Acquiring Expert Knowledge for the Design of Conceptual Information Systems

Gerd Stumme

Technische Universität Darmstadt, Fachbereich Mathematik
Schloßgartenstr. 7, D–64289 Darmstadt, stumme@mathematik.tu-darmstadt.de

Abstract. Conceptual Information Systems unfold the conceptual structure of data stored in relational databases. In the design phase of the system, conceptual hierarchies have to be created which describe different aspects of the data. In this paper, we describe two principal ways of designing such conceptual hierarchies, *data driven design* and *theory driven design*, and discuss advantages and drawbacks. The central part of the paper shows how *Attribute Exploration*, a knowledge acquisition tool developed by B. Ganter can be applied for narrowing the gap between both approaches.

1 Introduction

Conceptual Information Systems ([20], [21]) unfold the conceptual structure of data stored in relational databases. A *Conceptual Information System* consists of the relational database together with conceptual hierarchies. These hierarchies, called *conceptual scales*, are used to support navigation through the data. Conceptual Information Systems are based on the mathematical theory Formal Concept Analysis ([10]). The management system TOSCANA visualizes arbitrary combinations of conceptual scales and allows on-line interaction with the database to analyze and explore data conceptually. TOSCANA has been developed at the Technische Universität Darmstadt and is, for four years now, also marketed by NaviCon Gesellschaft für Begriffliche Wissensverarbeitung mbH. There are more than 30 Conceptual Information Systems implemented up to now, including an information system about laws and regulations in civil engineering ([7]), a library retrieval system ([14]) and an information system about flight movements ([12]). The use of Conceptual Information Systems gave rise to new theoretical questions which now dominate the research in Formal Concept Analysis. The demand of integrating knowledge acquisition tools in the design process of Conceptual Information Systems appeared for instance during the development of a Conceptual Information System about IT security.

For most applications, the Conceptual Information System is designed in a discursive process involving a domain expert and a knowledge engineer. Beside the database design, the conceptual scales have to be generated. Both steps require knowledge about the domain and about the structure of conceptual scales. In order to obtain interesting and non-trivial insights from the data, it is crucial that the domain expert is intensively involved in the design process. On the other

hand, it has been observed that the time a domain expert is expected to spend for the design is one of the most critical factors for the decision of a company whether to implement a Conceptual Information System. Hence, one important requirement is to make the knowledge acquisition from the domain expert more efficient.

In order to keep the scales in a suitable size, they are, in some applications, designed to fit the actual data, and are not conform to all possible updates of the underlying database. These scales are derived semi-automatically from the actual data, thus their design needs less expertise — and time — from the domain expert. If an update violating the structure of the scale happens, then the user is warned, and he has to redraw the scale. If there are only small changes, the re-drawing can be done automatically, but due to the lack of acceptable drawing algorithms for lattices, large changes cannot be recovered automatically, and have to be effectuated by the knowledge engineer. If the latter is not part of the company in which the system is implemented, then these eventualities should be covered in advance. Hence, a second requirement is the stability of the conceptual scales against all possible updates of the underlying database. This requirement will be obsolete when acceptable drawing algorithms for lattices are developed, but this evolution is not in sight in the next future.

In this paper, we describe two principal ways of designing conceptual scales, *data driven design* and *theory driven design*, and discuss advantages and drawbacks with respect to the two requirements. The central part of the paper shows how *Attribute Exploration*, a knowledge acquisition algorithm developed by B. Ganter, can be applied in order to narrow the gap between both approaches. Attribute Exploration determines implications (functional dependencies) between attributes in an interactive session. Its typical application is in Mathematics, where mathematical theorems or counter-examples, resp., are asked from the mathematician in a systematic way in order to obtain a complete theory about specific mathematical structures.

In the next section, we describe the basics of Conceptual Information Systems and illustrate them by means of examples. Section 3 discusses the two principal ways of preparing a Conceptual Information System: theory driven design and data driven design. The design of the underlying database scheme is not topic of this paper. In Section 4, we describe the algorithm of Attribute Exploration and show by means of an example how it can be applied to the design of Conceptual Scales.

2 Conceptual Information Systems

Conceptual Information Systems provide a multi-dimensional conceptually structured view on data stored in relational databases. Conceptual Information Systems are similar to On-Line Analytical Processing (OLAP) tools, but focus on qualitative (i.e. non-numerical) data. The analog to OLAP dimensions are hierarchies of concepts. They are based on Formal Concept Analysis ([23], [10]), a mathematical theory modeling the concept of 'concept' as discussed in Philoso-

phy since the logic of Port Royal ([3]) and described in the German Industrial Standards DIN 2330 and DIN 2331. There, a concept is understood as a unit of thought consisting of two parts: its extension and its intension ([22]). The extension consists of all objects belonging to the concept, and the intension of all attributes common to all the objects. In OLAP terminology, intensions of concepts correspond to coordinates addressing a cell, and extensions to entries of cells of a data cube. *Formal concepts* as defined below act as knots tying together the extensional and the intensional aspect of the data.

Each conceptual scale is generated from a *formal context*, a binary relation which allocates subsets of the attribute domains of the database to attributes which are meaningful to the analyst. The derived conceptual hierarchy can be an arbitrary lattice. It is displayed by a *Hasse diagram* which provides a universal and intuitively readable visualization of the data. By combining Hasse diagrams and zooming into them, operations similar to slicing, pivoting, drill-down and drill-up are supported ([17]). In the next section, we provide the mathematical background. Readers not familiar to mathematical notation may directly skip to the example.

2.1 The Mathematical Background: Formal Concept Analysis

Definition. A *(formal) context* is a triple $\mathbb{K} := (G, M, I)$ where G and M are sets and I is a relation between G and M. The elements of G and M are called *objects* and *attributes*, respectively, and $(g, m) \in I$ is read *"the object g has the attribute m"*.

For $A \subseteq G$, we define $A' := \{m \in M \mid \forall g \in A\colon (g, m) \in I\}$. For $B \subseteq M$, we define dually $B' := \{g \in G \mid \forall m \in B\colon (g, m) \in I\}$. Now a *(formal) concept* is a pair (A, B) such that $A \subseteq G$, $B \subseteq M$ and $A' = B$, $B' = A$. (This is equivalent to A and B being maximal with $A \times B \subseteq I$.) The set A is called the *extent* and the set B the *intent* of the concept.

Each formal context gives rise to a conceptual hierarchy, called *concept lattice* of $\mathbb{K}$ and denoted by $\underline{\mathfrak{B}}(\mathbb{K})$. The hierarchical subconcept–superconcept–relation of concepts is formalized by

$$(A, B) \leq (C, D) :\Longleftrightarrow A \subseteq C \quad (\Longleftrightarrow B \supseteq D) \ .$$

Theorem 1 (cf. [10]). *The set of all concepts of the context $\mathbb{K}$ together with this order relation is a complete lattice. I. e., for each set (A_t, B_t), $t \in T$, of concepts, a least common superconcept and a greatest common subconcept exist. They are computed as follows:*

$$\bigvee_{t\in T}(A_t, B_t) = ((\bigcup_{t\in T} A_t)'', \bigcap_{t\in T} B_t) \ , \qquad \bigwedge_{t\in T}(A_t, B_t) = ((\bigcap_{t\in T} A_t, (\bigcup_{t\in T} B_t)'')$$

The first equation describes the aggregation along the subconcept-superconcept-hierarchy: The extent of the least common superconcept is the closure by $''$ of the set union $\bigcup_{t\in T} A_t$. Because of the symmetry of the definition, attributes

M 3.8	M 3.7	M 3.6	M 3.5	M 3.4	M 3.3	M 3.2	M 3.1	
					×			Personalausfall
							×	Unzureichende Kenntnis über Regelungen
				×			×	Vertraulichkeits-/Integritätsverlust von Daten durch Fehlverhalten der IT-Benutzer
×	×		×	×		×	×	Fahrlässige Zerstörung von Gerät oder Daten
	×		×			×	×	Nichtbeachtung von IT-Sicherheitsmaßnahmen
			×	×			×	Fehlerhafte Nutzung des IT-Systems
×	×	×	×			×		Manipulation/Zerstörung von IT-Geräten oder Zubehör
×	×	×	×			×		Manipulation von Daten oder Software
			×				×	Social Engineering

M 3.1: Geregelte Einarbeitung/Einweisung neuer Mitarbeiter
M 3.2: Verpflichtung der Mitarbeiter auf Einhaltung einschlägiger Gesetze, Vorschriften und Regelungen
M 3.3: Vertretungsregelungen
M 3.4: Schulung vor Programmnutzung
M 3.5: Schulung zu IT-Sicherheitsmaßnahmen
M 3.6: Geregelte Verfahrensweise beim Ausscheiden von Mitarbeitern
M 3.7: Anlaufstelle bei persönlichen Problemen
M 3.8: Vermeidung von Störungen des Betriebsklimas

Fig. 1. Formal context about perils and counter-measures concerning IT security in Human Resources

can be aggregated in an analogous way by *descending* the hierarchy (cf. second equation). Again, the appropriate aggregation is not set union, but its closure by $''$. This allows the investigation of *implications (functional dependencies)* between the attributes:

Definition. For two sets $X, Y \subseteq M$ of attributes, *the implication* $X \to Y$ *holds* in a formal context, if each object having all attributes in X also has all attributes in Y (i. e., $X' \subseteq Y'$, or equivalently $Y \subseteq X''$).

These implications play an important role in data analysis, and are also crucial for knowledge acquisition by Attribute Exploration (cf. Sect. 4). [1]

Example: The following example is taken from an information system about IT security ([16]). In the 'IT–Grundschutzhandbuch' of the Bundesamt für Sicherheit in der Informationstechnik ([4]), perils to certain objects, such as e. g. infra structure, telecommunication, human resources, are listed, and counter-measures are discussed. The presented information system is for demonstration purpose only, but a similar, more praxis oriented system with a higher level of detail is offered by NaviCon. The design of conceptual scales for the latter gave rise to this paper.

[1] A remark for readers who are familiar with association rules ([1]): Implications are association rules with minsupp=0 and minconf=1. In the framework of this paper, other association rules than implications are of no importance, because the concep-

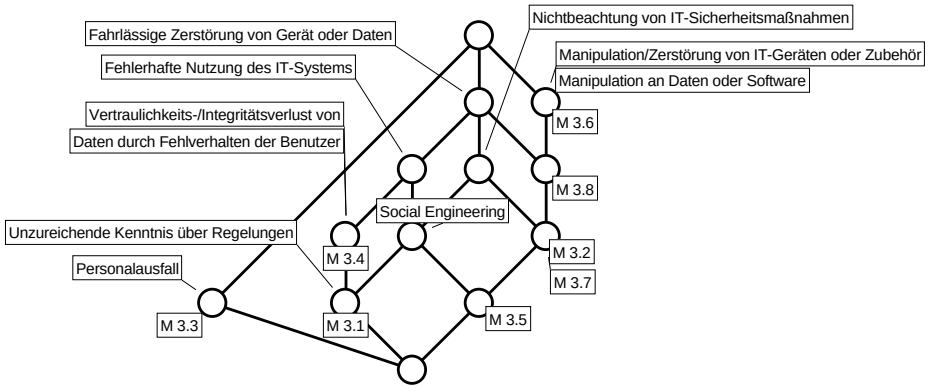

Fig. 2. Hasse diagram of the formal context in Fig. 1

The table for human resources from [4] is given in Fig. 1. It can be understood as a formal context, where the perils 'Personalausfall' (Staff drop out), ..., 'Social Engineering' are the attributes, and the counter-measures M 3.1, ..., M 3.8 are the objects. The relation assigns to each peril possible counter-measures. The context has 13 formal concepts. For instance, there is one concept having M 3.2, M 3.5, M 3.7, and M 3.8 in its extent, and 'Fahrlässige Zerstörung von Gerät oder Daten' (negligent destruction of machines or data), 'Manipulation/Zerstörung von IT-Geräten oder Zubehör' (manipulation of IT tools or accessories), and 'Manipulation an Daten oder Software' (manipulation on data or software) in its intent.

The concept lattice of that formal context is shown in Fig 2. Each circle stands for a formal concept, and the subconcept-superconcept hierarchy can be read by following ascending paths of straight line segments. The intent [extent] of each concept is given by all labels reachable from that context by ascending [descending] paths of straight line segments. For instance, the concept mentioned above is the one labeled by M 3.8.

In such a diagram, we can read the implications between the attributes. For determining the conclusion of an implication, one determines the greatest common subconcept of the premise (the concept where "the attributes of the premise first meet" by descending the diagram), and collects all attributes listed above. I.e., the implication $X \to Y$ holds if and only if $\bigvee_{m \in X} (\{m\}', \{m\}'') \leq (\{n\}', \{n\}'')$ for all $n \in Y$. The concept $(\{m\}', \{m\}'')$ is the concept which is labeled by the attribute m. For instance, we have that each counter-measure against both 'Fehlerhafte Nutzung des IT-Systems' (misuse of the IT system)

tual scales to be created shall cover *all* possible combinations, not only the frequent ones.

Bauteil	Bauteileart	Nennweite	DichtWerkst	Wanddicke
Rohr DIN 2448- 13 CrMo 4 4 -355,6x8,0	Rohr	350		8
Rohr DIN 2448- 13 CrMo 4 4 -355,6x8,8	Rohr	350		8,8
Rohr DIN 2448- 13 CrMo 4 4 -355,6x11,0	Rohr	350		11
Rohr DIN 2448- 13 CrMo 4 4 -406,4x8,8	Rohr	400		8,8
Rohr DIN 2448- 13 CrMo 4 4 -406,4x11,0	Rohr	400		11
Rohr DIN 2448- 13 CrMo 4 4 -406,4x14,2	Rohr	400		14,2
Flansch C 15x21,3 DIN 2631 - St 37-2	Vorschweißflansch	15	Weichgumm	2
Flansch C 20x26,9 DIN 2631 - St 37-2	Vorschweißflansch	20	Weichgumm	2,3
Flansch C 25x33,7 DIN 2631 - St 37-2	Vorschweißflansch	25	Weichgumm	2,6
Flansch C 32x42,4 DIN 2631 - St 37-2	Vorschweißflansch	32	Weichgumm	2,6

Fig. 3. Part of a many-valued context

and 'Manipulation an Daten oder Software' (the only counter-measure against both perils simultaneously is M 3.5) is also a counter-measure against the perils 'Manipulation/Zerstörung von IT-Geräten oder Zubehör', 'Nichtbeachtung von IT-Sicherheitsmaßnahmen' (ignoring of IT security measures), and 'Fahrlässige Zerstörung von Gerät oder Daten'.

2.2 The conceptual data model of Conceptual Information Systems: Many-valued contexts and conceptual scales

Often attributes are not one-valued as in the previous example, but allow a range of values. This is modeled by a *many-valued context*. In order to obtain a concept lattice, a many-valued context is 'translated' into a one-valued context by *conceptual scales*. (Remark that 'conceptual' is used in two different meanings in the heading!)

Definition 2. A *many-valued context* is a tuple $(G, M, (W_m)_{m\in M}, I)$ where G and M are sets of *objects* and *attributes*, resp., W_m is a set of *values* for each $m \in M$, and $I \subseteq G \times \bigcup_{m\in M}(\{m\} \times W_m)$ such that $(g, m, w_1) \in I$ and $(g, m, w_2) \in I$ imply $w_1 = w_2$. A *conceptual scale* for an attribute $m \in M$ is a context $\mathbb{S}_m := (G_m, M_m, I_m)$ with $W_m \subseteq G_m$. The context (G, M_m, J) with $gJn :\Longleftrightarrow \exists w{\in}W_m\colon (g, m, w){\in}I \wedge (w, n){\in}I_m$ is called the *realized scale* for the attribute $m \in M$.

Example: Figure 3 shows a part of a many-valued context about pipes. The total context consists of 240 pipes, 2428 curved pipes, 560 T-parts, 348 flanges, and 385 restricted fittings, and of 54 attributes. The objects are listed in the column 'Bauteil' (Part). In Fig. 4, the realized scale for the attribute 'Bauteileart' (Part type) is given. Since there are almost 4000 objects, the diagram does not display their names, but contingents only.

Conceptual Information Systems consist of a many-valued context together with a collection of conceptual scales. The many-valued context is implemented

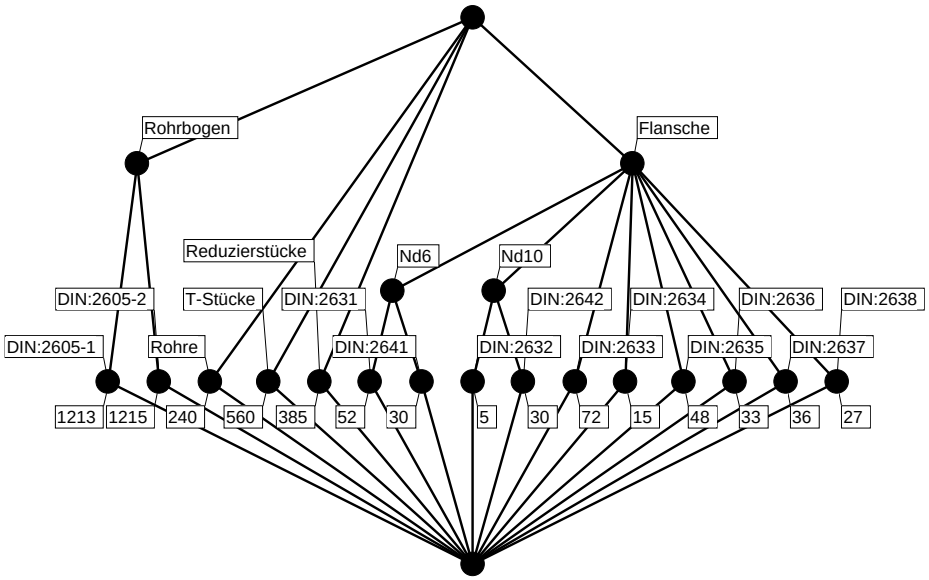

Fig. 4. Realized scale 'Part type'

as a relational database. The collection of the scales is called *conceptual scheme* ([20], [15]). It is written in the description language ConScript ([19]). Beside the contexts of the conceptual scales, the conceptual scheme also contains the layout of their line diagrams. The layout has to be provided in advance, since, in general, well readable line diagrams cannot be generated fully automatically.

For Conceptual Information Systems, the management system TOSCANA ([13], [21]) has been developed. Based on the paradigm of conceptual landscapes of knowledge ([24]), TOSCANA supports the navigation through the data by using the conceptual scales like maps designed for different purposes and in different granularities. We illustrate the navigation procedure by the pipeline system.

Example: The context in Fig. 3 and the conceptual scale in Fig. 4 are part of a Conceptual Information System on pipelines ([18]). It shall support the engineer by choosing suitable parts for a projected pipeline system. Let us assume that he needs a pipe which has an inner diameter of about 100 mm and a wall thickness of about 4 mm. Starting with the scale 'Part type' in Fig. 4, he finds the concept labeled with the attribute 'Rohre' (Pipes), and sees that he can choose among 240 different pipes. By *zooming into* that context with the scale 'Inner diameter', see Fig. 5), he can see the distribution of the 240 pipes according to their inner diameter. Each concept stands for an interval. Since the engineer is interested in pipes with about 100 mm inner diameter, he chooses the 8th concept from the right at the bottom level, which stands for the interval 90–110 mm. By taking a

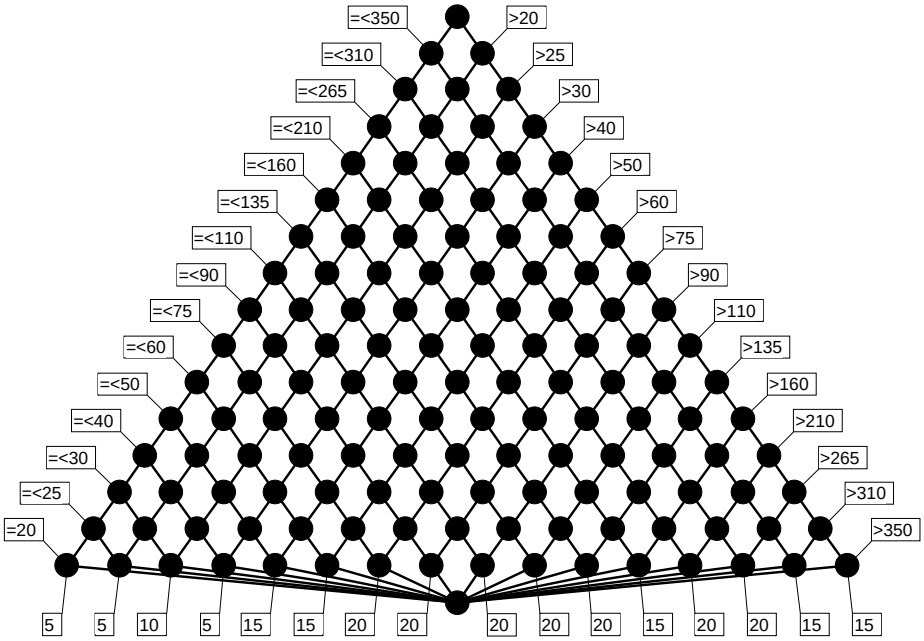

Fig. 5. Realized scale 'Inner diameter' after zooming into the concept labeled by 'Rohre' in Figure 4

concept which is higher in the hierarchy, he could have continued with an interval of larger width. By zooming into the chosen concept with the next conceptual scale (e. g.'Wall thickness'), the engineer can drill-down further until he obtains a small number of parts which are suitable for the projected pipeline system. By clicking on the numbers, he can obtain the names of the parts, and can then drill-down to the original data given in the database or to additional information such as DIN standards.

3 Preparation of Conceptual Information Systems

The preparation of a conceptual information system consists basically of two steps. First, the underlying many-valued context has to be designed and implemented as a database system. Second, the conceptual scales have to be created. Both are non-trivial tasks which require expertise in the domain of interest as well as in the knowledge representation techniques of formal concept analysis. Hence conceptual information systems are usually designed in a discursive process involving both domain experts and knowledge engineers ([2]). This process is described in detail in [7] for a system about laws and regulations in civil engineering.

In this paper, we focus on the second step of the preparation. We assume that the many-valued context is already given. The task is then to design adequate conceptual scales. We discuss the two basic ways.

3.1 Theory Driven Design

The first step in designing scales driven by theory is to choose attributes meaningful to the user. They need not to be the domain values of the database, but are usually on a more general level. For instance, the user is often not interested in exact numerical values but only in certain ranges: In a medical application, the physician is not interested in the exact pH level of the blood, but only if the pH level is pathological or even dangerous.

The second step is to assign the domain values to the attributes. Here, the knowledge engineer has to bring in his expertise about conceptual hierarchies, since domain experts always tend to scale nominally. In the medical example, for instance, a longer discussion revealed that a dangerous pH level is also understood as pathological, hence a bi-ordinal scale (with a third attribute 'pH level normal') was chosen.

Figures 4 and 5 show two theory driven scales. While the scale in Fig. 4 is specially designed for the application, the inter-ordinal scale in Fig. 5 is a standard scale that is used in many applications. Typically, database attributes of type `string` need an individual design, while numerical types as `integer` or `real` allow the use of standard scales. There is a broad variety of standard scales that can be used, e. g., nominal scales, ordinal scales, and inter-ordinal scales. In the latter case, only the number of intervals to be considered and the interval boundaries have to be fixed. It is planned to release the knowledge engineer from implementing such standard scales by implementing *parametrized scales* which adopt themselves to the actual range of the values. Naturally this approach fails for free-text entries such as those in 'Bauteileart' in Fig. 3. Here the conceptual structure in the data has to be determined in a discursive process.

3.2 Data Driven Design

While theory driven design is typically (but not exclusively) applied to many-valued attributes, data driven design is only possible for the data type `boolean`. In that case, the attributes of the database are usually also the attributes of the conceptual scale. While there is normally one conceptual scale for each many-valued attribute, some one-valued (i. e., Boolean) attributes are grouped together in order to form one conceptual scale. The task for knowledge engineer and domain expert is to find a suitable grouping of the attributes. Groups should not be too large since the size of the scale may be exponential to the number of attributes; neither too small in order not to hide dependencies between the attributes. Typically there are between five and ten attributes. But before all, it is important that attributes addressing similar topics are grouped together. Therefore it is possible that attributes appear in more than one scale.

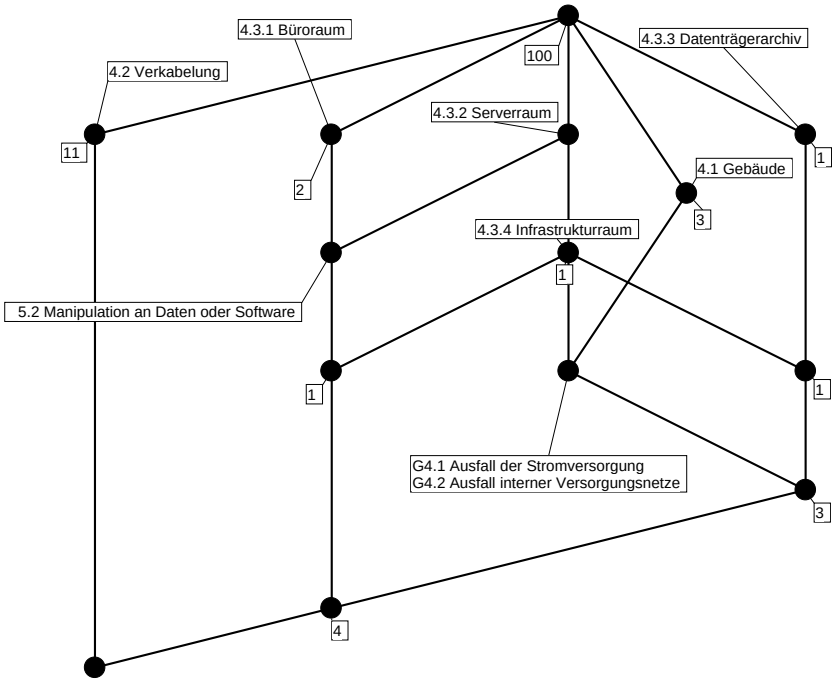

Fig. 6. Realized scale 'Rooms' for the IT Security System

Having obtained a suitable grouping, a conceptual scale has to be designed for for each of the sets of attributes. In a first approach, one could assume that there are no valid implications between the attributes. This leads to a scale that is conform to all possible updates of the database. However, this scale will be too large for more than five attributes, since for n attributes the number of concepts of such a *Boolean scale* scale is 2^n. Therefore, *data driven design* takes into account all implications which hold for the actual entries in the database. It is supported by DOKUANA, a tool developed by NAVICON. An example for a data driven scale is the scale 'Rooms' of the IT security system (Fig. 6). It shows the distribution of perils according to locations. Instead of $2^6 = 64$ possible concepts it only consists of 14 concepts.

When knowledge engineer and domain expert agreed on a data driven design process, then the design can be performed by the former without any support by the latter, once the grouping of the attributes is decided. Hence data driven design is efficient for the client in the way that he does not have to invest much time of the domain expert. This is an important argument in marketing Conceptual Information Systems.

The big disadvantage of this approach however is that scales need not be consistent with updates of the database. New entries can contradict to the functional dependencies used for the design of the conceptual scales. If there are not too many new concepts (up to ten at the moment), and if the structure of the new scale does not differ too much from the original scale, then the diagram can be re-drawn automatically. However, if the change is more complex, then the layout has to be done manually, which usually requires the expertise of the knowledge engineer. Hence, an important research task is the development of fully automatic lay-out algorithms for lattices. Unfortunately, satisfying answers are not in sight in the next future.

For applications provided to a remote client or for time critical applications it is therefore important to prepare the scales such that future updates of the database are covered. Hence all possible combinations of attributes have to be determined before handing over the information system. In the next section we discuss how this task can be performed in a systematic way by involving the domain expert as less as possible.

4 Extending Scales by Attribute Exploration

The data driven design of a conceptual scale provides us with all combinations of attributes which occur as concept intents for the actual data. Then the question arises which combinations may occur additionally. As we pointed out in the last section, the powerset of the attribute set would cover all eventualities, but is in general too large for practical applications. Hence we have to find a subset of the powerset, which contains all possible combinations, by systematically inquiring the domain expert.

The solution to that problem is *Attribute Exploration* ([8], [10]), an interactive knowledge acquisition algorithm developed by B. Ganter. The algorithm is implemented in the program ConImp ([5]) of P. Burmeister. It benefits from the fact that the requested set of intents must be closed under set intersection (cf. to first equation in Theorem 1). The knowledge is acquired from the domain expert in a dialogue in which he has to answer questions of the form "Does each possible object in the database having attributes $x_1, \ldots x_n$ necessarily have the attributes $y_1, \ldots y_m$ as well? (I.e., "Does the implication $\{x_1, \ldots, x_n\} \rightarrow \{y_1, \ldots, y_m\}$ hold?") Either the expert confirms the implication, or he has to provide a counter-example.

Details about Attribute Exploration can be found in [8] and [10]. Here, we only give a short summary: The algorithm uses the fact that, for a given formal context, the implications $P \rightarrow P''$, where P is a *pseudo-intent* (see below), are sufficient (and even minimal) for describing the structure of the concept lattice. This set of implications is called the *Duquenne-Guigues-basis* ([6], [11]).

The algorithm asks the implications in such a sequence that pseudo-intents determined once remain pseudo-intents even after adding the counter-examples to the context:

Definition. A set $P \subseteq M$ of attributes is called a *pseudo-intent*, if $P \neq P''$ and if for each pseudo-intent $Q \subset P$ the inclusion $Q'' \subseteq P$ holds.

For a set $X \subseteq M$ and a set $\mathcal{L}$ of implications, we define $\mathcal{L}^*(X)$ as the closure of X under repeated application of $X \mapsto X \cup \bigcup\{B \mid A \to B \in \mathcal{L}, A \subset X, A \neq X\}$.

For sake of simplicity, we assume now that $M = \{1, 2, \ldots, n\}$. For $i \in M$, we define $X <_i Y :\iff i \in Y \setminus X$ and $X \cap \{1, \ldots, i-1\} = Y \cap \{1, \ldots, i-1\}$. Furthermore we define a *lectic order* on the subsets of M by $X < Y :\iff \exists i \in \{1, \ldots, n\} : X <_i Y$.

Algorithm: Let (G, M, I) be the formal context determined by data driven design. The set M contains the attributes that are used as labels in the diagram. The set G contains strings which are used as `where`-parts of SQL-statements which TOSCANA generate in order to query the database.

1. The first intent or pseudo-intent is the empty set.
2. For a given intent or pseudo-intent X one obtains the next intent or pseudo-intent in the lectic order by letting $i := n$, and decreasing i until $X <_i X^* := \mathcal{L}^*((X \cap \{1, 2, \ldots, i-1\} \cup \{i\})$ holds. X^* is then the next intent or pseudo-intent in the lectic order.
3. IF $X^* = M$ then Stop.
4. If X^* is an intent, then let $X := X^*$ and go to 2).
5. If X^* is a pseudo-intent, then ask the user "Does the implication $X^* \to X^{*\prime\prime}$ hold?" If the answer is "Yes", then add the implication to $\mathcal{L}$. Let $X := X^*$ and go to 2). If the answer is "No", then the user has to provide a counter-example. Add the counter-example to G and go to 4).

The dialogue is optimal in the sense that the number of confirmed implications is minimal. The complexity of the algorithm is, for each concept, cubic in the number of attributes and objects. As the number of the concepts can grow exponentially in the number of attributes, the overall complexity of the algorithm is exponential. However, as the number of attributes for one scale is usually between five and ten, and the attributes are normally not totally independent, the number of questions is tolerable in all practical applications.

Attribute Exploration can handle the answer "I don't know", but for designing a conceptual scale, finally each of the implications has to be either confirmed or rejected. As "I don't know" indicates that there *may be* objects that violate the implication, only the interpretation of these answers as "no" assures that the scale will be consistent for all possible updates of the database.

Example: The 'IT-Grundschutzhandbuch' provides for each object the relationship between its related perils and counter-measures. The relationship between objects and perils is not given explicitly in the handbook. Since the data tables are designed locally — for each single object — only, there may be groups of objects sharing the same perils which are not identified in the book. Figure 6 provides us with the scale considering only the actual entries in the handbook.

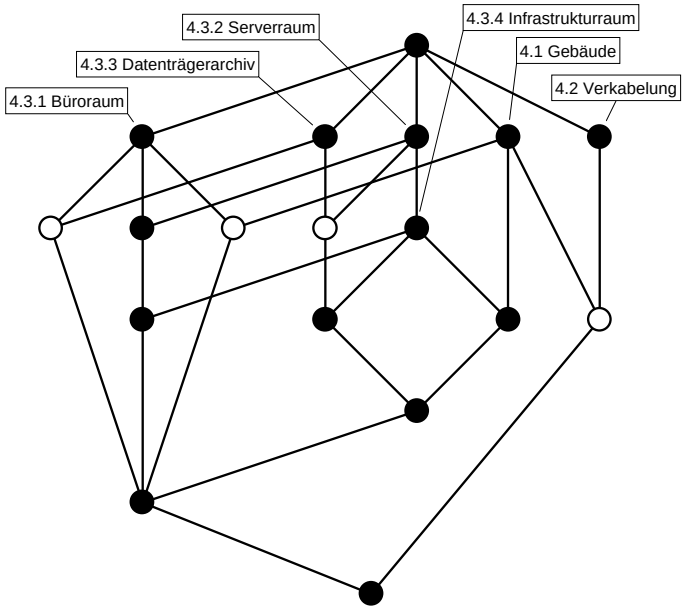

Fig. 7. The scale 'Rooms' extended by Attribute Exploration

For determining all possible combinations of objects, we applied Attribute Exploration to that scale. The exploration dialogue consisted of twelve questions, starting with:

> "Is each possible peril for 'Gebäude' (building) and 'Archiv' (archive) necessarily a peril for 'Serverraum' (server room) and 'Infrastrukturraum' (infrastructure room)?" — "Yes" — "Is each possible peril for 'Infrastrukturraum' necessarily a peril for 'Serverraum?" — "Yes" — "Is each possible peril for 'Infrastrukturraum' necessarily a peril for 'Serverraum?" — "Yes" — "Is each possible peril for 'Serverraum' and 'Archiv' necessarily a peril for 'Infrastrukturraum?" — "No." ...

For the last question, one can e. g. provide the attribute 'Datenträgervernichtung' (destruction of storage media) as counter-example. In this example, nine of the twelve implications were accepted, and three denied. The resulting scale is shown in Figure 7. It consists of 18 concepts, while the 'worst case', the Boolean scale, has $2^6 = 64$ concepts and is too large for a useful visualization. The black circles indicate how the data driven scale is embedded in the scale determined by Attribute Exploration. This example also shows that the diagram has to be re-layouted for remaining readable.

5 Outlook

We have shown how the gap between data driven and theory driven design can be bridged (or at least narrowed) by applying Attribute Exploration. This knowledge acquisition process serves three purposes. Firstly, it makes knowledge acquisition from the domain expert more efficient, since it starts with the actual data (instead from the scratch) and solves the remaining questions in a systematic and somehow minimal way. Secondly, it allows to prepare the scales so that all eventual updates of the database are covered a priori. Hence the system can be run without support of the knowledge engineer. Thirdly, the process provides to the domain expert a better understanding of the data by making his knowledge explicit.

A restraint for the approach is however that a certain knowledge about the dependencies between the attributes must be present. Although the answers during the dialogue do not have to be infallible (since TOSCANA provides a warning if an accepted implication is violated), they must however be confident to a certain degree. For instance, the Library Retrieval System at the 'Zentrum für interdisziplinäre Technikforschung' at the Technische Universität Darmstadt ([14]) is also based on a data driven design. In that application, books and journals are objects, and catchwords are attributes. The conceptual scales produced by data driven design are almost all near to Boolean scales, i. e., almost all combinations of catchwords are possible. In this application, the experts were not able to answer the questions with a certain confidence, since for each remaining combination of catchwords one could imagine a book having exactly those catchwords. Such applications would profit enormously from automatic layout algorithms for lattices.

An automatic layout algorithm would also provide the possibility to choose on-line Boolean attributes (e. g., catchwords) of the database and to let the resulting conceptual scale be drawn on the fly. The development of layout algorithms is one of the most urgent research tasks for advancing the commercial application of conceptual knowledge processing.

Attribute Exploration determines the structure of the conceptual scale only; but it does not indicate which concepts may be labeled by objects. Each concept which potentially is labeled gives rise to a SQL-query. Hence it may be of interest for time-critical applications to minimize the number of these concepts. For determining them, a variation of Attribute Exploration called *Clause Exploration* ([9]) can be applied. As this knowledge acquisition procedure generates more questions than Attribute Exploration, it has to be examined for each application if the extra work during the design phase is really necessary.

References

1. R. Agrawal, T. Imielinski, A. Swami: Mining association rules between sets of items in large databases. *Proc. ACM SIGMOD*, 1993
2. U. Andelfinger: *Diskursive Anforderungsanalyse: ein Beitrag zum Reduktionsproblem bei Systementwicklungen in der Informatik*. Peter Lang, Frankfurt 1997.

3. A. Arnauld, P. Nicole: La logique ou l'art de penser — contenant, outre les règles communes, plusieurs observations nouvelles, propres à former le jugement. 3^o édit. reveüe & augm. P., Ch. Saveux, 1668
4. Bundesamt für Sicherheit in der Informationstechnik: *IT–Grundschutzhandbuch 1996. Maßnahmenempfehlungen für den mittleren Schutzbedarf.* Bundesanzeiger, Köln 1996
5. P. Burmeister: ConImp – Programm zur formalen Begriffsanalyse einwertiger Kontexte. TH Darmstadt 1987 (latest version 1995)
6. V. Duquenne:Contextual implications between attributes and some properties of finite lattices. In: B. Ganter, R. Wille, K.E. Wolff (eds.): *Beiträge zur Begriffsanalyse.* B. I.-Wissenschaftsverlag, Mannheim 1987, 213–239
7. D. Eschenfelder, W. Kollewe, M. Skorsky, R. Wille: Ein Erkundungssystem zum Baurecht: Methoden der Entwicklung eines TOSCANA-Systems. In: G. Stumme, R. Wille (eds.): *Begriffliche Wissensverarbeitung: Methoden und Anwendungen.* Springer, Heidelberg 1998 (to appear)
8. B. Ganter: Algorithmen zur Begriffsanalyse. In: B. Ganter, R. Wille, K. E. Wolff (eds.): *Beiträge zur Begriffsanalyse.* B. I.-Wissenschaftsverlag, Mannheim, Wien, Zürich 1987. 241–254
9. B. Ganter, R. Krauße: *Pseudo models and propositional Horn inference.* (in preparation)
10. B. Ganter, R. Wille: *Formal Concept Analysis: Mathematical Foundations.* Springer, Heidelberg 1999 (Translation of: *Formale Begriffsanalyse: Mathematische Grundlagen.* Springer, Heidelberg 1996)
11. J.-L. Guigues, V. Duquenne: Familles minimales d'implications informatives resultant d'un tableau de données binaires. *Math. Sci. Humaines* **95**, 1986, 5–18
12. U. Kaufmann: *Begriffliche Analyse von Daten über Flugereignisse — Implementierung eines Erkundungs- und Analysesystems mit TOSCANA.* Diplomarbeit, TU Darmstadt, 1996
13. W. Kollewe, M. Skorsky, F. Vogt, R. Wille: TOSCANA — ein Werkzeug zur begrifflichen Analyse und Erkundung von Daten. In: R. Wille, M. Zickwolff (eds.): *Begriffliche Wissensverarbeitung — Grundfragen und Aufgaben.* B.I.–Wissenschaftsverlag, Mannheim 1994
14. T. Rock, R. Wille: Ein TOSCANA-System zur Literatursuche. In: G. Stumme and R. Wille (eds.): *Begriffliche Wissensverarbeitung: Methoden und Anwendungen.* Springer, Berlin-Heidelberg (to appear)
15. P. Scheich, M. Skorsky, F. Vogt, C. Wachter, R. Wille: Conceptual data systems. In: O. Opitz, B. Lausen, R. Klar (eds.): *Information and classification.* Springer, Heidelberg 1993, 72–84
16. H. Söll: *Begriffliche Analyse triadischer Daten: Das IT-Grundschutzhandbuch des Bundesamtes für Sicherheit in der Informationstechnik.* Diplomarbeit, TU Darmstadt 1998
17. G. Stumme: On-Line Analytical Processing with Conceptual Information Systems. *Proc. 5th Intl. Conf. on Foundations of Data Organization*, 12.–13. November 1998, 117–126 (to be published by Kluwer)
18. N. Vogel: *Ein Begriffliches Erkundungssystem für Rohrleitungen.* Diplomarbeit, TH Darmstadt 1995
19. F. Vogt: *Datenstrukturen und Algorithmen zur Formalen Begriffsanalyse: Eine C++–Klassenbibliothek.* Springer, Heidelberg 1996
20. F. Vogt, C. Wachter, R. Wille: Data analysis based on a conceptual file. In: H.-H. Bock, P. Ihm (eds.): *Classification, data analysis, and knowledge organization.*

Springer, Heidelberg 1991, 131–140
21. F. Vogt, R. Wille: TOSCANA – A graphical tool for analyzing and exploring data. In: R. Tamassia, I. G. Tollis (eds.): *Graph Drawing '94*, Lecture Notes in Computer Sciences **894**, Springer, Heidelberg 1995, 226–233
22. H. Wagner: Begriff. In: H. M. Baumgartner, C. Wild (eds.): *Handbuch philosophischer Grundbegriffe.* Kösel Verlag, München 1973, 191–209
23. R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.): *Ordered sets.* Reidel, Dordrecht–Boston 1982, 445–470
24. R. Wille: Conceptual landscapes of knowledge: A pragmatic paradigm of knowledge processing. In: *Proceedings of the international conference on knowledge retrieval, use, and storage for efficiency*, Vancouver, Kanada, 11.–13. 8. 1997, 2–13