse relation as a black box with connectors; connectors symbolising classifying variables to the left of the black box and connectors symbolising dependent variables to its right. In this way we can rapidly investigate large numbers of different scenarios.

Figure 4 depicts the same functional dependency as the one represented by the fact type derived in figure 3, now as a 'black box'.

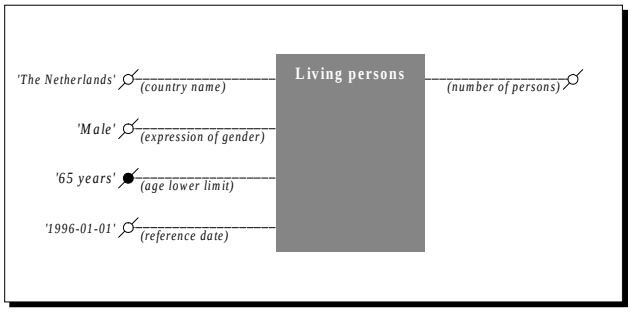

Figure 4. Counting the number of living persons

Figure 5 shows an elementary relation that expresses the dependency of the average income of living persons on a similar set of classifying variables.

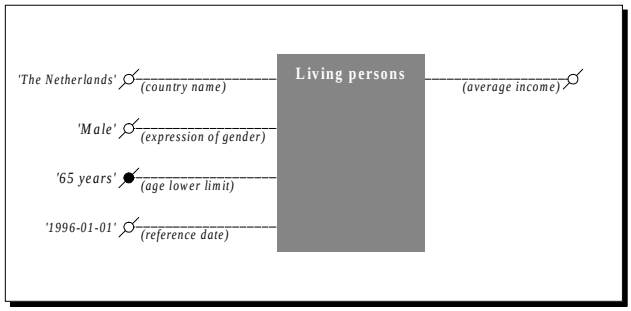

Figure 5. Calculating the average income

Given an identical set of independent variables, the black boxes in figures 4 and 5 can be united within a single black box, as is shown in figure 6.

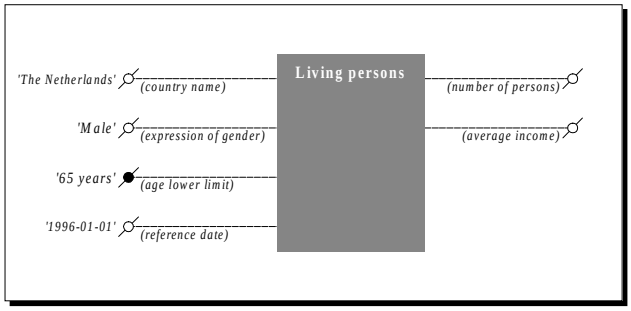

Figure 6. The number of persons and their income

Considering the meaning of unbound roles in aggregations, we unite all elementary relations in which independent variables depend on fully identical sets of independent classifying variables, or subsets thereof, into composite relations. The result is a set of data warehouse relation in which the combination of all independent variables represents the key to several dependent variables. Because of the degree of freedom we experience in clustering elementary data warehouse relations into composite ones and because every conceivable aggregation level can be maintained within a single relation, a data warehouse usually consists of a few relations only.

4.3 Transitive derivability

In a transitive derivation, at least one independent variable is itself derived. An average is always a transitively derived variable; the sum and the number of observations from which the average are calculated both represent derived variables. When the object counted is the same as the object of summation, a single black box suffices. However, when the object counted differs from the object summed up, e.g. the tax administration dividing the total amount spent for verifying tax declarations in some region by the number of taxable persons living in that region in order to obtain some measure of efficiency, two black boxes that share some or all of the independent variable with equal values are required. Such derivations demand a particular order of evaluation. Figure 7 shows such a scenario.

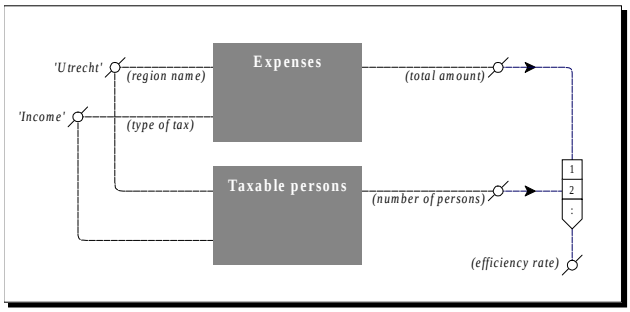

Figure 7. Transitive derivation.

4.4 Evolving data warehouse objects

If the history of the evolution of concepts in a database is registered, timeline attributes represent classifying variables that can be treated accordingly. Only dependent variables that have been modified between the last and the before last database extraction, will be considered for updating the data warehouse. To prevent counting unchanged dependent values more than once, only those dependent variables that have changed may be registered in the newly created data warehouse tuple.

Otherwise, if no history is maintained in the database, the moment of database extraction may provide a suitable timestamp. Here, the specific dependency of a dependent variable from its classifying variables in the database must be compared with its equivalent in the data warehouse. Series of changes between two successive extractions will go unnoticed, however. The classifying variable in this case can be a simple timestamp, or a derived status variable that expresses a stage in the lifecycle of the data warehouse object at the moment of extraction. Here also, only modified dependent variables may be registered, in order to prevent counting the others more than once.

Selecting several related data warehouse objects, e.g., to calculate the elapsed time between two successive extractions, may be difficult in some data analysis applications and near impossible in others. If such a variable is relevant, it is good practice to calculate it at the moment of extraction and store it as a dependent variable in the data warehouse object concerned.

4.5 Classifying variables and Drill paths

Some data warehouse relation may be the result of looking up classifying attributes at several successive levels of parent relations. If the foreign key attributes have been included as classifying attributes, aggregation of dependent variables over those foreign key attributes, provides clean sums, counts, averages and so on for a particular class within the enveloping parent category.

Such a hierarchy of embracing categories represents a useful means to traverse a data warehouse relation from the highest aggregation level to the lowest, or to perform *drill down*. Such hierarchies are called *drill paths*.

If no other provisions are made, the nature and origin of classifying variables in a data warehouse relation are lost. Therefore, in a data warehouse, the knowledge that several classifying variables represent a drill path, must be maintained explicitly.

4.6 Aggregation levels in the data warehouse

The designer of a data warehouse can define a single data warehouse relation that may hold every conceivable aggregation level. An undefined classifying variable in a data warehouse tuple means that every derived variable in the same tuple represents the aggregate of all individual dependent variables that occur in conjunction with a defined value of the classifying variable at a lower aggregation level. In this way, all aggregation levels of a data warehouse object – ranging from the most elementary level to the highest aggregation level – can be unambiguously maintained within a single data warehouse relation.

An aggregated value is best associated with several other derived values that describe the original population and that facilitate further aggregation, whenever needed. The number of observations from which the aggregate has been derived must be known. Also maintaining the lowest and the highest observed value and the median of all observations may come in handy; after derivation such information may get lost. Next to that, the sum and the sum of squares of all observations must be stored at least. In this way, the mean, the variance and the standard deviation can always be calculated. When more recent observations are added to a data warehouse relation, the new set of aggregates can be computed without having to recur to the original values.

At the level of elementary observations, by definition, the number of observations equals one and the sum equals the value observed.

5. Conclusions

This article describes the outline of an approach for the analysis and design of data warehouses. As such, a data warehouse is the database of a management information system. Where [Inmon93] just explains *what* must be done, this article explains *how* this can be achieved. The approach in this article aims at procedural precision, completeness and prescriptiveness. As an outline it may not be perfect in every respect. At its very best, it is a guide for practical application that leaves ample room for further research and improvement. An educationally sound publication about this subject would rather fill a book and would not fit into an article such as this.

The analysis and design of a data warehouse based on principles derived from information theory offers several remarkable and possibly new perspectives. A data warehouse relation may be considered as a set of grouped elementary functions sharing the same set of classifying variables or a subset thereof. The largest set of classifying variables serves as the primary key to the data warehouse relation. If a classifying variable has an undefined value, all dependent variables represent aggregates of original observations over all tuples that feature a defined value of that classifying variable. In this respect, a data warehouse relation differs essentially from a relation according to [Codd90] in which undefined primary key attributes are inadmissible.

Every aggregation procedure can be described completely and correctly with a logical expression – possibly in the

form of an existential graph – provided, the identity and meaning of the concepts involved is known and well understood. Apart from the derived concept, an existential graph that represents a derivation rule shows concepts that provide either classifying variables or dependent variables or concepts that link these together. The derived concept usually features one role, whose value is the result of counting, accumulation, averaging, or any other computation and that functionally depends on the conjunction of all classifying variables distinguished in the derivation. The excellent qualities of existential graphs such as simplicity, readability and logical completeness make them indispensable for the analysis of derivation rules.

The special meaning associated with undefined classifying variables in data warehouse relations offers the designer several degrees of freedom. He may sensibly combine several aggregation levels of a data warehouse object in a single data warehouse relation. This helps to prevent a combinatory explosion of conceivable aggregation levels. Likewise, successive stages in the life cycle of a data warehouse object along with all relevant data can be registered and exploited in a single data warehouse relation.

Transitive derivability occurs and requires attention. The data warehouse designer is compelled to consider a particular order of derivation.

The fancy horse and the workhorse portrayed in this article have proven to reinforce each other rather than compete with one another. Looking-up classifying variables in a hierarchy of enveloping relations provides a valuable first scan. The rule-based approach helps us to identify and investigate more intricate derivations.

References

[Nijssen94] **Nijssen G.M., Schouten H.** *Matemataal voor bedrijfseconomie en bedrijfskunde*. PNA Publishing, Beutenaken, 1994. ISBN 90-5540-002-5.

[Hartsh60] **Hartshorne C., Weiss P. (editors).** *Collected papers of Charles Sanders Peirce.* Volume IV: *The simplest mathematics.* Book II: *Existential graphs.* Pages 293 - 464. Harvard University Press, Cambridge, Massachusetts, USA, 1960.

[Creasy89] **Creasy P.** *ENIAM – A more complete conceptual schema language.* Proceedings of the fifteenth international conference on very large databases, Amsterdam. (1989).

[Schouten94] **Schouten H.** *The rules of the game – How to formally specify the NIAM information analysis method.* Working papers of the Second International NIAM Conference, Albuquerque, New Mexico, USA, 1994.

[Inmon93] **Inmon W.H.** *Building the Data Warehouse.* John Wiley & Sons, Inc.1993. ISBN 0-471-56960-7.

[Bakema94] **Bakema G.P., Zwart J.P.C., Lek H. van der.** *Fully communication oriented NIAM.* Working papers of the Second International NIAM Conference, Albuquerque, New Mexico, USA, 1994.

[Codd90] **Codd E.F.** *The relational model for database management – Version 2.* Addison-Wesley Publishing Company, Inc., 1990. ISBN 0-201-14192-2.

[Kaplan96] **Kaplan R.S., Norton D.P.** *Using the balanced scorecard as a strategic management system.* Harvard Business Review, January-February 1996.

[Schouten99] **Schouten, H.** *A repository for existential graphs.* Journal of Conceptual Modeling, April 1999. http://www.inconcept.com/jcm