What can FCA do for database linkkey extraction? (problem paper)

Manuel Atencia[1,2], Jérôme David[1,2], and Jérôme Euzenat[2,1]

[1] Univ. Grenoble Alpes & [2] INRIA,
Grenoble, France
http://exmo.inria.fr

Abstract. Links between heterogeneous data sets may be found by using a generalisation of keys in databases, called linkkeys, which apply across data sets. This paper considers the question of characterising such keys in terms of formal concept analysis. This question is natural because the space of candidate keys is an ordered structure obtained by reduction of the space of keys and that of data set partitions. Classical techniques for generating functional dependencies in formal concept analysis indeed apply for finding candidate keys. They can be adapted in order to find database candidate linkkeys. The question of their extensibility to the RDF context would be worth investigating.

We aim at finding correspondences between properties of two RDF datasets which allows for identifying items denoting the same individuals. This is particularly useful when dealing with linked data [8] for finding equality links between data sets.

Because the RDF setting raises many additional problems, we restrict ourselves here to databases. The problem is illustrated by the two (small) book relations of Table 1 (from [5], p.116). We would like to characterise a way to identify items on the same line while not (wrongly) identifying any other pair of items.

bookstore relation					library relation				
id	firstname	title	lastname	lang	year	author	orig	translator	wid
id	fn	tt	ln	lg	y	a	o	t	w
					1845	Poe	Raven	Baudelaire	a1
					1845	Poe	Raven	Mallarmé	a2
3	E.	Gold bug	Poe	en	1843	Poe	Gold Bug	Baudelaire	b
4	T.	On murder	Quincey	en	1827	Quincey	On murder	Schwob	c
5	T.	Kant	Quincey	en	1827	Quincey	Kant	Schwob	d
6	T.	Confessions	Quincey	en	1822	Quincey	Confessions	Musset	e
7	J.-J.	Confessions	Rousseau	fr					
8	T.	Confessions	Aquinus	fr					

Table 1. Two relations with, on the same lines, those tuples that represents the same individual (the line after attribute names are abbreviations).

For that purpose, we have defined linkkeys [5, 2] and we would like to formulate the linkkey extraction problem in the framework of formal concept analysis [6].

We first present this problem in the context of database candidate key extraction where one looks for sets of attributes and the sets of equality statements that they generate. We formulate this problem as the computation of a concept lattice. Then we turn to an adaptation of linkkeys to databases and show that the previous technique cannot be used for extracting the expected linkkeys. Instead we propose an adaptation.

1 Candidate keys in databases

A relation $\mathcal{D} = \langle \mathcal{A}, \mathcal{T} \rangle$ is a set of tuples $\mathcal{T}$ characterised by a set of attributes $\mathcal{A}$. A key is a subset of the attributes whose values identify a unique tuple.

Definition (key) Given a database relation $\mathcal{D} = \langle \mathcal{A}, \mathcal{T} \rangle$, a key is a subset of the attributes $K \subseteq \mathcal{A}$, such that $\forall t, t' \in \mathcal{T}, (\forall p \in K, p(t) = p(t')) \Rightarrow t \approx t'$.

Classically, keys are defined from functional dependencies. A set of attributes A is functionally dependent from another K, if equality of the attributes of K determines equality for the attributes of A. If the equality between tuples is the same thing as the equality for all attribute values, then a key is simply those sets of attributes of which $\mathcal{A}$ is functionally dependent.

However, we have not used the equality between tuple $(=)$ but a particular $\approx$ relation. The reason is that we do not want to find keys for the database with $=$, but with an unknown relation $\approx$ which is to be discovered.

The statements $t \approx t'$ are those equality statements that are generated by the key. The $\approx$ relation must contain $=$ $(t = t' \Rightarrow t \approx t')$ and be an equivalence relation (this is by definition if it is the smallest relation satisfying the key).

From a key K of a relation $\langle \mathcal{A}, \mathcal{T} \rangle$, it is easy to obtain these statements through the function $\gamma : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{T} \times \mathcal{T}}$ such that $\gamma(K) = \{t \approx t' | \forall p \in K, p(t) = p(t')\}$. γ is anti-monotonic $(\forall K, K' \subseteq \mathcal{A}, K \subseteq K' \Rightarrow \gamma(K) \supseteq \gamma(K'))$.

We define *candidate key extraction* as the task of finding the minimal sets of attributes which generate a partition of the set of tuples.

Definition (candidate key) Given a database relation $\mathcal{D}$, a candidate key is a key such that none of its proper subsets generate the same partition. $\kappa(\mathcal{D})$ is the set of candidate keys.

Those candidate keys which generate the $singletons(\mathcal{T})$ partition are called normal candidate keys and their set noted $\hat{\kappa}(\mathcal{D}) = \{K \in \kappa(\mathcal{D}) | \forall (t \approx t') \in \gamma(K), t = t'\}$.

The problem of candidate key extraction is formulated in the following way:

Problem: Given a database relation $\mathcal{D}$, find $\kappa(\mathcal{D})$.

This problem is usually not considered in databases. Either keys are given and used for finding equivalent tuples and reducing the table, or the table is assumed without redundancy and keys are extracted. In this latter case, the problem is the extraction of normal candidate keys.

Using lattices is common place for extracting functional dependencies [9, 4] and the link to extract functional dependencies with formal concept analysis has already been considered [6] and further refined [10, 3].

In fact, this link can be fully exploited for extracting candidate keys instead of finding functional dependencies.

It consists of defining[1] a formal context $enc(\langle \mathcal{A}, \mathcal{T} \rangle) = \langle \mathfrak{P}_2(\mathcal{T}), \mathcal{A}, I \rangle$ such that: $\forall p \in \mathcal{A}, \forall \langle t, t' \rangle \in \mathfrak{P}_2(\mathcal{T})$,

$$\langle t, t' \rangle Ip \text{ iff } p(t) = p(t')$$

The (formal) concepts of this encoding, that we denote by the set $FCA(enc(\mathcal{D}))$, associate a set of attributes to a set of pairs of tuples. These pairs of tuples are tuples that cannot be distinguished by the values of the attributes, i.e., our $\approx$ assertions. The candidate keys are the minimal elements of the intent which generate exactly the corresponding partition[2]. $\kappa(\mathcal{D}) = \bigcup_{c \in FCA(enc(\mathcal{D}))} \mu_{\subseteq}\{K \subseteq intent(c) | \gamma(K) = \gamma(intent(c))\}$.

For any key $K \in \kappa(\mathcal{D})$, $\gamma(c)$ is the reflexive, transitive and symmetric closure of the extent of its concept.

If this method is applied to the data sources of Table 1, the result is displayed in Figure 1.

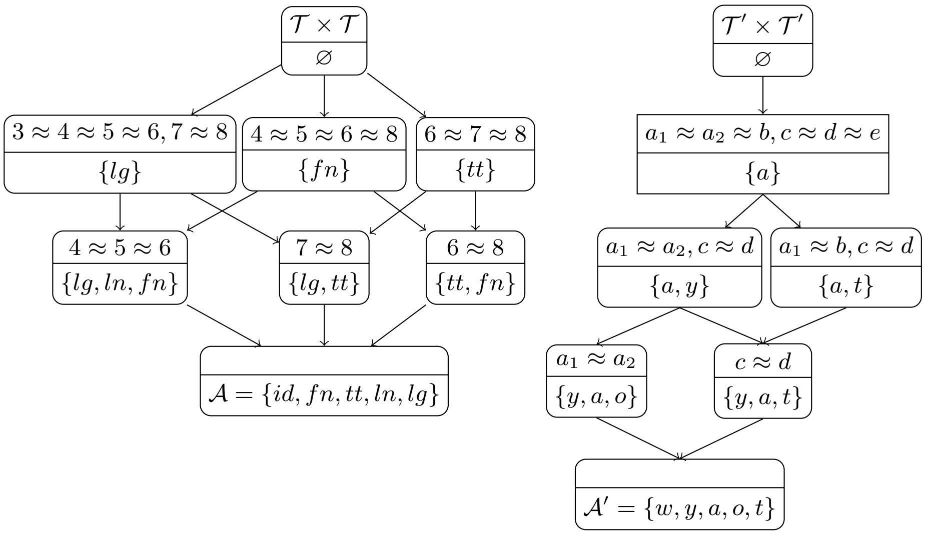
$\mathcal{T} \times \mathcal{T}$
$\varnothing$
$3 \approx 4 \approx 5 \approx 6, 7 \approx 8$
$\{lg\}$
$4 \approx 5 \approx 6 \approx 8$
$\{fn\}$
$6 \approx 7 \approx 8$
$\{tt\}$
$4 \approx 5 \approx 6$
$\{lg, ln, fn\}$
$7 \approx 8$
$\{lg, tt\}$
$6 \approx 8$
$\{tt, fn\}$
$\mathcal{A} = \{id, fn, tt, ln, lg\}$
$\mathcal{T}' \times \mathcal{T}'$
$\varnothing$
$a_1 \approx a_2 \approx b, c \approx d \approx e$
$\{a\}$
$a_1 \approx a_2, c \approx d$
$\{a, y\}$
$a_1 \approx b, c \approx d$
$\{a, t\}$
$a_1 \approx a_2$
$\{y, a, o\}$
$c \approx d$
$\{y, a, t\}$
$\mathcal{A}' = \{w, y, a, o, t\}$

Fig. 1. Example of the key concept lattice for the bookstore (left) and library (right) relations.

As presented in Table 2, this may generate several candidate keys for the same concept ($\{o, t\}$ and $\{w\}$ for the maximal partition in the library dataset; in the bookstore dataset, the concept of extent $4 \approx 5 \approx 6$ has two candidate keys $\{ln\}$ and $\{lg, fn\}$ and the maximal partition has three candidate keys $\{id\}$, $\{tt, ln\}$ and $\{tt, fn, lg\}$).

This answers positively to our first question: it is possible to extract keys, i.e., generating $\kappa(\mathcal{D})$ from data with some help from formal concept analysis.

[1] For an arbitraty total strict order $<$ on $\mathcal{T}$, $\mathfrak{P}_2(\mathcal{T}) = \{\langle t, t' \rangle \in \mathcal{T}^2 \mid t < t'\}$.

[2] $\mu_R E = \{X \in E | \forall X' \in E, \neg(X' R X))\}$.

intent	potential keys	candidate keys	extent
$\{id, fn, tt, ln, lg\}$	id. . . ,	$\{id\}$, $\{tt, ln\}$, $\{tt, fn, lg\}$	$\varnothing$
$\{fn, ln, lg\}$	fnlnlg, fnlg, fnln, lgln, ln	$\{ln\}$, $\{fn, lg\}$	$\{4 \approx 5 \approx 6\}$
$\{tt, lg\}$	ttlg	$\{tt, lg\}$	$\{7 \approx 8\}$
$\{fn, tt\}$	fntt	$\{fn, tt\}$	$\{6 \approx 8\}$
$\{lg\}$	lg	$\{lg\}$	$\{3 \approx 4 \approx 5 \approx 6, 7 \approx 8\}$
$\{fn\}$	fn	$\{fn\}$	$\{4 \approx 5 \approx 6 \approx 8\}$
$\{tt\}$	tt	$\{tt\}$	$\{6 \approx 7 \approx 8\}$
$\varnothing$	$\varnothing$	$\varnothing$	$\mathcal{T} \times \mathcal{T}$
$\{w, y, a, t, o\}$	w, . . . ot, yot, aot, yaot, wyaot	$\{o, t\}$, $\{w\}$	$\varnothing$
$\{y, a, o\}$	o, ao, at, yo, yao	$\{o\}$	$\{a_1 \approx a_2\}$
$\{y, a, t\}$	yt, yat	$\{y, t\}$	$\{c \approx d\}$
$\{y, a\}$	y, ya	$\{y\}$	$\{a_1 \approx a_2, c \approx d\}$
$\{a, t\}$	t	$\{t\}$	$\{a_1 \approx b, c \approx d\}$
$\{a\}$	a	$\{a\}$	$\{a_1 \approx a_2 \approx b, c \approx d \approx e\}$
$\varnothing$	$\varnothing$	$\varnothing$	$\mathcal{T}' \times \mathcal{T}'$

Table 2. The list of concepts extracted from the bookstore (top) and library (bottom) relations. All intent should be completed by their subsets.

2 Database linkkey extraction

Consider that, instead of one relation, we are faced with two relations from two different databases which may contain tuples corresponding to the same individual.

We assume that candidate attribute pairs are already available through an alignment A which expresses equivalences between attributes of both relations. In this example, $A = \{\langle \textsf{lastname}, \textsf{author}\rangle, \langle \textsf{title}, \textsf{orig}\rangle, \langle \textsf{id}, \textsf{wid}\rangle\}$. Our goal is to find those which will identify the same individuals (tuples) in both databases.

2.1 Linkkeys for databases

Linkkeys [5] have been introduced for generating equality, a.k.a. sameAs, links between RDF datasets. We present a simplified notion of linkkey which is defined over relations. **Definition (Linkkey)** Given two relations $\mathcal{D} = \langle \mathcal{A}, \mathcal{T}\rangle$ and $\mathcal{D}' = \langle \mathcal{A}', \mathcal{T}'\rangle$ and an alignment $A \subseteq \mathcal{A} \times \mathcal{A}'$. $LK \subseteq A$ is a linkkey between $\mathcal{D}$ and $\mathcal{D}'$ iff $\forall t, t' \in \mathcal{T}, \mathcal{T}'$; $(\forall \langle p, p'\rangle \in LK, p(t) = p(t')) \Rightarrow t \approx t'$. The set of linkkeys between $\mathcal{D}$ and $\mathcal{D}'$ with respect to A is denoted $\kappa_A(\mathcal{D}, \mathcal{D}')$.

This definition may be rendered independent from A by assuming $A = \mathcal{A} \times \mathcal{A}'$, so any attribute of one relation may be matched to any other.

2.2 Strong linkkey extraction

One way to deal with this problem is to start with keys: either candidate keys or normal candidate keys. For that purpose, we define $\kappa(\mathcal{D})/A$ as the operation which replaces, in all candidate keys of $\mathcal{D}$, each occurrence of an attribute in a correspondence of A by this correspondence[3].

A first kind of linkkeys that may be extracted are those which are normal candidate keys in their respective relations. They are called strong linkkeys and may be obtained by selecting normal candidate keys that contain only attributes mentioned in the alignment (replacing the attribute by the correspondence) and to intersect them, i.e., $\hat{\kappa}(\mathcal{D})/A \cap \hat{\kappa}(\mathcal{D}'))/A$. Strong linkkeys have the advantage of identifying tuples matching across relations without generating any links within the initial relations.

In the example of Table 1, there is one such strong linkkey: $\{\langle \mathsf{id}, \mathsf{wid}\rangle\}$. Indeed, the normal candidate keys for the bookstore relation are $\{\mathsf{id}\}$, $\{\mathsf{title}, \mathsf{lastname}\}$, or $\{\mathsf{title}, \mathsf{firstname}, \mathsf{lang}\}$ and, for the library relation they are $\{\mathsf{wid}\}$ and $\{\mathsf{orig}, \mathsf{translator}\}$. Since, translator has no equivalent in the bookstore relation (through A), only $\{\langle \mathsf{id}, \mathsf{wid}\rangle\}$ can be used. Unfortunately, it does not identify any equality statement as this happens very often with databases surrogates (this may have been worse if both relations used integers as identifiers: identifying false positives).

This scheme may be relaxed by trying to extract linkkeys from all candidate keys. In this way one would simply use $\kappa(\mathcal{D})/A \cap \kappa(\mathcal{D}')/A$. In our example, this does not bring further linkkeys.

2.3 Candidate linkkey extraction

The technique proposed above, does indeed generate linkkeys, but does not generate all of them: linkkeys may rely on sets of attributes which are not candidate keys. Indeed, one interesting linkkey for the relations above is $\{\langle \mathsf{lastname}, \mathsf{author}\rangle, \langle \mathsf{title}, \mathsf{orig}\rangle\}$.

Surprisingly, it does not use a normal candidate key of the library relation and not even a candidate key of the bookstore relation as $\{\mathsf{author}, \mathsf{orig}\}$ generates the same links as $\{\mathsf{orig}\}$ in this relation. However, when applied to the elements of $\mathcal{T} \times \mathcal{T}'$ this linkkey generates non ambiguous links, i.e., links which do not entail new links within a relation (this would have been different if a tuple $\langle \mathsf{year} = 1822, \mathsf{author} = \mathsf{Quincey}, \mathsf{orig} = \mathsf{Confessions}, \mathsf{translator} = \mathsf{Baudelaire}\rangle$ were present in the library relation).

Such linkkeys may be found by the same type of technique as before. It consists of defining a formal context $enc(\langle \mathcal{A}, \mathcal{T}\rangle, \langle \mathcal{A}', \mathcal{T}'\rangle, A) = \langle \mathcal{T} \times \mathcal{T}', A, I\rangle$ such that: $\forall \langle p, p'\rangle \in A, \forall \langle t, t'\rangle \in \mathcal{T} \times \mathcal{T}'$,

$$\langle t, t'\rangle I \langle p, p'\rangle \text{ iff } p(t) = p'(t')$$

γ is redefined to deal with subsets of alignments and generate $\approx$ assertions on $\mathcal{T} \cup \mathcal{T}'$. But, in order for $\approx$ to remain an equivalence relation it will be necessary to close $\approx$ on $\mathcal{T} \cup \mathcal{T}'$ and not only on $\mathcal{T} \times \mathcal{T}'$. Indeed if two tuples of $\mathcal{T}$ are found equal to a tuple of $\mathcal{T}'$, then by transitivity, they should be equal as well.

Again, candidate linkkeys are the minimal elements of the intent which generate exactly the corresponding set of links. $\kappa_A(\mathcal{D}, \mathcal{D}') = \bigcup_{c \in FCA(enc(\mathcal{D}, \mathcal{D}', A))} \mu_{\subseteq}\{K \subseteq intent(c) | \gamma(K) = \gamma(intent(c))\}$.

[3] This assumes that the alignment is one-to-one. This assumption is necessary for this subsection of the paper only.

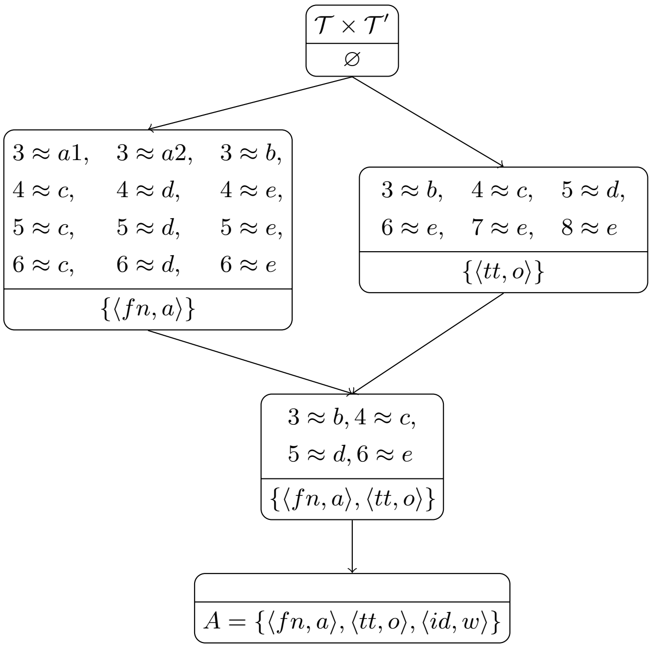
$\mathcal{T} \times \mathcal{T}'$
$\varnothing$
$3 \approx a1,\ 3 \approx a2,\ 3 \approx b,\ 4 \approx c,\ 4 \approx d,\ 4 \approx e,\ 5 \approx c,\ 5 \approx d,\ 5 \approx e,\ 6 \approx c,\ 6 \approx d,\ 6 \approx e$
$\{\langle fn, a\rangle\}$
$3 \approx b,\ 4 \approx c,\ 5 \approx d,\ 6 \approx e,\ 7 \approx e,\ 8 \approx e$
$\{\langle tt, o\rangle\}$
$3 \approx b, 4 \approx c, 5 \approx d, 6 \approx e$
$\{\langle fn, a\rangle, \langle tt, o\rangle\}$
$A = \{\langle fn, a\rangle, \langle tt, o\rangle, \langle id, w\rangle\}$

Fig. 2. Linkkey concept lattice for the relations of Table 1.

This technique, applied to the example of Table 1, generates the lattice of Figure 2. It can be argued that the candidate linkkeys $\{\langle fn, a\rangle, \langle tt, o\rangle\}$ and $\{\langle id, w\rangle\}$ are better than the others because they do not generate other statements within the relations. Indeed, $\{\langle tt, o\rangle\}$ generates $6 \approx 7 \approx 8$, and $\{\langle fn, a\rangle\}$ generates $a_1 \approx a_2 \approx b$, $c \approx d \approx e$ and $4 \approx 5 \approx 6$.

3 Conlusion and further work

We introduced, in the context of the relational model, the notions of candidate keys and linkkeys and we discussed potential links with formal concept analysis. These are only a few elements of a wider program. Problems were expressed in the relational framework because they are simpler. Our ambition is to provide an integrated way to generate links across RDF data sets using keys and it may be worth investigating if the proposed formal concept analysis framework can be extended to full RDF data interlinking.

Plunging this in the context of RDF requires further developments:

- considering that values do not have to be syntactically equal but may be found equal with respect to some theory: this may be a simple set of equality statement ("étudiant"="student") or may depend on RDF Schemas or OWL ontologies;
- considering several tables depending on each others together (this is related to Relational Concept Analysis [7] and could use the notion of foreign keys);

– considering that RDF attributes are not functional and hence yield a more general type of keys [1].

Once this is integrated within a common theoretical framework, a full solution requires work before and after running formal concept analysis:

– Before, it is necessary to use ontology/database matching [5] and to proceed to value normalisation.
– After, it is necessary to select among these potential or candidate keys those which are the more accurate [2].

Acknowledgements

This work has been partially supported by the ANR projects Qualinca (12-CORD-0012 for Manuel Atencia), and Lindicle (12-IS02-0002 for all three authors), and by grant TIN2011-28084 (for Manuel Atencia and Jérôme David) of the Ministry of Science and Innovation of Spain, co-funded by the European Regional Development Fund (ERDF).

Thanks to the anonymous reviewers for helping clarifying the paper.

References

1. Manuel Atencia, Michel Chein, Madalina Croitoru, Michel Chein, Jérôme David, Michel Leclère, Nathalie Pernelle, Fatiha Saïs, François Scharffe, and Danai Symeonidou. Defining key semantics for the RDF datasets: experiments and evaluations. In *Proc. 21st International Conference on Conceptual Structures (ICCS), Iasi (RO)*, pages 65–78, 2014.
2. Manuel Atencia, Jérôme David, and Jérôme Euzenat. Data interlinking through robust linkkey extraction. In *Proc. 21st european conference on artificial intelligence (ECAI), Praha (CZ)*, 2014.
3. Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli. Characterizing functional dependencies in formal concept analysis with pattern structures. *Annals of mathematics and artificial intelligence*, 2014. To appear.
4. János Demetrovics, Leonid Libkin, and Ilya Muchnik. Functional dependencies in relational databases: A lattice point of view. *Discrete Applied Mathematics*, 40(2):155–185, 1992.
5. Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer, Heidelberg (DE), 2nd edition, 2013.
6. Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer, Berlin, New York, Paris, 1999.
7. Mohamed Rouane Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. Relational concept analysis: mining concept lattices from multi-relational data. *Annals of Mathematics and Artificial Intelligence*, 67(1):81–108, 2013.
8. Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.
9. Mark Levene. A lattice view of functional dependencies in incomplete relations. *Acta cybernetica*, 12(2):181–207, 1995.
10. Stéphane Lopes, Jean-Marc Petit, and Lotfi Lakhal. Functional and approximate dependency mining: database and FCA points of view. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):93–114, 2002.