

Querying Relational Concept Lattices

Zeina Azmeh, Marianne Huchard, Amedeo Napoli, Amine Mohamed Rouane Hacene, Petko Valtchev

▶ **To cite this version:**

Zeina Azmeh, Marianne Huchard, Amedeo Napoli, Amine Mohamed Rouane Hacene, Petko Valtchev. Querying Relational Concept Lattices. CLA: Concept Lattices and their Applications, Oct 2011, Nancy, France. pp.377-392. lirmm-00646409

HAL Id: lirmm-00646409

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00646409

Submitted on 29 Nov 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Querying Relational Concept Lattices

Z. Azmeh[1], M. Huchard[1],
A. Napoli[2], M. Rouane-Hacene[3], and P. Valtchev[3]

[1] LIRMM, 161, rue Ada, F-34392 Montpellier Cedex 5
[2] LORIA, B.P. 239, F-54506 Vandœuvre-lès-Nancy
[3] Dépt. d'informatique, UQÀM, C.P. 8888, Succ. Centre-Ville Montréal, Canada

Abstract. Relational Concept Analysis (RCA) constructs conceptual abstractions from objects described by both own properties and inter-object links, while dealing with several sorts of objects. RCA produces lattices for each category of objects and those lattices are connected via relational attributes that are abstractions of the initial links. Navigating such interrelated lattice family in order to find concepts of interest is not a trivial task due to the potentially large size of the lattices and the need to move the expert's focus from one lattice to another. In this paper, we investigate the navigation of a concept lattice family based on a query expressed by an expert. The query is defined in the terms of RCA. Thus it is either included in the contexts (modifying the lattices when feasible), or directly classified in the concept lattices. Then a navigation schema can be followed to discover solutions. Different navigation possibilities are discussed.

Keywords: Formal Concept Analysis, Relational Concept Analysis, Relational Queries.

1 Introduction

Recently [1], we worked on the problem of selecting suitable Web services for instantiating an abstract calculation workflow. This workflow can be seen as a DAG whose nodes are abstract tasks (like *book a hotel room*) and directed edges are connections between the tasks, which often correspond to a data flow (like connecting *reserve a train ticket* and *book a hotel room*: train dates and timetable are transmitted from *reserve a train ticket* to *book a hotel room*). The selection is based on quality-of-service (QoS) properties like response time or availability and on the composability quality between services chosen for neighbor tasks in the workflow. Besides, we aim at identifying and storing a set of backup services adapted to each task. To be efficient in the replacement of a failing Web service by another, we want to organize each set of backup Web services by a partial order that expresses the quality criteria and helps to choose a good trade-off for instantiating the abstract workflow. Analyzing such multi-relational data is a complex problem, which can be approached by various methods including querying, visualization, statistics, or rule extraction (data mining).

We proposed an approach based on Relational Concept Analysis (an iterative version of Formal Concept Analysis) to solve this problem, because of its multi-relational nature. Web services are filtered and grouped by tasks they may satisfy (*e. g.* the Web services for booking a hotel room). In formal contexts (one for each task), we associate the Web services and their QoS criteria. For example, the service *HotelsService* by *lastminutetravel.com* would be described by *low response time*, *medium availability* (classical scaling is applied to the QoS values). In relational contexts we encode the composability levels in each directed edge of the workflow. Given an edge of the workflow, the composition quality depends on the way output data of the source task cover input data of the ending task, and the need for data adaptation. A relational context encodes for example the relation *Adaptable-Fully-Composable* between services for *reserve a train ticket* and services for *book a hotel room*. In this relation *TravelService* by *puturist.com* is connected to *HotelsService* by *lastminutetravel.com* if output data of *TravelService* can be used, with a slight adaptation, to fill input data of *HotelsService*.

The concept lattice family we obtain (one Web service lattice for each task of the workflow) makes it possible: (1) to select a Web service for each task based on QoS and composability criteria, (2) to memorize classified alternatives for each task.

Due to the nature of our problem, we are interested in classifying independently the Web services corresponding to the tasks and not classifying the solutions. By solution, we mean a set of Web services, each of which can instantiate a task of the workflow. If a particular service fails or is no more available, the goal is to constitute a new working combination out of the old one, with the smallest number of service replacements. To the best of our knowledge, this problem area has not been investigated in-depth prior to our study, especially in the context of Relational Context Analysis [7, 6]. Therefore, we believe that it would be useful to generalize and report what we learned in our experience. In more general terms, we have multi-relational data and a question which contains variables we want to instantiate, and we aim at:

- Finding a specific set of objects that satisfy the query. An answer is composed of objects, each object instantiates one variable;
- Classifying, for each variable, the objects depending on the way they satisfy (or not) the query, to find alternative answers.

In this paper, we put the problem in a more general framework, which assumes an unrestricted relational context family and a query given by an expert. The query can be seen as a DAG, where some nodes are labelled by variables and some others are labelled by objects. The nodes roughly correspond to the formal (object-attribute) contexts and the edges correspond to the relational (object-object) contexts. A set of lattices is built using Relational Concept Analysis and the existential scaling operator. We assume that an expert gives a total ordering of the edges of the DAG. Then an algorithm navigates the lattices following this ordering. This navigation allows us to determine objects that answer the query.

These objects with their position in the lattices are what the expert wants to explore, to extract a solution and store the alternatives.

In the following, Section 2 reminds the main principles of Relational Concept Analysis (RCA). Section 3 defines the model of queries in the RCA framework that we consider in this paper. Section 4 presents and discusses an algorithm that navigates the concept lattice family using a query. Related work is presented in Section 5 and we conclude in Section 6.

2 Background on Relational Concept Analysis

For FCA, we use the notations of [4]. In RCA [5], the objects are classified not only according to the attributes they share, but also according to the links between them. Let us take the following case study. We consider a list of countries, a list of restaurants, a list of Mexican dishes, a list of ingredients, and finally a list of salsas. We impose some relations between these entities {Country, Restaurant, MexicanDish, Ingredient, Salsa}, such that: a Country "has" a Restaurant; a Restaurant "serves" a MexicanDish; a MexicanDish "contains" an Ingredient; an Ingredient is "made-in" a Country; and finally a Salsa is "suitable-with" a MexicanDish. We express these entities and their relations by the DAG in Fig. 1. We capture an instantiation of this entity-relationship diagram in a relational context family.

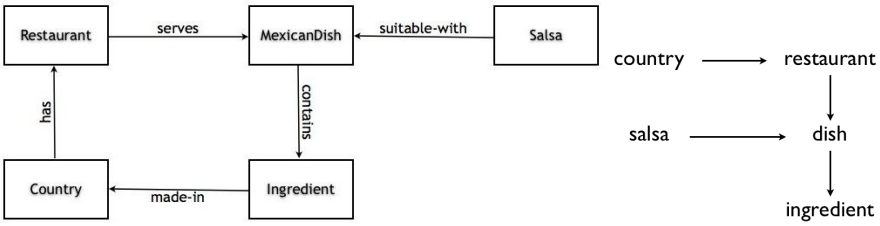


Fig. 1. The entities of the Mexican food example (left). The query schema (right)

Definition 1. *A relational context family* ***RCF*** *is a pair* $(\mathbb{K}, R)$ *where* $\mathbb{K}$ *is a set of formal (object-attribute) contexts* $K_i = (O_i, A_i, I_i)$ *and* R *is a set of relational (object-object) contexts* $r_{ij} \subseteq O_i \times O_j$*, where* O_i *(domain of* r_{ij}*) and* O_j *(range of* r_{ij}*) are the object sets of the contexts* K_i *and* K_j*, respectively.*

The RCF corresponding to our example contains five formal contexts and five relational contexts, illustrated in Table 1 (except the *made-in* relational context, which is not used in this paper for sake of simplicity). An RCF is used in an iterative process to generate at each step a set of concept lattices. First concept lattices are built using the formal contexts only. Then, in the following steps, a scaling mechanism translates the links between objects into

Table 1. Relational Context Family for mexican dishes

Country	ca	en	fr	lb	mx	es	us	America	Asia	Europe
Canada	×							×		
England		×								×
France			×							×
Lebanon				×					×	
Mexico					×			×		
Spain						×				×
USA							×	×		

Restaurant	r1	r2	r3	r4	r5	r6	r7
Chili's	×						
Chipotle		×					
El Sombrero			×				
Hard Rock				×			
Mi Casa					×		
Taco Bell						×	
Old el Paso							×

MexicanDish	d1	d2	d3	d4	d5	d6
Burritos	×					
Enchiladas		×				
Fajitas			×			
Nachos				×		
Quesadillas					×	
Tacos						×

Ingredient	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11	i12
chicken	×											
beef		×										
pork			×									
vegetables				×								
beans					×							
rice						×						
cheese							×					
guacamole								×				
sour-cream									×			
lettuce										×		
corn-tortilla											×	
flour-tortilla												×

Salsa	s1	s2	s3	s4	mild	medium-hot	hot
Fresh Tomato	×				×		
Roasted Chili-Corn		×				×	
Tomatillo-Green Chili			×			×	
Tomatillo-Red Chili				×			×

contains	chicken	beef	pork	vegetables	beans	rice	cheese	guacamole	sour-cream	lettuce	corn-tortilla	flour-tortilla
Burritos	×	×	×	×	×	×	×	×	×	×		×
Enchiladas	×						×		×		×	
Fajitas	×	×		×			×	×	×	×		×
Nachos				×	×		×	×				
Quesadillas	×	×					×				×	×
Tacos	×	×			×		×			×	×	×

has	Chili's	Chipotle	El Sombrero	Hard Rock	Mi Casa	Taco Bell	Old el Paso
Canada	×	×	×	×		×	
England		×		×		×	
France			×	×			×
Lebanon	×			×		×	
Mexico	×				×	×	
Spain				×		×	
USA	×	×	×	×	×	×	

serves	Burritos	Enchiladas	Fajitas	Nachos	Quesadillas	Tacos
Chili's			×		×	×
Chipotle	×					×
El Sombrero	×	×	×	×	×	×
Hard Rock			×	×		
Mi Casa	×	×		×	×	×
Taco Bell	×			×	×	×
Old el Paso						×

suitable-with	Burritos	Enchiladas	Fajitas	Nachos	Quesadillas	Tacos
Fresh Tomato	×	×	×	×	×	×
Roasted Chili-Corn	×			×		
Tomatillo-Green Chili	×			×		
Tomatillo-Red Chili	×	×	×	×	×	×

conventional FCA attributes and derives a collection of lattices whose concepts are linked by relations. For example, the existential scaled relation (that we will use in this paper) captures the following information: if an object o_s is linked to another object o_t, then in the scaled relation, this link is encoded in a relational attribute assigned to o_s. This relational attribute states that o_s is linked to a concept, which clusters o_t with other objects. This is used to form new groups, for example the group (See *Concept*_84) of restaurants, which serve at least one dish containing sour cream (such dishes are grouped in *Concept*_75). The steps are repeated until reaching the stability of lattices (when no more new concepts are generated). For mexican dishes, four lattices of the concept lattice family are represented in Figures 3 and 4. The ingredient lattice is presented in Fig. 2.

Definition 2. *Let* $r_{ij} \subseteq O_i \times O_j$ *be a relational context. The* ***exists*** *scaled relation* $r_{ij}^{\exists}$ *is defined as* $r_{ij}^{\exists} \subseteq O_i \times \mathfrak{B}(O_j, A, I)$*, such that for an object* o_i *and a concept* c*:* $(o_i, c) \in r_{ij}^{\exists} \iff \exists\ x, x \in o_i' \cap Extent(c)$*.*

In this definition, A is any set of attributes maybe including relational attributes, which are defined below.

Definition 3. *A relational attribute* $(s\ r\ c)$ *is composed of a scaling operator* s *(for example exists), a relation* $r \in R$*, and a concept* c*. It results from scaling a relation* $r_{ij} \in R$ *where* $r_{ij} \subseteq O_i \times O_j$*. It expresses a relation between the objects* $o \in O_i$ *with the concepts of* $\mathfrak{B}(O_j, A, I)$*. An existential relational attribute is denoted by* $\exists r_{ij}\ c$ *where* $c \in \mathfrak{B}(O_j, A, I)$*.*

For example: the *Concept_50* in the Country lattice owns the relational attribute $\exists$*has Concept_60*. This expresses that each country in *Concept_50* (Canada and USA) *has* at least a restaurant in *Concept_60* extent (El Sombrero or Mi Casa).

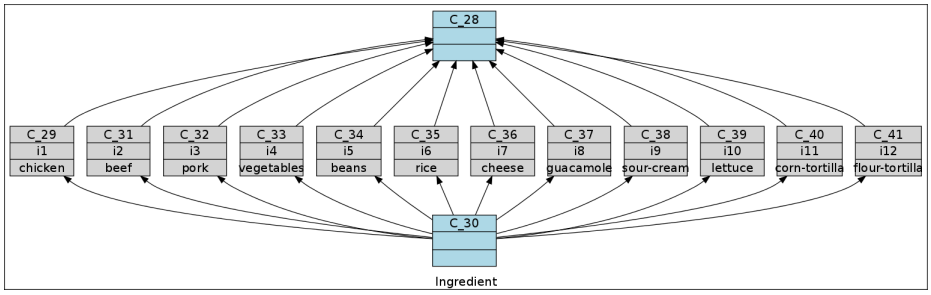


Fig. 2. The concept lattice for ingredients of the RCF in Table 1 (concepts names are reduced to C_n).

3 Introducing Relational Queries

In this section, we define the notion of *query* and *answer to a query*. First (section 3.1) we recall simple queries that help navigating concept lattices [7]. Then (section 3.2), we generalize to relational queries that lead the navigation across a concept lattice family.

3.1 Simple queries

Definition 4. *A query (including its answer) on a context* $K = (O, A, I)$*, denoted by* $q|_K$ *(or* q *when it is not ambiguous), is a pair* $q = (o_q, a_q)$*, such that* o_q *is the query object(s) i.e. the set of objects satisfying the query (or the answer set), and* a_q *is the set of attributes defining the constraint of the query. By definition, we have:* $o_q' \supseteq a_q$*, where* $a_q \subseteq A$*.*

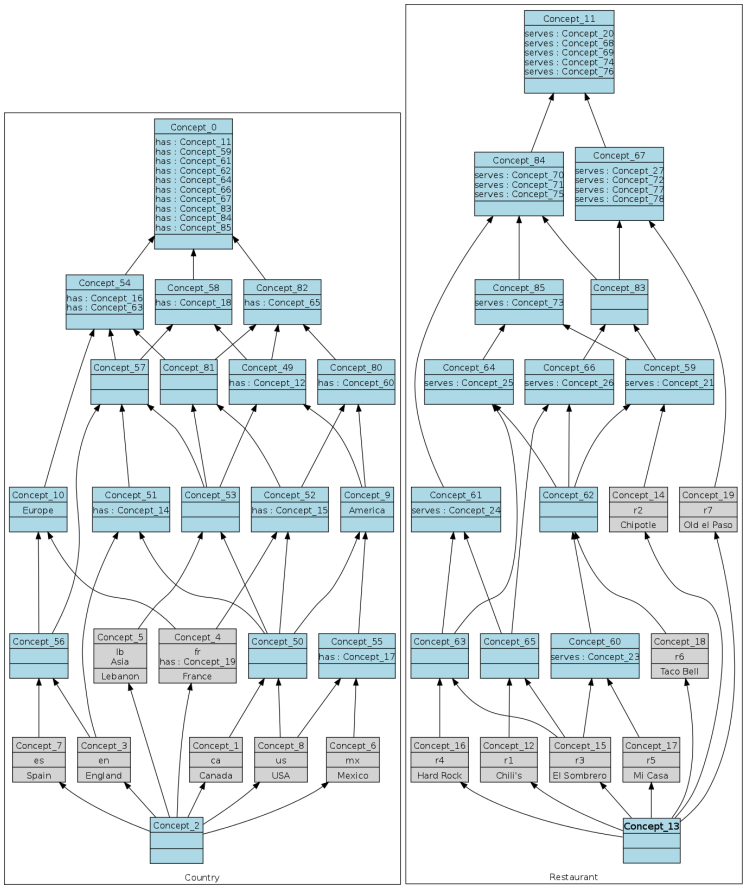


Fig. 3. Country and restaurant lattices for `exists` and the RCF in Table 1.

For example $q|_{K_{country}} = (\{England, France, Spain\}, \{Europe\})$ is a query on the country context (in Table 1), asking for countries in Europe. Another example $q|_{K_{MexicanDish}} = (\{\}, \{rice, corn\text{-}tortilla\})$

When a_q is closed, solving the query consists in finding the concept $C = (a'_q, a_q)$. To ensure that such a concept exists, a virtual query object ov_q that satisfies $ov'_q = a_q$ can be added to the context (as an additional line). Then, three types of answers can be interesting: the more precise answers are in a'_q, less constrained (with less attributes) answers are in extents of super-concepts of C, more constrained (with more attributes) answers are in extents of sub-concepts of C. When a_q is not closed, and we don't use the virtual query object, searching for answers needs to find first the more general concept C whose intent contains a_q. Now we will define more generally what we mean by relational queries.

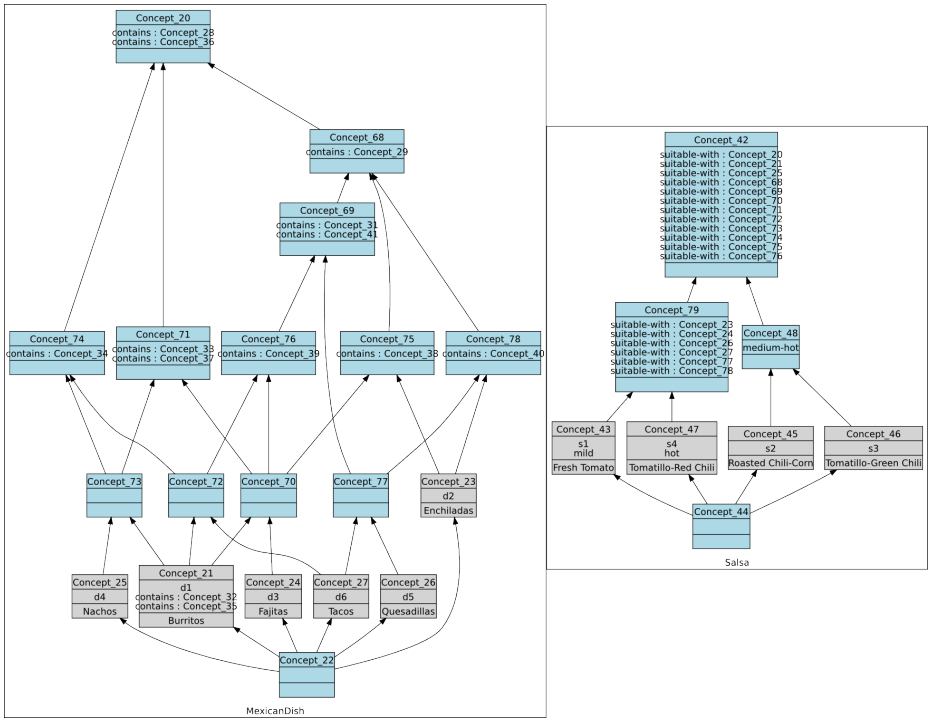


Fig. 4. Dishes and salsa lattices for `exists` and the RCF in Table 1.

3.2 Relational queries

In this study, a relational query is composed of several simple queries, to which we add relational constraints. The relational constraints are expressed via virtual query objects (variables), one for each formal context, where we want to find an object. A virtual query object may have relations (according to the relational contexts) with objects of other contexts, as well as with other virtual query objects.

Definition 5. *A relational query Q on a relational context family $(\mathbb{K}, R)$ is a pair $Q = (A_q, O_{vq}, R_q)$, where:*

1. *A_q is a set of simple queries*
   $A_q = \{q|_{K_i} = (o_q|_{K_i}, a_q|_{K_i}) \mid q|_{K_i}$ is a query on $K_i \in \mathbb{K}\}$
2. *There is a one-to-one mapping between A_q and O_{vq}, where O_{vq} is the set of virtual query objects.*
3. *R_q is a set of relational constraints $R_q = \{(o_{vq|_{K_i}}, r_{ij}, O_q)\}$, where $o_{vq|_{K_i}}$ is the virtual object associated with $q|_{K_i}$, $O_q \subseteq O_j \cup \{o_{vq|_{K_j}}\}$, with $o_{vq|_{K_j}}$ is the virtual object associated with K_j.*

For example, let us consider the following query: I am searching for a country with the attribute "fr", a restaurant in this country serving Mexican dish containing (chicken, cheese, and corn-tortilla), and a salsa which is "hot" and suitable with the dish. This query can be translated into a relational query $Q_{example} = (A_q, O_{vq}, R_q)$ as follows: $A_q = \{q_{country}, q_{rest.}, q_{dish}, q_{salsa}\}$, $a_{q_{country}} = \{fr\}$, $a_{q_{rest.}} = a_{q_{dish}} = \emptyset$, $a_{q_{salsa}} = \{hot\}$.
$O_{vq} = \{o_{vq_{dish}}, o_{vq_{country}}, o_{vq_{rest.}}, o_{vq_{salsa}}\}$
$R_q = \{(o_{vq_{dish}}, \textit{contains}, \{\textit{chicken}, \textit{cheese}, \textit{corn-tortilla}\}), (o_{vq_{country}}, \textit{has}, \{o_{vq_{rest.}}\}), (o_{vq_{rest.}}, \textit{serves}, \{o_{vq_{dish}}\}), (o_{vq_{salsa}}, \textit{suitable-with}, \{o_{vq_{dish}}\})\}$.
By definition, a query corresponds to the data model, and must respect the schema of the RCF (see in Fig. 1).

An answer to the relational query is included in the answers of the simple queries. For our example, the answers of the simple queries would be $o_{q_{country}} = \{France\}$, $o_{q_{rest.}}$ contains all the restaurants, $o_{q_{dish}}$ contains all the dishes, $o_{q_{salsa}} = \{Tomatillo\text{-}Red\ Chili\}$. If we consider these objects connected with the relations, this forms what we call the maximal answer graph. In this graph, we are interested in the subgraphs that cover the query (they have at least one object per element of A_q). These subgraphs are included in the graph of Fig. 5. There are various interesting forms of answer: having exactly one object per element of A_q, or having several objects per element of A_q.

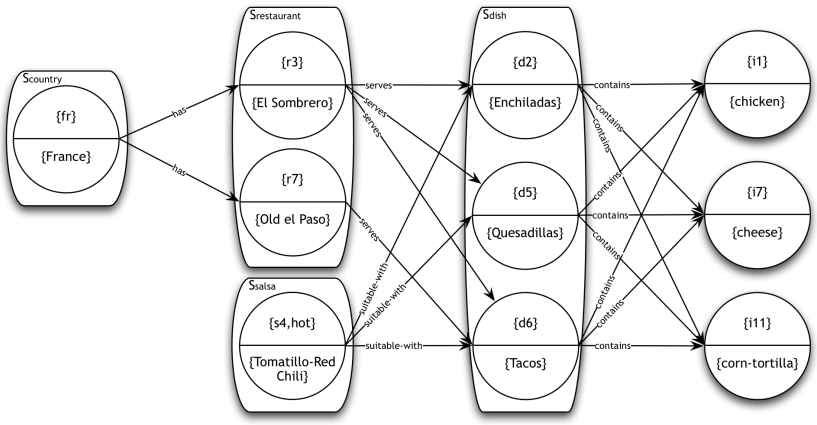


Fig. 5. The subgraph containing all the answers with the relations between the objects corresponding to the relational query example.

Definition 6. *An answer to a relational query $Q = (A_q, O_{vq}, R_q)$ is a set of objects X having a unique object per each context that is involved in the query:*

$$X =< o_i \mid o_i \in O_i \textit{ with } 1 \ \leq \ i \ \leq \ n >$$

These objects satisfy the query $Q = (A_q, O_{vq}, R_q)$, when they have the requested attributes: $\forall\ q|_{K_i} \in A_q,\ \exists\ o_i \in X\ :\ o'_i \supseteq a_{q|_{K_i}}$

and they are connected as expected:
$\forall\ (o_{vq|_{K_i}}, r, O_q) \in R_q$ *with* $r \subseteq O_i \times O_j,\ (and\ thus:\ O_q \subseteq O_j \cup \{o_{vq|_{K_j}}\})$ *and* $\forall\ o \in O_q,\ we\ have:$

1. *if* $o \in O_j,\ we\ have\ (o_i, o) \in r$
2. *if* $o = o_{v_{q|_{K_j}}},\ we\ have\ (o_i, o_j) \in r\ with\ o_j \in X \cap O_j$

For our example, the set of answers to the relational query, is:
$\{\{France, El\ Sombrero, Enchiladas, Tomatillo\text{- }Red\ Chili\}, \{France, El\ Sombrero, Quesadillas, Tomatillo\text{-}Red\ Chili\}, \{France, El\ Sombrero, Tacos, Tomatillo\text{-}Red\ Chili\}, \{France, Old\ el\ Paso, Tacos, Tomatillo\text{-}Red\ Chili\}\}$.

Answers can be provided with an aggregated form which can be found in lattices, as we explain below. They allow us to discover sets of equivalent objects relatively to the answer. E.g. $\{Enchiladas, Quesadillas, Tacos\}$ are equivalent objects if we choose $France$ and $ElSombrero$.

Definition 7. *An aggregated answer to a query* $Q = (A_q, O_{vq}, R_q)$ *is the set AR containing the sets* S_i*, such that:*

- *there is a one-to-one mapping between AR and* A_q *which maps each* $q|_{K_i}$ *to a set* S_i
- $\forall\ q|_{K_i} \in A_q,\ \forall\ o_i \in S_i,\ o_i' \supseteq q|_{K_i}$ *(objects of* S_i *have the requested attributes)*
- *when* $(o_{vq|_{K_i}}, r, O_q) \in R_q$
  - *if* $o_{vq|_{K_j}} \in O_q,\ r \subseteq O_i \times O_j,\ thus:\ \forall\ o_i \in S_i,\ \forall\ o_j \in S_j,\ S_j \in AR,\ we\ have\ (o_i, o_j) \in r$ *(virtual objects are connected if requested)*
  - *for each* $o_j \in O_q \cap O_j$ *we have:* $(o_i, o_j) \in r$ *(connections with particular objects are satisfied).*

For example, an aggregated answer for our query is $\{S_{country}, S_{rest.}, S_{dish}, S_{salsa}\}$ $= \{\{France\}, \{ElSombrero\}, \{Enchiladas, Quesadillas, Tacos\}, \{Tomatillo\text{-}RedChili\}\}$

4 Navigating a Concept Lattice Family w.r.t. a Query

In this section, we explain how the navigation between the concept lattices can be guided by a relational query. Following relational attributes that lead us from one lattice to another, we navigate a graph whose nodes are the concept lattices. In a first subsection, we propose an algorithm which gives a general navigation schema that applies to concept lattices built with the existential scaling. Then we present several variations of this navigation algorithm.

4.1 A query-based navigation algorithm

Our approach for navigating the concept lattices along the relational attributes is based on the observations made during an experimental use of RCA, for finding the appropriate Web services to implement an abstract calculation workflow [1]. We consider an RCF and a query that respects the RCF relations. From our experience, we observed that an expert often expresses his query by a phrase, where the chronology of the principal verbs (relations) gives a natural path for the query flow. This will be our hypothesis. Let us consider the query previously specified: *I am searching for a country, with the basic attribute "fr", that has a restaurant which serves dishes containing chicken, cheese and corn-tortilla; I am searching for a hot salsa suitable with this dish.* In order to simplify the notation, we use the same notation for queries $q|_{k_i}$ and the virtual objects $o_{vq|_{K_i}}$.

The query path is a total ordering of the arcs of the query (the query itself is a DAG in general). For our example, the path is the total ordering for R_q given by $\{(q_{country}, has, \{q_{restaurant}\}), (q_{restaurant}, serves, \{q_{dish}\}), (q_{dish}, contains, \{chicken, cheese, corn\text{-}tortilla\}), (q_{salsa}, suitable\text{-}with, \{q_{dish}\})\}$. Each arc corresponds to a relation used in the query. All the relations involved inside a query are covered by this path. This translation of the expert query determines a composition on the relations. The query path does not always correspond to a directed chain in the object graph (*e.g.* dishes are the end of two of the considered relations (serves and suitable-with)).

We propose the algorithms 1 to 3 (an additional procedure is needed which combines two others) for navigating through a concept lattice family using queries. During the exploration, we fill a set X by objects that will constitute an answer at the end (at most one object for each formal context). In this section, the algorithm is presented as an automatic procedure. Its use to guide an expert in its manual exploration of the data is discussed afterwards.

Algorithm 1 identifies three main cases:

- line 2, the arc connects two query objects, *e.g.* $(q_{country}, has, \{q_{restaurant}\})$;
- line 5, the arc connects a query object to original objects *e.g.* $(q_{dish}, contains, \{chicken, cheese, corn\text{-}tortilla\})$;
- line 8, the arc connects a query object to another query object and to original objects *e.g.* $(q_{dish}, contains, \{q_{ingredient}, chicken, cheese, corn\text{-}tortilla\})$.

Each of these cases considers, for a given arc a, whether the partial answer X already contains a source object or (inclusively) a target object.
When the arc connects a query object to another query object $a = (q|_{K_s}, r_{st}, q|_{K_t})$, (Algorithm 2), four cases are possible.

- X does not contain any object for K_s and any o_t for K_t: we identify the highest concept that introduces the attributes of $q|_{K_s}$ and we select an object in its extent (lines 3-5). Then the algorithm continues on the next conditional statement (to find a target).

- X contains an object o_s for K_s and an object o_t for K_t selected in previous steps: we just check if o_s owns the relational attribute pointing at the object concept introducing o_t, that is γo_t (line 8)[1].
- X contains only an object o_s for K_s. We should find a target. We identify, under the meet of the concepts that introduce the attributes of $q|_{K_t}$, one of the lowest concepts to which o_s points (lines 12-14). We select a target in its extent.
- X contains only an object o_t for K_t. We should find a source. We identify the meet of the concepts that introduce the attributes of $q|_{K_s}$ and the relational attribute that points to o_t (lines 20-23). We select a source in its extent.

When the arc connects a query object to original objects $a = (q|_{K_s}, r_{st}, O_q)$ (Algorithm 3):

- Either X contains an object for K_s and we need to check if the relational attributes confirm that this object is connected to all the original objects in O_q) (line 4);
- Or we have to select an object for K_s, owning the attributes of the query $q|_{K_s}$ and owning the relational attributes ending in the concepts introducing the original objects (line 9-11).

The algorithm for the last case is a combination of the algorithms for the two other cases. Note that whenever a condition is not verified, we have to backtrack, this is not specified in the algorithm for sake of simplicity. If the query path forms also a directed chain in the entity-relationship diagram, the main algorithm is a depth-first search. But in the general case, in some steps, when we consider an arc, we assigned to X an object for the end of the arc, and we need to find a source object.

For example, we start with the arc $(q_{country}, has, \{q_{restaurant}\})$ where the query path begins. We have to identify a source object o_s satisfying the query $\{fr\}$ (Definition 4). For example, we choose the object *France* appearing the extent of $Concept_4$, whose intent contains *fr*.

We extract the relational attributes of $o_s = France$, having the form $\exists r_{st}\ C$). They are in practice in the lattices denoted by $r : C$. For example, we obtain has:*Concept_19*, *has:Concept_15*, *has:Concept_60*, *has:Concept_16*, etc. We keep the relational attributes with the concepts satisfying the target query in the corresponding lattice and discard the rest. In our example, the $q_{restaurant}$ is empty. A relational attribute with the smallest concept (C_t) is the one to consider that leads us to find a solution. We choose *Concept_15* among the available smallest concepts. Let $\exists\ r_{st}\ C_t$ be the selected relational attribute (if it exists). The object o_t must be in the extent of C_t. In our example, we select *El Sombrero*.

Then we consider the query-to-query arc $(q_{restaurant}, serves, \{q_{dish}\})$. Given that an object is selected for $K_{restaurant}$, we look for a possible target object, led by the query $q_{dish} = \emptyset$ and the relational attributes owned by the object

[1] We remind that γo is the object concept introduced by o.

concept $Concept_15$ which introduces *El Sombrero*. Suppose we choose (line 13) a relational attribute that targets one of the minimum concepts, namely $serves : Concept_23$ (but $serves : Concept_26$ or $serves : Concept_25$ are also possible). This leads us to $Concept_23$, in the extent of which we select *Enchiladas*.

Dealing with the next arc $(q_{dish}, contains, \{chicken, cheese, corn\text{-}tortilla\})$ involves, since we have already selected a dish, to verify (Algorithm 3, line 4) that, the object concept $\gamma Enchiladas$ owns all the relational attributes that go to object concepts introducing chicken, cheese, and corn-tortilla. These are $contains : \gamma\ chicken = Concept_29$, $contains : \gamma\ cheese = Concept_36$ and $contains : \gamma\ corn - tortilla = Concept_40$ and they are indeed inherited by $\gamma Enchiladas = Concept_23$.

When the arc $(q_{salsa}, suitable\text{-}with, \{q_{dish}\})$ is considered, the target (Enchiladas) is in X. Thus we identify a source in the extent of the $Concept_47$, which satisfies the target query $\{hot\}$. Its intent contains $suitable - with : Concept_23$ which is *Enchiladas*. A target object (Tomatillo-Red Chili) is selected in the extent of $Concept_47$. The answer is now complete.

Algorithm 1: Navigate(RCF, Q, P_Q) $//P_Q = (a_k) \mid a_k = r_{ij}\ and\ r_{ij} \in R_Q$

```
Data: (𝕂, R): an RCF; Q = (A_q, O_vq, R_q): a query on (𝕂, R); and a query path
Result: X: an object set (answer for Q) or fail
foreach arc a ∈ P_Q do                                                    1
    if a = (q|_{K_s}, r_st, q|_{K_t}) then                                2
        Case_pure_query                                                   3
    else                                                                  4
        if a = (q|_{K_s}, r_st, O_q) with O_q ⊆ O_t then                  5
            Case_pure_objects                                             6
        else                                                              7
            if a = (q|_{K_s}, r_st, q|_{K_t}) with q|_{K_t} ∈ O_q then    8
                Case_query_and_objects                                    9
```

4.2 Variations about the algorithm

Integrating queries into the contexts. One approach that was investigated in the case of simple queries consists of integrating the virtual query object in the context, then building the concept lattice. This can also be done for relational queries. A relational query $Q = (A_q, O_{vq}, R_q)$ can be integrated into an RCF by adding the virtual query objects $o_{v_{q|_{K_i}}}$ into the context K_i. Each virtual query object $o_{vq|_{K_i}}$ owns the attributes of the query $a_{q|_{K_i}}$ and for each arc $(o_{v_{q|_{K_i}}}, r_{ij}, o_{v_{q|_{K_j}}})$, the relational context of r_{ij} is enriched by a line for $o_{v_{q|_{K_i}}}$,

Algorithm 2: Case_pure_query

Let $a = (q|_{K_s}, r_{st}, q|_{K_t})$ 1
if *// X does not contain a source and a target for the current arc a* 2
$X \cap O_s = \emptyset$ and $X \cap O_t = \emptyset$ **then**
  *// **select** a source in the extent of a concept that verifies the source query* 3
  Let C_s be the highest concept having Intent $(C_s) \supseteq q|_{K_s}$
  select $o_s \in Extent(C_s)$ 4
  $X \leftarrow X \cup \{o_s\}$ 5
if *// X contains a source and a target for the current arc a* 6
$X \cap O_s = \{o_s\}$ and $X \cap O_t = \{o_t\}$ **then**
  // verify that the source is connected to the target 7
  check $\exists r_{st}\ \gamma_{o_t} \in Intent(\gamma_{o_s})$ 8
else 9
  if *// X contains a source for the current arc a* 10
  $X \cap O_s = \{o_s\}$ **then**
    // select a target in the extent of a concept that verifies the target query 11
    and is connected to the source
    Let C_t be the highest concept having Intent $(C_t) \supseteq q|_{K_t}$ 12
    and $C_t \in \min(C \mid \exists\ (\exists r_{st}\ C) \in Intent(\gamma o_s))$ 13
    select $o_t \in Extent(C_t)$ 14
    $X \leftarrow X \cup \{o_t\}$ 15
  else 16
    // X contains a target for the current arc a 17
    // select a source in the extent of a concept that verifies the source query 18
    and is connected to the target 19
    Let $o_t \in X \cap O_t$ 20
    Let C_s be the highest concept having Intent $(C_s) \supseteq q|_{K_s}$ 21
    and $\exists r_{st}\ \gamma o_t\ \in Intent(C_s)$ 22
    select $o_s \in Extent(C_s)$ 23
    $X \leftarrow X \cup \{o_s\}$ 24

Algorithm 3: Case_pure_objects

Let $a = (q|_{K_s}, r_{st}, O_q)$ with $O_q \subseteq O_t$ 1
if *// X contains a source for the current arc a* 2
$X \cap O_s = \{o_s\}$ **then**
  // verify that the source is connected to the objects in O_q 3
  check $\forall o \in O_q, \exists r_{st}\ \gamma_o \in Intent(\gamma_{o_s})$ 4
else 5
  // X does not contain a possible source 6
  // select a source in the extent of a concept that verifies the source query 7
  // and is connected to the target objects 8
  Let C_s be the highest concept having Intent $(C_s) \supseteq q|_{K_s}$ 9
  and $\forall o \in O_q, \exists r_{st}\ \gamma_o \in Intent(C_s)$ 10
  select $o_s \in Extent(C_s)$ 11
  $X \leftarrow X \cup \{o_s\}$ 12

a column for $o_{v_{q|K_j}}$ and the relation $(o_{v_{q|K_i}}, o_{v_{q|K_j}})$[2]. We generate the corresponding concept lattice family, considering the existential scaling [3]. Locating the highest concept that introduces all the attributes of each query of each concerned context, now is much more easy because it introduces the virtual query object. Then, we can navigate in a similar way as before.

Opportunities of browsing offered by the exploration. As we explained before, the algorithm described in the previous section can be understood as an automatic procedure to determine a solution to a query. Nevertheless, it is more interesting to use it as a guiding method for the exploration of data by a human expert. Each object selection is a departure point for inspecting the objects of the selected concept, and, explore the neighborhood, going up by relaxing constraints or going down by adding constraints.

A point in favor of the lattices is that they do not only give us a solution, but they also classify the objects of the solutions and provide a navigation structure. They also carry other information about the objects which can be useful for the expert: attributes that objects of the answer set have necessarily, attributes that appear simultaneously as attributes of the answer, etc.

In our Web service application, we preferred the solution which integrates the query in RCF because it was easier to identify the answers. The lattices show how the existing objects match and differ from the query, thanks to the factorization of attributes between the query and the existing objects. Nevertheless, having several queries at the same time would not be efficient. Thus, the solution has been used only for specific problems. An incremental algorithm can be used to introduce the query, which enlightens the process of modifying the lattice and highlights the structure of the data. We can keep the original lattice (before query integration), and save the query objects together with the resulting concepts in an auxiliary structure. This way, we can always easily go back to the original lattices.

5 Related Work

ER-compatible data, e.g., relational databases, and concept lattices have a long history of collaboration. First attempts to apply FCA to that sort of data go back to the introduction of concept graphs by R. Wille in the mid-90s [8]. The standard approach is rooted in the translation of an ER model into a power-context family (PCF) where basically everything is represented within a formal context [9]. Thus, inter-object links of various arities (i.e., tuples of different sizes) are reified and hence become formal objects of a dedicated context (one per arity). The overall reasoning is therefore uniformly based on the formal concepts.

[2] See our example in Table `http://www.lirmm.fr/~huchard/RCA_queries/mexicoExistsWithQuery.rcft.html`)

[3] It is represented in Figure `http://www.lirmm.fr/~huchard/RCA_queries/mexicoExistsWithQuery.rcft.svg`

While this brings an undeniable mathematical strength in the formalization of the data processing and, in particular, querying, there are some issues with the expressiveness. Indeed, while formal concepts are typically based on a doubly universal quantification, the relational query languages mostly apply existential one.

Alternatives to the PCF in the interpretation of concept graphs have been proposed that involve the notions of nested graphs and cuts [2]. It was shown that the resulting formalism, called Nested Query Graphs, have the same expressive power over relational data as first order predicate logic and hence can be used as a visual representation of most mainstream SQL queries.

Existing approaches outside the concept graphs-based paradigm (see [3, 6]) follow a more conventional coding schema. Here inter-object links are modeled either through a particular sort of formal attributes or they reside in a different binary tables that match two sorts of individuals among them (instead of matching a set of individuals against a set of properties). Our own relational concept analysis framework is akin to this second category of approaches, hence our querying mechanisms are closer in spirit to those presented in the aforementioned papers.

For instance, in [3], the author proposes a language modeled w.r.t. SPARQL (the query language associated with the RDF language) to query relational data within the logical concept analysis (LCA) framework. The idea is to explore the relation structure of the data, starting from a single object and following its links to other objects. The language admits advanced constructs such as negation and disjunction and therefore qualifies as a fully-fledged relational query language.

Recently, a less expressive language has been proposed in [6] for the browsing of a relational database content while taking advantage of the underlying conceptual structure. As the author himself admits, the underlying data format used to ground the language semantics, the linked context family, is only slightly different from our own relational context family construct. The queries are limited here to conjunctions and existential quantifiers, yet variables are admitted. Consequently, query topologies are akin to general graphs: In actuality, the browsing engine comprises a factorization mechanism enabling the discovery of identical extensions in the query graph which are subsequently merged.

The downside of remaining free of the extensive commitments made by the concept graphs formalism both in terms of syntax and of semantics is the lack of unified methodological and mathematical framework beneath this second group of approaches. As a result, these diverge on a wide range of aspects which makes their in-depth comparison a hard task.

First, there is an obvious query language expressiveness gap: On that axis, the two extremes are occupied by the LCA- and the RCA-based approaches, respectively, the former being the most expressive and the latter, the less expressive one. Then, the role played by the concept lattices vs. the query resolution is specific in each case. While in the LCA-based approach the concepts seem to be formed on the fly, in [6] the author seems to imply that they are constructed beforehand. Despite this distinction, in both cases the concept lattice is a sec-

ondary structure that supports query resolution. In our own approach however, lattices are not only constructed prior to querying, but they also incorporate relational information in the intents of their concepts. In this sense, they are the primary structures whereas the queries are intended as navigational support.

6 Conclusion

In this paper, we have presented a query-based navigation approach that helps an expert to explore a concept lattice family. The approach was based on an application of Relational Concept Analysis to the selection of suitable Web services for instantiating an abstract service composition. There are many perspectives of this work. In our Web service experience, we tested other scaling operators (like the *covers* operator) that offers other results, and helps to find more easily the aggregate answers. The query language can be made more expressive (including quantifiers). For example, we can request dishes containing *only* {chicken, cheese, ...}, which means that the universal scaling operator shall be used in the RCA process for this particular relation. Besides, the query path can be calculated, rather than being defined by the expert, suggesting more efficient exploration paths.

References

1. Azmeh, Z., Driss, M., Hamoui, F., Huchard, M., Moha, N., Tibermacine, C.: Selection of composable web services driven by user requirements. In: ICWS. pp. 395–402. IEEE Computer Society (2011)
2. Dau, F., Correia, J.H.: Nested concept graphs: Applications for databases and mathematical foundations. In: Contribution to ICCS 2003. Skaker Verlag (2003)
3. Ferré, S.: Conceptual navigation in RDF graphs with SPARQL-Like Queries. In: Kwuida, L., Sertkaya, B. (eds.) ICFCA. LNCS, vol. 5986, pp. 193–208. Springer (2010)
4. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundations. Springer-Verlag (1999)
5. Huchard, M., Rouane-Hacene, M., Roume, C., Valtchev, P.: Relational concept discovery in structured datasets. Ann. Math. Artif. Intell. 49(1-4), 39–76 (2007)
6. Kötters, J.: Object configuration browsing in relational databases. In: Valtchev, P., Jäschke, R. (eds.) ICFCA. Lecture Notes in Computer Science, vol. 6628, pp. 151–166. Springer (2011)
7. Messai, N., Devignes, M.D., Napoli, A., Smaïl-Tabbone, M.: Querying a bioinformatic data sources registry with concept lattices. In: Dau, F., Mugnier, M.L., Stumme, G. (eds.) ICCS. LNCS, vol. 3596, pp. 323–336. Springer (2005)
8. Wille, R.: Conceptual graphs and formal concept analysis. In: Lukose, D., Delugach, H.S., Keeler, M., Searle, L., Sowa, J.F. (eds.) ICCS. Lecture Notes in Computer Science, vol. 1257, pp. 290–303. Springer (1997)
9. Wille, R.: Formal concept analysis and contextual logic. In: Hitzler, P., Scharfe, H. (eds.) Conceptual Structures in Practice. pp. 137–173. Chapman and Hall/CRC (2009)