Conceptual Data Warehouse Design

Bodo Hüsemann, Jens Lechtenbörger, Gottfried Vossen
Institut für Wirtschaftsinformatik
Universität Münster, Steinfurter Straße 109
D-48149 Münster, Germany
bodo.huesemann@uni-muenster.de
{lechten,vossen}@helios.uni-muenster.de

Abstract

A *data warehouse* is an integrated and time-varying collection of data derived from operational data and primarily used in strategic decision making by means of online analytical processing (OLAP) techniques. Although it is generally agreed that warehouse design is a non-trivial problem and that multidimensional data models and star or snowflake schemata are relevant in this context, hardly any methods exist to date for deriving such a schema from an operational database. In this paper, we fill this gap by showing how to systematically derive a conceptual warehouse schema that is even in *generalized multidimensional normal form*.

1 Introduction

A *data warehouse* is generally understood as an integrated and time-varying collection of data primarily used in strategic decision making by means of online analytical processing (OLAP) techniques. It is essentially a database that stores integrated, often historical, and aggregated information extracted from multiple, heterogeneous, autonomous, and distributed information sources. Although it is generally agreed that warehouse design is a non-trivial problem and that multidimensional data models and star or snowflake schemata are relevant in this context, hardly any methods exist to date for deriving such a schema from an operational database. In this paper, we recall the notion of *multidimensional normal form (MNF)* [LAW98] as a means to describe "good" warehouse schemata, and we show how to systematically derive a conceptual warehouse schema in MNF from a given operational schema.

Traditional database design methods [BCN92, Vos99] structure the design process into a sequence of phases or steps. It is common to start with *requirements analysis and specification*, then do *conceptual design*, then *logical design*, and finally *physical design*; this broad description can be refined in a variety of ways. As a central issue during a design process, a *conceptual* schema for the database under design is established which is then transformed into the "language of a *logical* data model as the basis for physical implementation. For data warehouses, no corresponding methodology is yet in sight, although there is a large body of literature that discusses how to derive schemas based on intuitive principles. We remedy this situation by presenting a method for conceptual warehouse design that is in line with traditional database design, and that fits into a modeling process which follows classical approaches. An important point for our exposition is that traditional design methods have clearly defined goals and objectives, such as completeness w.r.t. a coverage of the underlying application, minimality of resulting schemata, freedom of redundancies, readability, etc. Some of these requirements can even be made formally precise. Most notably, relational *dependency theory* provides insight to understanding the reasons for redundancies in database relations, and formalizes normal forms and normalization as a way to avoid them; moreover, there are algorithmic approaches (such as the synthesis algorithm [Vos99]) for constructing normalized schemata.

For two reasons, the above is in remarkable contrast with what can be called warehouse design. First, current practice in data warehousing and its applications marks a radical departure from the principles of normalized schema design. Indeed, a common understanding of a "well-

Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000)
Stockholm, Sweden, June 5-6, 2000
(M. Jeusfeld, H. Shu, M. Staudt, G. Vossen, eds.)
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-28/

designed" warehouse schema is that the schema under consideration should have the form of a "star", i.e., it should consist of a central *fact table* that contains the facts of interest to an OLAP application, and that is connected to a number of *dimension tables* through referential integrity constraints based on the various dimension keys. Since dimensions can be composed of attribute hierarchies, it is often the case that dimension tables are unnormalized, and their normalization results in what is known as a *snowflake* schema. As an example, the conceptual schema shown in Figure 1 may be perceived as a conceptual snowflake schema that corresponds to the operational schema of Figure 2. Second, the research community has not been paying too much attention so far to (a) developing complete design methods for data warehouses in general or for conceptual design in particular, and (b) establishing guidelines for good schema design or integrity constraints within the context of multidimensional models. As a result, there appears to be a discrepancy between traditional database design as applied to operational databases, and the design principles that apply to data warehouses.

We remedy this situation by presenting a phase-oriented design process for data warehouses that is modeled after the traditional design process, where we use an example from the banking domain as our running example. Concretely, we will show how to obtain the warehouse schema shown in Figure 1 from the conceptual operational database schema in Figure 2. The most important phase, the phase of *conceptual design*, has to sort out *dimensions*, corresponding *dimension hierarchies*, and *measures*, and has to determine which attribute from the underlying database(s) belongs where. Our contribution here is threefold: First, we establish guidelines for answering the question of whether an attribute is a dimension level or a property attribute. Second, we propose a graphical formalism for conceptual warehouse design that captures this distinction in an appropriate way. Third, we show how *generalized multidimensional normal form (GMNF)*, originally proposed in [LAW98], can be obtained for a warehouse schema under design.

The organization of this paper is as follows: In Section 2, we briefly recall the database design process, and we review previous approaches on warehouse design and normal forms for warehouse schemata. We then introduce necessary terminology in Section 3, before we propose and illustrate our approach to conceptual warehouse design in Section 4. Afterwards, we show in Section 5 that our design method does indeed produce schemata in GMNF. Finally, concluding remarks and plans for future research appear in Section 6.

2 Related Research

Database design, i.e., the task of mapping a given real-world application to the formal data model of a given database management system, is ia well-understood process; following [BCN92], database design can roughly be seen as being done in four steps, with the result of a database schema which can be processed by a database management system. This process comprises the phases *requirements analysis*, *conceptual schema design*, *logical schema design*, and *physical schema design*.

Concerning *data warehouse design*, there is a general agreement that at least a conceptual or logical modeling activity should precede the actual implementation [WB97, AGS97, CT98, GMR98]. Typically, the modeling activity is based on a multidimensional model (see [BSHD98, VS99, PJ99] for comparisons of various multidimensional models), whereas the implementation is carried out either within relational or multidimensional databases [CD97]. However, most of these models were developed without an embedding into a design process and thus without guidelines on how to use them or what to do with the resulting schemata. Notable exceptions in this respect are the approaches of [CT98] and [GR98], which we summarize next.

In [CT98] a design method is presented that starts from an existing E/R schema, derives a multidimensional schema, and provides implementations in terms of relational tables as well as multidimensional arrays. The derivation of the multidimensional schema is structured into the steps (1) identification of facts and dimensions, (2) restructuring of the E/R schema, (3) derivation of a dimensional graph, and (4) translation into the multidimensional model. In the first step, facts along with their measures have to be selected, and afterwards dimensions for a fact are identified by navigating the schema. Then, in the second step, the initial E/R schema is restructured in order to express facts and dimensions explicitly, thus arriving at a schema that can be mapped to the multidimensional model. From this point on, the remaining steps provide natural translations of the E/R schema into an multidimensional schema and then into a target database schema.

The work of [GR98] presents a complete warehouse design method which resembles the traditional database design and consists of the following steps: (1) analysis of the information system, (2) requirement specification, (3) conceptual design (following the method of [GMR98]), (4) workload refinement and schema validation, (5) logical design, (6) physical design. We note here that the design of a conceptual schema is carried out by producing a so-called *fact schema* for each fact, which, following [GMR98], can be derived from an E/R schema using an algorithmic procedure, which starting from facts navigates through the schema along x-to-one relationships in order to determine dimensions and their hierarchies.

Concerning these two approaches, we critically remark the following. In [CT98], conceptual and logical design are mixed, and a "logical" multidimensional model is presented. We argue (in accordance with, e.g., [WB97]) that

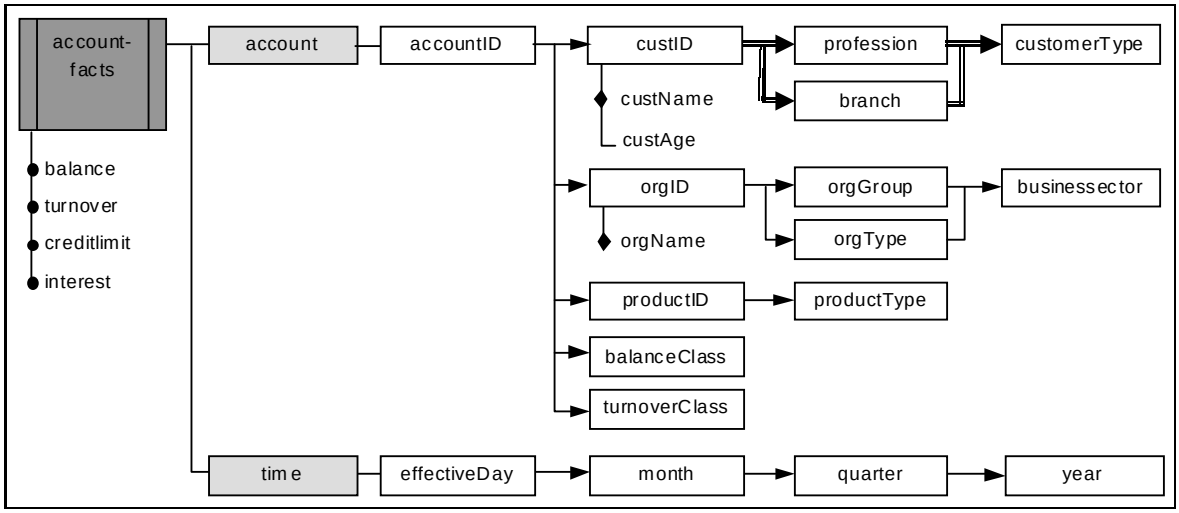

Figure 1: Conceptual Multidimensional Schema.

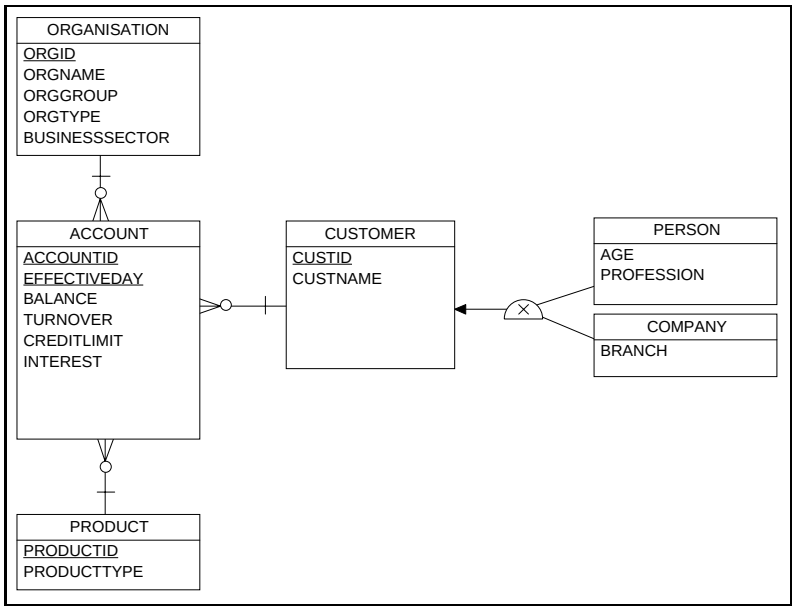

Figure 2: Sample Conceptual Operational Schema.

the conceptual design should be independent of the target database system, whereas the logical design should be tailored towards the chosen target system (here: relational or multidimensional database system). As a consequence, what is called logical design in [CT98] should actually be called conceptual design instead, and the implementation activity of [CT98] should be divided into a logical and a physical design phase, as in [GR98]. Moreover, the crucial first two steps of [CT98] are presented by examples only, and remain rather vague. In contrast, [GR98] contains an algorithmic procedure to perform the corresponding activities. However, neither approach provides any formal justification for the applied method, and nor does it supply criteria according to which schema quality could be measured.

In summary, well-established design phases for multidimensional databases are still missing, and while normal forms have a long tradition in the area of relational databases as guidelines for good schema design, research on quality factors for multidimensional database schemata seems to have started only recently. Indeed, the work presented in [LS97] can be regarded as a first step in this respect. The authors argue that *summarizability*, i.e., guaranteed correctness of aggregation results, is of most importance for OLAP queries. Hence, any multidimensional schema should be set up in such a way that summarizability is obtained to the highest possible degree. Moreover, if summarizability is violated along certain aggregation paths then the schema should express this constraint clearly. Finally, [LS97] provides criteria that imply summarizability.

Following and extending this line of thought, [LAW98] defines the *multidimensional normal form* (MNF) of multidimensional schemata, which guarantees summarizability within one-dimensional contexts. However, [LAW98] illustrates by an example taken from [Leh98] that the multidimensional normal form might be too restrictive for certain application scenarios and defines the *generalized multidimensional normal form* (GMNF). As argued in [LAW98], a schema in GMNF ensures summarizability in a context-sensitive manner and supports an efficient physical database design. However, so far there is no design method that guarantees multidimensional normal forms.

3 Terminology and notation

We briefly describe a graphical notation for conceptual multidimensional data modeling next. To avoid misunderstandings resulting from the variety of terminology in multidimensional databases, we first clarify some terms and then use these throughout the rest of the paper.

3.1 Basic multidimensional structures

Facts represent atomic information elements in a multidimensional database. A fact consists of quantifying values stored in *measures* and a qualifying context which is determined through *(terminal) dimension levels*. Each dimension level contains a set of instances or *elements*. An *aggregation path* is a subsequence of dimension levels, which starts in a terminal dimension level and ends in an implicit dimension level `All` containing a single element <`all`>. Given an aggregation path $(d_1,\ldots,d_n)$, where d_i is a dimension level, $1 \le i \le n$, we assume that every element of d_i *belongs to* (or is associated to) at most one element of d_{i+1}, $1 \le i \le n-1$; moreover, we say that d_{i+1} is a higher (aggregation) level than d_i. A dimension is structured in terms of one or more aggregation paths that share the same terminal dimension level. Therefore, each dimension comprises at least one terminal dimension level and the implicit level `All`. For example, a dimension `time` might consist of the single aggregation path (`day, month, year`), where the element <2000/03/29> of dimension level `day` is associated to element <2000/03> of level `month`, which in turn belongs to element <2000> of level `year`. In slight abuse of terminology, the graph consisting of all aggregation paths for a given dimension is called *dimension hierarchy* (although a dimension level might have more than one parent level). We assume that sets of dimension levels of distinct dimensions are disjoint.

A *property attribute* describes additional information related to a dimension level (e.g., the property attribute `custName` for dimension level `custID` in Figure 1). An *optional* property attribute (e.g., `custAge`) needs not be specified for each element of the corresponding dimension level and therefore may contain <null> values. A property attribute can be used to delimit the resulting set of a multidimensional query, but does not determine its aggregation level.

A *fact schema* represents the dimensional context for a set of facts that share the same terminal dimension levels; we use graphical schemata such as the one shown in Figure 1 to express fact schemata. Note that we avoid the term *fact table* during conceptual design, as "table" suggests logical storage in (relational) tables, which concerns logical design decisions only.

3.2 Dimension hierarchies

Dimension hierarchies are classified into two basic types. A *simple* hierarchy consists of exactly one linear aggregation path within a dimension (e.g., path `day`→`month`→`year` in dimension `time` of Figure 3).

A *multiple* dimension hierarchy contains at least two different aggregation paths in a dimension (e.g., the dimension `account` of Figure 3). A group of aggregation paths rooted in a common dimension level d is called *alternative*, if every element of d belongs to exactly one element of each higher level (e.g., the paths of Figure 3 starting in dimension level `orgID`). Moreover, we allow *optional* groups of aggregation paths. Here, we assume that some

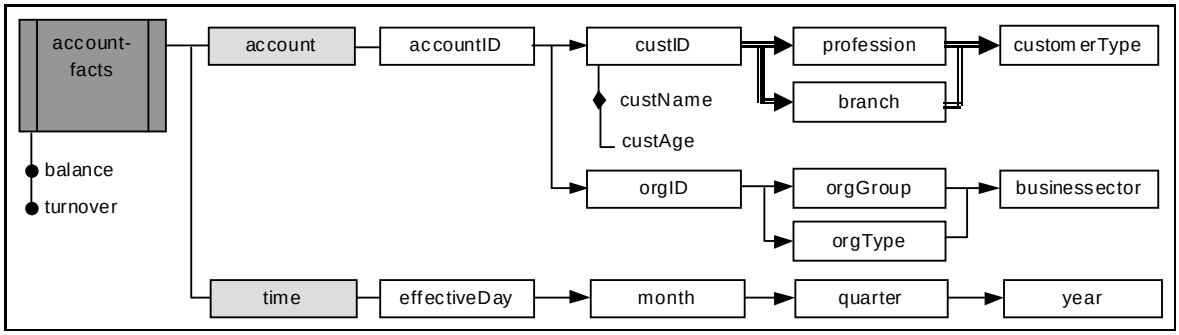

Figure 3: Simplified Fact Schema `account facts` with measures {`balance, turnover`} and dimensions {`account, time`}.

dimension level d_l has two or more higher levels such that for every element of d_l there is exactly one element in a higher level to which that element belongs. For example, in Figure 3 the group of aggregation paths starting in dimension level `custID` is optional, and each element of dimension level `custID` is related to either a `profession` or a `branch`); within an optional group of aggregation paths a dimension level such as `custID` is called *split level* whereas `customerType` is called *join level*.

In our graphical notation, simple hierarchies and alternative groups of aggregation paths are indicated by simple arrows, pointing from the lower dimension level to the higher one. Optional groups of aggregation paths are specified by double lined arrows. A mandatory property attribute is connected via a diamond to its dimension level, and an optional property attribute without.

Intuitively, aggregations along different aggregation paths in an alternative group yield the same result. However, if a group is optional then aggregations along different paths within the group represent (partial) aggregation results for mutually disjoint subsets of the elements of the terminal dimension level.

We point out that if some element of a dimension level does not belong to an element of some higher level then all data related to that element is lost at the higher aggregation level, resulting in erroneous query results. To avoid such partial mappings we propose the insertion of an element <`other`> into the higher dimension level. After that we allocate every nonrelated element of the lower level to the element <`other`> of the higher one.

4 Data Warehouse Design

In this section, we propose an approach to data warehouse design that resembles the traditional database design process. In Section 4.1 we briefly sketch the overall approach, then we present the conceptual design phase in detail in Section 4.2; we refer to [Hue99] for more information on the remaining phases.

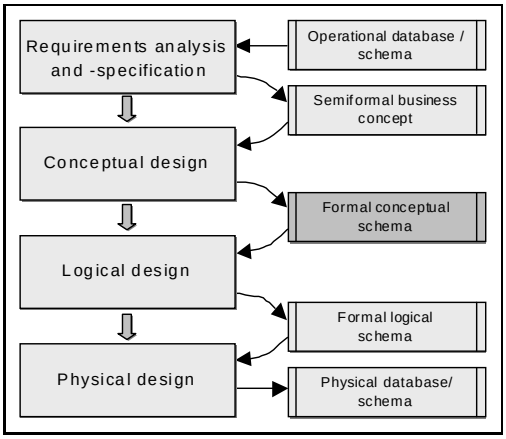

Figure 4: Process Model For Data Warehouse Design.

4.1 A process model for Data Warehouse design

The data warehouse design process comprises four sequential phases just like the classical database design process. In Figure 4 these phases along with their respective input and output are shown.

Requirements analysis and specification

The operational E/R schema delivers basic information to determine the multidimensional analysis potential, where we assume that such a schema is available to the warehouse designer (if it is not, it could be obtained, for example, through techniques of reverse engineering as described in [FV95]). In this phase business domain experts select strategically relevant operational database attributes and specify the purpose to use them as dimensions and/or measures. For each attribute it is necessary to decide whether it contains optional data or not. Furthermore, additional complementary requirements in the form of complex derived measures are added. The resulting requirements specification contains a tabular listing of attributes along with their multidimensional purpose. Supplementary informal information (such as relevant business integrity constraints) can

be added in a textual appendix, which also includes typical strategic queries.

Conceptual design

The conceptual design phase performs a transformation of the semi-formal business requirements specification into a formalized conceptual multidimensional schema. The formalization results in a graphical multidimensional diagram, which comprises fact schemata with their related measures and dimension hierarchies. For each measure of a fact scheme the summarizability constraints are formalized in an tabular appendix. In Section 4.2 we propose a phase model to derive such diagrams and appendices in a straightforward way starting from the requirements specification.

Logical design

The logical design phase converts the conceptual schema to a logical one with respect to the target logical data model (mostly relational or multidimensional). The logical schemata are generated according to transformation rules, which *only* refer to the developed conceptual diagrams and summarizability constraints.

Physical design

The data warehouse design process ends in a physical implementation of the logical schemata with respect to the individual properties of the target database system, including physical optimization techniques such as commonly known indexing strategies, partitioning etc., as well as OLAP-specific adjustments like relational denormalization of dimension tables, pre-aggregations or special use of bitmap indexes.

4.2 Conceptual Data Warehouse Modeling

As we mentioned already, the aim of the conceptual schema design phase is to produce a graphical multidimensional schema, which for each measure expresses its multidimensional context in terms of relevant dimensions and their hierarchies.

We assume that a global operational E/R schema such as the one shown in Figure 2 exists, which describes the available source information. Moreover, we assume that requirements analysis has been carried out together with domain experts, and that the global operational E/R schema has been analyzed to determine interesting measures, dimensions, and initial OLAP queries. The output of this phase consists of (a) tables such as the extract concerning account information shown in Table 1 (which contains an informal description for each relevant attribute and indicates whether the attribute may be used as measure or dimensional attribute and whether the attribute is optional or not) and (b) standard multidimensional queries such as "show the average turnover by year and product group" or "how many current accounts are managed online."

We note that Table 1 contains additional attributes that represent multidimensional requirements which are not part of the operational schema (e.g., {`month`, `customerType`}). In particular, the new dimension level `customerType` is introduced in order to represent the specialization hierarchy concerning different types (e.g. business and private) of customers as shown in Figure 2. These attributes extend operational data and augment multidimensional aggregation possibilities according to multidimensional analysis needs. Further additional attributes are obtained by an examination of those attributes that are specified both as measure and dimension. In our case, we note that related to business management it may not be useful to query an aggregation level on discrete `balance` and `turnover` values. Thus, we define new dimension levels `balanceClass, turnoverClass` that represent analysis intervals; moreover, the corresponding original attributes {`balance, turnover`} are then rated as non-dimensional.

A process model to conceptual data warehouse design

We subdivide the process phase of conceptual data warehouse design into three sequential phases:

1. context definition of measures,
2. dimensional hierarchy design, and
3. definition of summarizability constraints.

Context definition of measures

Given the set $M = \{m_1, \ldots, m_k\}$ of measures defined during requirements analysis and the set D of dimensional attributes, every fact can be perceived as an element of a graph of some *function* from dimension levels to measures. Hence, the conceptual design phase starts by determining *functional dependencies* (FDs)[1] from dimension levels to measures. First, we determine a (minimal) key $D_i \subseteq D$ for each measure m_i; then we define the set F_{Key} as consisting of all FDs of the form $D_i \rightarrow m_i$ so obtained. Thus, given an FD $D_i \rightarrow m_i \in F_{Key}$, the dimension levels in D_i functionally determine measure m_i, but are not functionally determined by any other level. Hence, they qualify as terminal dimension levels that are used as roots of dimension hierarchies. For each terminal dimension level we define a corresponding dimension.

In our running example, we hence obtain the FD (`accountID, effectiveDay`) $\rightarrow$ `balance` $\in F_{Key}$ for measure `balance`. Furthermore, all measures m_i, m_j with

[1] Since functional dependencies are a well-known concept in relational databases, we assume the reader to be familiar with it; otherwise see, for example, [BCN92, Vos99].

Table 1: Requirements Specification.

Attribute	Description	M	D	O
effectiveDay	data import date	no	yes	no
month	time aggregation	no	yes	no
quarter	time aggregation	no	yes	no
year	time aggregation	no	yes	no
accountID	account key	no	yes	no
balance	balance at effective day	yes	no	no
balanceClass	balance classification	no	yes	no
turnover	turnover at effective day	yes	no	no
turnoverClass	turnover classification	no	yes	no
creditlimit	creditlimit of the account	yes	no	no
interest	interest rate	yes	no	no
custID	customer key	no	yes	no
custName	costumer name	no	yes	no
custAge	age of a private customer	no	yes	yes
customerType	classification of customers	no	yes	no
profession	profession of a private customer	no	yes	yes
branch	branch of a business customer	no	yes	yes
productID	product description	no	yes	no
productType	classification of products	no	yes	no
orgID	attending organizational unit	no	yes	no
orgName	name of a organizational unit	no	yes	no
orgGroup	grouping of organizational units	no	yes	no
orgType	classification of organizational units	no	yes	no
businessector	classification of orgGroup and orgType	no	yes	no

$D_i = D_j$ are grouped into the same *fact schema*, as they share the same dimensional context.

Table 2 shows the result of this process applied to measures `balance`, `turnover`, `creditlimit`, and `interest` occurring in Table 1. All measures are functionally dependent on the same terminal dimension levels, namely `accountID` and `effectiveDay`, which in turn belong to the dimensions `account` and `time`, resp. Hence, all measures are grouped into a common fact schema `accountfacts`. At this point, we start the *graphical* conceptual design by modeling fact schemata up to terminal dimension levels (see Figure 5).

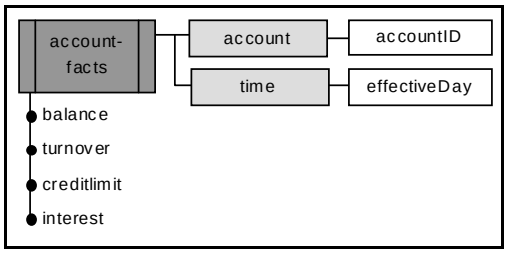

Figure 5: First Part of the Conceptual Schema.

Dimensional hierarchy design

In the next step, we gradually develop the dimension hierarchies for each dimension. To this end, we determine all FDs between dimension levels belonging to a dimension *dim* with terminal dimension level d_j as follows: Suppose we are given dimension levels $d_k, d_l \in D$ s.t. $d_k \rightarrow d_l$ is a valid FD and there exists a (potentially transitive) functional dependence of d_k on d_j, then we add $d_k \rightarrow d_l$ to the set F_{dim}.

In our example, fact schema `accountfacts` includes the `accountID` and `effectiveDay` as terminal dimension levels. Starting with terminal dimension level `effectiveDay` of dimension `time` we determine the following FDs:

$$F_{time} = \{\texttt{effectiveDay} \rightarrow \texttt{month}, \texttt{month} \rightarrow \texttt{quarter}, \texttt{quarter} \rightarrow \texttt{year}\}$$

Graphically, we derive the simple dimension hierarchy shown in Figure 6.

Table 2: Functional Dependencies between Terminal Dimension Levels and Measures.

Fact schema	Measure	Dimension	Terminal dimension level
accountfacts	balance, turnover, creditlimit, interest	account	accountID
		time	effectiveDay

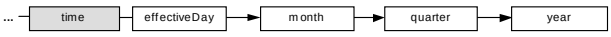

Figure 6: Simple Dimension Hierarchy `time`.

The dimension levels of dimension `account` exhibit the following FDs:

$F_{account}$ = { `accountID` → `orgID`,
`accountID` → `custID`, `accountID` → `turnoverClass`,
`accountID` → `balanceClass`,
`accountID` → `productID`,
`productID` → `productType`,
`orgID` → `orgGroup`, `orgID` → `orgType`,
`orgType` → `businessector`,
`orgGroup` → `businessector`,
`orgID` → `orgName`,
`custID` → `profession`, `custID` → `branch`,
`profession` → `customerType`,
`branch` → `customerType`,
`custID` → `custName`, `custID` → `custAge` }

In the following, we use the dimension `account` (see Figure 7) as running example for the derivation of complex dimension hierarchies. In a first step, property attributes and dimension levels have to be distinguished according to analysis requirements (recall that property attributes may only be used for selections but not for aggregations). For example, `orgName` represents a property attribute, since aggregations according to `orgName` are meaningless to business analysts. Similarly, `custName` and `custAge` are identified as property attributes, whereas all remaining dimensional attributes represent dimension levels.

Next, in a second step, a rough approximation of the dimension hierarchy is obtained by building a directed graph whose nodes are dimension levels. This graph contains an edge from dimension level d_i to level d_j, if $d_i \neq d_j$ and $d_i \rightarrow d_j$ is a non-transitive FD, i.e., if $d_i \rightarrow d_j$ holds and there is no dimension level d_k ($d_k \neq d_i, d_j$) such that $d_i \rightarrow d_k \rightarrow d_j$ holds.

The graph obtained so far is now augmented with property attributes: Property attribute d_p is attached to dimension level d_l if the FD $d_l \rightarrow d_p$ is non-transitive. The information whether a property attribute is optional or not can be retrieved from the requirements specification.

Finally, multiple dimension hierarchies have to be checked whether they contain optional groups of aggregation paths or not. If all dimension levels are mandatory according to the requirements specification then all groups of aggregation paths are alternative. If, however, some dimension levels are optional (such as `profession` and `branch` in our example) then these levels introduce groups of optional aggregation paths. Assume that d_l is a (mandatory) split level that functionally determines the optional dimension levels $d_{c_1}, \ldots d_{c_k}$. These optional levels are now grouped by building disjoint subsets of $\{d_{c_1}, \ldots d_{c_k}\}$ such that the elements of levels in each group form a complete and disjoint partitioning of the elements of split level d_l.

In our example, `custID` is the split level, and {`profession`, `branch`} are the optional dimension levels which have to be grouped. Clearly, each element of `custID` belongs either to `profession` (in case of a private customer) or to `branch` (in case of a business customer), which implies that the aggregation paths from `custID` to `profession` and `branch` form an optional group of aggregation paths. Moreover, `customerType` is the join level for this group as `customerType` is functionally dependent on `profession` and `branch`. We point out that in general, however, (a) multiple groups may arise and (b) it may be necessary to introduce new optional levels in order to ensure completeness for each group. Furthermore, we require that each group has a join level whose elements indicate for each element of the split level to which optional level of the group it belongs. For example, if `customerType` contains the elements <`business customer`> and <`private customer`> then each customer, i.e., each element of `custID`, belongs to `branch` if the `customerType` is <`business customer`> and to `profession` otherwise. Thus, `customerType` is a join level that satisfies our requirements. In general, it might be necessary to introduce such a join level explicitly.

Definition of summarizability constraints

As already argued in, e.g., [LS97, GMR98, PJ99], not all possible aggregations of measures within a certain application scenario make sense in general. For example, given a group of customers, summing over as well as taking averages of account balances may be perfectly reasonable; similarly, the computation of average ages may be reasonable,

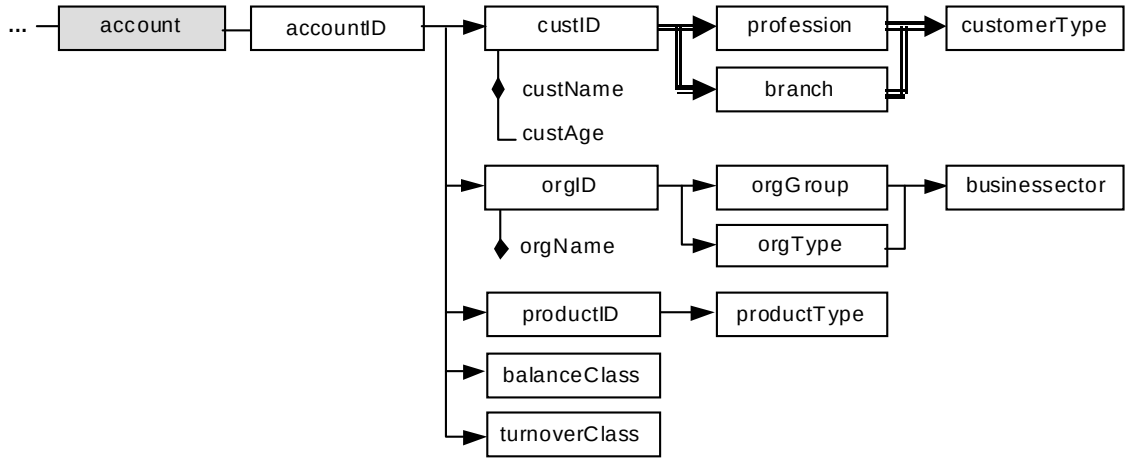

Figure 7: Multiple Dimension Hierarchy `account`.

whereas the sum of customer ages will probably be meaningless. Consequently, a conceptual model should provide means to distinguish *meaningful* aggregations of measures from *meaningless* ones, as this information helps analysts in formulating their queries. In particular, the warehouse schema should express explicitly which measure may be aggregated along what dimension hierarchy according to what aggregation function. Clearly, we could integrate this information into the graphical fact schemata by connecting each pair of measures and terminal attributes by an edge that is labeled with all meaningful (or, alternatively, by all meaningless) aggregate functions. However, for the schema shown in Figure 1 we would already have to add at least 8 edges, thereby reducing readability. Hence, we refrain from representing meaningful aggregations within graphical schemata, but propose to use *summarizability appendices* as described next.

Along the lines of [PJ99] we distinguish four degrees of increasing restriction levels for measures within the context of dimension levels as shown in Table 3. Given a pair (m,d) of a measure m and a dimension level d, we associate restriction level 1, if all aggregate functions may be applied to roll-up m from dimension level d to every functionally dependent higher level. Restriction level 2 is associated to those pairs (m,d), where all aggregate functions but the SUM-Operator are applicable (e.g., age information along the customer dimension as explained above). The restriction level 3 represents the highest limitation, where aggregation is still possible, but only in terms of counting (e.g., for constant measures). Finally, degree 4 states that no aggregation function is permissible.

Given the graphical, conceptual schemata so far, the next step is to define restriction levels for all measures along the different aggregation paths in every fact schema: For each pair of measures and dimension levels we define a restriction level such that every multidimensional query based on the allowed aggregation functions along every path is meaningful. We declare a chosen restriction on a given dimension level implicitly valid for all dependent higher dimension levels. This does not exclude the possibility to define further constraints with greater restriction level on higher dimension levels.

The results of this process are arranged within a summarizability appendix such as the one shown in Table 4, which contains the restriction levels for measures in the fact schema of Figure 1.

The measure `balance` is additive in the dimension level accountID, so we assign restriction level 1 to it for all aggregation paths starting in `accountID`. In contrast summing up `balance`-values in the dimension level `effectiveDay` is meaningless. Nevertheless the analysis of average values or other statistical characterizing quantities makes sense, so restriction level 2 is allocated for the dimension level `effectiveDay`.

5 Discussion

Compared to the approaches [CT98, GMR98], our analysis of FDs corresponds to the "schema navigation" performed in those approaches. However, we start by an identification of measures and proceed algorithmically from there on, whereas [CT98, GMR98] start by a rather vague identification of facts. Moreover, our method is justified by multidimensional normal forms as we will show next.

First, we recall terminology related to multidimensional normal forms from [LAW98], suitably adapted to our framework. We point out that [LAW98] distinguishes weak FDs (which amount to partial functions) from non-weak ones (denoting total functions), whereas we (equivalently) make use of FDs in the usual sense but distinguish optional from mandatory attributes.

A *dimensional schema* is a set of dimensional attributes D where for all $d_i \in D$ there is $d_j \in D \setminus \{d_i\}$ such that we

Table 3: Classification of Restriction Levels.

Restriction level	Applicable aggregate functions
1	{ SUM, AVG, MIN, MAX, STDDEV, VAR, COUNT }
2	{ AVG, MIN, MAX, STDDEV, VAR, COUNT }
3	{ COUNT }
4	{ }

Table 4: Summarizability Appendix for Fact Schema `accountfacts`.

Fact schema	Measure	Dimension levels	Restriction level
account facts	balance	accountID	1
		effectiveDay	2
	turnover	accountID	1
		effectiveDay	1
	creditlimit	accountID	2
		effectiveDay	2
	interest	accountID	2
		effectiveDay	2

have an FD of the form either $d_i \to d_j$ or $d_j \to d_i$. A *multidimensional schema* is a pair $M = (\{D_1, \ldots, D_k\}, S)$, where $\{D_1, \ldots, D_k\}$ is a set of dimensional schemata and S is a measure that is functionally determined by the attributes occurring in $\{D_1, \ldots, D_k\}$. A dimensional attribute $d_t \in D$ is *terminal*, if there is no $d \in D \setminus \{d_t\}$ such that $d \to d_t$. A dimensional attribute $d \in D \setminus \{d_t\}$ is a *category attribute* if d is mandatory and there is $d' \in D \setminus \{d_t, d\}$ such that $d' \to d$ or d' is mandatory and $d \to d'$; all other attributes are *property attributes*. Let d_t be a terminal attribute, d_p a property attribute, and d_c a category attribute of a common dimension. An element c of d_c is a *context of validity* of d_p if (a) for each element of d_t belonging to c there is a value for d_p and (b) for each element of d_t not belonging to c there is no value for d_p.

A dimensional schema D is in *dimensional normal form* if (a) there is exactly one terminal attribute $d_t \in D$, (b) the elements of d_t are complete (i.e., all real-world concepts are captured), and (c) all dimensional attributes are mandatory. A multidimensional schema $M = (\{D_1, \ldots, D_k\}, S)$ is in *generalized multidimensional normal form (GMNF)* if the following conditions are satisfied: (1) For each property attribute $d_p \in D_i$ there is an element of a category attribute $d_c \in D_i$ denoting the context of validity of d_p. (2) Each dimensional schema restricted to category attributes is in dimensional normal form. (3) The dimensions are orthogonal to each other, i.e., there are no FDs among attribute from distinct dimension schemata. (4) Measure S is full functionally determined by the set of terminal attributes of the dimensions.

Now we are ready to relate our approach to the concepts introduced in [LAW98]:

Lemma 5.1

1. *Each dimension hierarchy determined during conceptual design is a dimensional schema that is rooted in exactly one terminal attribute.*
2. *The dimensions are orthogonal to each other.*
3. *In dimensions without optional hierarchies all dimension levels are mandatory.*
4. *If a dimension contains an optional hierarchy then the elements of the join level represent contexts of validity for property attributes.*
5. *All measures are full functionally determined by the set of terminal dimension levels of the dimensions.*

Using this lemma, we can show the following:

Theorem 5.1 *The conceptual design described in Section 4 produces fact schemata in generalized multidimensional normal form.*

It is instructive to note that the distinction of property and category attributes in the sense of [LAW98] is not necessary in order to characterize schemata in GMNF.

Theorem 5.2 *A fact schema is in GMNF if the following conditions are satisfied.*

1. *For each optional dimension level d_l in dimension d there is an element of a mandatory higher level d_c denoting the context of validity of d_l.*

2. *Each dimensional schema restricted to mandatory attributes is in dimensional normal form.*

3. *The dimensions are orthogonal to each other.*

4. *All measures are full functionally determined by the set of terminal dimension levels of the dimensions.*

6 Conclusions

In this paper, we have outlined an approach to conceptual warehouse design which incorporates new guidelines to address specific warehouse needs and which can naturally be embedded into the traditional database design process. In this respect, the generalized multidimensional normal form can be regarded as a quality factor that avoids *aggregation anomalies* (similar to the avoidance of update anomalies by traditional normal forms for operational databases) and is therefore of most importance to guarantee correct query results.

Previously, the method proposed in [GR98] seems to have been the only existing approach that covers a complete data warehouse design starting from source integration and requirements analysis, very much in resemblance of traditional database design. We believe that their overall approach is well-suited to data warehouse design, since the separation of design phases is essential to any design process, within the context of relational databases or otherwise. For example, it is the requirements analysis that allows to identify facts, measures, and typical queries, which are basic ingredients for the following design steps. In contrast, starting with the conceptual design the approach [CT98] indicates only that facts and dimensions have to be identified and selected but contains no hints as how to achieve this.

Concerning further details on logical schema design, the reader is referred to [Hue99], which focuses on the relational model, and develops a straightforward transformation procedure from the conceptual to the relational model, where fact schemata are mapped to fact relations and each dimension level is mapped to a separate table. Interestingly, the final schema turns out to be a fully normalized snowflake schema (if only a single fact schema occurs) or galaxy schema (if multiple fact schemata occur and share some dimensions).

Acknowledgement. The authors are grateful to Thomas Löchte of zeb/rolfes-schierenbeck/associates for a number of discussions on the subject of this paper.

References

[AGS97] R. Agrawal, A. Gupta, S. Sarawagi, “Modeling multidimensional databases,” Proc. 13th ICDE 1997, 232–243.

[BCN92] C. Batini, S. Ceri, S. B. Navathe, *Conceptual Database Design - An Entity–Relationship Approach*, Benjamin/Cummings, Redwood City, 1992.

[BSHD98] M. Blaschka, C. Sapia, G. Höfling, B. Dinter, “Finding your way through multidimensional data models,” DEXA Workshop on Data Warehouse Design and OLAP Technology 1998.

[CT98] L. Cabibbo, R. Torlone, “A logical approach to multidimensional databases,” Proc. 6th EDBT 1998, LNCS 1377, 183–197.

[CD97] S. Chaudhuri, U. Dayal, “An Overview of Data Warehousing and OLAP Technologies,” Proc. ACM SIGMOD 1997, 65–74.

[FV95] C. Fahrner, G. Vossen, “A Survey of Database Design Transformations Based on the Entity-Relationship Model,” Data & Knowledge Engineering 15, 1995, 213–250.

[GMR98] M. Golfarelli, D. Maio, S. Rizzi, “Conceptual design of data warehouses from E/R schemes,” Proc. 32th HICSS 1998.

[GR98] M. Golfarelli, S. Rizzi, “A methodological framework for data warehousing design,” ACM Workshop on Data Warehousing and OLAP, 1998.

[Hue99] B. Hüsemann, “OLAP-Datenmodelle für Controlling im Finanzdienstleistungsbereich,” Diplomhausarbeit, Institut für Wirtschaftsinformatik, Universität Münster, September 1999.

[Leh98] W. Lehner, “Modeling large scale OLAP scenarios,” Proc. 6th EDBT 1998, LNCS 1377, 153–167.

[LAW98] W. Lehner, J. Albrecht, H. Wedekind, “Normal forms for multidimensional databases,” Proc. 10th SSDBM 1998, 63–72.

[LS97] H. Lenz, A. Shoshani, “Summarizability in OLAP and statistical databases,” Proc. 9th SSDBM 1997, 132–143.

[MQM97] I.S. Mumick, D. Quass, B.S. Mumick, “Maintenance of Data Cubes and Summary Tables in a Warehouse,”, Proc. ACM SIGMOD 1997, 100–111.

[PJ99] T.B. Pedersen, Ch.S. Jensen, “Multidimensional Data Modeling for Complex Data,” Proc. ICDE 1999, 336–345.

[SBHD98] C. Sapia, M. Blaschka, G. Höfling, B. Dinter, “Extending the E/R Model for the Multidimensional Paradigm,” ER Workshops 1998, 105–116.

[VS99] P. Vassiliadis, T. Sellis, “A Survey on Logical Models for OLAP Databases,” ACM SIGMOD Record 28(4) 1999, 64–69.

[Vos99] G. Vossen, *Data Models, Database Languages and Database Management Systems*, 4th edition (in German), Oldenbourg, 2000.

[WB97] M.-C. Wu, A. P. Buchmann, “Research issues in data warehousing,” Proc. 7th BTW 1997, 61–82.