Cubes of Concepts: Multi-dimensional Exploration of Multi-valued Contexts

Sébastien Ferré, Pierre Allard, Olivier Ridoux

Cubes of Concepts: Multi-Dimensional Exploration of Multi-Valued Contexts

Sébastien Ferré, Pierre Allard, and Olivier Ridoux

IRISA/Université de Rennes 1, Campus de Beaulieu, 35042 Rennes cedex, France
Firstname.Lastname@irisa.fr

Abstract. A number of information systems offer a limited exploration in that users can only navigate from one object to another object, e.g. navigating from folder to folder in file systems, or from page to page on the Web. An advantage of conceptual information systems is to provide navigation from concept to concept, and therefore from set of objects to set of objects. The main contribution of this paper is to push the exploration capability one step further, by providing navigation from set of concepts to set of concepts. Those sets of concepts are structured along a number of dimensions, thus forming a cube of concepts. We describe a number of representations of concepts, such as sets of objects, multisets of values, and aggregated values. We apply our approach to multi-valued contexts, which stand at an intermediate position between many-valued contexts and logical contexts. We explain how users can navigate from one cube of concepts to another. We show that this navigation includes and extends both conceptual navigation and OLAP operations on cubes.

Keywords: formal concept analysis, information systems, data exploration, navigation, multi-valued context, multi-dimensional analysis, OLAP cubes.

1 Introduction

Navigation is a convenient way for exploring data, compared to querying, because it provides guiding to users during their search, and supports exploratory search [16]. At each step of a navigation path, users are located at a navigation place, and a number of navigation links are suggested to them in order to reach neighbour navigation places. However, most existing systems suffer from a number of limitations, e.g., file systems and the Web. A first limitation is that navigation places are objects (e.g., files and folders, Web pages), and users can therefore only jump from object to object. This makes it difficult to group and compare objects. A second limitation is that the navigation graph is manually drawn. As a consequence, the navigation graph is generally very sparse, and requires from users chance or expertise or both to find their way to the searched objects. For example, if photos are organized in a file system first by date then by topic, it is easy enough to find photos by date, difficult to find photos by topic, and impossible to find them by depicted person. A third limitation is

that navigation links may lead to empty results, e.g., an empty folder. This is a frustrating experience for users who have to resort to tedious trial-and-error. A fourth limitation is that the navigation path must be linear. At each step, only one navigation link can be chosen. If users want to explore several navigation links, they have to choose and explore one, then to move backward in the navigation history, and choose a second one, and so on.

Conceptual navigation [14,3,15,6,7] overcomes the first three limitations by relying on Formal Concept Analysis (FCA) [13]. Users navigate from concept to concept, and hence from set of objects to set of objects. The navigation graph is the concept lattice, which is automatically derived from data, a formal context, and therefore automatically adapts to changes in data. Finally, it is easy to prevent empty results, by removing the bottom concept from the navigation graph. Regarding the fourth limitation, if the concept lattice is small enough, the line-diagram enables to view all navigation paths at a glance. In practice, however, the concept lattice is much too large to be visualized, and only the current concept and its neighbours are displayed. Existing conceptual navigation systems follow that principle, and only allow for linear navigation paths.

The contribution of this paper is to extend conceptual navigation to non-linear navigation paths. Concretely, this means that users can choose a set of navigation links at some step, leading to a set of concepts. As several multi-choices can be done in sequence, a navigation place becomes a multi-dimensional set of concepts, a *cube of concepts*. We show that our approach covers and extends the multi-dimensional analysis of OLAP [5,20]. We define cubes of concepts on *multi-valued contexts*, which we introduce in this paper. They extend many-valued contexts by allowing several values for the attribute of an object, rather than only one. The advantages of using multi-valued contexts are (1) more general results that directly apply to many-valued contexts and binary contexts, and (2) a data model close to databases [4], OLAP [5,20], the Semantic Web [2], and Logical Information Systems [10].

After introducing useful notations on mappings, multisets, tuples, and FCA in Section 2, Section 3 defines multi-valued contexts, and value domains. Section 4 defines cubes of concepts to be used as navigation places, and cube transformations as navigation links. Those cube transformations are proved safe, i.e. not leading to empty results. Section 5 discusses the representation of cubes of concepts in Abilis. Section 6 compares cubes of concepts and their transformations to OLAP cubes and operators. Section 7 compares our approach to other approaches combining FCA and OLAP.

2 Preliminaries

A *mapping* M from a set A to a set B is a set of couples $a \mapsto b$, where $a \in A$ and $b \in B$, and such that $(a, b_1) \in M \wedge (a, b_2) \in M \Rightarrow b_1 = b_2$. It is a partial function from A to B that is defined in extension. Its domain is noted $dom(M)$, and its range is noted $ran(M)$. A *multiset* M over a set A is a mapping from A to natural numbers $\mathbb{N}$. It generalizes sets by allowing elements to occur several

times in a multiset: $a \mapsto n \in M$ means that a has *multiplicity* n in M. For convenience, we define the following notations on a multiset M: $M^\alpha = \{a \mapsto n \in M \mid n \geq \alpha\}$ is the filtering of multiset M by a minimum multiplicity α, $M_Y = \{a \mapsto n \in M \mid a \in Y\}$ is the restriction of the domain of M to Y, and $\#M = \Sigma\{n \mid a \mapsto n \in M\}$ is the cardinal of multiset M.

Given a *tuple* $\overline{x}$ of dimension n, we denote by x_i the i-th component of the tuple. The notation $\overline{x}$ is a shorthand for $x_1 \dots x_n$. We define the following notations on tuples: $\overline{x} + x' = x_1 \dots x_n x'$ is the extension of a tuple $\overline{x}$ by an element x', $\overline{x} - i = x_1 \dots x_{i-1} x_{i+1} \dots x_n$ is the restriction of a tuple $\overline{x}$ by removing the i-th element, $\overline{x}[i \leftarrow x'] = x_1 \dots x_{i-1} x' x_{i+1} \dots x_n$ is the replacement of the i-th element by x', and $dom_i(X) = \{x_i \mid \overline{x} \in X\}$ is the domain of the i-th elements, where X is a set of tuples of same dimension n.

We here recall some basic notations for Formal Concept Analysis (FCA) [13]. However, we adopt a somewhat novel presentation in that *intension* is made a particular case of a more general notion: the *index*. A *formal context* is a triple $K = (O, A, I)$, where O is the set of objects, A is the set of attributes, and $I \subseteq O \times A$ is the incidence relation between objects and attributes. The *extension* of a set of attributes Y is defined as the set of objects that have all the attributes in Y, i.e., $ext(Y) = \{o \in O \mid \forall a \in Y : (o, a) \in I\}$. By abuse of notation, we will denote by $ext(\{a\})$ the expression $ext(a)$ for . The *index* of a set of objects X is defined as the multiset of attributes of objects in X, where the multiplicity of each attribute is the number of its objects in X, i.e.,

$$index(X) = \{a \mapsto n \mid a \in A, n = \#(X \cap ext(a)) \neq 0\}.$$

The *intent* of a set of objects X is then defined as the domain of the index of X filtered by the maximum frequency, i.e., $int(X) = dom(index(X)^{\#X})$. A *concept* c is a pair (X, Y) s.t. $X = ext(Y)$ and $Y = int(X)$: $ext(c) = X$ is called the *extent* of c, and $int(c) = Y$ is called the *intent* of c.

3 Multi-Valued Contexts

In order to better represent real datasets and complex data, a number of extensions or generalizations of FCA have been proposed: many-valued contexts [12], logical contexts [9], pattern structures [11], to cite only a few of them. In particular, many-valued contexts are equivalent to tables in relational databases. Logical contexts (and pattern structures) are more general, but miss the distinction between attributes and values, which will be useful in this paper. Conversely, logical contexts enable to describe a photo as depicting both Alice and Bob, which is not possible in a many-valued context where the photo is an object, "depicts" is an attribute, and persons are values: "depicts" is a *multi-valued* attribute. The term "many-valued" means that an attribute can take one among many values, whereas the term "multi-valued" here means that an attribute can take multiple values at once. Multi-valued attributes do not exist in relational databases because those are normalized into several tables (one table for photos, and one table for the "depicts" relation). As FCA generally handles a single table, the

context, it is important to allow for multi-valued attributes. This leads us to the definition of *multi-valued contexts* as a set of triples (object,attribute,value), like graphs in the RDF data model [2].

Definition 1 (multi-valued context). *A* multi-valued context *is a quadruple* $K = (O, A, V, T)$*, where* O *is the set of objects,* A *is the set of (valued) attributes,* V *is the set of values, and* $T \subseteq O \times A \times V$ *is a set of triples. Each triple* (o, a, v) *means that* v *is a value taken by the atribute* a *on object* o*.*

As the set $O \times A \times V$ is equivalent to $O \times A \to \mathcal{P}(V)$, a multi-valued context can also be represented like a many-valued context, but allowing for zero, one or several values in each cell. The following table defines a multi-valued context K_e of 6 photos described by the persons they depict, the year they were taken, and their size in pixels. It has 19 triples.

object	person	date	size
1	Alice	2010-03-19	1.1M
2	Bob	2010-07-13	3.1M
3	Charlie	2011-01-30	1.2M
4	Alice, Bob	2011-04-28	3.2M
5	Alice, Charlie	2011-08-20	1.3M
6	Alice	2011-11-11	

This example illustrates the fact that an object can have several values for an attribute (e.g., the depicted persons of photo 4), or no value at all (e.g., the size of photo 6). This example is kept small for illustration purposes, but our approach is designed for datasets with hundreds to thousands of objects, and tens of attributes.

3.1 Value Domains and Attribute-Value Schemas

In practice, there are interdependencies between attributes and values on one hand, and between values on the other hand. Each attribute expects values from a given domain, and the values of a domain can be organized into a generalization ordering (e.g., locations, date intervals). We first define domains of values in order to formalize hierarchies of values, and aspects related to OLAP such as granularity levels, and aggregators.

Definition 2 (value domain). *A* value domain *is a structure* $D = (V, \sqsubseteq, \top, \Lambda, \Gamma)$*, where:*

- V *is the set of values,*
- $\sqsubseteq$ *is a partial ordering (called* subsumption*) that represents the generalization ordering between values, and* $\prec$ *is the corresponding covering relation,*
- $\top$ *is a distinguished value, more general than any value,*
- $\Lambda \subseteq \mathcal{P}(V)$ *is a set of granularity levels, each level* $\lambda \subseteq V$ *being defined as a subset of values,*
- Γ *is a set of aggregators over multi-sets of values.*

The example multi-valued context K_e uses three value domains, respectively for persons, dates, and sizes.

person Values are either persons (e.g., Alice, Bob, Charlie) or groups (e.g., family, friends). A person is subsumed by every group it belongs to. Persons and groups constitute the two granularity levels: $\Lambda = \{individual, group\}$. The only applicable aggregators are the count ($\gamma(M) = \#M$) and the distinct count ($\gamma(M) = \#dom(M)$).

date Values are dates at four granularity levels: days (e.g., 2010-03-19), months (e.g., 2010-03), years (e.g., 2010), and calendar weeks (e.g., 2010:42). Date values represent intervals of time, and are hierarchically organized by inclusion. Applicable aggregators are, in addition to the count and distinct count, the earliest date, the latest date, and the median date.

size Values are natural numbers at various precisions (e.g., 1M, 1.2M, 1234k). There is a granularity level for each level of precision: units (e.g., 1234567), tens (e.g., 1234.56k), ..., millions (e.g., 1M). Size values represent integer intervals, and are hierarchically organized by inclusion. Applicable aggregators are, in addition to those of dates, the sum and the average.

There are different ways to define the hierarchy of values of a value domain. A first way is to use scale contexts [12] when the set of values is finite and fixed. A second way is to define logics as in Logical Concept Analysis (LCA, [10]), which works well for infinite domains (e.g., integers, dates, geographic shapes), and allows for the dynamic and automatic insertion of new values in the hierarchy.

Subsumption can be extended from values to granularity levels: for any two levels λ_1, λ_2, we have $\lambda_1 \sqsubseteq \lambda_2$ iff for every $v_1 \in \lambda_1$, there exists $v_2 \in \lambda_2$ s.t. $v_1 \sqsubseteq v_2$. While levels are generally disjoint and totally ordered, they need not be. A classical example is the date domain, with four disjoint levels: days, months, weeks, and years. The level "day" is subsumed by the levels "month" and "week", both of which are in turn subsumed by the level "year", but neither "month" is subsumed by "week" nor the converse. Sometimes, there are no natural levels, and we want to define them from the topology of the hierarchy. Each level corresponds to a certain depth in the hierarchy, and they may overlap.

Definition 3 (topological levels). *Let $D = (V, \sqsubseteq, \top, \Lambda, \Gamma)$ be a value domain. The granularity level $\lambda(p)$ denotes the topological level at depth p, and is recursively defined as follows for every $p > 0$:*

- $\lambda(0) = \{\top\}$,
- $\lambda(p+1) = \{v \in V \mid \exists v' \in \lambda(p) : v \prec v'\} \cup \{v \in \lambda(p) \mid \nexists v' \in V : v' \prec v\}$.

3.2 Attribute Contexts and Feature Context

A multi-valued attribute is a binary relationship between objects and values. A formal context can therefore be derived for each attribute, with the help of a value domain for handling subsumption between values. It is called an *attribute context*, and it plays the same role as a realized scale.

	person			date		size	
photo	Alice	Bob	Charlie	2010	2011	1M	3M
1	×			×		×	
2		×		×			×
3			×		×	×	
4	×	×			×		×
5	×		×		×	×	
6	×				×		

Table 1. The feature context of the multi-valued context K_e on photos (only some levels of values are shown).

Definition 4 (attribute context). *Let $K = (O, A, V, T)$ be a multi-valued context, $a \in A$ be an attribute, and D a value domain for a. The* attribute context *of a is defined as $K_a = (O, V_a, I_a)$, where $I_a = \{(o, v) \in O \times V_a \mid (o, a, v') \in T, v' \sqsubseteq_a v\}$.*

In the attribute context K_a, $ext_a(Y)$ is the set of objects having all values of Y as values of the attribute a (directly or by subsumption), e.g., the set of photos depicting a set of persons together; $index_a(X)$ is the mappings from values in V_a to their frequency as the value of the attribute a over the set of objects X, e.g., the distribution of persons that are depicted in a given set of photos. Finally, $int_a(X)$ is the set of values in V_a occurring in all objects in X through attribute a, e.g., the set of persons depicted by all elements of given set of photos.

The feature context K_F is the apposition (see p. 30 in [13]) of all attribute contexts, distinguishing values by prefixing them with attributes. This is the natural translation of a multi-valued context into a formal context.

Definition 5 (feature context). *Let $K = (O, A, V, T)$ be a multi-valued context, and $(D_a)_{a \in A}$ be a collection of value domains. The* feature context *is defined as $K_F = (O, F, I_F)$, where the set of features F, and the incidence relation I_F between objects and features is defined by:*

- $F = \{(a, v) \mid a \in A, v \in V_a\}$,
- $I_F = \{(o, (a, v)) \mid (o, a, v') \in T, v' \sqsubseteq_a v\}$.

In the feature context, the extension $ext_F(Y)$ returns the set of objects having a set of features Y; and the index $index_F(X)$ returns the distribution of features over a set of objects X. Table 1 contains a partial representation of the feature context of K_e, showing only one level of values for each attribute (persons, years, and millions of pixels). The FCA operations of the feature context can be related to the FCA operations of attributes contexts as in Lemma 1.

Lemma 1. *The following equations holds for every multi-valued context K:*

1. $ext_F((a, v)) = ext_a(v)$,
2. $int_F(X) = \bigcup_{a \in A} \{(a, v) \mid v \in int_a(X)\}$,
3. $index_F(X) = \bigcup_{a \in A} \{(a, v) \mapsto n \mid v \mapsto n \in index_a(X)\}$.

4 Cubes of Formal Concepts as Navigation Places

In conceptual navigation, navigation places are formal concepts. We are here interested in navigating in the concept lattice of the feature context because it contains all the information of a multi-valued context, and its associated value domains. In the following, we assume a multi-valued context $K = (O, A, V, T)$ together with value domains D_a for each $a \in A$, and its derived feature context $K_F = (O, F, I_F)$. In this paper, we start from the framework of Logical Information Systems (LIS, [10]) with a restriction to conjunctive queries, whereas disjunction and negation are generally available in LIS. For the sake of simplicity, we will stick in this paper to this restriction, but the following results easily extent to Boolean queries. In LIS, the current concept c is defined by $ext(c) = ext_F(q)$, where $q \subseteq F$ is the current query. This query is the result of the successive choices of a single feature (*single-choices*) among those suggested by the system, starting from the empty set, and hence from the top concept. We now want to extend conceptual navigation to choices of multiple features (*multi-choices*), in order to escape the linearity limitation, and to provide multi-dimensional analysis. This raises the following questions: What is a natural multi-choice? What is a navigation place after performing a multi-choice? What defines such a navigation place?

Starting from the example multi-valued context K_e, suppose a user wants to look at photos by depicted person. He can select in turn each person, successively visiting the concepts 1456, 24, and 35 (for convenience, we name concepts after their extension: 24 is the concept whose extension is $\{2, 4\}$). This is tedious and unpractical for comparing the three subsets of photos. Alternately, the user could perform a multi-choice of the three persons, leading him to the set of the three concepts, indexed by the three persons. This results in the mapping $\{Alice \mapsto 1456, Bob \mapsto 24, Charlie \mapsto 35\}$. From there, the user wants to focus on photos depicting Alice, and performs a single choice of the feature $(person, Alice)$. The effect of this single-choice applies to each of the three above concepts, leading to the mapping $\{Alice \mapsto 1456, Bob \mapsto 4, Charlie \mapsto 5\}$. Bob and Charlie persist as indices because they are depicted with Alice on some photos. Finally, the user performs a second multi-choice of sizes at the level of millions (1M and 3M). The effect of this multi-choice applies to each concept of the last mapping, producing a mapping of mappings or, in a more compact form, a mapping from couples (person, size) to concepts: $\{(Alice, 1M) \mapsto 15, (Alice, 3M) \mapsto 4, (Bob, 3M) \mapsto 4, (Charlie, 1M) \mapsto 5\}$. Only concepts whose extension is not empty are retained, so that not all combinations of a person and a size are present.

This example scenario shows that a natural multi-choice is a granularity level of some attribute, which we define as an *axis* in analogy with graph axes.

Definition 6 (axis). *An* axis $x = a/\lambda$ *combines an attribute* $a \in A$ *and a granularity level* $\lambda \in \Lambda_a$ *for that attribute. In the following,* $a(x)$ *denotes the attribute of the axis,* $\lambda(x)$ *denotes the level of the axis, and* $index_x(X)$ *is a shorthand for* $(index_a(X))_\lambda$*, the* a*-index over objects* X*, restricted to the level* λ*.*

The example scenario also shows that navigation places are now mappings from tuples of values to concepts of the feature context. Those mappings are equivalent to n-dimensional arrays, where n is the size of tuples. Because the dimension of those arrays can be any natural number, and in analogy with OLAP, we call them *cubes of concepts*. Section 7 compares them to OLAP cubes. What determines such a cube is a sequence of single-choices and multiples-choices. The sequence of single-choices amounts to a set of features, the *query* q, and the sequence of multi-choices amounts to a tuple of axes, the dimension tuple $\overline{d}$.

Definition 7 (cube of concepts). *Given a query $q \subseteq F$, and a tuple $\overline{d}$ of n axes as dimensions, the* cube of concepts *is defined as a mapping from* coordinates *(tuples of values) to concepts:*

$$Cube(q, \overline{d}) = \{\overline{v} \mapsto c \mid \overline{v} \in \prod_{i=1}^{n} \lambda(d_i), ext(c) = ext_F(q) \cap \bigcap_{i=1}^{n} ext((a(d_i), v_i)) \neq \emptyset\}.$$

The above example shows that different concepts in a cube of concepts may overlap, and even that a same concept can appear at different coordinates. This comes from multi-valued contexts, where an object can have several values on a same axis (e.g., Alice and Bob as depicted persons). The example also shows that some coordinates, i.e. some combinations of values, may be missing in the cube. The reason is that either no object matches this combination, or an object has no value on some axis (e.g., photo 6 has no size value). Those are key differences with OLAP cubes (Section 7).

It is possible to generalize some definitions from concepts to cubes.

Definition 8. *The extension of the cube of concepts is the union of the extensions of the concepts. The intent and index of a cube of concepts can be derived from its extension as usual.*

$$ext(C) = \bigcup_{\overline{v} \mapsto c \in C} ext(c) \quad int(C) = int_F(ext(C)) \quad index(C) = index_F(ext(C))$$

Beware that the extension of a cube is not necessarily a concept extension, because concept extensions are not closed under set union. However, it is important not to close it, so that the index $index(ext(C))$ properly reflects the contents of the cube, and not the larger $ext_F(int(C))$ that may contain objects not visible in the cube.

In LIS, navigation links are defined by query transformations [7], rather than by the covering relation of the concept lattice. Query transformations are the addition or the removal of a feature, and combinations such as the replacement of a feature. Adding a feature f to a query q, noted $q + f$, provides downward navigation in the concept lattice. This gives access not only to lower neighbours, but also to concepts deeper in the lattice. All relevant features are suggested as navigation links, and not only those leading to a lower neighbour. Removing a feature f, noted $q - f$, provides upward navigation in the concept lattice. Of course, features can be removed in a different order they were added to the

query. Addition and removal can be combined to provide sideward navigation, e.g., shifting from photos of Alice in 2010 to photos of Bob in 2010, and then to photos of Bob in 2011.

In this paper, because navigation places are cubes of concepts, we define navigation links as transformations of cubes of concepts. As a cube of concepts is made of a query q and a tuple of dimensions $\overline{d}$, the above query transformations equally apply to cubes of concepts. A second way to transform a cube is to change the dimensions of the cube. Possible transformations are:

- $\overline{d} + d'$: the addition of a dimension d',
- $\overline{d} - i$: the removal of a dimension d_i,
- $\sigma(\overline{d})$: a permutation σ on $\overline{d}$ to change the ordering of dimensions.

An important property of conceptual navigation is *safeness*, i.e. to suggest only navigation links that lead to concepts whose extension is not empty. This is important to avoid dead-ends, trial-and-error navigation, and hence frustration for users. With cubes of concepts, navigation is safe if it never leads to empty cubes. The following theorem states the conditions under which a cube transformation is *safe*.

Theorem 1. *Let $C = Cube(q, \overline{d})$ be a cube of concepts. The specializing cube transformations are safe under the following conditions:*

- *$Cube(q + f, \overline{d})$ if $f \in index_F(ext(C))$,*
- *$Cube(q, \overline{d} + d')$, if $index_{d'}(ext(C)) \neq \emptyset$.*

The generalizing transformations $Cube(q - f, \overline{d})$ and $Cube(q, \overline{d} - i)$, as well as the permutation transformation $Cube(q, \sigma(\overline{d}))$, are necessarily safe.

Proof. We give the proofs for the specializing transformations. The proof for other transformations are trivial.

- Proof for $Cube(q + f, \overline{d})$:
  $f \in index_F(ext(C))$
  $\Rightarrow ext_F(f) \cap ext(C) \neq \emptyset$
  $\Rightarrow ext_F(f) \cap \bigcup_{\overline{v} \mapsto c \in C} ext(c) \neq \emptyset$ (Definition 8)
  $\Rightarrow \exists \overline{v} \mapsto c \in C : ext_F(f) \cap ext(c) \neq \emptyset$ (Definition 7)
  $\Rightarrow \exists \overline{v} \in \prod_i \lambda(d_i) : ext_F(f) \cap (ext_F(q) \cap \bigcap_i ext((a(d_i), v_i))) \neq \emptyset$
  $\Rightarrow \exists \overline{v} \in \prod_i \lambda(d_i) : ext_F(q + f) \cap \bigcap_i ext((a(d_i), v_i)) \neq \emptyset$
  (because $ext(f) \cap ext(q) = ext(q \cup \{f\}) = ext(q + f)$)
  $\Rightarrow \exists \overline{v} \mapsto c' \in Cube(q + f, \overline{d})$
  $\Rightarrow Cube(q + f, \overline{d}) \neq \emptyset$.
- Proof for $Cube(q, \overline{d} + d')$:
  $index_{d'}(ext(C)) \neq \emptyset$
  $\Rightarrow index_{a(d')}(ext(C))_{\lambda(d')} \neq \emptyset$ (Definition 6)
  $\Rightarrow \exists v' \in \lambda(d') : v' \in index_{a(d')}(ext(C))$
  $\Rightarrow \exists v' \in \lambda(d') : (a(d'), v') \in index_F(ext(C))$
  $\Rightarrow \exists v' \in \lambda(d') : ext_F((a(d'), v')) \cap ext(C) \neq \emptyset$ (Lemma 1)

$\Rightarrow \exists \overline{v} \mapsto c \in C : \exists v' \in \lambda(d') : ext((a(d'), v')) \cap ext(c) \neq \emptyset$ (Definition 8)
$\Rightarrow \exists \overline{v} \in \prod_i \lambda(d_i) : \exists v' \in \lambda(d') : ext((a(d'), v')) \cap (ext_F(q) \cap \bigcap_i ext((a(d_i), v_i))) \neq \emptyset$
$\Rightarrow \exists \overline{v} + v' \in (\prod_i \lambda(d_i)) \times \lambda(d') : ext_F(q) \cap ((\bigcap_i ext((a(d_i), v_i))) \cap ext((a(d'), v'))) \neq \emptyset$
$\Rightarrow \exists \overline{v} + v' \mapsto c' \in Cube(q, \overline{d} + d')$
$\Rightarrow Cube(q, \overline{d} + d') \neq \emptyset.$ □

The condition for the addition of a feature to the query is the same as previously known in LIS. It establishes the feature index $index_F(ext(C))$ of a cube C as the set of features that can be added to the query. The condition for the addition of a dimension involves the indexes for each axis, which are included in the feature index (see Lemma 1). Therefore, the extension of conceptual navigation from concepts to cubes of concepts does not entail any increase in the size of suggested navigation links. A suggested dimension is simply an axis that shares values with the feature context. This is consistent with multi-choices being sets of choices.

5 Representation and Interaction in Abilis

In LIS systems, e.g. Camelis and Abilis[1], the current concept c is represented by the query q that defines it, the extension $ext(c)$ of the concept as a list of objects, and the feature index $index(ext(c))$ over that extension, which includes the intension $int(c)$ of the current concept. The feature index is organized by attribute, like facets in Faceted Search [18], and is displayed as a tree to reflect the generalization ordering between values. The query can be transformed by selecting features in the index. A feature is removed from the query if it belongs to it, otherwise it is added to the query.

Multi-dimensional conceptual navigation with cubes of concepts has been implemented in Abilis, along with rich capabilities to represent cubes of concepts. With a cube of concepts $C = Cube(q, \overline{d})$, we still have a query q, an extension $ext(C)$, and an index $index_F(ext(C))$ over that extension. The index is also the support of interaction by suggesting features and axes to be added or removed from the cube. The important difference is that a navigation place is a set of concepts projected at some coordinates instead of a single concept. Each concept defines a unit of knowledge, and has many possible concrete representations. The two obvious representations of a concept are its extension and its intention. Other useful representations are attribute indexes, and also aggregations of attribute indexes. By analogy with OLAP, we call a possible representation of a concept a *measure*, even if OLAP measures are generally limited to aggregated values.

Definition 9 (measure). *A* measure *is defined as any function from a concept to a piece of data representing some aspect of that concept. Given a multi-valued*

[1] Abilis is a Web interface (try it at `http://ledenez.insa-rennes.fr/abilis/`) on top of Camelis (download it at `http://www.irisa.fr/LIS/ferre/camelis/`)

context K, the measures for a concept c that we have implemented in Abilis are the extension $ext(c)$ (noted ext), the count $\#ext(c)$ (noted count), the index over some axis $index_x(ext(c))$ (noted x), and the aggregated index over some axis $\gamma(index_x(ext(c)))$ (noted $\gamma(x)$).

For example, the concept 1456 has the following representations.

measure	result
ext	$\{1, 4, 5, 6\}$
$count$	4
$person/individual$	$\{Alice \mapsto 4, Bob \mapsto 1, Charlie \mapsto 1\}$
$date/year$	$\{2010 \mapsto 1, 2011 \mapsto 3\}$
$size/million$	$\{1M \mapsto 2, 3M \mapsto 1\}$
$sum(size/million)$	$5M$

In Abilis, an extension is represented as a list of objects, which can be displayed only in part if too long. An index over some axis is a multiset of values, and can therefore be represented as a tag cloud, where the font size renders the multiplicity of values. An aggregated index is generally a numerical value, but it could be anything. For example, in a domain where values are geometrical shapes, an aggregated value can be a geometrical shape (e.g., union, centroid, buffer area). If those geometrical shapes are geo-located, they can be rendered on a map.

The definition of a cube of concepts can be refined as a *cube of concept measures*, which is defined by a tuple of measures in addition to the query and dimensions.

Definition 10 (cube of concept measures). *Given a query $q \subseteq F$, a tuple $\overline{d}$ of n axes as dimensions, and a tuple of p measures $\overline{m}$, the* cube of concept measures *is defined as*

$$Cube(q, \overline{d}, \overline{m}) = \{\overline{v} \mapsto (m_j(c))_{j \in 1..p} \mid \overline{v} \mapsto c \in Cube(q, \overline{d})\}.$$

After describing the possible representations of individual concepts, we need to describe the possible representations of cubes of concepts. In other words, how to represent the multi-dimensional structure of a cube in rich and flexible ways. Abilis provides the following structures:

arrays Arrays can be used for all dimensions. They use values as row/column labels, and their cells can contains arbitrary representations of dimensions and measures. There are three kinds of arrays: horizontal arrays, vertical arrays, and two-dimensional arrays (spreadsheets). The later represents two dimensions at a time.

bar charts Bar charts can be used to represent the last dimension when the measure is an aggregated numerical value. There are horizontal and vertical bar charts.

pie charts Pie charts can be used in the same conditions as bar charts.

maps Maps can be used to represent a geographical dimension.

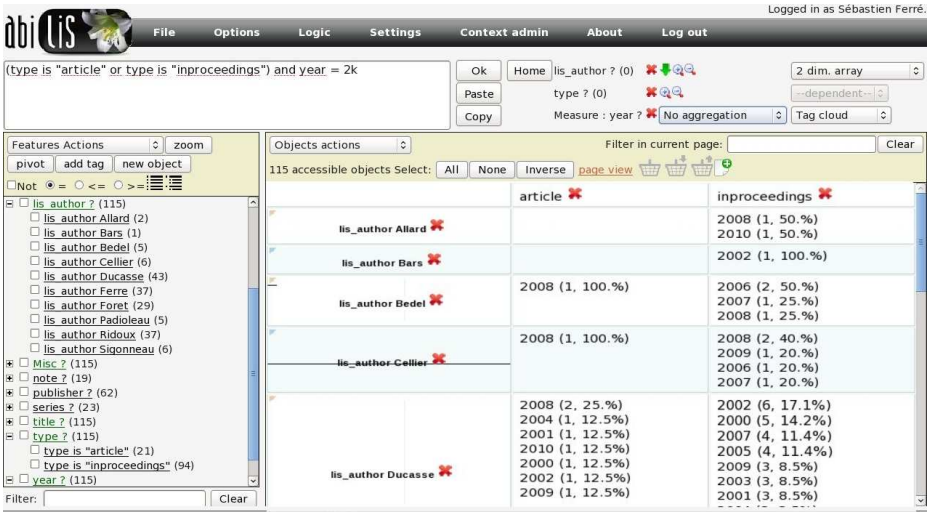

Fig. 1. Screenshot of Abilis showing the distribution over years of publications in journals and conferences since 2000 by author and by type of publication.

Figure 1 is a screenshot of our prototype Abilis. It displays a cube of concept measures showing the distribution over years of publications in journals and conferences since 2000 by author and by type of publication. The query is at the top left, the feature index is at the bottom left, the selected dimensions and measure are at the top right along with representation choices, and the cube itself, here a two-dimensional array of tag clouds, is at the bottom right. There are many possible ways to navigate to this view from the initial cube $Cube(\emptyset, (), ext)$, because the definition of the query, dimensions, and measures can be interleaved arbitrarily. Here is a possible scenario. Initially, the list of all 208 publications of the context is displayed in a zero-dimensional cube. Then, this list is grouped by LIS team author by choosing the attribute `lis_author` as a dimension. This results in a vertical array of lists of publications indexed by author. Note that a same publication may appear in several lists (in several cells of the array) because a publication may have several authors (`lis_author` is multi-valued). Then, the results are restricted to journal and conference papers by selecting the two features `type is "article"` and `type is "inproceedings"`. Note that disjunction (and negation) is available in Abilis, like in other LIS systems. Then, by choosing the attribute `year` as a measure, lists of papers are replaced by tag clouds of years in each cell. From there, it would be possible to apply an aggregator such as average, minimum or maximum. Those tag clouds can be restricted to years since 2000 by adding the feature `year = 2k` to the query. Finally, in order to get a finer analysis of the distribution of years of publication, the attribute `type` is selected as an additional dimension, which results in a two-dimensional array of tag clouds. Note how the attribute `type` is used both as a dimension and in the query, and how the attribute `year` is used both as a measure and in the query.

6 Comparison with OLAP

OLAP (On-Line Analytical Processing) [5,20] is an approach to the multidimensional exploration of data, and it is part of the domain of business intelligence. OLAP does not add any expressiveness compared to relational databases, and in fact is less expressive, but it makes it much easier and quicker to perform aggregative queries than with SQL. The principle is to let users navigate from view to view, and in this respect, it follows the same goals as conceptual navigation. In this section, we compare our approach to OLAP, and we show that it covers all OLAP representations and operations, and that it allows for more flexibility and expressiveness than OLAP. However, it should be noted that there is a trade-off between expressiveness and efficiency, and that some of the restrictions seen in OLAP are useful to accelerate some computations.

In OLAP, a same structure, called an *OLAP cube*, is used both for representing data and for representing views. An OLAP cube is a multidimensional database that is defined by n dimensions (e.g., date and place), p measures (e.g., sales), and a mapping from tuples of n values (e.g., December 2011 and Rennes) to aggregated values for the measure (e.g., total sales). Therefore, an OLAP cube is equivalent to a cube of concept measures, where measures are all aggregated indexes. In our approach, data is represented as a multi-valued context, from which many different cubes can be defined by varying dimensions and measures. Moreover, elements of the cubes need not be aggregated values, but can also be sets of objects (extension), and multisets of values (indexes). In fact, the objects from which an OLAP cube may have been defined have been lost because they have been aggregated in the preprocessing stage. On the contrary, multi-valued contexts are object-centered.

The object-centered approach allows for more flexibility and heterogeneity in data, such as multi-valued attributes or missing values. Suppose we have the cube of the sum of the size of photos, by person: $Cube(\emptyset, person/individual, sum(size/million)) = \{Alice \mapsto 5M, Bob \mapsto 6M, Charlie \mapsto 2M\}$. This is a valid OLAP cube. Now, suppose we want to aggregate those sizes to get the total size of photos. The result in OLAP would be $13M$, whereas the correct result, as given by our approach, is $9M$. This is because a photo can have several persons, and this is why OLAP assumes a strict partition between the values of a dimension.

OLAP defines a number of operators on cubes that play the same role as our cube transformations. We translate each of those operators in our approach:

Slice The selection of a sub-cube of dimension $n-1$ by fixing the value of some dimension. In our approach, this is equivalent to adding a feature to the query, and removing a dimension. A difference is that any feature can be selected, whether it belongs to a dimension or a measure or none.

Add dimension The addition of a dimension. In OLAP, this is restricted to predefined dimensions, while in our approach, this can be any attribute.

Remove dimension The removal of a dimension. In OLAP, this necessarily entails an aggregation.

Pivot The swap of two dimensions. This is a particular case of permutation in our approach.

Drill-down The change of a dimension level for a finer granularity level (e.g., from years to months or weeks). In our approach, this is equivalent to removing a dimension axis a/λ, and adding the axis a/λ', where $\lambda' \prec \lambda$. Abilis provides a direct navigation link for drill-down.

Roll-up The converse of drill-down, i.e. the change of a dimension level for a coarser granularity level (e.g., from months to years).

This demonstrates the flexiblity and expressiveness of our approach. All attributes can be used in the query, dimensions, and measures, and a same attribute can play several roles at the same time. An attribute can even be used twice as a dimension, which makes sense when the attribute is multi-valued. For example, the cube $Cube(\emptyset, (person/individual, person/individual), count)$ displays the number of photos for each couple of persons (and for each person on the diagonal of the array).

Finally, our approach allows for drill-down and roll-up on the measures that are based on an axis, because an axis is parameterized by a level. This is particularly useful when the measure is an index, i.e. a multiset of values. For instance, we can display for each person, the multiset of the dates of their photos, and those dates can be displayed at the level of years or months or weeks or days. This has proved useful for discovering functional dependencies and association rules in multi-valued contexts [1].

7 Related Work

We compare our approach to other approaches combining FCA and OLAP. Stumme [19] describes *conceptual OLAP* for conceptual information systems. His approach is very similar to OLAP, except for the definition of hierarchies of dimension values, and therefore has the same limitations as OLAP (see Section 6). A many-valued context defines the multi-dimensional space. Each attribute defines a dimension, whose hierarchy of values is the conceptual scale derived from a scale context. Those conceptual scales have no defined levels, apart from the default topological levels as in Definition 3. The measures, called variables, are defined out of the context, as functions from objects of the many-valued context to values. Therefore, dimensions and measures are strongly separated, and only aggregated measures are available. The only structures for representing cubes are the line-diagrams of the conceptual scales, which have to be designed in advance for better presentation. Each line-diagram represents one dimension, and nested line-diagrams are used to represent multi-dimensional cubes. Measure values appear as labels of the concepts of the innermost line-diagrams. Nested line-diagrams are an alternative to nested arrays. They need more space but they better expose complex hierarchies of values.

Penkova and Korobko [17] apply standard FCA on cube schemas, instead of on cubes themselves. They start from a formal context where objects are measures, attributes are dimensions, and the incidence relation is the compatibility

relation between measures and dimensions. For instance, in a dataset about the activities of a scientific organization, the dimension "journal name" is compatible with the measure "number of published paper" but not with "number of established conferences". A formal concept is a *maximal* cube schema: no dimension can be added without removing measures. The concept lattice can be used to guide users on the addition of dimensions (moving downward) or measures (moving upward). This approach is interesting when different kinds of objects are mixed (e.g., established conferences and published papers), and some dimensions only apply to some kinds of objects. In our approach, the feature index offers the same benefits by showing for each attribute whether it applies to all objects in the current cube, or only to a subset. And this comes without the usual limitations with OLAP: assymetry between dimensions and measures, only aggregated values as measures, etc. Applying our approach to the above examples, objects would be the individual published papers and established conferences, and their number would be obtained by choosing *count* as a measure. Other measures would allow to visualize the objects themselves, or the distribution of authors for papers.

8 Conclusion

The contribution of this paper is to extend conceptual navigation from single-concept views to cubes-of-concepts views. This means that, at each navigation step, the view is not limited to a single concept, but to a set of concepts, organized into a multi-dimensional cube. The single-concept view is a special case of a cube of concepts, having dimension 0. An important difference with OLAP is that each cube cell is a concept, which can be represented by a number of measures: a set of objects (the extension), a multiset of values (an attribute index), or an aggregated value. Finally, the cube need not cover the whole dataset, but can focus on a given subset, which is defined by a query.

Our approach generalizes OLAP cubes and navigation between cubes by relaxing a number of constraints. Cubes are derived from an object-centered multi-valued context, where no distinction is made between dimensions and measures, and where objects are not aggregated *a priori*. Objects can have several values for a same attribute. A same attribute can be used in a query, as a dimension, and as a measure at the same time. Drill-down and roll-up equally apply to dimensions and measures.

Our approach retains some constraints from OLAP relative to relational databases in terms of expressiveness. In particular, in a multi-valued context, the entities of a relational database must clearly be separated between objects and values. In another work, we have extended conceptual navigation and LIS to relational data from the Semantic Web [8]. Our goal is now to join the two extensions into one: multi-dimensional *and* relational conceptual navigation.

References

1. Allard, P., Ferré, S., Ridoux, O.: Discovering functional dependencies and association rules by navigating in a lattice of OLAP views. In: Kryszkiewicz, M., Obiedkov, S. (eds.) Concept Lattices and Their Applications. pp. 199–210. CEUR-WS (2010)
2. Antoniou, G., van Harmelen, F.: A Semantic Web Primer. MIT Press (2004)
3. Carpineto, C., Romano, G.: A lattice conceptual clustering system and its application to browsing retrieval. Machine Learning 24(2), 95–122 (1996)
4. Codd, E.F.: A relational model of data for large shared data banks. Communications of the ACM 13(6), 377–387 (Jun 1970)
5. Codd, E., Codd, S., Salley, C.: Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate. Codd & Date, Inc, San Jose (1993)
6. Ducrou, J., Eklund, P.: An intelligent user interface for browsing and search MPEG-7 images using concept lattices. Int. J. Foundations of Computer Science, World Scientific 19(2), 359–381 (2008)
7. Ferré, S.: Camelis: a logical information system to organize and browse a collection of documents. Int. J. General Systems 38(4) (2009)
8. Ferré, S.: Conceptual navigation in RDF graphs with SPARQL-like queries. In: Kwuida, L., Sertkaya, B. (eds.) Int. Conf. Formal Concept Analysis. pp. 193–208. LNCS 5986, Springer (2010)
9. Ferré, S., Ridoux, O.: A logical generalization of formal concept analysis. In: Mineau, G., Ganter, B. (eds.) Int. Conf. Conceptual Structures. pp. 371–384. LNCS 1867, Springer (2000)
10. Ferré, S., Ridoux, O.: An introduction to logical information systems. Information Processing & Management 40(3), 383–419 (2004)
11. Ganter, B., Kuznetsov, S.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) Int. Conf. Conceptual Structures. pp. 129–142. LNCS 2120, Springer (2001)
12. Ganter, B., Wille, R.: Conceptual scaling. In: Roberts, F. (ed.) Applications of combinatorics and graph theory to the biological and social sciences, pp. 139–167. Springer-Verlag (1989)
13. Ganter, B., Wille, R.: Formal Concept Analysis — Mathematical Foundations. Springer (1999)
14. Godin, R., Missaoui, R., April, A.: Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods. International Journal of Man-Machine Studies 38(5), 747–767 (1993)
15. Lindig, C.: Concept-based component retrieval. In: IJCAI Work. Formal Approaches to the Reuse of Plans, Proofs, and Programs. Morgan Kaufmann (1995)
16. Marchionini, G.: Exploratory search: from finding to understanding. Communications of the ACM 49(4), 41–46 (2006)
17. Penkova, T., Korobko, A.: Constructing the integral OLAP-model based on formal concept analysis. In: MIPRO, International Convention. pp. 1544 –1548. IEEE (2011)
18. Sacco, G.M., Tzitzikas, Y. (eds.): Dynamic taxonomies and faceted search. The information retrieval series, Springer (2009)
19. Stumme, G.: Conceptual on-line analytical processing. In: Tanaka, K., Ghandeharizadeh, S., Kambayashi, Y. (eds.) Information Organization and Databases, The Kluwer International Series in Engineering and Computer Science, vol. 579, pp. 191–203. Springer US (2001)
20. Vassiliadis, P., Sellis, T.K.: A survey of logical models for OLAP databases. SIGMOD Record 28(4), 64–69 (1999)