1-1-2005

Dynamic schema navigation using Formal Concept Analysis

Jon R. Ducrou
University of Wollongong, jond@uow.edu.au

Bastian Wormuth
Darmstadt University of Technology

Peter W. Eklund
University of Wollongong, peklund@uow.edu.au

Follow this and additional works at: <https://ro.uow.edu.au/commpapers>

Part of the [Business Commons](#), and the [Social and Behavioral Sciences Commons](#)

Recommended Citation

Ducrou, Jon R.; Wormuth, Bastian; and Eklund, Peter W.: Dynamic schema navigation using Formal Concept Analysis 2005, 398-407.
<https://ro.uow.edu.au/commpapers/1234>

Dynamic schema navigation using Formal Concept Analysis

Abstract

This paper introduces a framework for relational schema navigation via a Web-based browser application that uses Formal Concept Analysis as the metaphor for analysis and interaction. Formal Concept Analysis is a rich framework for data analysis based on applied lattice and order theory. The application we develop, D-SIFT, is intended to provide users untrained in Formal Concept Analysis with practical and intuitive access to the core functionality of Formal Concept Analysis for the purpose of exploring relational database schema. D-SIFT is a Web-based information systems architecture that supports natural search processes over a preexisting database schema and its content.

Keywords

Dynamic, Schema, Navigation, Using, Formal, Concept, Analysis

Disciplines

Business | Social and Behavioral Sciences

Publication Details

Ducrou, J. R., Wormuth, B. & Eklund, P. W. (2005). Dynamic schema navigation using Formal Concept Analysis. *Data Warehousing and Knowledge Discovery* (pp. 398-407). Germany: Springer.

A Dynamic Schema Navigation using Formal Concept Analysis

Jon Ducrou¹, Bastian Wormuth², and Peter Eklund¹

¹ School of Economics and Information Systems
The University of Wollongong
Northfields Avenue, Wollongong, NSW 2522, Australia
`jrd990@uow.edu.au, peklund@uow.edu.au`

² Darmstadt University of Technology, Department of Mathematics,
Schloßgartenstr. 7, 64289 Darmstadt, Germany;
`bastian@wormuth.info`

Abstract. This paper introduces a framework for relational schema navigation via a Web-based browser application that uses Formal Concept Analysis as the metaphor for analysis and interaction. Formal Concept Analysis is a rich framework for data analysis based on applied lattice and order theory. The application we develop, D-SIFT, is intended to provide users untrained in Formal Concept Analysis with practical and intuitive access to the core functionality of Formal Concept Analysis for the purpose of exploration of relational database schema. D-SIFT is an information systems architecture that supports natural search processes over a predefined database schema and its attribute values. This enables the user to build concept lattices interactively through the selection and refinement of dynamic definitions of search boundaries, (via interaction with an object “*zoom*” feature), and dynamic selection of search scales, (via interaction with an attribute “*filter*” feature), based on the attribute values contained within the database. In detail, the paper presents the architecture of the D-SIFT browser and illustrates the resulting D-SIFT-systems on example database. The two examples presented illustrate the generality of system integration outcomes from D-SIFT to schema browsing using Formal Concept Analysis. The Conceptual Information Systems that result from applying the D-SIFT architecture present a new workflow for building and interacting with Formal Concept Analysis-based information systems. This workflow more closely aligns with dynamic schema interaction an increasingly popular technique used in conceptual modeling and analysis.

Introduction

This paper presents a new application framework for relational schema navigation using Formal Concept Analysis (FCA). The initial idea behind the framework is the simplification of existing application development frameworks for FCA, in particular the way humans process standard searches in FCA. The software prototype – called D-SIFT (Dynamic Simple Intuitive FCA Tool) – consolidates various features that have been introduced by other applications [1,2,3].

D-SIFT allows users to define query elements based on database schema and contents, using one of two query modalities, to visualise structures in the data as a concept lattice. The interface allows dynamic creation of lattice diagrams and allows the user to add and remove attributes from the displayed concept lattice according to his current preferences. D-SIFT implements the classical features of FCA software with so-called *mandatory attributes*. The user is able to restrict the displayed object set by the selection of mandatory attributes. The resulting concept lattice is limited to objects which share these attributes. This process closely resembles iterative search in information retrieval, whereby the user starts from one or two keywords and progressively refines the result set by the addition of further (or different) keywords.

D-SIFT is also more easily accessible as a platform than existing FCA frameworks. The required plug-ins used are provided in standard configurations of most Web browsers, and the underlying database complies with the CSV file format (text files with comma-separated entries).

1 Formal Concept Analysis background

Formal Concept Analysis [4] has a long history as a technique of data analysis ([?], [?]) conforming to the idea of Conceptual Knowledge Processing. Data is organized as a table and is modeled mathematically as a many-valued context, (G, M, W, I_w) where G is a set of objects, M is a set of attributes, W is a set of attribute values and I_w is a relation between G , M , and W such that if $(g, m, w_1) \in I_w$ and $(g, m, w_2) \in I_w$ then $w_1 = w_2$. In the table there is one row for each object, one column for each attribute, and each cell is either empty or asserts an attribute value.

A refined organization over the data is achieved via conceptual scales. A conceptual scale maps attribute values to new attributes and is represented by a mathematical entity called a formal context. A formal context is a triple (G, M, I) where G is a set of objects, M is a set of attributes, and I is a binary relation between the objects and the attributes, i.e. $I \subseteq G \times M$. A conceptual scale is defined for a particular attribute of the many-valued context: if $\mathbb{S}_m = (G_m, M_m, I_m)$ is a conceptual scale of $m \in M$ then we define $W_m = \{w \in W \mid \exists (g, m, w) \in I_w\}$ and require that $W_m \subseteq G_m$. The conceptual scale can be used to produce a summary of data in the many-valued context as a derived context. The context derived by $\mathbb{S}_m = (G_m, M_m, I_m)$ w.r.t. to plain scaling from data stored in the many-valued context (G, M, W, I_w) is the context (G, M_m, J_m) where for $g \in G$ and $n \in M_m$

$$gJ_m n \Leftrightarrow \exists w \in W : (g, m, w) \in I_w \\ \text{and } (w, n) \in I_m$$

Scales for two or more attributes can be combined in a derived context. Consider a set of scales, $\mathbb{S}_m$, where each $m \in M$ gives rise to a different scale. The new

attributes supplied by each scale can be combined:

$$N := \bigcup_{m \in M} M_m \times \{m\}$$

Then the formal context derived from combining these scales is:

$$gJ(m, n) \Leftrightarrow \exists w \in W : (g, m, w) \in I_w \\ \text{and } (w, n) \in I_m$$

Several general purpose scales exist such as ordinal and nominal scales. A nominal scale defines one formal attribute for each value that a many valued attribute can take. An ordinal scale can be used on a many-valued attribute for which there is a natural ordering, for example, $\text{size} \leq 1024, \text{size} \leq 4096$ and $\text{size} \leq 40\text{Mb}$.

The derived context is then displayed to the user as a lattice of concepts. A concept of a formal context (G, M, I) is a pair (A, B) where $A \subseteq G$, $B \subseteq M$, $A = \{g \in G \mid \forall m \in B : (g, m) \in I\}$ and $B = \{m \in M \mid \forall g \in A : (g, m) \in I\}$. For a concept (A, B) , A is called the extent and is the set of all objects that have all of the attributes in B , similarly, B is called the intent and is the set of all attributes possessed in common by all the objects in A . As the number of attributes in B increases, the concept becomes more specific, i.e. a specialization ordering is defined over the concepts of a formal context by:

$$(A_1, B_1) \leq (A_2, B_2) :\Leftrightarrow B_2 \subseteq B_1$$

In this representation more specific concepts have larger intents and are considered “less than” ($<$) concepts with smaller intents. The analog is achieved by considering extents, in which case, more specific concepts have smaller extents. The partial ordering over concepts is always a complete lattice [4].

For a given concept $C = (A, B)$ and its set of lower covers $(A_1, B_1) \dots (A_n, B_n)$ with respect to the above $<$ ordering the object contingent of C is defined as $A - \bigcup_{i=1}^n A_i$. We shall refer to the object contingent simply as the contingent in this paper.

Creating a Conceptual Information System from a Database

D-SIFT takes a user-supplied *comma separated values* database (CSV) and provides an interface to the database as a conceptual information system. For this reason the input format of D-SIFT closely aligns with a typical export format from a relational database management system (RDBMS) and common applications like Excel and OpenOffice.

The CSV format is simple, easy to read and edit. It is a common optional output format for most modern and legacy applications and database systems. CSV files are forced to contain only data that can be expressed as text; this

caters to the input requirements of D-SIFT. To translate the CSV database into a Conceptual Information System, the user indicates how D-SIFT should treat each field. This requires the user to indicate a field which is each entry's identifier (entity in RDBMS modeling terms) and then group the remaining fields into nominal or numerical scale models.

In order to extract objects with meaningful names, the user identifies the field which provides an identifier for the database (e.g. a candidate key such as *name* in a database of people). Nominal data, in FCA terms, is usually text (e.g. *names* or *locations* in the people database), and sometimes represents boolean values (e.g. attributes such as *gender* or attributes with values such as *yes/no*). Numerical data is represented by numbers which, over the scope of the entire field, have some form of ordering (e.g. a schema attribute such *length in meters* with some entries longer than others).

There are instances where database attributes consisting of numeric data should not be scaled ordinally; for example identifiers such as social security numbers, which may or may not be indicative of an order over the data values. D-SIFT also gives the option to drop fields that are not of interest to the user, by tagging those fields (e.g. *comment* or *ID* fields).

Interaction with the CSV file described to this point in the text allows D-SIFT to collect enough information to construct the context and scale information for the Conceptual Information System.

Using D-SIFT

D-SIFT intends to offer the user a flexible tool for viewing the various structures and relationships that are present in a database. The user only needs some understanding of the data they are viewing; enough to understand the objects being dealt with and the meaning of attributes, and some level of ability reading lattice diagrams. The owner of a database should know its content and user testing has shown that users can quickly become competent at reading lattice diagrams with little or no formal training [5].

The user constructs queries by selecting query elements of interest and assigning them to one of two lists; *Zoom* or *Filter*.

Query elements are made up of one or more *nominally-scaled* or *numerically-scaled* attributes. Nominally-scaled attributes comprise attribute groups and an attribute value. Numerically-scaled attributes comprise an attribute group, a *size* and an *order*. The size of a numerically scaled attribute can be thought of as the number of intervals which will be produced, while the order specifies the way in which the values should be compared. The orders are of three types, *Ordinal Up*, *Ordinal Down* and *Interordinal*. Ordinal Up and Down correspond to comparisons based on $\geq$ and $\leq$ respectively. Interordinal generates both Ordinal Up and Down simultaneously (See Fig. 1).

The *Zoom* list should be populated with query elements that are 'required'. The elements of the *Zoom* list are used to restrict the object set of the context to

Fig. 1. Concept lattices showing the different types of numeric scaling available via the D-SIFT interface are (from left to right) Ordinal Up, Ordinal Down and Interordinal. All are shown with a size of 3.

only objects with elements in the list. This is a conjunctive query so if dichotomous elements are in the *Zoom* list, the object set will be empty. Query elements in the *Zoom* list will always appear as attributes of the topmost concept of the diagram. Ordinal element groups can not be added, nor can two element values from the same element group. The *Filter* list should be populated with query elements that are ‘of interest’. The elements in the *Filter* list are used to restrict the attribute set of the context. This means that only elements in the *Filter* list (and any from the *Zoom* list) will feature in the resulting lattice. It is these attributes that will be used to show structural relations in the database.

Using this query building paradigm of ‘required’ and ‘of interest’, the user can perform exploratory tasks against the database. The simplest example of which is the idea of a ‘search’ for an object that meets several criteria, or aids in the discovery of the ‘next best’ when the exact result is unavailable. In our examples, a database concerning cellphones, we imagine a potential customer of a new phone. The user may have a rough idea of the technical features but no understanding which of these he really needs or wants. The case scenario follows the user’s looking at all the features – or specifying known features. After obtaining an overview of the data, the user can sort the features into those that are essential and the remaining features as softer constraints on the search. The user may have already encountered dichotomous features, but not knowing which to eliminate may continue to use both. Step-by-step the user will make decisions and compromises before selecting the phone with the features that he desires most and satisfy the search criteria. The last part of the search process will require many comparisons and iterations when exploring the information landscape with multiple dichotomous attributes.

As more *Filter* elements are added the complexity of the resulting lattice will most likely increase exponentially. To counter this complexity increase, which can make the diagram difficult to understand, elements of *Filter* can be promoted to *Zoom*. This will decrease the object set and decrease the number of attributes used to show structure of the data, which in turn reduces the complexity of the lattice diagram.

The advantage of having the structure as a lattice is that the user can ‘see’ relationships. Of these relationships it is easiest to see relations such as mutual exclusivity and implication. Figure 2 shows a simple lattice diagram. The user

can see that **mp3player:yes** and **chat:yes** are mutually exclusive (there are no phones with both an *MP3 player* and a *chat* function) because the point where the concepts join (reading the diagram downwards) has an extent size of 0. Also, it can be seen that **chat:yes** implies **games:yes** (every phone with a *chat* function also has *games*). In a search context, where the desired result has all query elements in the *Filter* list, it can be seen that there are 0 total matches (bottommost concept extent size is 0), but there are 7 phones that meet 2 of the query elements (the concepts directly above the bottommost concept).

Fig. 2. Diagram generated from the Phones database with **mp3player:yes**, **games:yes** and **chat:yes** as *Filter* elements.

Case Scenario One: The Exploration Method

We now demonstrate the ideas described in the previous section with respect to a more concrete interaction scenario. In this scenario, the user knows every feature considered important (and desirable) in a new cellphone. In this case scenario the user wants:

GPRS support	Infrared capabilities	Built-in MP3 player	Built-in organiser
Vibration alert	Voice-dial	WAP support	

The user adds all the corresponding attributes as ‘filter’ attributes. The resulting line diagram from these filter attributes is too large for the user to make an instant decision, but the line diagram gives an overview of the search space and it is possible to conclude the following from it:

1. The top-most concept has a contingent of 60, therefore there are 60 phones with none of the desired features.
2. The bottom-most concept has an empty extent, therefore there is no phone with all desired features.

3. The attributes `vibe:yes`, `voicedial:yes` and `wap:yes` are most common in this diagram³.

This knowledge leads the user to zoom on the three common attributes, which would seem a good way to reduce the complexity of the data while still maintain the majority of the phones.

The result above shows that at least one desired feature can not be kept, and that the selection is from 7 phones in 3 groups – each group has one of the desired features missing.

- The *Siemens SL 42*, *Siemens SL 45* and *Siemens SL 45i* do not have GPRS Support.
- The *Ericsson T65* does not have infrared capabilities.
- The *Motorola Accompli 008*, *Nokia 6310* and *Nokia 8310* do not have a built-in MP3 player.

At this point the user could decide that infrared capability is the least desired feature and opt for the *Ericsson T65* as the phone to purchase.

³ Recognizable by the fact they are darker in colour - indicating a large extent compared to other concepts. This is the coloring style used in [6].

Case Scenario Two: Attributes Addition Method

The user knows that two things he definitely wants in a cellphone are predictive text and infrared capabilities. He starts the search and adds **t9dict:yes** and **infrared:yes** as filter attributes.

This produces the simple lattice above showing 27 phones on the bottom concept. This means there are 27 phones with both predictive text and infrared capabilities. The list of 27 phones is too large for the user to reach a decision straight away so he promotes **t9dict:yes** and **infrared:yes** to zoom attributes. The user decides that an organiser and a long stand-by time are also important features which would influence his purchase decision, so adds the corresponding attributes as filters on the data. When adding stand-by time – the aim being to emphasis phones with a greater stand-by time – he configures the **stand-by** attribute to be ‘Ordinal Up’ resulting in the following diagram.

After looking at the generated lattice above, the user decides enough phones come with an organiser to warrant adding **organiser:yes** as a ‘zoom’ attribute. He realizes that a long stand-by time might come at the cost of increased phone

weight. To ensure that the phone he gets is not too heavy for his needs, he adds **weight** with the order ‘Ordinal Down’ so that lighter phones are emphasised resulting the in following diagram.

The diagram above allows the user to quickly choose an optimum weight/stand-by time combination. It is easy to see in the above diagram that the phone most suitable to the requirements specified is the *Nokia 8310*.

Further Research

At this point D-SIFT can perform the basic FCA operations against data quickly and dynamically. The final version of the user interface for the selection of mandatory attributes (*zooming*) is planned to be similar to TOSCANA and TOSCANA-J where by clicking a concept thereby selects the concept's intent as a restriction on the objects. This represents a minor implementation extension to the existing D-SIFT.

Furthermore, we are investigating the possibility of using the human input coded in conceptual scales from already existing Toscana Systems to support the user search and exploration. After parsing the .CSX file of a Toscana System, D-SIFT could “offer” groups of attributes as in Toscana Systems. Then the interaction can start from a given diagram, extending and changing it using the existing dynamic creation features of D-SIFT.

The last issue of further research addresses the use of the multi-context as introduced in [7,8]. On the basis of several contexts sharing sets of attributes or objects, the user would be enabled to “jump” from the perspective of one formal context to the corresponding perspective in another context by the use of coherence mappings.

Conclusion

This paper has presented the architecture of the D-SIFT browser and illustrates the resulting D-SIFT-systems on two case scenarios against a database

of cellular phones. The two examples demonstrate the generality of system integration outcomes from D-SIFT. The Conceptual Information Systems which result from applying the D-SIFT architecture present a new workflow for building and interacting with Formal Concept Analysis-based information systems. The workflow more closely aligns with dynamic schema interaction used in conceptual modeling.

References

1. Peter Becker, Richard J. Cole: Querying and analysing document collections with Formal Concept Analysis. In: Proceedings of the 8th Australasian Document Computing Symposium, Canberra (2003)
2. Hereth Correia, J., Kaiser, T.B.: A Mathematical Model for TOSCANA-Systems: Conceptual Data Systems. In Eklund, P., ed.: Concept Lattices, Heidelberg, Berlin, New York, Springer (2004)
3. Becker, P.: Multi-dimensional Representations of Conceptual Hierarchies. In Stumme, G., Mineau, G., eds.: Proceedings of the 9th International Conference on Conceptual Structures, Laval (2001)
4. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
5. Eklund, P., Ducrou, J., Brawn, P.: Concept lattices for information visualization: Can novices read line diagrams. In Eklund, P., ed.: Proceedings of the 2nd International Conference on Formal Concept Analysis - ICFCA'04, Springer (2004)
6. Becker, P., Hereth, J., Stumme, G.: ToscanaJ - an open source tool for qualitative data analysis. In: Advances in Formal Concept Analysis for Knowledge Discovery in Databases. Proc. Workshop FCAKDD of the 15th European Conference on Artificial Intelligence (ECAI 2002). Lyon, France. (2002)
7. Dörfllein, S.K., Wille, R.: Coherence Networks of Concept Lattices: The Basic Theorem. Number 3403 in LNCS, Heidelberg - Berlin - New York, Springer (2005) to appear in the proceedings of the ICFA05.
8. Wille, R.: Conceptual Structures of Multicontexts. In Eklund, P.W., Ellis, G., Mann, G., eds.: Conceptual Structures: Knowledge Representation as Interlingua. Number 1115 in LNAI, Heidelberg - Berlin - New York, Springer (1996) 23–29
9. Wille, R., Zickwolff, M.: Grundlagen einer Triadischen Begriffsanalyse. In Stumme, G., Wille, R., eds.: Begriffliche Wissensverarbeitung: Methoden und Anwendungen, Heidelberg - Berlin - New York, Springer (2000) 125–150
10. Rock, T., Wille, R.: Ein TOSCANA—Erkundungssystem zur Literatursuche. In Stumme, G., Wille, R., eds.: Begriffliche Wissensverarbeitung: Methoden und Anwendungen, Berlin-Heidelberg, Springer (2000) 239–253
11. Dieberger, A., Dourish, P., Höök, K., Resnick, P., Wexelblat, A.: Social navigation: Techniques for building more usable systems. *ACM Transactions on Human-Computer Interaction* **7** (2004) 26–58
12. Horvotz, E., Kadie, C., Paek, T., Hovel, D.: Models of attention in computing and communication: From principles to applications. *CACM* **46** (2003) 52–59
13. Carpineto, C., Romano, G.: Concept Data Processing: Theory and Practice. Wiley (2004)
14. Carpineto, C., Romano, G.: Order-theoretical ranking. *Journal of the American Society for Information Sciences (JASIS)* **7** (2000) 587–601

15. Cole, R., Eklund, P., Stumme, G.: Document retrieval for email search and discovery using formal concept analysis. *Journal of Experimental and Theoretical Artificial Intelligence* **17** (2003) 257–280
16. Cole, R., Eklund, P.: Browsing semi-structured web texts using formal concept analysis. In: *Proceedings 9th International Conference on Conceptual Structures*. Volume 2120 of LNAI., Berlin, Springer (2001) 319–332
17. Eklund, P., Ducrou, J., Brawn, P.: Concept lattices for information visualization: Can novices read line diagrams. In Eklund, P., ed.: *Proceedings of the 2nd International Conference on Formal Concept Analysis - ICFCA'04*, Springer (2004)
18. Wille, R.: Conceptual landscapes of knowledge: A pragmatic paradigm for knowledge processing. In Gaul, W., Locarek-Junge, H., eds.: *Classification in the Information Age*, Springer (1999) 344–356
19. Wille, R.: Introduction to Formal Concept Analysis. In Negrini, G., ed.: *Modelli e modellazione, Models and Modelling*, Roma, Consiglio Nazionale delle Ricerche, Istituto di Studi sulle Ricerche e Documentazione Scientifica (1997)
20. Colomb, R.: *Information Spaces: the Architecture of Cyberspace*. Springer (2002)
21. Kaiser, T.: *Conceptual Data Systems - Providing a Mathematical Basis for TOSCANA-Systems* (2003) Diploma thesis, University of Technology Darmstadt.
22. Wormuth, B.: *A Conceptual Information System for Topic Maps* (2004) Diploma thesis, University of Technology Darmstadt.
23. Davey, D., Priestly, H.: *Introduction to Lattices and Order*. second edn. Press Syndicate of the University of Cambridge (2002)