On-demand Relational Concept Analysis

Alexandre Bazin, Jessie Carbonnel, Marianne Huchard, Giacomo Kahn, Priscilla Keip, Amirouche Labib Ouzerdine

On-demand Relational Concept Analysis

Alexandre Bazin[1], Jessie Carbonnel[2], Marianne Huchard[2], Giacomo Kahn[3], Priscilla Keip[4], and Amirouche Ouzerdine[2]

[1] Université de Lorraine, CNRS, Inria, LORIA, F54000 Nancy, France
[2] LIRMM, CNRS and Université de Montpellier, Montpellier France
[3] Université d'Orléans, INSA Centre Val de Loire, LIFO, Orléans, France
[4] CIRAD, UPR AIDA, F-34398 Montpellier, France
AIDA, Univ Montpellier, CIRAD, Montpellier, France
`contact@alexandrebazin.com`, `jcarbonnel@lirmm.fr`, `huchard@lirmm.fr`
`giacomo.kahn@isima.fr`, `keip@lirmm.fr`, `ouzerdine@lirmm.fr`

Abstract. Formal Concept Analysis (FCA) and its associated conceptual structures are used to support exploratory search through conceptual navigation. Relational Concept Analysis (RCA) is an extension of Formal Concept Analysis to process relational datasets. RCA and its multiple interconnected structures represent good candidates to support exploratory search in relational datasets, as they are enabling navigation within a structure as well as between the connected structures. However, building the entire structures does not present an efficient solution to explore a small localised area of the dataset, to retrieve the closest alternatives to a given query. In these cases, generating only a concept and its neighbour concepts at each navigation step appears as a less costly alternative. In this paper, we propose an algorithm to compute a concept, and its neighbourhood, in connected concept lattices. The concepts are generated directly from the relational context family, and possess both formal and relational attributes. The algorithm takes into account two RCA scaling operators and it is implemented in the RCAExplore tool.

Keywords: Relational Concept Analysis, Formal Concept Analysis, Exploratory Search, On-demand Generation, Local Generation

1 Introduction

Many datasets in thematic areas like environment or product lines comprise databases complying with a relational data model. Typical applications in which we are currently involved concern issues relative to watercourse quality[5] (Fresqueau project), the inventory and use of pesticidal, antibacterial and antifungal plants[6] (Knomana project), and the analysis and representation of complex product lines [6]. In these applications, there is a wide range of question forms, such as classical querying, establishing correlations between descriptions of objects from

[5] http://dataqual.engees.unistra.fr/
[6] http://www.cirad.fr/en/news/all-news-items/articles/2017/science/identifying-plants-used-as-natural-pesticides-in-africa-knomana

several categories or case based reasoning. These questions can be addressed by complementary approaches including conceptual classification, knowledge pattern and rule extraction, or exploratory search [23, 26]. In the Knomana project, for example, one main purpose will be, after the ongoing plant inventory, to support farmers, their advisors, local entrepreneurs or researchers in selecting plants of immediate interest for agricultural crop protection and animal health. As users will face large amounts of data with no prior knowledge of the data, and will mainly formulate potentially imprecise queries, without an immediate answer, exploratory search seems to be a suitable approach.

Previous work [17, 7, 10, 14, 11] has shown that Formal Concept Analysis (FCA) is a relevant support for data exploration. FCA is a data analysis framework organising datasets composed of objects described by attributes in canonical structures called concept lattices, which have good properties for exploratory search. We expect Relational Concept Analysis (RCA), an FCA extension to handle several datasets connected by relationships, to be beneficial as well. Considering RCA for relational dataset exploration brings issues relative to (a) the use of scaling (logical) operators describing the relationships, (b) the iterative process to build the concept lattices, and (3) the presence of several concept lattices connected via relational attributes. Despite this additional complexity, RCA helps the user to concentrate on the classification of objects of several categories, where the object groups (concepts) are described by intrinsic attributes and by their relations to object groups of other categories. Besides, the relational attributes offer a support to navigate between the object groups of the different categories, while the concept lattices offer a (by-specialisation) navigation between object groups of the same category.

There are several complementary strategies to implement dataset exploration using RCA. One may consist in exhaustively computing concept lattices (and related artefacts like implication rules) at several steps, using several logical operators and considering only some of the object categories and some of the inter-categories relationships [5]. Another strategy, which is followed here, consists in an on-demand computation of a concept and its neighbourhood comprising its upper, lower and relational covers.

The next section presents the main principles of Relational Concept Analysis (Section 2). The on-demand computation of a concept and its neighbourhood is presented in Section 3, and its integration in the RCAExplore tool is presented in Section 4. Section 5 illustrates the algorithm with the example introduced in Section 2. Related work is exposed in Section 6. We conclude the paper with a few perspectives in Section 7.

2 Relational Concept Analysis

Formal Concept Analysis (FCA) [15] is a data analysis framework to structure objects described by attributes in a canonical structure called a concept lattice. It revolves around the notions of formal *contexts* and *concepts*.

Definition 1. *A* formal context *is a triple* $\mathbb{K} = (\mathcal{O}, \mathcal{A}, \mathcal{I})$ *in which* $\mathcal{O}$ *is a set of* objects*,* $\mathcal{A}$ *a set of* attributes*, and* $\mathcal{I} \subseteq \mathcal{O} \times \mathcal{A}$ *an* incidence relation *stating "which objects possess which attributes".*

Two *derivation operators* (both denoted by $(\cdot)'$) are defined on a formal context $(\mathcal{O}, \mathcal{A}, \mathcal{I})$. Let O be a set of objects and A a set of attributes. Then:

$$O' = \{a \in \mathcal{A} \mid \forall o \in O, (o, a) \in \mathcal{I}\}$$

$$A' = \{o \in \mathcal{O} \mid \forall a \in A, (o, a) \in \mathcal{I}\}$$

The composition of these two operators forms a *Galois connection* and thus a closure operator. We say that a set S is *closed* when $S = S''$.

Definition 2. *The attribute (resp. object) set* X *is called a* generator *of the set* X''*. It is a* minimal generator *if none of its subsets have the same closure.*

Definition 3. *A* formal concept *is a pair* $(A, B) \in 2^{\mathcal{O}} \times 2^{\mathcal{A}}$ *such that* $A = B'$ *and* $B = A'$*. The set* A *is called the* extent *and* B *the* intent *of the concept.*

A concept is thus a maximal set of objects sharing a maximal set of attributes and corresponds to a maximal rectangle of crosses (up to permutation of rows and columns) when the context is represented as a crosstable. The set $\mathcal{C}_{\mathbb{K}}$ of all concepts of the context $\mathbb{K}$, together with the order induced by the set-inclusion relation (denoted by $\leq_s$) on either of the components (but usually the extents), forms a complete lattice called the *concept lattice* of $\mathbb{K}$. Figure 1 depicts two concept lattices. It uses a condensed representation in which an attribute (resp. an object) is introduced in the greatest (resp. the smallest) concept having this attribute in the lattice.

Thus, a concept inherits the attributes from its super-concepts, and the objects from its sub-concepts. For instance, concept *C_DM_tools_0* of the concept lattice on the left-hand side of Fig. 1 has for intent {*OS:Windows, DM:Conceptual*}, and for extent {*Astah, ErwinDM, Magic Draw, ER/Studio*}.

Definition 4. *Let* (A, B) *be a formal concept of a formal context* $\mathbb{K}$*. A formal concept* (A_2, B_2) *is called a* cover *of* (A, B) *if* A *and* A_2 *are comparable and there is no concept* (A_3, B_3) *with* A_3 *between* A *and* A_2*. It is an* upper cover *when* $A \subset A_2$ *and a* lower cover *otherwise.*

Definition 5. *A concept is an* object-concept *(resp.* attribute-concept*) if it is the smallest (resp. greatest) concept which extent (resp. intent) contains a particular object (resp. attribute).*

In Fig. 1 (left-hand side), *C_DM_tools_0* is an attribute-concept, while *C_DM_tools_2* is an object-concept, and *C_DM_tools_3* is both.

Relational Concept Analysis (RCA) [18, 19] is an adaptation of FCA to process relational datasets. A relational dataset is composed of several categories of objects described by both their own (intrinsic) attributes and their relationships with objects of other categories. As input, RCA takes a Relational Context

Family (RCF), gathering a set of formal contexts (representing the different categories of objects and their attributes) and a set of relational contexts (representing relationships between objects of different categories), where each relational context defines links between the objects of two formal contexts.

Definition 6 (Relational Context Family). *A Relational Context Family is a pair* $(\boldsymbol{K}, \boldsymbol{R})$ *such that:*
- $\boldsymbol{K} = \{\mathbb{K}_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)\}$ *is a set of formal contexts (object-attribute relations)*
- $\boldsymbol{R} = \{r_k\}, r_k \subseteq \mathcal{O}_i \times \mathcal{O}_j$ *is a set of relational contexts (object-object relations), with* $\mathcal{O}_i$ *being the set of objects of context* $\mathbb{K}_i$ *(the source context) and* $\mathcal{O}_j$ *being the set of objects of context* $\mathbb{K}_j$ *(the target context).*

Table 1. (top) Two formal contexts: (left-hand side) Data Modelling tools (DM_tools) and (right-hand side) DataBase Management Systems (DBMS). (bottom) Relational context stating which DM_tools *support* which DBMS

$\mathbf{K}_s =$

DM_ tools	OS:Windows	OS:Mac OS	OS:Linux	DM:Conceptual	DM:Physical	DM:Logical	DM:ETL
Astah	×	×	×	×			
Erwin DM	×			×	×	×	
ER/Studio	×			×	×	×	×
Magic Draw	×	×	×	×	×	×	
MySQL Workbench	×	×	×		×		

DBMS	DT:Enum	DT:Set	DT:Geometry	DT:Spatial	DT:Audio	DT:Image	DT:Video	DT:XML	DT:JSON	DT:Period
MySQL	×	×	×							
Oracle				×	×	×	×	×		
PostgreSQL	×		×					×	×	
Teradata	×		×					×	×	×

$\mathbf{R}_s =$

support	MySQL	Oracle	PostgreSQL	Teradata
Astah	x	x		
Erwin DM	×	×		×
ER/Studio	×	×	×	×
Magic Draw	×	×	×	
MySQL Workbench	×			

The three contexts of Table 1 present an example of an RCF $(\mathbf{K}_s, \mathbf{R}_s)$ taken from the software product line domain. Table 1 (top) displays two formal contexts. The one on the left-hand side presents five Data Modelling tools (*DM_ tools*) against seven attributes representing their compatible operating systems (*OS:*), and the data models (*DM:*) the tools may manage. The table on the right-hand side describes four DataBase Management Systems (*DBMS*) according to the data types (*DT:*) they may handle. Table 1 (bottom) presents a relational context stating which Data Modelling tools *support* which DataBase Management Systems.

The RCA process uses two steps iteratively. The first is building the concept lattices of all the formal contexts in $\mathbf{K}$, ignoring the relations contained in the relational contexts. The two concept lattices associated with Table 1 (top) are presented in Fig. 1. The second step is taking the relations into account by augmenting the formal contexts with new attributes that represent information about those relations. These *relational attributes* take the form of a $\rho r.C$

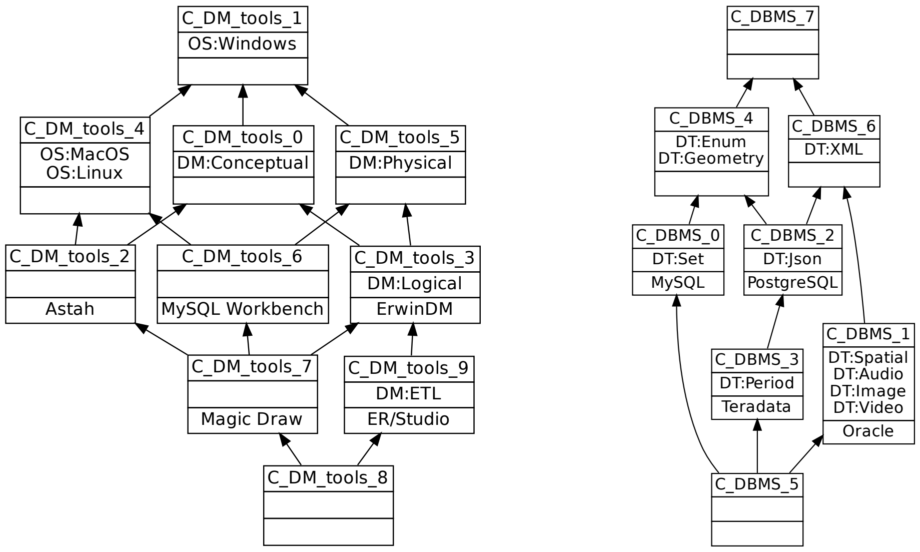

Fig. 1. (left) concept lattice of DM_tools, (right) concept lattice of DBMS

construct in which ρ is a *scaling operator* (sometimes called *quantifier*), r is a relation in the RCF and C is a formal concept of the lattice of the target of r built in the first step. In our example, we may introduce the relational attribute $\exists\ support.(C\_DBMS\_2)$ to characterise the *DM_ tools* that support at least one *DBMS* offering Json and XML. Given two formal contexts $\mathbb{K}_i, \mathbb{K}_j \in \mathbf{K}$ and a relational context $r \subseteq \mathcal{O}_i \times \mathcal{O}_j$, the application of RCA extends the set of attributes $\mathcal{A}_i$ with a set of relational attributes representing links to the concepts of $\mathbb{K}_j$. The extended attribute set is denoted by $\mathcal{A}_i^+$ and the incidence relation $\mathcal{I}_i$ is extended and denoted $\mathcal{I}_i^+$ to take these new attributes into account by associating them to objects of $\mathcal{O}_i$ depending on the relation r, the concept C involved in the relational attribute and the scaling operator ρ. In this paper, we focus on two scaling operators: the *existential* operator $\exists$, associating an object o to the relational attribute $\exists r.(C)$ if o is linked to at least one object of the extent of C by r, and the *universal strict* operator $\exists\forall$, associating an object o to $\exists\forall r.(C)$ if all the objects linked to o by r are included in the extent of C, and $r(o) \neq \emptyset$.

Table 2 shows *DM_ tools*$^+$, the formal context about *DM_ tools* extended by the relational context *support* and the scaling operator $\rho = exist$. According to the relation *support*, the Data Modelling tool `Magic Draw` supports the DBMS `PostgreSQL`. Now let us consider the concept *C_ DBMS_ 2* in the concept lattice (right-hand side) of Figure 1. Its extent contains `PostgreSQL`. As one object in the concept's extent is linked to `Magic Draw` through the relation *support*, then `Magic Draw` is associated with the relational attribute $\exists\ support(C\_DBMS\_2)$.

Table 2. Formal context *DM_ tools* extended according to the relation *support*

DM_tools^{+}	OS:Windows	OS:Mac OS	OS:Linux	DM:Conceptual	DM:Physical	DM:Logical	DM:ETL	∃ support(C_DBMS_3)	∃ support(C_DBMS_1)	∃ support(C_DBMS_0)	∃ support(C_DBMS_2)
Astah	×	×	×	×					×	×	
Erwin DM	×			×	×	×		×	×	×	×
ER/Studio	×			×	×	×	×	×	×	×	×
Magic Draw	×	×	×	×	×	×			×	×	×
MySQL Workbench	×	×	×		×					×	

The concept lattice associated with a formal context $\mathbb{K}^{+} = (\mathcal{O}, \mathcal{A}^{+}, \mathcal{I}^{+})$ then structures the objects from $\mathcal{O}$ both by their attributes and their relations to other sets of objects through the relational attributes. Figure 2 presents the extended concept lattice corresponding to the extended formal context *DM_ tools*$^{+}$, according to the relation *support* and the *existential* scaling operator.

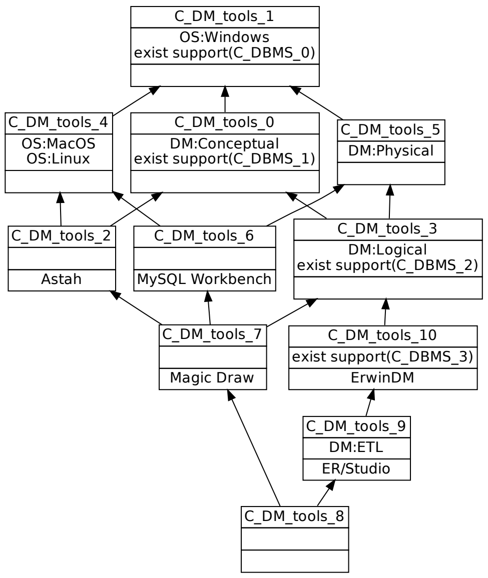

Fig. 2. Concept lattice of the extended context *DM_ tools*$^{+}$

In this way, for complex data models including more than one relation, RCA produces a succession of concept lattices, extended at each step by the new abstractions obtained at the previous step. Let $\mathbf{K}^0 = \mathbf{K} = \{\mathbb{K}_i^0\}$ denote the contexts of the initial RCF and $\mathbf{L}^0$ the family of their lattices. We will use $\mathbf{K}^n = \{\mathbb{K}_i^{(n-1)+}\}$ and $\mathbf{L}^n$ to denote the state of the contexts and lattices after n applications of the previously described two steps.

In the following, we introduce some exploration algorithms in a RCF. In order to do so, we define the notion of relational cover of a concept, and we also use the notion of *transversal* from hypergraph theory.

Definition 7. *Let $\mathbb{K}_i^n$ be a formal context in the RCF $(\boldsymbol{K}^n, \boldsymbol{R})$ and (A, B) a formal concept of $\mathbb{K}_i^n$. A* relational cover *of (A, B) is a concept C such that $\rho r.C \in B$.*

Definition 8. *Let $\mathcal{F}$ be a family of subsets of a ground set G. A* transversal *is a set of elements of G that intersects every set of $\mathcal{F}$. A transversal is* minimal *when none of its proper subsets is a hitting set (i.e. a transversal itself).*

3 The Exploration Algorithm

In this section, we present algorithms for computing a concept from connected lattices, along with its neighbouring concepts. It enables exploration by minimal steps, which show the closest alternatives to the concept representing a given query. Then, the user may choose one of these alternatives and pursue his/her exploration (i.e., taking a step in the exploration) by investigating the neighbouring concepts, which can be in the same lattice, or in a connected one.

With the example given on Data Modelling tools, let us consider a user who wants to find a tool for Linux (`OS:Linux`) dedicated to physical data models `DM:physical`. It is easy to compute independently the concept $C\_DM\_tools\_6$ from Fig. 2 gathering the corresponding tools. Let us suppose now that the user is not satisfied by the proposed tools, they can consider removing a constraint (like using Linux or MacOS) which leads them to consider $C\_DM\_tools\_5$. Reversely, the user may want to add constraints like (1) a tool also considering logical data models (`DM:Logical`); (2) or a tool admitting Json data types (`DT:Json`) from concept C_DBMS_2. In these two cases, considering $C\_DM\_tools\_3$ is relevant. Thus, the purpose of the algorithm is to show $C\_DM\_tools\_6$ and all possible similar solutions present and gathered in the neighbouring concepts. It aims to avoid the inherent complexity and running time of computing the whole lattices, and it is also designed to assist users during their data exploration task.

The algorithm takes as input a RCF $(\mathbf{K}^n, \mathbf{R})$ and a formal concept $C = (A, B) \in \mathbf{L}^n$. Its output is the concept $(A, B^+) \in \mathbf{L}^{(n+1)}$ and its upper, lower and relational covers, where B^+ is B enhanced with relational attributes. Meanwhile, the RCF is updated with the relational attributes for a next step.

3.1 Redefining Derivation Operators

In order to compute the new concept and its covers in $\mathbf{L}^{(n+1)}$, we need the contexts of $\mathbf{K}^{n+1}$. The explicit knowledge of all the relational attributes of those contexts would require the computation of all the concepts in the contexts of the input $\mathbf{K}^n$. This is too time-consuming. We would prefer to manipulate only a minimal number of relational attributes allowing us to derive, on-the-fly, the other relational attributes.

Any object described by an attribute $\rho r.(X,Y)$ is also necessarily described by all the attributes of the form $\rho r.(X_2,Y_2)$ such that $(X_2,Y_2) \in \mathbf{L}^n$ with $X \subseteq X_2$ and/or $Y_2 \subseteq Y$. As such, intents can be represented without loss of information by their relational attributes constructed from attributes-wise maximal concepts. However, two problems arise with such a representation: (1) the set intersection cannot be used to compute the intent of a set of objects anymore, and (2) if only maximal relational attributes are explicitly present in the context, the extent of a set of attributes cannot be computed through a simple test of set inclusion. To remedy this, we provide three algorithms to use on sets of attributes (both intrinsic and relational) with only the maximal relational attributes given explicitly.

The Ex algorithm (Algorithm 1) computes the extent, in the formal context $\mathbb{K}_i$, of an attribute set A represented by its maximal relational attributes. For each object o and attribute $\rho r.(X,Y) \in A$, it checks whether $r(o)$ and X intersect in the correct way (depending on the scaling operator). The algorithm runs in $O(|\mathcal{O}_i| \times |A| \times |\mathcal{O}_j|)$ where $\mathcal{O}_j$ is the biggest set of objects in the RCF.

Algorithm 1: Ex($\mathbb{K}_i, A$)

```
  Input: $\mathbb{K}_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $A \subseteq \mathcal{A}_i$ a set of attributes
  Output: Computes the extent of a set of attributes $A$
1 $O \leftarrow \mathcal{O}_i$
2 foreach $a \in A$ do
3     if $a \sim \exists\forall r.(X,Y)$ then
4         $O \leftarrow \{o \in O \mid r(o) \subseteq X\}$
5     else if $a \sim \exists r.(X,Y)$ then
6         $O \leftarrow \{o \in O \mid r(o) \cap X \neq \emptyset\}$
7     else
8         $O \leftarrow \{o \in O \mid (o,a) \in \mathcal{I}_i\}$
9 return O
```

Let A and B be two attribute sets represented by their maximal relational attributes. A relational attribute $\exists r.(X,Y)$ is in the intersection of A and B if and only if there exist two attributes $\exists r.(X_2,Y_2) \in A$ and $\exists r.(X_3,Y_3) \in B$ such that $X \subseteq X_2$ and $X \subseteq X_3$. The same holds for the $\exists\forall$ scaling operator. The Intersect algorithm (Algorithm 2) uses this property to compute the maximal relational attributes of the intersection of A and B by recursively intersecting the intents of the concepts used to build the relational attributes in A and B. The fact that, at any given time, the depth of all the relational attributes is bounded ensures that the recursion ends. The algorithm runs in $O((|\mathcal{A}_i|^2 \times Q_{Ex}(\mathbb{K}_j, \mathcal{A}_j))^{n+1})$ with $\mathcal{A}_i$ the biggest attribute set in the RCF, $Q_{Ex}(\mathbb{K}_j, \mathcal{A}_j)$ the complexity of Algorithm 1 with, as inputs, $\mathbb{K}_j$ and $\mathcal{A}_j$ such that $\mathbb{K}_j$ is the context in the RCF for which $|\mathcal{O}_j| \times |\mathcal{A}_j|$ is the biggest and n the maximal depth of a relational attribute, i.e. the current number of times RCA's steps have been applied.

Algorithm 2: INTERSECT($\mathbb{K}_i, A, B$)

```
Input: $\mathbb{K}_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $A, B \subseteq \mathcal{A}_i$ two attribute sets
Output: The relational intersection of $A$ and $B$
1  $R \leftarrow A \cap B$
2  $\mathcal{F} \leftarrow \emptyset$
3  foreach $a_1 \sim \exists r.(X_1, Y_1) \in B$ with $r \subseteq \mathcal{O}_i \times \mathcal{O}_j$ and $\mathbb{K}_j = (\mathcal{O}_j, \mathcal{A}_j, \mathcal{I}_j)$ do
4      foreach $a_2 \sim \exists r.(X_2, Y_2) \in A$ do
5          $\mathcal{F} \leftarrow \mathcal{F} \cup \{\exists r.(\text{EX}(\mathbb{K}_j, \text{INTERSECT}(\mathbb{K}_j, Y_1, Y_2)), \text{INTERSECT}(\mathbb{K}_j, Y_1, Y_2))\}$
6  $R \leftarrow R \cup \text{MAX}(\mathcal{F}, \subseteq_{\mathcal{A}_i})$
7  $\mathcal{F} \leftarrow \emptyset$
8  foreach $a_1 \sim \exists\forall r.(X_1, Y_1) \in B$ with $r \subseteq \mathcal{O}_i \times \mathcal{O}_j$ and $\mathbb{K}_j = (\mathcal{O}_j, \mathcal{A}_j, \mathcal{I}_j)$ do
9      foreach $a_2 \sim \exists\forall r.(X_2, Y_2) \in A$ do
10         $\mathcal{F} \leftarrow$
             $\mathcal{F} \cup \{\exists\forall r.(\text{EX}(\mathbb{K}_j, \text{INTERSECT}(\mathbb{K}_j, Y_1, Y_2)), \text{INTERSECT}(\mathbb{K}_j, Y_1, Y_2))\}$
11 $R \leftarrow R \cup \mathcal{F}$
12 return $R$
```

IN (Algorithm 3) uses INTERSECT to compute the intent, in the formal context $\mathbb{K}_i$, of a set O of objects described by their maximal relational attributes. It starts with the set of all explicitly known attributes and intersects it with the description of each object in the context $\mathbb{K}_i$. It runs in $O(|O| \times Q_{Int})$ where Q_{Int} is the complexity of INTERSECT.

Algorithm 3: IN($\mathbb{K}_i, O$)

```
Input: $\mathbb{K}_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $O \subseteq \mathcal{O}_i$ a set of objects
Output: Computes the intent of a set of objects $O$
1  $A \leftarrow \mathcal{A}_i$
2  foreach $o \in O$ do
3      $A \leftarrow \text{INTERSECT}(A, Intent(\{o\}))$
4  return A
```

3.2 Computing the Closed Neighbourhood

Now that we have redefined the derivation operators on implicitly known relational contexts, we are able to compute the upper, lower and relational covers of a concept.

The easiest are the relational covers. A concept (X, Y) is a relational cover of a concept (U, V) if and only if $\rho r.(X, Y)$ is a maximal relational attribute in V. Upper covers are easy too. Candidates can be generated by adding an object – the set of which we have perfect knowledge of – to the current extent and computing the corresponding concept. The covers are the candidates that have the smallest extent. Computing the lower covers is more challenging. They could be computed by adding attributes to the intent but the full set of relational attributes is only known implicitly. We chose to, instead, remove objects. The lower covers of (X, Y) being the concepts with the maximal extents that are contained in X and do not contain any of the minimal generators of X, a simple way to compute them would be to remove minimal transversals of the minimal generators.

```
Algorithm 4: GrowContext($\mathbb{K}_i, r, \rho, o, OC_j$)
Input: $\mathbb{K}_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $r \subseteq \mathcal{O}_i \times \mathcal{O}_j$ a relational context, $\rho$
       a scaling operator, $o \in \mathcal{O}_i$ an object, $OC_j$ the set of object-concepts of
       $\mathbb{K}_j = (\mathcal{O}_j, \mathcal{A}_j, \mathcal{I}_j)$
Output: Extends the context $\mathbb{K}_i$ and adds the crosses, giving $\mathbb{K}_i^+$
1 if $\rho == \exists$ then
2     foreach $(X, Y) \in OC_j$ such that $r(o) \cap X \neq \emptyset$ do
3         $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \exists r.(X, Y)$
4         $\mathcal{I}_i \leftarrow \mathcal{I}_i \cup (o, \exists r.(X, Y))$
5 if $\rho == \exists\forall$ then
6     $Y \leftarrow$In$(\mathbb{K}_j, r(o))$
7     $X \leftarrow$Ex$(\mathbb{K}_i, Y)$
8     $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \exists\forall r.(X, Y)$
9     $\mathcal{I}_i \leftarrow \mathcal{I}_i \cup (o, \exists\forall r.(X, Y))$
```

Algorithm 4 (GrowContext) takes as input a context, a relation, a scaling operator, an object o and the set of object-concepts [2] of o. It constructs new relational attributes, adds them to the context and completes the incidence relation accordingly. For the scaling operator $\exists$, each relation r and each object-concept (X, Y) in the target of r give rise to a new relational attribute $\exists\ r.(X, Y)$ that is added to the context $\mathbb{K}_i$. As object-concepts contain the irreducible elements of the lattice, relational attributes constructed with them are enough to reconstruct all the other possible relational attributes. For the scaling operator $\exists\forall$, the new added relational attribute $\exists\forall\ r.C$ is built thanks to Algorithms 3 and 1, roughly speaking by computing the intent of $r(o)$ and the associated extent.

Algorithm 5 computes the closed neighbourhood of a concept C. It takes as input a set of formal contexts $\mathbf{K} = (\mathbb{K}_1, \ldots, \mathbb{K}_w)$ of a RCF, a strategy $\mathcal{S} = \{(r,\rho)_{lj}, \ldots\}$, $l, j \in \{1, \ldots, w\}$ and a starting concept C from a context $\mathbb{K}_i$. A strategy is an assignment of scaling operators to relations, which corresponds to a user choice. The goal is to compute (or complete) the intent corresponding to the extent of C, as well as its upper, lower and relational covers, in the extended context $\mathbb{K}_i^+$. For each $(r,\rho)_{ij} \in \mathcal{S}$ such that $r : \mathbb{K}_i \mapsto \mathbb{K}_j$, the first loop (Lines 1 to 4) computes OC_j the object-concepts of $\mathbb{K}_j$ and calls Algorithm 4 to obtain the extended context. In Line 5, the intent of concept C is extended with the relational attributes added during the previous loop. The next loop (Lines 6 to 8) computes the relational covers $\mathcal{R}$ of concept C. For each relational attribute in the intent of C, the corresponding concept (in the target context) is added to the cover. In Lines 9 to 11, the lower covers $\mathcal{L}$ of C are computed by removing from the extent of C a minimal transversal of the set of minimal generators of C's extent. Finally, the upper covers $\mathcal{U}$ of C are computed in Lines 12 to 14. Candidates are created by adding an object o to the extent of C. Only the extent-wise minimal resulting concepts are kept.

Algorithm 5: INCREMENTAL($\mathbf{K}, \mathcal{S}, C, \mathbb{K}_i$)

```
   Input: $\mathbf{K} = \{\mathbb{K}_1, \ldots, \mathbb{K}_w\}$, $\mathcal{S} = \{(r,\rho)_{lj}, \ldots\}, l, j \in \{1, \ldots, w\}$ a strategy,
          $C = (O, A)$ a concept of $\mathbb{K}_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$
   Output: $C, \mathcal{U}, \mathcal{R}, \mathcal{L}$ the completed concept $C$ and its closed relational
           neighbourhood
1  foreach $(r,\rho)_{ij} \in \mathcal{S}$ do
2      $OC_j \leftarrow$ ObjectPoset($\mathbb{K}_j$)
3      foreach $o \in \mathcal{O}_i$ do
4          GrowContext($\mathbb{K}_i, r, \rho, o, OC_j$)
5  $A \leftarrow$ In($\mathbb{K}_i, O$)
6  $\mathcal{R} \leftarrow \emptyset$
7  foreach $a \sim \rho r.(X_1, Y_1) \in A$ do
8      $\mathcal{R} \leftarrow \mathcal{R} \cup \{(X_1, Y_1)\}$
9  $\mathcal{L} \leftarrow \emptyset$
10 foreach $T \in minTrans(minGen(O))$ do
11     $\mathcal{L} \leftarrow \mathcal{L} \cup \{(O \setminus T, \text{In}(\mathbb{K}_i, O \setminus T))\}$
12 $\mathcal{U} \leftarrow \emptyset$
13 foreach $o \in \mathcal{O}_i \setminus O$ do
14     $\mathcal{U} \leftarrow \mathcal{U} \cup \{(\text{Ex}(K_i, \text{In}(\mathbb{K}_i, O \cup \{o\})), \text{In}(\mathbb{K}_i, O \cup \{o\}))\}$
15 $\mathcal{U} \leftarrow$ Min($\mathcal{U}, \subseteq_{\mathcal{O}_i}$)
16 return $C, \mathcal{U}, \mathcal{R}, \mathcal{L}$
```

4 Implementation

In this section we present RCAExplore[7], the tool in which the proposed algorithms are integrated. We quote the different algorithms that helped for the implementation, and we explain how they concretely work in the RCAExplore tool. A version of RCAExplore with the implemented algorithms is currently available on demand to the authors and will soon be available online.

RCAExplore is a framework for FCA and RCA. It provides to the user the ability to finely tune the RCA process. At each step, the user can select the formal contexts and the relational contexts that interest them and modify their content if they want. The user can also select one or more quantifiers to process each relation, and choose for each selected formal context a particular algorithm to build the associated conceptual structure. RCAExplore implements 5 different algorithms (*fca, iceberg, ares, acposet* and *ocposet*) to build the corresponding conceptual structures. *fca* builds a concept lattice, *iceberg* builds an iceberg lattice [21], *ares, acposet* and *ocposet* respectively build the partially ordered sets induced by the concepts that introduce objects or attributes (AOC-poset), attributes (AC-poset) or objects (OC-poset). The *ocposet* algorithm is called for each relation to extend the source formal context (Algorithm 5, Line 2).

The on-demand algorithms proposed in this paper build only a part of the concept lattice (i.e., a concept and its neighbours) on which the user wants to focus on. It is integrated into RCAExplore as a new way of navigating, where the user gradually progresses into the lattice without getting lost. First, the user selects the formal context from which they want to start the navigation. This formal context must have at least one attribute or one relation toward another formal context of the RCF. Then the user selects the attribute set around which they want to build the first concept. For this concept, the relational, upper and lower covers are computed as mentioned in Section 3. To compute the lower cover, we needed to compute minimal transversals of minimal generators (Algorithm 5, Lines 10,11). For this purpose, we implemented the MTMiner algorithm [20]. Once covers are calculated, the user can choose another concept among those calculated to continue the navigation. If this new concept contains a relational attribute, the user can either switch to the corresponding lattice and build what is around the concept introducing this relational attribute, or stay in the same lattice and resume the process.

5 Illustrative Example

In this section, we illustrate the defined algorithms and we briefly introduce the real case studies on which we tested them. We consider the RCF $(\mathbf{K}_s, \mathbf{R}_s)$ with $\mathbf{K}_s = \{DM\_tools, DBMS\}$ and $\mathbf{R}_s = \{support\}$ presented in Section 2. We apply the strategy $\{(support, \exists)\}$.

[7] http://dataqual.engees.unistra.fr/logiciels/rcaExplore

User Request (step 1) Let us consider a user who initially wants to select a data modelling tool that runs on Windows (*OS:Windows*) and that handles conceptual data models (*DM:Conceptual*). Traditional FCA may compute the formal concept associated with these 2 attributes (i.e., *C_DM_tools_0*, left-hand side of Fig. 1) and inform the user that the corresponding tools are `Astah`, `Erwin DM`, `Magic Draw` and `ER/Studio`. Now the user wants to get insight into the neighbourhood of the proposed solution, e.g. to know what are the specificities of the tools in case the functional needs or the working environment evolves.

Algorithm (run 1) Let us apply our algorithms on this concept to 1) retrieve the supported DBMS (relational cover) and 2) find the closest alternatives to the query (lower and upper covers). We call Algorithm 5 with the following parameters: $\textsc{Incremental}(\mathbf{K}_s, \{(support, \exists)\}, C\_DM\_tools\_0, DM\_tools)$.

In Algorithm 5, Lines 1 to 4 extend the context of *DM_tools* with the relational attributes representing the object-concepts of *DBMS* (*support*'s target context). In our case, we have only one relation (*support*, ∃) visited at Line 1. In Line 2, OC_j takes the object-concepts of *DBMS*, i.e., concepts 0, 1, 2 and 3 from the concept lattice on the right-hand side of Fig. 1: the whole OC-poset is shown in the right-hand side of Fig. 3. Then, the loop on Lines 3 and 4 considers the 5 objects of *DM_tools*, on which GrowContext is called. Each object o_i of *DM_tools* is associated to the relational attributes representing the concepts of OC_j having in their extents at least one object linked to o_i.

As $support(\texttt{Astah}) = \{MySQL, Oracle\}$, the relational attributes $\exists support(C\_DBMS\_0)$ (corresponding to *MySQL* object-concept) and $\exists support(C\_DBMS\_1)$ (corresponding to *Oracle* object-concept) are added to *DM_tools* and associated to `Astah`. We apply the same process with the other objects. After Line 4 execution, we obtain the extended context presented in Table 2.

Line 5 updates the intent of the input concept to add the relational attributes: the intent of *C_DM_tools_0* is now {*OS:Windows, DM:Conceptual,* $\exists support(C\_DBMS\_0)$, $\exists support(C\_DBMS\_1)$}. The concepts of *DBMS* corresponding to the added relational attributes, i.e. the concepts C_DBMS_0 and C_DBMS_1, form the relational cover of the input concept (Lines 6 to 8).

Then, (Lines 9 to 11) we compute the minimal generators of the extent of *C_DM_tools_0*, which are {`Erwin DM`, `Astah`} and {`ER/Studio, Astah`}. Their minimal transversals are {`Astah`} and {`Erwin DM,ER/Studio`}. The two concepts having {`Astah,Magic Draw`} and {`Erwin DM,ER/Studio,Magic Draw`} for extent represent the lower cover of *C_DM_tools_0* (respectively *C_DM_tools_2* and *C_DM_tools_3* in Fig. 2).

Finally, in Lines 12 to 14, we consider the objects of *DM_tools* that are not in *C_DM_tools_0*'s extent, i.e., `MySQL Workbench`. For each one of them, we compute the concept corresponding to their union with *C_DM_tools_0*'s extent, and we obtain in this case the concept *C_DM_tools_1* of Fig. 2, which represents the upper cover of *C_DM_tools_0*.

With the implementation of the algorithms in RCAExplore, we obtain, as output, the concept $C\_DM\_tools\_0$ and its relational, upper and lower covers as shown in left-hand side of Fig. 3, where the input concept is highlighted.

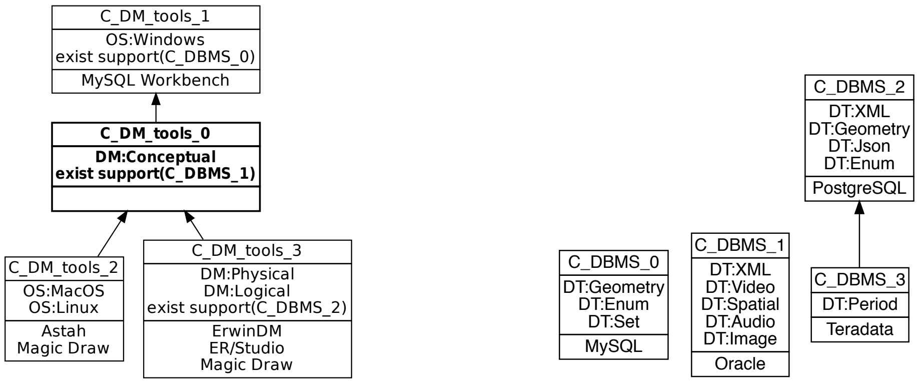

Fig. 3. (left) first iteration on DM_tools, (right) OC-poset on DBMS (results obtained with RCAExplore extension to on-demand algorithm)

User Request (step 2) The user can now select another concept from which they want to continue the navigation, e.g. $C\_DM\_tools\_1$, because they think that the constraints of handling conceptual data models (*DM:Conceptual*) may not be so important.

Algorithm (run 2) In this case, the incremental algorithm is called again on the chosen concept, and computes its covers as shown in left-hand side of Fig. 4. In this example, the concept $C\_DM\_tools\_1$ has no upper cover, being the top concept of the lattice, as we can confirm thanks to Fig. 2. The concept $C\_DM\_tools\_2$ is a sub-concept of $C\_DM\_tools\_4$ but their connection is not visible in Fig. 4 because their covers have not been computed yet. By selecting one of those concepts, their covers will be computed and the connection will be visible.

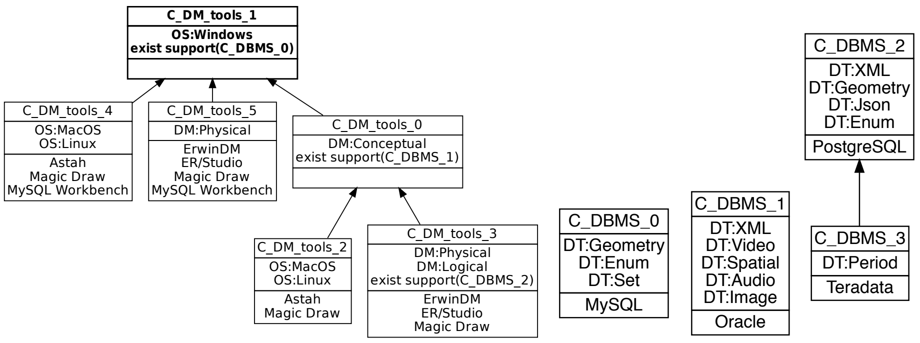

Fig. 4. (left) second iteration on DM_tools, (right) OC-poset on DBMS (results obtained with RCAExplore extension to on-demand algorithm)

User Request (step 3) and Algorithm (run 3) In the follow-up, the user could continue exploring the concepts of DM_tools, or they could jump to a relational cover, which is one concept of $DBMS$, e.g. from $C\_DM\_tools\_1$ to C_DBMS_0. In this case, the user focuses on the kind of DBMS that are supported by the group of considered DM_tools. The algorithm will build the covers of C_DBMS_0, that are C_DBMS_4 and C_DBMS_5, from Fig. 1 (right-hand side). There will be no relational cover, as $DBMS$ is not the source of an object-object relation.

6 Related Work

Lattice structures are among the first structures used as a support for exploratory search [17], and this task has later attracted a lot of attention in Formal Concept Analysis theory [9]. Many works focus on *conceptual neighbourhood* to present both information related to a query and its closest variants [1, 10, 16]. Another approach proposes to retrieve cousin concepts [8] which are similar yet not comparable concepts. In this paper, we consider RCA to retrieve the conceptual neighbourhood in interconnected lattices, structuring both intrinsic and relational attributes.

The exponential growth of concept lattices is well-known [15]. As a consequence, the main limitation of FCA-based exploratory search lies in the complexity and computation of the structures [7]. Many solutions have been proposed to reduce the complexity of conceptual navigation. Some authors propose to prune the concept lattice to restrict the explorable dataspace, by computing iceberg concept lattices [27], or by applying constraints to bound the final structure [7]. To ease the navigation, the authors of [24] seek to extract more simplified browsable structures; they first extract a tree from the concept lattice, and then reduce the obtained tree using clustering and fault-tolerance methods. The tool `SearchSleuth` [10] enables FCA-based exploratory search for web queries, a field where the domain cannot be entirely processed using FCA and concept lattices. To tackle this issue, they generate a new formal context specific to a query at each navigation step. In a previous work [3], we proposed to compute the conceptual neighbourhood of a query in a sub-order of the concept lattice restricted to the attribute- and object-concepts (attribute-object-concept poset), a condensed alternative to concept lattices. At each step, only the conceptual neighbourhood is computed. In the present work, we also generate the conceptual neighbourhood on-the-fly, but this time in interconnected concept lattices.

Mimouni et al. [25] use RCA to structure, query and browse a collection of legal documents. First, they build interconnected lattices representing different types of legal documents referring to each other. Then, their approach allows for the retrieval of the concept corresponding to a user query, and to explore variations of this query by navigation in the neighbour concepts. In their approach, they compute all the lattices during the first step.

Ferré and Hermann [12] propose *Query-based Faceted Search* and an implementation in the tool `SEWELIS`, that allows to browse relational datasets in the form of RDF files. Also, Ferré et al. [13] propose RLCA, a relational extension of *Logical Formal Analysis*, an adaptation of FCA to describe objects by formulas of ad-hoc logics instead of binary attributes. While RCA computes connected yet separate concept lattices, one per sort of objects, RLCA gathers the objects, their descriptions and their relations to other objects in one structure, giving two complementary points of view on data. Besides, Ferré [14] introduces *abstract conceptual navigation* as an abstraction describing user guidance in a dynamic space of concepts connected by navigation links. Our approach fits into this vision.

7 Conclusion

In this paper, we proposed algorithms to compute the conceptual neighbourhood of a query in connected concept lattices generated with RCA. First, we redefined the traditional FCA derivation operators to take into account relational attributes. Then, we presented a way to compute the relational, upper and lower covers of a given concept in extended lattices, without computing all structures. Two RCA scaling operators, i.e., existential and universal strict, may be used. We illustrated how the algorithms work on a running example from the domain of software product line engineering. Also, we implemented our algorithms in RCAExplore, an existing tool to handle relational context families with relational concept analysis.

In the future, we plan to perform a systematic scalability study on real datasets from the projects Fresqueau and Knomana and from available product descriptions [4]. As the computation of minimal transversals of minimal generators may be a limitation while applying our algorithms on large datasets, we envision to develop metaheuristics to reduce the time complexity of their computation. We will also work on the presentation of the selectable attributes to the user. Our tests of the on-demand algorithm on the Fresqueau dataset showed that the extended formal contexts can have an important number of relational attributes, and it is difficult for the user to choose one of them. We also collected concrete questions from the Knomana project partners as they have some real exploration tasks in their domain. We evaluated the potentiality of constructing the whole conceptual structure on their data [22]. As a next step, we will qualitatively evaluate the benefits of the on-demand approach. In this evaluation, we will consider partial contexts to reduce the user focus with regard to their request, and we will implement strategies to keep records of the data exploration.

Acknowledgement. The authors warmly thank Xavier Dolques who helped us during the implementation in RCAexplore. This work was supported by the INRA-CIRAD Glofoods metaprogramme (Knomana project) and by the French National Research Agency under the Investments for the Future Program, referred as ANR-16-CONV-0004.

References

1. Alam, M., Le, T.N.N., Napoli, A.: LatViz: A New Practical Tool for Performing Interactive Exploration over Concept Lattices. In: Proc. of CLA'16. pp. 9–20 (2016)
2. Arévalo, G., Berry, A., Huchard, M., Perrot, G., Sigayret, A.: Performances of galois sub-hierarchy-building algorithms. In: Proc. of ICFCA'07. pp. 166–180 (2007)
3. Bazin, A., Carbonnel, J., Kahn, G.: On-Demand Generation of AOC-Posets: Reducing the Complexity of Conceptual Navigation. In: Proc. of ISMIS'17. pp. 611–621 (2017)
4. Ben Nasr, S., Bécan, G., Acher, M., Bosco, J.F.F., Sannier, N., Baudry, B., Davril, J.M.: Automated Extraction of Product Comparison Matrices From Informal Product Descriptions. J. of Systems and Software 124, 82 – 103 (2017)
5. Braud, A., Dolques, X., Huchard, M., Ber, F.L.: Generalization effect of quantifiers in a classification based on relational concept analysis. Knowl.-Based Syst. 160, 119–135 (2018)
6. Carbonnel, J., Huchard, M., Nebut, C.: Towards the extraction of variability information to assist variability modelling of complex product lines. In: Proc. of VAMOS'18. pp. 113–120 (2018)
7. Carpineto, C., Romano, G.: Exploiting the Potential of Concept Lattices for Information Retrieval with CREDO. J. of Universal Comp. Sci. 10(8), 985–1013 (2004)
8. Codocedo, V., Lykourentzou, I., Napoli, A.: A semantic approach to concept lattice-based information retrieval. Ann. Math. Artif. Intell. 72(1-2), 169–195 (2014)
9. Codocedo, V., Napoli, A.: Formal Concept Analysis and Information Retrieval - A Survey. In: Proc. of ICFCA'15. pp. 61–77 (2015)
10. Ducrou, J., Eklund, P.W.: SearchSleuth: The Conceptual Neighbourhood of an Web Query. In: Proc. of CLA'07. pp. 249–259 (2007)
11. Dunaiski, M., Greene, G.J., Fischer, B.: Exploratory search of academic publication and citation data using interactive tag cloud visualizations. Scientometrics 110(3), 1539–1571 (2017)
12. Ferré, S., Hermann, A.: Reconciling faceted search and query languages for the semantic web. Int. J. of Metadata, Semantics and Ontologies 7(1), 37–54 (2012)
13. Ferré, S., Ridoux, O., Sigonneau, B.: Arbitrary relations in formal concept analysis and logical information systems. In: Proc. of ICCS'05. pp. 166–180. Springer (2005)
14. Ferré, S.: Reconciling Expressivity and Usability in Information Access - From Filesystems to the Semantic Web. Habilitation thesis, Matisse, Univ. Rennes 1 (2014), habilitation à Diriger des Recherches (HDR), defended on November 6th
15. Ganter, B., Wille, R.: Formal Concept Analysis - mathematical foundations. Springer (1999)
16. Godin, R., Gecsei, J., Pichet, C.: Design of a Browsing Interface for Information Retrieval. In: Proc. of SIGIR'89. pp. 32–39 (1989)
17. Godin, R., Saunders, E., Gecsei, J.: Lattice model of browsable data spaces. Inf. Sci. 40(2), 89–116 (1986)
18. Hacene, M.R., Huchard, M., Napoli, A., Valtchev, P.: A Proposal for Combining Formal Concept Analysis and Description Logics for Mining Relational Data. In: Proc. of ICFCA'07. pp. 51–65 (2007)
19. Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Relational concept discovery in structured datasets. Ann. Math. Artif. Intell. 49(1-4), 39–76 (2007)
20. Hébert, C., Bretto, A., Crémilleux, B.: A Data Mining Formalization to Improve Hypergraph Minimal Transversal Computation. Fundam. Inform. 80, 415–433 (11 2007)

21. Jäschke, R., Hotho, A., Schmitz, C., Ganter, B., Stumme, G.: TRIAS - an algorithm for mining iceberg tri-lattices. In: Proc. of ICDM'06. pp. 907–911 (2006)
22. Keip, P., Gutierrez, A., Huchard, M., Le Ber, F., Sarter, S., Silvie, P., Martin, P.: Effects of input data formalisation in Relational Concept Analysis for a data model with a ternary relation. In: Proc. of ICFCA'19 (to appear) (2019)
23. Marchionini, G.: Exploratory search: from finding to understanding. Comm. ACM 49(4), 41–46 (2006)
24. Melo, C.A., Grand, B.L., Aufaure, M.: Browsing Large Concept Lattices through Tree Extraction and Reduction Methods. Int. J. of Intelligent Information Technologies 9(4), 16–34 (2013)
25. Mimouni, N., Nazarenko, A., Salotti, S.: A conceptual approach for relational ir: Application to legal collections. In: Proc. of ICFCA'15. pp. 303–318 (2015)
26. Palagi, É., Gandon, F.L., Giboin, A., Troncy, R.: A survey of definitions and models of exploratory search. In: ACM Workshop ESIDA@IUI. pp. 3–8 (2017)
27. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with Titanic. Data Knowledge Engineering 42(2), 189–222 (2002)