On-demand Relational Concept Analysis

Alexandre Bazin, Jessie Carbonnel, Marianne Huchard, Giacomo Kahn

On-demand Relational Concept Analysis

Alexandre Bazin[1], Jessie Carbonnel[2], Marianne Huchard[2], and Giacomo Kahn[3]

[1] Le2i, Université Bourgogne Franche-Comté, Dijon, France
[2] LIRMM, CNRS and Université de Montpellier, Montpellier France
[3] LIMOS, Université Clermont Auvergne, Clermont-Ferrand, France
contact@alexandrebazin.com,{jcarbonnel,huchard}@lirmm.fr,giacomo.kahn@isima.fr

Abstract. Formal Concept Analysis and its associated conceptual structures have been 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, for instance 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 extended 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. We illustrate it on an example.

Keywords: Relational Concept Analysis, Formal Concept Analysis, On-demand 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[4] (Fresqueau project), the inventory and use of pesticidal, antibacterial and antifungal plants[5] (Knomana project), and the analysis and representation of product lines [4]. In these applications, there is a wide range of question forms, such as classical querying, establishing correlations between descriptions of objects from several categories or case based reasoning. These questions can be addressed by complementary approaches including conceptual classification building, knowledge pattern and rule extraction, or exploratory search [17, 20]. In the Knomana project,

[4] http://engees-fresqueau.unistra.fr/presentation.php?lang=en
[5] http://www.cirad.fr/en/news/all-news-items/articles/2017/science/identifying-plants-used-as-natural-pesticides-in-africa-knomana

for example, one main purpose will be, after the ongoing inventory, to support farmers, their advisors, local entrepreneurs or researchers in selecting plants of immediate interest for agricultural crop protection and animal health. As such users will face large amounts of data, and mainly will formulate general, potentially imprecise, and potentially inaccurate queries without prior knowledge of the data, exploratory search will be a suitable approach in this context.

Previous work [14, 5, 7, 11, 8] has shown that Formal Concept Analysis may be a relevant support for data exploration and we expect Relational Concept Analysis (RCA) to be beneficial as well. Considering RCA for relational dataset exploration brings issues relative to the use of the scaling (logical) operators, the iterative process and 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 explore datasets 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. 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. Section 4 illustrates the algorithm with the example introduced in Section 2. Related work is exposed in Section 5. We conclude the paper with a few perspectives in Section 6.

2 Relational Concept Analysis

Formal Concept Analysis (FCA) [12] allows to structure a set of objects described by attributes in a canonical structure called a concept lattice. It is based on a formal context $K = (\mathcal{O}, \mathcal{A}, \mathcal{I})$, where $\mathcal{O}$ is the set of objects, $\mathcal{A}$ the set of attributes, and $\mathcal{I}$ an incidence relation stating "which objects possess which attributes". From this context, the application of FCA extracts a finite set C_K of formal concepts (X, Y) such that $X = \{o \in \mathcal{O} \mid \forall a \in Y, (o, a) \in \mathcal{I}\}$ is the concept's extent, and $Y = \{a \in \mathcal{A} \mid \forall o \in X, (o, a) \in \mathcal{I}\}$ is the concept's intent. The concept lattice is obtained by ordering the concepts of C_K by the set-inclusion order on their extents. We call an *object-concept* (resp. *attribute-concept*) the lowest (resp. the greatest) concept in the lattice possessing an object (resp. an attribute).

Relational Concept Analysis (RCA) [15, 16] is an adaptation of FCA to process relational datasets. A relational dataset is composed of several sorts of objects described by both their own attributes and their relationships with other objects. As input, RCA takes a Relational Context Family (RCF), gathering a set of formal contexts and a set of relational contexts defining links between the objects of different formal contexts.

Definition 1 (Relational Context Family). *A Relational Context Family is a pair* $(\boldsymbol{K}, \boldsymbol{R})$ *such that:*
- $\boldsymbol{K} = \{K_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)\}$ *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* $\mathcal{O}_i$ *and* $\mathcal{O}_j$ *being sets of objects (respectively of* K_i *and* K_j*).* K_i *is called the source context and* K_j *the target context.*

The three contexts of Table 1 present an example of 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 5 Data Modelling tools (*DM_tools*) against 7 attributes representing their compatible operating systems (*OS:*), and the data models (*DM:*) the tools may manage. The table on the right-hand side describes 4 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.

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	x	x	x	x			
Erwin DM	x			x	x	x	
ER/Studio	x			x	x	x	x
Magic Draw	x	x	x	x	x	x	
MySQL Workbench	x	x	x		x		

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

$\mathbf{R}_s =$

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

Applying RCA on the contexts of $\mathbf{K}$ builds, in a first time, one concept lattice per context (i.e., sort/category of objects), without taking links into account. The two concept lattices associated with Table 1 (top) are presented in Fig. 1. In a second time, RCA introduces links between objects of different lattices depending

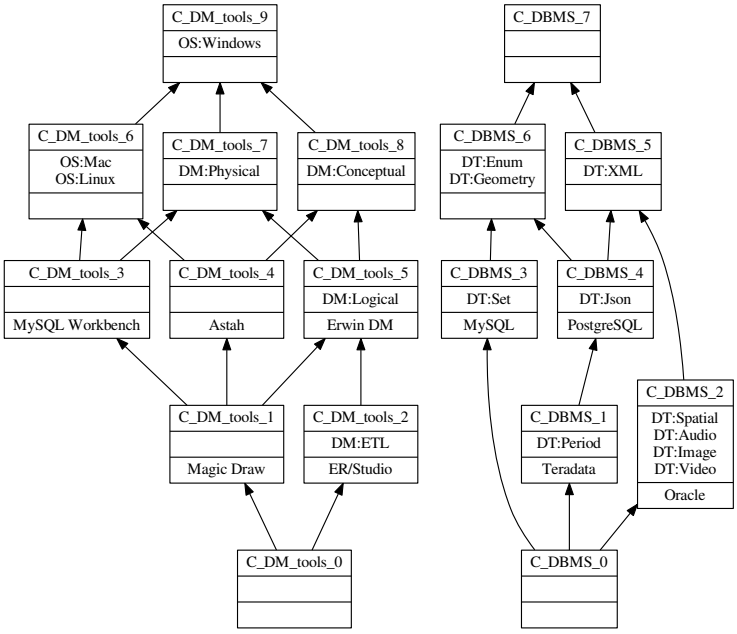

Fig. 1. (left) concept lattice of DM_tools, (right) concept lattice of DBMS

on the relations expressed in $\mathbf{R}$. These links take the form of *relational attributes*; they introduce the abstractions (i.e., concepts) from the target context into the source context through a specific relation and a specific *scaling operator*. In our example, we may introduce the relational attribute $\exists\ support.(C\_DBMS\_4)$ to characterise the *DM_ tools* that support at least one *DBMS* offering Json and XML. More generally, given two formal contexts $K_i, K_j \in \mathbf{K}$ 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 K_j. The extended attribute set is denoted $\mathcal{A}_i^+$. Then, the incidence relation $\mathcal{I}_i$ is extended to take into account these new attributes (denoted $\mathcal{I}_i^+$), by associating them to each object of $\mathcal{O}_i$ depending on the relation r, the concept (denoted C) involved in the relational attribute and a scaling operator ρ. A relational attribute is thus of the form "$\rho\ r.(C)$". In this paper, we focus on two scaling operators: the *existential* operator (denoted $\exists$), 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; the *universal strict* operator (denoted $\exists\forall$), 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 $r(o) \neq \emptyset$.

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

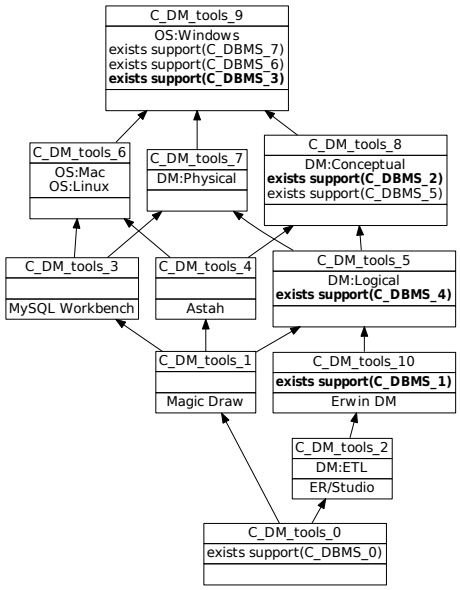

Fig. 2. Concept lattice of the extended context DM_tools^{+}

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. At step 0, the concept lattices in the set $\mathbf{L}_0$ are the ones built from the initial formal contexts from $\mathbf{K}$. At step n, the formal contexts in the set $\mathbf{K}_n$ are extended depending on the concepts of the concept lattices in $\mathbf{L}_{n-1}$ and the relations expressed in $\mathbf{R}$.

3 The Exploration Algorithm

In this section, we present an algorithm for taking a step in an exploration. It considers an RCF potentially extended at previous steps, a starting concept C from a context $\mathcal{K}_i$ of the RCF and an exploration *strategy* which consists in choosing a set of relations of the RCF (with $\mathcal{K}_i$ as a source) provided with scaling operators. The objective of one step is to complete the intent corresponding to the extent of C, as well as compute its upper, lower and relational covers. Meanwhile, the RCF is updated with the relational attributes for a next step.

Redefining Derivation Operators The explicit knowledge of all the relational attributes of a context requires the computation of all the concepts of the target contexts. However, we cannot afford what amounts to the exhaustive computation of the relational concepts of multiple contexts. 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 $\rho\ r.(X, Y)$ (instead of $\rho\ r.((X, Y))$ by abuse of notation) is also necessarily described by all the attributes of the form $\rho\ r.(X_2, Y_2)$ where $Y_2 \subseteq Y$. As such, intents can be represented without loss of information by their relational attributes constructed from attributes-wise maximal concepts. However, a problem arises with such a representation: the set intersection cannot be used to compute the intent of a set of objects anymore. Similarly, 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.

INTERSECT takes as input two sets of attributes A and B represented by their maximal relational attributes. It outputs the set of maximal relational attributes of their intersection. A relational attribute $\exists r.(X, Y)$ is in the intersection of A and B if and only if there exists 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. As such, intersecting the intents of the concepts in the attributes of A and B and keeping the maximal ones results in the maximal relational attributes. It uses EX (Algorithm 3).

Algorithm 1: INTERSECT$(\mathcal{K}_i, A, B)$

Input: $\mathcal{K}_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $A \subseteq \mathcal{A}_i$ an attribute set, $B \subseteq \mathcal{A}_i$ the intent of an object o
Output: The relational intersection of the attribute set A and the intent of o

$A_2 \leftarrow A \cap B$
$\mathcal{F} \leftarrow \emptyset$
foreach $a_1 \sim \exists r.(X_1, Y_1) \in B$ *with* $r \subseteq \mathcal{O}_i \times \mathcal{O}_j$ *and* $\mathcal{K}_j = (\mathcal{O}_j, \mathcal{A}_j, \mathcal{I}_j)$ **do**
  foreach $a_2 \sim \exists r.(X_2, Y_2) \in A$ **do**
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{\exists r.(\text{EX}(\mathcal{K}_j, \text{INTERSECT}(\mathcal{K}_j, Y_1, Y_2)), \text{INTERSECT}(\mathcal{K}_j, Y_1, Y_2))\}$
$A_2 \leftarrow A_2 \cup \text{MAX}(\mathcal{F}, \subseteq_{\mathcal{A}_i})$
$\mathcal{F} \leftarrow \emptyset$
foreach $a_1 \sim \exists\forall r.(X_1, Y_1) \in B$ *with* $r \subseteq \mathcal{O}_i \times \mathcal{O}_j$ *and* $\mathcal{K}_j = (\mathcal{O}_j, \mathcal{A}_j, \mathcal{I}_j)$ **do**
  foreach $a_2 \sim \exists\forall r.(X_2, Y_2) \in A$ **do**
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{\exists\forall r.(\text{EX}(\mathcal{K}_j, \text{INTERSECT}(\mathcal{K}_j, Y_1, Y_2)), \text{INTERSECT}(\mathcal{K}_j, Y_1, Y_2))\}$
$A_2 \leftarrow A_2 \cup \mathcal{F}$
return A_2

IN uses INTERSECT to compute the intent of a set 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 $\mathcal{K}_i$.

EX computes the extent of a set of maximal relational attributes A. 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).

Algorithm 2: $\textsc{In}(\mathcal{K}_i, O)$

Input: $\mathcal{K}_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $O \subseteq \mathcal{O}_i$ a set of objects
Output: Computes the intent of a set of objects O

```
A ← A_i
foreach o ∈ O do
    A ← INTERSECT(A, Intent({o}))
return A
```

Algorithm 3: $\textsc{Ex}(\mathcal{K}_i, A)$

Input: $\mathcal{K}_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $A \subseteq \mathcal{A}_i$ a set of attributes
Output: Computes the extent of a set of attributes A

```
O ← O_i
foreach a ∈ A do
    if a ∼ ∃∀r.(X, Y) then
        foreach o ∈ O do
            if r(o) ⊈ X then
                O ← O \ o
    else if a ∼ ∃r.(X, Y) then
        foreach o ∈ O do
            if r(o) ∩ X = ∅ then
                O ← O \ o
    else
        foreach o ∈ O do
            if (o, a) ∉ I_i then
                O ← O \ o
return O
```

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 computes the closed neighbourhood of a concept C. It takes as input a set of formal contexts $\mathbf{K} = (\mathcal{K}_1, \ldots, \mathcal{K}_w)$ 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 $\mathcal{K}_i$. 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 $\mathcal{K}_i^+$.

For each $(r, \rho)_{ij} \in \mathcal{S}$ such that $r : \mathcal{K}_i \mapsto \mathcal{K}_j$, the first loop (Lines 1 to 4): computes OC_j the object-concepts of $\mathcal{K}_j$; then, each object-concept $(X, Y) \in OC_j$, relation r and scaling operator ρ give rise to a new relational attribute $\rho\, r.(X, Y)$ that is added to the context $\mathcal{K}_i$ with GrowContext.

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 4: RCA$(\mathbf{K}, \mathcal{S}, C, \mathcal{K}_i)$

Input: $\mathbf{K} = \{\mathcal{K}_1, \ldots, \mathcal{K}_w\}$, $\mathcal{S} = \{(r, \rho)_{lj}, \ldots\}, l, j \in \{1, \ldots, w\}$ a strategy, $C = (O, A)$ a concept of $\mathcal{K}_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$

Output: $C, \mathcal{U}, \mathcal{R}, \mathcal{L}$ the completed concept C and its closed relational neighbourhood

```
1  foreach (r, ρ)_ij ∈ S do
2  |  OC_j ← ObjectPoset(K_j)
3  |  foreach o ∈ O_i do
4  |  |  GrowContext(K_i, r, ρ, o, OC_j)
5  A ← In(K_i, O)
6  R ← ∅
7  foreach a ∼ ρr.(X_1, Y_1) ∈ A do
8  |  R ← R ∪ {(X_1, Y_1)}
9  L ← ∅
10 foreach T ∈ minTrans(minGen(O)) do
11 |  L ← L ∪ {(O \ T, In(K_i, O \ T))}
12 U ← ∅
13 foreach o ∈ O_i \ O do
14 |  U ← U ∪ {(Ex(K_i, In(K_i, O ∪ {o})), In(K_i, O ∪ {o}))}
15 U ← Min(U, ⊆_{O_i})
16 return C, U, R, L
```

Algorithm 5: GROWCONTEXT($\mathcal{K}_i, r, q, o, OC_j$)

Input: $\mathcal{K}_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $r \subseteq \mathcal{O}_i \times \mathcal{O}_j$ a relational context, ρ a scaling operator, $o \in \mathcal{O}_i$ an object, OC_j the set of object-concepts of $\mathcal{K}_j = (\mathcal{O}_j, \mathcal{A}_j, \mathcal{I}_j)$

Output: Extends the context $\mathcal{K}_i$ and adds the crosses

1 **if** $\rho == \exists$ **then**
2 **foreach** $(X, Y) \in OC_j$ *such that* $\exists obj \in r(o)$, $obj \in X$ **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 $X \leftarrow$IN$(\mathcal{K}_j, r(o))$
7 $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \exists\forall r.(\text{EX}(\mathcal{K}_i, X), X)$
8 $\mathcal{I}_i \leftarrow \mathcal{I}_i \cup (o, \exists\forall r.(\text{EX}(\mathcal{K}_i, X), X))$

4 Example

In this section, we illustrate the defined algorithms. We consider the RCF $(\mathbf{K}_s, \mathbf{R}_s)$ with $\mathbf{K}_s = \{DM\_tools, DBMS\}$ and $\mathbf{R}_s = \{support\}$ as presented in Section 2. We decide to apply the strategy $\{(support, \exists)\}$.

Let us imagine that a user wants to select a data modelling tool that runs on Windows (*OS:Windows*) and that handles logical and conceptual data models (*DM:Logical* and *DM:Conceptual*). Traditional FCA may compute the formal concept associated with these 3 attributes (i.e., *C_DM_tools_5*, left-hand side of Fig. 1), and inform the user that 1) the corresponding tools are `Erwin DM`, `Magic Draw` and `ER/Studio`, and that 2) all these tools also handle *DM:Physical*.

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): RCA$(\mathbf{K}_s, \{(support, \exists)\}, C_DM_tools_5, DM_tools)$.

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*, $\exists$) visited at Line 1. In Line 2, OC_j takes the object-concepts of *DBMS* , i.e., concepts 1, 2, 3 and 4 from the right-hand side of Fig. 1. 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 with o_j.

As $support(\texttt{Astah}) = \{MySQL, Oracle\}$, $\exists support(C\_DBMS\_3)$ (*MySQL* object-concept) and $\exists support(C\_DBMS\_2)$ (*Oracle* object-concept) are added to *DM_tools* and associated to `Astah`. At the end of Line 4, we obtain the extended context presented in Table 2.

Line 5 updates the intent of the input concept to take into account the relational attributes: {*OS:Windows, DM:Conceptual, DM:Physical, DM:Logical,* $\exists sup.(C\_DBMS\_2)$, $\exists sup.(C\_DBMS\_3)$, $\exists sup.(C\_DBMS\_4)$}). The con-

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	∃ sup.(C_ DBMS_ 1)	∃ sup.(C_ DBMS_ 2)	∃ sup.(C_ DBMS_ 3)	∃ sup.(C_ DBMS_ 4)
Astah	x	x	x	x					x	x	
Erwin DM	x			x	x	x		x	x	x	x
ER/Studio	x			x	x	x	x	x	x	x	x
Magic Draw	x	x	x	x	x	x			x	x	x
MySQL Workbench	x	x	x		x					x	

cepts of *DBMS* corresponding to the relational attributes of C (C_DBMS_2 to 4) 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, which are {`Erwin DM`, `Magic Draw`} and {`ER/Studio, Magic Draw`}. Their minimal transversals are {`Magic Draw`} and {`Erwin DM, ER/Studio`}. The two concepts having {`Magic Draw`} and {`Erwin DM,ER/Studio`} for extent represent the lower cover (respectively $C\_DM\_tools\_1$ and $C\_DM\_tools\_2$ in Fig. 2).

Finally, in Lines 12 to 14, we consider the objects of *DM_ tools* that are not in C's extent, i.e., `MySQL Workbench` and `Astah`. For each one of them, we compute the concept corresponding to their union with C's extent, and we obtain the two concepts *C_ DM_ tools_ 7* and *8* of Fig. 2. They represent the upper cover of C.

5 Related Work

Lattice structures are among the first structures used as a support for exploratory search [14], and this task has later attracted a lot of attention in Formal Concept Analysis theory [6]. Many works focus on *conceptual neighbourhood* to present both information related to a query and its closest variants [13, 7, 1]. 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 [12]. As a consequence, the main limitation of FCA-based exploratory search lies in the complexity and computation of the structures [5]. 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 [21], or by applying constraints to bound the final structure [5]. To ease the navigation, the authors of [18] 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` [7] 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 [2], 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. [19] 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 [9] 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. [10] 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.

6 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 the 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.

In the future, we plan to study the properties of the algorithm and to implement it to perform exploratory search in relational datasets. A scalability study on real datasets from the projects Fresqueau and Knomana and from available product descriptions [3] is then envisioned. To this end, we will generate random queries and exploration paths. We also are collecting concrete questions from the Knomana project partners for having real exploration tasks in their domain and qualitatively evaluate the benefits of the approach.

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 the 13th Int. Conf. on Concept Lattices and Their Applications (CLA'16). pp. 9–20 (2016)
2. Bazin, A., Carbonnel, J., Kahn, G.: On-Demand Generation of AOC-Posets: Reducing the Complexity of Conceptual Navigation. In: Proc. of the 23rd Int. Symp. on Foundations of Intelligent Systems (ISMIS'17). pp. 611–621 (2017)

3. 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)
4. Carbonnel, J., Huchard, M., Nebut, C.: Analyzing Variability in Product Families through Canonical Feature Diagrams. In: Proc. of the 29th Int. Conf. on Software Engineering & Knowledge Engineering (SEKE'17). pp. 185–190 (2017)
5. 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)
6. Codocedo, V., Napoli, A.: Formal Concept Analysis and Information Retrieval - A Survey. In: Proc. of the 13th Int. Conf. on Formal Concept Analysis (ICFCA'15). pp. 61–77 (2015)
7. Ducrou, J., Eklund, P.W.: SearchSleuth: The Conceptual Neighbourhood of an Web Query. In: Proc. of the 5th Int. Conf. on Concept Lattices and Their Applications (CLA'07). pp. 249–259 (2007)
8. 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)
9. 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)
10. Ferré, S., Ridoux, O., Sigonneau, B.: Arbitrary relations in formal concept analysis and logical information systems. In: Proc. of the 13th Int. Conf. on Conceptual Structures (ICCS'05). pp. 166–180. Springer (2005)
11. 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
12. Ganter, B., Wille, R.: Formal Concept Analysis - mathematical foundations. Springer (1999)
13. Godin, R., Gecsei, J., Pichet, C.: Design of a Browsing Interface for Information Retrieval. In: Proc. of the 12th Int. Conf. on Research and Development in Information Retrieval (SIGIR'89). pp. 32–39 (1989)
14. Godin, R., Saunders, E., Gecsei, J.: Lattice model of browsable data spaces. Inf. Sci. 40(2), 89–116 (1986)
15. 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 the 5th Int. Conf. on Formal Concept Analysis (ICFCA'07). pp. 51–65 (2007)
16. 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)
17. Marchionini, G.: Exploratory search: from finding to understanding. Comm. ACM 49(4), 41–46 (2006)
18. 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)
19. Mimouni, N., Nazarenko, A., Salotti, S.: A conceptual approach for relational ir: Application to legal collections. In: Proc. of the 13th Int. Conf. on Formal Concept Analysis (ICFCA'15). pp. 303–318 (2015)
20. 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)
21. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with Titanic. Data Knowledge Engineering 42(2), 189–222 (2002)