

Looking at the World Thru Colored Glasses

Floris Geerts¹, Anastasios Kementsietsidis², and Heiko Müller³

¹ ADReM Research Group, University of Antwerp, Antwerpen, Belgium
`floris.geerts@ua.ac.be`

² IBM Research - Thomas J. Watson Research Ctr, Hawthorne, NY, USA
`akement@us.ibm.com`

³ Intelligent Sensing and Systems Laboratory, CSIRO, Hobart, Australia
`heiko.mueller@csiro.au`

Abstract. There are two central issues in the curation of (scientific) databases: annotation management and archiving. Both issues have been addressed by the Edinburgh database group and led to the MONDRIAN annotation management system and the XARCH archiving system, respectively. In this paper, we present an application of MONDRIAN to represent and query the history of evolving databases. We show how the annotation model and query language underlying MONDRIAN not only allows to answer queries about how individual data values change over time, but also allows to capture and query structural changes that occur in a database over time, beyond the querying functionalities that XARCH currently offers.

1 Introduction

In recent years, one has seen a vast increase in the number of curated databases. This is particularly true for databases that originate from scientific research and from governmental agencies. Common examples include IUPHARDB, the official database of the IUPHAR Committee on Receptor Nomenclature and Drug Classification [2], and the CIA World Factbook, a comprehensive resource of demographic data [1], among others.

Annotations play an important role in database curation. Indeed, the curation process usually involves manual collection, verification, and aggregation of existing data sources by a dedicated group of curators. Annotations can, for example, represent opinions of curators about the quality of data or suggested changes, record information about the provenance of data, as well as indicate temporal information with regards to the validity of data. In general, annotations can be regarded as additions to the core data for which there is no dedicated place within the database schema. The importance of annotations has been recognized by several research efforts in which the problems of maintaining and querying annotated databases have been addressed [7,3,13,14,20,9,16].

Equally important in the curation process is the ability to store, manipulate and query different versions of the data. Indeed, it is common that curated data evolves and gets updated at a regular pace and proper archiving of the data is

required [4]. Furthermore, users want to query the history of the data, rather than simply querying a single old version of the data. A prime example of this is the CIA World Factbook [1] where queries like “How did the population of China change over the past 15 years?” are common¹. To deal with these challenges, different approaches and systems for archiving and querying data have been developed in recent years [6,19,8,18,21].

In this paper we marry two ideas, both developed when the authors were doing time in the database group in Edinburgh. The first idea concerns the modeling of annotated databases by means of colors, to represent the annotations, and blocks, to represent the data items to which the colors are associated. The resulting annotation management system, MONDRIAN, was reported in [13,12]. The second idea concerns the archiving of data by means of keys and XML. More specifically, in the XARCH system [19], different versions of an evolving database are merged into a single well-defined hierarchical data format. Furthermore, temporal information is stored as annotations of elements in the archive. The goal of this paper is to show that by modeling time-stamped XML as so-called color relations in the MONDRIAN system, and by using the corresponding color algebra as query language, we obtain a flexible mechanism to store, manipulate and query annotated historical data. Using the CIA World Factbook as an example, we show that this approach enables to answer queries over archives that XARCH currently cannot answer.

The remainder of this paper is structured as follows: We review the MONDRIAN annotation management system in Section 2. In Section 3, we describe the archiving system XARCH. Finally, in Section 4 we combine both systems to represent and query annotated evolving databases.

2 The MONDRIAN Annotation System

Most existing approaches for annotation management deal with annotations of individual values or records [7,3,13,14,20,9,16]. In many cases, however, it is of importance to be able to annotate sets of values. In data integration, for example, one wants to use annotations to provide evidence for the correctness of associations between values. Likewise, it is often important to group and annotate values that have a semantic or temporal relationship. We illustrate how the MONDRIAN annotation system models such complex annotations by means of the following example:

Example 1. Consider the relations in Fig. 1(a)-(c) taken from the CIA World Factbook of 2011. The relations list for each country (a) the highest-valued imported products; (b) the most important import trading partners; and (c) the highest-valued exported products. Figure 1(d) shows an integrated relation in which countries are associated with their trading partners and corresponding imported products. Annotations in this relation represent evidence for associations,

¹ This query is one of Peter Buneman’s favorite queries. Others include the shoe and hat size of individuals, which are outside the scope of this paper.

country	imports
Brazil	Chemical prod.
Brazil	Electronics
Chile	Chemicals
Chile	Electrical equip.
Chile	Vehicles

country	partner
Brazil	S. Korea
Brazil	US
Chile	Germany
Chile	S. Korea
Chile	US

country	exports
Germany	Chemicals
Germany	Motor vehicles
Korea, South	Motor vehicles
Korea, South	Semiconductors
United States	Organic chemicals

(a) Imported products (b) Import trading partners (c) Exported products

country	partner	product	
Brazil	South Korea	Electronics	
Brazil	US	Chemicals	Verified by Eric
Chile	Germany	Motor vehicles	Verified by Mary
Chile	Germany	Chemicals	
Chile	South Korea	Electronics	Until June 2011
Chile	South Korea	Motor vehicles	

\$1.5 billion

S. Korea = Korea, South = South Korea

(d) Integrated relation of imported products

Fig. 1. Three relations (a), (b) and (c) on imports and exports taken from the CIA World Factbook (2011) and their integrated relation (d)

assumptions that were made during the integration process, as well as temporal information. Annotations are shown in the form of color blocks. Here, a block is a set of values for which an annotation exists. Colors are used to represent annotations for this block. In the figure, we also show the semantics of each color. For example, the \dots -colored block in the fifth tuple is valid “Until June 2011”. In the first and the last two tuples one block indicates that the association between *Brazil* (and *Chile* resp.) and *South Korea* is based on the assumption that the names *S. Korea*, *Korea*, *South*, and *South Korea* all represent the same country. As another example, the block in the third tuple indicates that the association has been verified by curator Eric. Note that not all annotations are shown (e.g., US and United States are regarded the same as well). □

Complex annotations like those shown in the previous example pose interesting challenges in terms of how they can be implemented on top of existing database management systems (DBMS). In MONDRIAN, a simple albeit efficient relational representation of color databases (i.e., databases that contain color blocks) was proposed. In a nutshell, the relation schemas are first extended with so-called block attributes, one for each attribute in the original schema; and second, a single color attribute (*col*) is attached to each relation. If *A* is an attribute in the original schema, we denote by A^b its corresponding block attribute. Intuitively, if a tuple *t* has a block covering attribute *A*, then A^b will be set to 1. Otherwise, the A^b -attribute of the tuple *t* is set to 0. Similarly, if *t* has a block of a certain color, then this color is simply recorded in its *col*-attribute. More specifically, let *R* be a relation consisting attributes A_1, \dots, A_n . For any









country	partner	product	country ^b	partner ^b	product ^b	col
Brazil	South Korea	Electronics	1	1	0	
Brazil	US	Chemicals	0	1	1	
Chile	Germany	Motor vehicles	1	1	1	
Chile	Germany	Motor vehicles	1	1	0	
Chile	Germany	Chemicals	1	1	0	
Chile	South Korea	Electronics	0	1	1	
Chile	South Korea	Electronics	1	1	0	
Chile	South Korea	Motor vehicles	1	1	0	

Fig. 2. Relational encoding of the color relation shown in Fig. 1(d)

relation scheme R , we define the relation scheme $\bar{R} = R \cup \{A_1^b, \dots, A_n^b\} \cup \{col\}$. Note that \bar{R} is the schema of the relational representation of the color databases. More generally, to every color block database schema \mathcal{S} we can associate the relational database schema $\bar{\mathcal{S}}$ which has precisely the same relation variables, but when relation variable x has relation scheme R in \mathcal{S} , then x has relation scheme \bar{R} in $\bar{\mathcal{S}}$.

Example 2. Consider the color relation shown in Fig. 1(d). The corresponding relational representation is shown in Fig. 2. In this figure, the colors represent the annotations as given in Fig. 1(d). □

The advantage of the relational encoding used by MONDRIAN is that it requires a minimal restructuring of the existing schema. Indeed, although the representation shown in Fig. 2 requires the addition of new attributes (for blocks and colors), one can equivalently work with a representation in which separate tables for blocks and colors are present. These are then linked by means of tuple, block and color identifiers as is common practice in DBMSs. In addition, annotations of the database imposes minimum overhead in terms of space [13]. Several indexing methods, such as bitmaps [17] and multi-dimensional indexes [10], are in place to better support the presence of color blocks in databases.

In addition to supporting complex annotations, MONDRIAN provides a query interface in which color blocks are treated as first-class citizens. More specifically, a color algebra was introduced in [13] that allows the user to focus on the annotated data, without knowing how these annotations are modeled in the underlying relational database system. More specifically, the color algebra, or CA for short, consists of the following operators:

- **Projection** (π_X^{col}): which behaves like a standard projection on the data part and simply inherits all color blocks, restricted to the attributes in X ;
- **Lower and Upper Block Selection** (Π_X^L, Π_X^U): which simply select all tuples and their color blocks covering all attributes in X (lower), or cover only attributes in X (upper);
- **Color Selection** (Σ_γ): which selects all tuples that contain a color block of color γ ;
- **Selection** ($\sigma_{A=B}$): which behaves like the standard selection on the data part and simply copies all color blocks in the selected tuples;

Table 1. Simulation of CA by conjunctive queries (CQ). If X is a set of attributes, then X^b denotes its corresponding set of block attributes. Furthermore, in the case of the tuple join, $x \bowtie y$, the letter R (S) refers to the relation scheme of the color block relation x (y), f (resp. g) renames the common block attributes A_i^b in $R \cap S$ to attributes of the form C_i^b (resp. D_i^b), and finally, **or** is the truth table of disjunction in which the third attribute contains the result of the union of the bits in first two attributes. The latter is needed to combine all blocks in the joined tuples.

CA	\mapsto CQ
$\pi_X^{col}(x)$	$\mapsto \pi_{X \cup X^b \cup \{col\}}(x)$
$\prod_X^1(x)$	$\mapsto \sigma_{\bigwedge_{A \in X} A^b=1}(x)$
$\prod_X^0(x)$	$\mapsto \sigma_{\bigwedge_{A \notin X} A^b=0}(x)$
$\Sigma_\gamma(x)$	$\mapsto \sigma_{col=\gamma}(x)$
$\sigma_{A=B}(x)$	$\mapsto \sigma_{A=B}(x)$
$x \cup y$	$\mapsto x \cup y$
$\rho_{A/B}(x)$	$\mapsto \rho_{A/B}(\rho_{A^b/B^b}(x))$
$x \bowtie y$	$\mapsto \pi_{R \cup S \cup R^b \cup S^b \cup \{col\}}(\rho_f(x)) \bowtie \rho_g(\rho_{col'/col''}(y)) \bowtie_{i=1}^p \mathbf{or}(C_i^b, D_i^b, A_i^p)$ \cup Same expression but with the roles of x and y reversed.
$x \boxtimes y$	$\mapsto x \bowtie y$
$\rightleftarrows_B^A(x)$	$\mapsto \rho_{A^b/B^b}(x)$

- **Union** (\cup): which behaves like the standard union on the data part and simply copies color blocks in the tuples in the component relations;
- **Block Join** (\boxtimes): which joins tuples together that agree on the data part *and* share the same color block on common attributes;
- **Tuple Join** (\bowtie): which joins tuples together that agree on the data part, irregardless of any color blocks that may be present;
- **Renaming** ($\rho_{A/B}$): which simply renames attributes; and finally
- **Block Switch** (\rightleftarrows_B^A): which switches color blocks involving covering attribute A by blocks involving attribute B .

Note that none of these operators access the color blocks explicitly. Indeed, their semantics does not rely on the relational representation used to model color databases. However, to make the semantics more precise we provide a translation from the operators in CA to conjunctive queries (CQ) over the relational representation described earlier (Table 1).

Example 3. Consider again the color relation shown in Fig. 1(d). Suppose that we want to find all the tuples that have a block of color “*Verified by Mary*”, or concern the country *Brazil*. Also, assume that we are only interested in keeping the $\{country, partner\}$ attributes from these tuples. Then, the CA expression

$$e = \pi_{country, partner}^{col}((\Sigma \text{“Verified by Mary”}(r)) \cup (\sigma_{country=\text{“Brazil”}}(r)))$$

returns the desired result. As another example, asking for all the tuples that have an annotation of the attribute *product* can be simply expressed in CA by means of

country	partner	country ^b	partner ^b	col
Brazil	South Korea	1	1	
Brazil	US	0	1	
Chile	Germany	1	1	

country	partner
Brazil	South Korea
Brazil	US
Chile	Germany

→ Verified by Eric

→ Verified by Mary

S. Korea = Korea, South = South Korea

(a) Relational representation of result
(b) Color result relation

Fig. 3. Query result as relational representation (a); and color relation (b)

$\Pi_{product}^L(r)$. When interested in all tuples that involve annotations exactly covering *country*, *partner*, then the CA expressions $\Pi_{country,partner}^L(\Pi_{country,partner}^U(r))$ would give the desired result. We refer to [13] for more examples. \square

The MONDRIAN system leverages the translation given in Table 1 by translating the user’s CA query into a CQ query on the underlying relational representation. Once the result of the CQ query is obtained, it is converted again into a color relation that is given to the user. We refer to [12] for more details of the MONDRIAN system. It is noteworthy to point out that the evaluation of CA expressions, by means of the intermediate translation step to the underlying RDBMS, is comparable in terms of query execution time when compared to its unannotated version [13].

Example 4. Consider the CA expression e from the previous example. To evaluate e on the color relation shown in Fig. 1(d), MONDRIAN first translates e into a CQ query q_e as specified by Table 1. It is readily verified that q_e equals

$$\pi_{country,partner,country^b,partner^b,col}(\sigma_{col=\text{“Verified by Mary”}}(\bar{r}) \cup \sigma_{country=\text{“Brazil”}}(\bar{r})),$$

where \bar{r} denotes the schema r extended with block and color attributes. The CQ query q_e is then executed on the relational representation shown in Fig. 2. The result of the q_e and the corresponding color relation is shown in Fig. 3. \square

We conclude the description of the MONDRIAN system by stating that CA (on color databases) has the same expressive power as CQ (over relational representations of color databases), even though CA does not access color blocks explicitly. Observe that Table 1 implies that for every CA-expression over \mathcal{S} , there exists an equivalent CQ-expression over $\bar{\mathcal{S}}$. The converse is also true:

Theorem 1 ([13]). *For every conjunctive query over $\bar{\mathcal{S}}$ whose result relation scheme is of the form \bar{R} for some relation scheme R , there exists an equivalent CA expression over \mathcal{S} .*

As final remark, we note that in the setting that blocks cover all the attributes, and thus only the colors matter, Theorem 1 is shown to hold even in the presence of negation [11].

3 The XARCH Archiving System

Curated databases are predominantly kept in well-organized hierarchical data formats. These data formats are often equipped with a key structure that pro-

vides a canonical identification for each element in the hierarchical data. For instance, a key for an element can be taken as the combination of the path in which the element occurs together with the values of some of its sub-elements. Buneman et al. [6] developed an archiving approach that takes advantage of such keys to maintain multiple versions of an evolving curated databases. The archiving system XARCH implements an extended version of the original approach that allows archiving databases of arbitrary size [15]. More specifically, in XARCH, multiple snapshots of an evolving database are merged into a single archive. Corresponding elements in different snapshots are identified based on their key values and stored only once in the resulting archive. Each snapshot is then given a unique identifier and each element is annotated with a timestamp that represents the sequence of snapshots in which the particular element was contained in. Archives in XARCH are currently stored as XML documents. We next provide a high-level description of the archiving process by means of the following example and refer to [6,15] for more details.

Example 5. Consider the following two (XML) entries from two releases of the CIA Factbook in 1992 (left) and 1998 (right). These entries concern the number of airports in Kazakhstan:

<pre> <country> <name>Kazakhstan</name> <category> <name>Communications</name> <property> <name>Airports</name> <value>NA</value> </property> </category> </country> </pre>	<pre> <country> <name>Kazakhstan</name> <category> <name>Transportation</name> <property> <name>Air Transport</name> <subprop> <name>Airports</value> <value>10</value> </subprop> </property> </category> </country> </pre>
---	--

In order to compare, merge and archive these two entries, a key specification is required to tell what the corresponding elements are in these two releases. The keys relevant for this fragment of the CIA Factbook are as follows:

$$\begin{aligned}
 k_1 &= (/country, \{name\}) \\
 k_2 &= (/country/name, \{\}) \\
 k_3 &= (/country/category, \{name\}) \\
 k_4 &= (/country/category/name, \{\}) \\
 k_5 &= (/country/category/property, \{name\}) \\
 k_6 &= (/country/category/property/name, \{\}) \\
 k_7 &= (/country/category/property/value, \{\}) \\
 k_8 &= (/country/category/property/subprop, \{name\}) \\
 k_9 &= (/country/category/property/subprop/name, \{\}) \\
 k_{10} &= (/country/category/property/subprop/value, \{\})
 \end{aligned}$$

We provide the formal definition of keys below. The archiver will merge corresponding nodes in both entries, in a top-down manner, starting from the `country`

nodes. The key k_1 specifies that elements nodes of type `country` are uniquely specified by the value of their `name` child node. Clearly, in order for this to make sense `/country/name` nodes should carry a unique value and have no further nodes below them (they are frontier nodes). For the two entries given above, “Kazakhstan” is the key for both `country` nodes and thus both nodes correspond to the same country. As a consequence, they will be merged in the archive. The resulting `country` node has timestamps $\{1992, 1998\}$. The key k_2 specifies that nodes of type `name` are identified by their own value. Again, the archiver merges both `name` nodes and the corresponding timestamps are inherited from its parent `country` node. As another example, k_3 says that `category` nodes are again identified by the value of their `name` child node. In the archive, we thus have two distinct nodes for the entries above, one for “Communications” (with timestamp $\{1992\}$) and one for “Transportation” (with timestamp $\{1998\}$). Proceeding along this way, until all nodes are processed, we obtain a timestamped XML documents that contains both entries. In case that more versions are available, the archiver will sequentially add each version to the archive computed so far. Figure 4 shows a minor modification of part of our current archive of the CIA World Factbook that contains the annual releases of the Factbook from 1992 to 2002. We only show the information related to the number of airports in Kazakhstan over these years. In the figure, element nodes are shown in square boxes together with their label (in angle brackets) and their key value (in square brackets). Timestamps are shown as edge labels. Note that elements (or edges) without an explicit timestamp inherit the timestamp of their parent. The two entries from 1992 and 1998 are embedded in the archive and are highlighted by bold edges. As already mentioned, frontier nodes that are used as key values, e. g., `/country/name`, can have only one text node as their child since objects are identified based on this value. Thus, the key value for each country is the value of this text node. Other frontier nodes, e. g., `/country/category/property/value`, may have multiple text nodes as children, each with a different (disjoint) timestamp. Indeed, these nodes are not used as key for any node and may have multiple values. \square

More formally, an archive \mathcal{A} is a tree with two types of nodes: (i) element nodes, and (ii) text nodes. Only element nodes may occur as internal nodes. In addition, each element node has a label and a key value. Element keys are defined using key constraints. Here, we consider only a limited form of key constraints and refer to [5,6] for an extended definition and for further details. A key specification K is a set of key definitions $k = (p, q)$, where p is an absolute path of element labels and q is a set of element labels. The key definition specifies for each element e reachable by path p how the key value is derived from the subtree of e . If q is empty then $key(e) = \perp$. Otherwise, $key(e)$ is an array of values; one for each the children of e reachable by path p/ℓ , for $\ell \in q$. Elements whose values are used as key values are called *key path values*. Note that (i) there has to be exactly one child of e for each label $\ell \in q$, and (ii) this child is a frontier node, i. e., it does not have other element nodes as children. All element keys are relative keys, i. e., the key identifies an element among its siblings (with the same label).

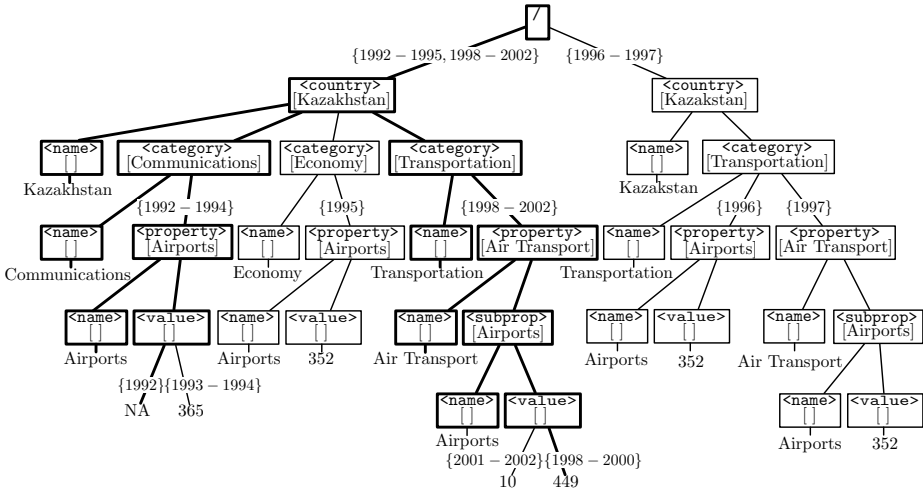


Fig. 4. Part of the archive of the CIA World Factbook (1992–2002)

Moreover, the concatenation of key values along the path from the root to an element e forms an absolute key for e . Let T denote the set of snapshot identifiers. Each node $n \in \mathcal{A}$ has a timestamp $time(n) \subseteq T$ that represents the set of snapshots that node was present in. Timestamps satisfy the following property: if a node n is a descendant of a node m in \mathcal{A} , then $time(n) \subseteq time(m)$. Since changes to databases are largely accretive and an element is likely to exist for a long time, we compactly represent its timestamp using time intervals rather than a sequence of version numbers. As illustrated in the previous example, the timestamps are assigned when merging different versions of the data.

Merging different snapshots into a single archive has several advantages: (i) any specific snapshot is retrievable from the archive in a single pass over the data, (ii) the storage space required is comparable to that of delta-based approaches that keep a sequence of records of changes between pairs of consecutive versions, (iii) tracking object history is easy. XARCH implements a query language (XAQL) for retrieval of individual database snapshots as well as queries over the history of data. The query language XAQL has a SQL-like syntax and it uses a restricted form of XPath expressions to filter output elements.

Example 6. The following XAQL query returns the name and population of European countries between years 2000 and 2008 in our current archive of the CIA World Factbook:

```
SELECT $c/name, $c/category[name = 'People']/property[name = 'Population']/value
FROM $c IN archive('CIAWFB')/country
VERSION 2000-2008
WHERE $c/category[name = 'Geography']/property[name = 'Map references']/value = 'Europe' □
```

XARCH has been successfully used to maintain the history of several curated databases. While the archiving approach generally works very well, it does make

the critical assumption that the structure of the archived database remains unchanged. This restriction, however, is almost certainly to become an issue when archiving over a long period of time.

Figure 4 shows a typical problem the archiver currently faces, that is, the change of (absolute) key values. Over the history of the Factbook, the information about the total number of airports in Kazakhstan (referred to as *Kazakstan* in 1996 and 1997) was first located under category *Communications*, then under category *Economy* and 1996 moved to category *Transportation*. In 1997, the Factbook started to group information about the airports, heliports, and airlines (not shown in Figure 4) as sub-properties under a new property *Air Transport*. Due to the change in its absolute key value the archiver maintains different elements at different levels of the tree about the number of airports in Kazakhstan. While this redundancy causes a slight storage overhead, a more severe problem occurs when querying the archive. Indeed, the same information can now be found under different paths in different snapshots.

As another example, more recently, the Factbook renamed category *People* into *People and Society*. Thus, the query shown in Example 6 would return an empty result for the more recent snapshots in our archive. Making matters worse, the population of China can now be found under different paths in the history of the Factbook. XAQL queries in XARCH are currently not able to handle such changes. That is, in order to retrieve to full history of the population of China one would have to issue two separate queries and manually merge the results. Moreover, XARCH currently does not provide any mechanism to maintain information about different key values for the same element. In the following, we show how MONDRIAN can be used to maintain and query historic data in the presence of key value changes.

4 Mondrianizing XARCH

We start by showing that an archive in XARCH is a special form of a color database where timestamps are represented as color blocks. That is, we can turn any archive into a color relation. Annotations in such a relation have temporal semantics, i. e., they represent the snapshots in which a data value was valid. Then, the color algebra can be used for temporal queries over the history of data. We can use additional annotations to capture the semantic relationships of different elements in an archive, and use this to better answer queries over the history of data. In comparison to XAQL this gives us (a) a formal query language, and (b) the ability to query data under key value changes.

Consider an archive \mathcal{A} following key specification K . For simplicity, assume that for all key definitions $(p, q) \in K$, q is either empty or contains exactly one label. When transforming \mathcal{A} into a color database we generate a single color relation with schema $\overline{R}^k = \{A_1, \dots, A_n\} \cup \{A_1^b, \dots, A_n^b\} \cup \{col\}$. The schema \overline{R}^k has exactly one attribute A_i for each key definition $(p, q) \in K$ where p does not identify a key path value. That is, if $p = p_1/\ell$ then $(p_1, \{\ell\})$ is not in K . We use Ψ to denote the mapping from path p in the key definition to the corresponding

attribute $A_i \in \overline{R^k}$. Figure 5(a) shows the mapping Ψ for the archive in Figure 4, while Figures 5(b) and (c) show the schema of the generated relation $\overline{R^k}$. For convenience, in our implementation we split $\overline{R^k}$ into two relations (as mentioned in Section 2), one relation for storing the archive raw data and another for storing just the annotations.

We now turn our attention on how to compute the instances shown in the figures and concentrate first on the raw data. Let r^k be in general the instance for schema R^k . We create one tuple in r^k for each text node n in \mathcal{A} that is not a child of a key path value. Let $value(n)$ denote the text value of n and let $\langle e_1, \dots, e_m \rangle$ denote the sequence of elements on the path to n . Furthermore, let $path(e)$ denote the path of element e . Then, for the text node n , we create a tuple t where for each $e \in \langle e_1, \dots, e_{m-1} \rangle$ we set $t[\Psi(path(e))] = key(e)$ and for e_m we set $t[\Psi(path(e_m))] = value(n)$. For attributes $A \in R^k$ where no element e with $\Psi(path(e)) = A$ exists in $\langle e_1, \dots, e_m \rangle$, we set $t[A] = \perp$. Figure 5(b) shows an instance r^k for the archive in Figure 4.

We next consider the creation of colors and blocks. For each of the tuples t in r^k and each snapshot identifier s in $time(e)$ with e the text element that warranted the creation of t in r^k , we create one tuple $\overline{r^k}$ with $t[col]$ representing the snapshot identifier s . This sets the color of the tuples. For the blocks, we set $t[A_i^b] = 1$, for $1 \leq i < m$ if $s \in time(e_i)$. Likewise, for A_m^b we use the timestamp of text node n to decide the value of $t[A_m^b]$. We illustrate the above construction by means of the following example.

Example 7. The instance $\overline{r^k}$ of $\overline{R^k}$ corresponding the CIA Factbook fragment shown in Figure 4 consists of the raw data instance shown in Figure 5(b) together with colored blocks represented by the first 11 tuples in Figure 5(c). For example, there are two blocks for the tuple with $tid=3$ of Figure 5(b), since the corresponding path is present in both 1994 and 1995. Similarly, there are three blocks for the tuple with $tid=6$ for the years 1998, 1999 and 2000. \square

In our example the timestamp of every internal node equals the union of timestamps of its children. This is not true in general, however. For example, in any snapshot of the CIA World Factbook a country may have a category without property elements or a property value without a text node child. Note that the key path values for a node, however, always have to exist, i.e., there are no missing values in element keys. To ensure that our transformation of an archive into a color relation is lossless we have to create a tuple in r^k for every element e whose timestamp contains snapshot identifiers that are not contained in the timestamp of any of e 's children. We set tuple values according to the sequence of elements on the path to e . That is, for every snapshot identifier $s \in time(e)$ that does not appear in any of the timestamps of the children of e we create one tuple $\overline{r^k}$ such that $t[col]$ represents s .

The advantage of modeling archives by means of color relations is that one can annotate the archived data, beyond the encoding of timestamps, at no additional cost.

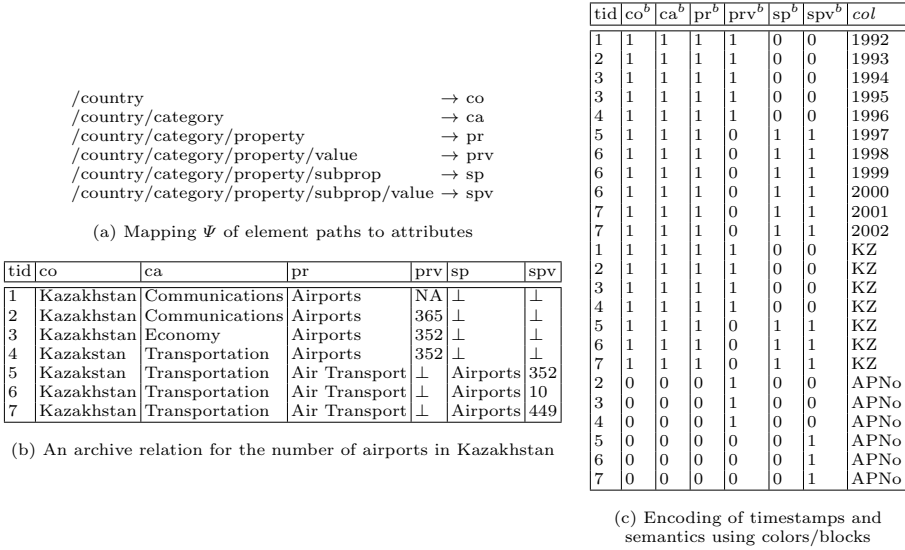


Fig. 5. A representation of archives in MONDRIAN

Example 8. For example, we already mentioned that there are (at least) two spellings, namely *Kazakstan* and *Kazakhstan*, for the same country. With colors and blocks, we alleviate such issues by introducing a new color *KZ* that corresponds to the ISO 3166 country code for this country. Then, we use this color to define 7 blocks that cover each of the tuples in Figure 5(b). Another *semantic* annotation can be used to denote that, in spite of the structural differences, all tuples in Figure 5(b) actually represent information about the number of airports in Kazakhstan. For that, we can use a new color *APNo* (for Airport Number) to create a block for each column (it can be a different column for each tuple) that is used to store the number of airports. The last six tuples in Figure 5(c) define these blocks. Notice that for the tuple with *tid*=1 in Figure 5(b) there is no block with color *APNo* in Figure 5(c) since the number of airports is not available (NA). □

We conclude this section by showing that the color algebra allows for the querying of the archived data beyond current capabilities of XAQL.

Example 9. We can use CA to query the annotated relation with timestamps. The following CA query retrieves the tuples between 1995 and 1997.

$$\Sigma_{\text{“1995”}}(r^k) \cup \Sigma_{\text{“1996”}}(r^k) \cup \Sigma_{\text{“1997”}}(r^k)$$

Notice that while the blocks annotate different columns in different tuples, this difference is not visible in the query itself. This is unlike the XARCH system where structural differences manifest themselves in the query expression. With this in place, the following simple CA expression retrieves all the different spellings of the country’s name:

$$\pi_{co}(\Sigma_{\text{“KZ”}}(r^k))$$

Notice that this query is not expressible in the XARCH system since unless we know explicitly the different spellings, we cannot identify the relevant parts of the tree that refer to this country (and not another one). As another example, the following CA expression retrieves all the tuples for which the number of airports in Kazakhstan is stored in the *prv* column:

$$\Sigma_{\text{“KZ”}}(\Pi_{prv}^U(r^k))$$

The query returns the tuples with *tid*=2, *tid*=3 and *tid*=4. Again, notice that we retrieve these numbers for different spellings of the country’s name, and for values that are located at different paths of the corresponding tree in the XARCH system. \square

All combined, this shows that MONDRIAN naturally allows for the modeling of curated evolving data and that the color algebra provides an elegant way to pose historical queries. The current paper provides only a proof-of-concept of our approach. We leave the experimental validation to future work.

References

1. <https://www.cia.gov/library/publications/the-world-factbook/index.html>
2. <http://www.iuphar-db.org>
3. Bhagwat, D., Chiticariu, L., Tan, W.C., Vijayvargiya, G.: An annotation management system for relational databases. In: Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), pp. 900–911 (2004)
4. Buneman, P., Cheney, J., Tan, W.C., Vansummeren, S.: Curated databases. In: Proceedings of the 27th Symposium on Principles of Database Systems (PODS), pp. 1–12 (2008)
5. Buneman, P., Davidson, S., Fan, W., Hara, C., Tan, W.C.: Keys for xml. In: Proceedings of the 10th International Conference on World Wide Web (WWW), pp. 201–210 (2001)
6. Buneman, P., Khanna, S., Tajima, K., Tan, W.C.: Archiving scientific data. ACM Trans. Database Syst. 29(1), 2–42 (2004)
7. Buneman, P., Khanna, S., Tan, W.C.: On propagation of deletions and annotations through views. In: Proceedings of the 21st Symposium on Principles of Database Systems (PODS), pp. 150–158 (2002)
8. Curino, C.A., Moon, H.J., Zaniolo, C.: Graceful database schema evolution: the prism workbench. Proc. VLDB Endow. 1(1), 761–772 (2008)
9. Eltabakh, M.Y., Aref, W.G., Elmagarmid, A.K., Ouzzani, M., Silva, Y.N.: Supporting annotations on relations. In: Proceedings of 12th International Conference on Extending Database Technology (EDBT), pp. 379–390 (2009)
10. Eltabakh, M.Y., Ouzzani, M., Aref, W.G., Elmagarmid, A.K., Laura-Silva, Y., Arshad, M.U., Salt, D., Baxter, I.: Managing biological data using bdbms. In: Proceedings of the 25th International Conference on Data Engineering (ICDE), pp. 1600–1603 (2008)
11. Geerts, F., den Bussche, J.V.: Relational completeness of query languages for annotated databases. J. Comput. Syst. Sci. 77(3), 491–504 (2011)

12. Geerts, F., Kementsietsidis, A., Milano, D.: \mathcal{M} MONDRIAN: A visual tool to annotate and query scientific databases. In: Ioannidis, Y., et al. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 1168–1171. Springer, Heidelberg (2006)
13. Geerts, F., Kementsietsidis, A., Milano, D.: Mondrian: Annotating and querying databases through colors and blocks. In: Proceedings of the 22nd International Conference on Data Engineering, ICDE (2006)
14. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: Proceedings of the 26th Symposium on Principles of Database Systems (PODS), pp. 31–40 (2007)
15. Koltsidas, I., Müller, H., Viglas, S.D.: Sorting hierarchical data in external memory for archiving. *Proc. VLDB Endow.* 1(1), 1205–1216 (2008)
16. Kostylev, E., Buneman, P.: Combining dependent annotations for relational algebra. In: Proceedings of the 15th International Conference on Database Theory, ICDT (2012)
17. Mavromatis, M.: Indexing in the MONDRIAN annotation management system. Master’s thesis, University of Edinburgh, United Kingdom (2006)
18. Moon, H.J., Curino, C.A., Deutsch, A., Hou, C.Y., Zaniolo, C.: Managing and querying transaction-time databases under schema evolution. *Proc. VLDB Endow.* 1(1), 882–895 (2008)
19. Müller, H., Buneman, P., Koltsidas, I.: Xarch: archiving scientific and reference data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), pp. 1295–1298 (2008)
20. Srivastava, D., Velegarakis, Y.: Intensional associations between data and metadata. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), pp. 401–412 (2007)
21. Wang, H.(W.), Liu, R., Theodoratos, D., Wu, X.: Efficient storage and temporal query evaluation in hierarchical data archiving systems. In: Bayard Cushing, J., French, J., Bowers, S. (eds.) SSDBM 2011. LNCS, vol. 6809, pp. 109–128. Springer, Heidelberg (2011)