Towards a Graph-Oriented Perspective for Querying Music Scores

Philippe Rigaux [1], **Virginie Thion** [2]

1. *Conservatoire National des Arts et Métiers, CEDRIC Laboratory, France*
   Philippe.Rigaux@cnam.fr
2. *Univ. Rennes, CNRS, IRISA, Lannion, France*
   Virginie.Thion@irisa.fr

ABSTRACT. Sheet music scores *have been the traditional way to preserve and disseminate Western classical music works for centuries. Nowadays, their content can be encoded in digital formats that yield a very detailed representation of music content expressed in the language of* music notation. *These* encoded (digital) scores *constitute an invaluable asset for digital library services such as search, analysis, clustering, and recommendations. In this paper, we propose a model of the musical content of digital score as graph data, which can be stored in a graph database management system. We then discuss the querying of such data through graph pattern queries. We also describe a proof-of-concept of the approach that allows uploading music scores in a Neo4j database, and expressing searches and analyses through graph pattern queries with the query language Cypher.*

RÉSUMÉ. *Depuis plusieurs siècles, la diffusion de la musique occidentale est assurée par la représentation des œuvres sous forme de partitions musicales. Ces partitions peuvent aujourd'hui être encodées dans des formats numériques offrant une représentation fine de leur contenu, ouvrant ainsi la voie à de nouvelles fonctionnalités. La notation musicale, même numérisée, reste cependant extrêmement complexe, notamment en raison de l'imbrication des aspects relatifs au contenu et de ceux qui décrivent la mise en forme de ce contenu. Dans cet article, nous proposons une modélisation formelle, sous forme de données graphe, du contenu purement musical des partitions numérisées. Cette modélisation vise à s'abstraire des aléas liés aux choix d'encodage et à la surchage consécutive aux informations de mise en forme. Nous discutons ensuite des fonctionnalités d'interrogation offertes par cette représentation. Nous fournissons également une réalisation concrète de ce cadre formel, sous la forme d'une implémentation dans le système Neo4j permettant l'interrogation des données* via *le langage de requête Cypher.*

KEYWORDS: *Music scores, Graph databases, Data model, Pattern queries*

MOTS-CLÉS : *Partitions musicales, Base de données graphe, Modèle de données, Interrogation à base de patrons*

---

1. Introduction

Music is an essential part of the world's cultural heritage. Even though audio files constitute the main access channel to music works nowadays, music has been preserved and disseminated as *sheet scores* for centuries. For a part of music production, sheet scores have been – and continue to be – the most complete and accurate way to encode the composer's intents, and to faithfully convey these intents to performers.

A sheet score is a complex semiotic object. In a single and compact layout, it combines a symbolic encoding of the music that must be produced with a sophisticated visual representation aiming at accurately representing the music content. As an illustration, Fig. 1 is the excerpt of the human-readable visualisation of a music score, extracted from a MEI dataset available in the NEUMA platform (Rigaux *et al.*, 2012; Neuma, 2022), which contains four musical voices.

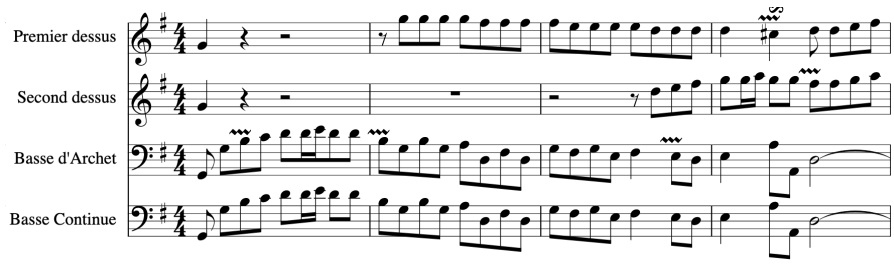


Figure 1. Excerpt of La Françoise*, François Couperin (NEUMA platform)*

Nowadays, digital score libraries store collections of music scores, most often in the form of images i.e., scans of sheet scores. We can cite IMSLP (IMSLP, 2022) or Gallica (Gallica, 2022). In general, we expect a digital library to be more than a simple repository of digital documents, encoded in a music-agnostic format that obscures their content (Foscarin *et al.*, 2021). Services that leverage digital representation are required, and at the very least a search engine that allows retrieval of documents that match patterns of interest *by content*, the extraction of relevant patterns and features, transformation (e.g., transposition), analysis (e.g., frequent patterns, quality defaults), *etc.* To supply such intelligent services, we need a digital representation that truly encodes the *music notation* embedded in a music score, and thereby gives fine-grained access to all its components. We will call *encoded scores* such documents. The most salient formats that exist at the moment are `**kern` (KernScores, 2022; kern, 2022), MusicXML (Good, 2001; MusicXML, 2022) and MEI (Rolland, 2002; MEI, 2022).

On the over hand, the need to handle *complex* data has led to the emergence of new types of data models. In the last few years, *graph databases* (Angles, Gutierrez, 2008; Angles, 2012; Webber *et al.*, 2013; Angles *et al.*, 2017; Bonifati *et al.*, 2018) have started to attract a lot of attention in the database community. Their basic purpose is to manage data natively modelled as a graph like e.g., social networks, biological, topological databases or bibliographic databases. As the content of an encoded score is composed of highly complex information, with connected items (a note follows

another one, it belongs to a voice, it also belongs to a measure, etc.), its graph-based representation seems a relevant approach to leverage existing encodings to a true data model apt at supporting analytic operations, including searches.

In this paper, we propose two complementary contributions. First, we model the content of encoded scores as a graph, which can then be stored in a graph database management system, and processed with graph pattern queries. We expose the formal structure of the model, illustrate the querying mechanism, and discuss its expressiveness. Second, we propose a proof-of-concept of the approach based on a tool called MUSYPHER, which allows producing the graph-based representation of an encoded score and then uploading it in a Neo4j graph database, in order to query such data with the concrete query language Cypher.

The paper is organized as follows. Section 2 presents a review of literature focused on recently proposed music score content models. Section 3 introduces our graph-based data model. Section 4 studies the properties of graph patterns queries. Section 5 then presents our implementation. Section 6 concludes and draws some perspectives of this work.

2. Related work

When modelling an encoded score, the literature provides two main approaches. The first one consists in a tree-shaped model. The second one sees the music content as a collection of time series, modelling musical events performed over a time period. In the following, we succinctly present these two approaches.

Tree-based modelling. Driven by the need of interoperability between systems and tools, semi-structured models have emerged for representing (Western) music scores. Widespread ones are MusicXML (Good, 2001; MusicXML, 2022) and MEI (Rolland, 2002; MEI, 2022). Their common characteristics is to encode the complete content of sheet scores (including graphical specifications that dictate how the notation content has to be visually rendered), and to organize, in the form of a n-ary ordered tree, the music events (e.g., notes, lyrics) according to the measure they belong to. Generic database query languages may be applied to such documents like XPath or XQuery (Ganseman *et al.*, 2008; Fournier-S'niehotta *et al.*, 2018). However, their complexity, mix of several concerns and the fact that a same information can be encoded with many syntactic variants hinder the definition of robust music-oriented query languages. We take as a starting point in the present paper the recent proposal of (Zhu *et al.*, 2022) to normalize the hierarchical structure of music score.

Times-series modelling. Another perspective consists in seeing the music content as time series of musical events (called "voices"). The *ScoreAlg* algebra, defined in (Fournier-S'niehotta *et al.*, 2018), is based on such a perspective. A voice is a function whose domain is a time measurement (relevant division of time for timestamping the musical events e.g., a time unit that represents the smallest interval between two musical events) and co-domain is the set of musical events. Voices are then synchro-

nized in order to form a complex score. *ScoreAlg* provides a closed form language based on operators for manipulating music scores information. Such a language, powerful but based on complex abstractions, may be difficult to use for the average user who is familiar with music notation but not the digitized encoding of music scores, e.g., a music performer (who retrieves music scores in order to play music with his band), a music analyst (who searches for similar patterns in the parts of a music score), or a musicologist (who conducts a philological study on Rameau's compositions including all their variants over time).

Graph-based modelling of the rhythm. Driven by the need to enhance an Optical Music Recognition (ORM) process, (Jin, Raphael, 2015) proposes an approach that represents a music content from a rhythmic perspective. The rhythm itself is a graph where each node models the occurrence of a rhythmically relevant symbol (note, rest, and bar line). Each symbol has a duration (a dotted eighth has duration of 3/16, while a bar line has a duration of 0). Nodes are connected either by their order of occurrence inside a voice (a symbol follows another one) or by coincidence edges (symbols of different voices are connected if they share the same onset time). This model focuses on the alignment of the music sheet symbols in order to refine music symbols recognition.

Our graph-based model follows the trend, explored in the above-mentioned papers, to abstract the content of encoded scores in order to expose a robust representation, focused on music content, and cleared from side information related to representation purposes. The graph representation captures both a tree-based representation that exposes the hierarchical nature of a score, and a time series perspective, close to the intuitive understanding of a music score as a flow of sound events.

3. Graph-based modelling of encoded scores

We now turn to the definition of a graph-based data model for modelling score content. It relies heavily on principles taken from music notation, seen as an expressive formal language that provides a powerful basis for modelling music content. Our model gives an abstract vision of digital music documents as structured objects, focusing on music content, cleared from side information related to representation purposes, and supports query functionalities developed in the forthcoming sections. It will be illustrated with the German anthem, *Das Lied der Deutschen*, composed by *Joseph Haydn* in 1797 (Haydn, 1797). The notation of this example is shown on Fig. 2.

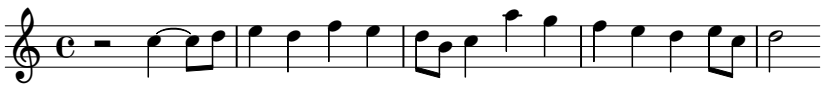


Figure 2. First notes of the German anthem, Das Lied der Deutschen *by Joseph Haydn (1797)*

To state it in a nutshell, we model music information as a mapping from a structured temporal domain to a set of (musical) *facts*. The temporal domain is a hierarchical structure, called *rhythmic tree*, that partitions a finite time range in non-overlapping

intervals. Each interval defined by a leaf of the rhythmic tree is associated with a music fact. Together they constitute a *musical event* (i.e., a fact that occurs during a specific temporal interval).

3.1. Preliminaries: the property graph model

In a graph database management system, the schema is a graph (nodes are entities and edges are relations between entities), and data is handled through graph-oriented operations and type constructors (Angles, Gutierrez, 2008; Angles, 2012; Webber *et al*., 2013; Angles *et al*., 2017). Our modelling relies the *property graph* data model (Angles *et al*., 2017; Bonifati *et al*., 2018), where nodes and edges may embed data in *properties* (key-value pairs). In terms of vocabulary, we assume the existence of the pairwise disjoint sets: a set $\mathcal{V}$ of *nodes*, a set $\mathcal{E}$ of edges.We also consider a set *Lab* of labels, and a set *Prop* of *properties* (a.k.a. *property keys*), and a set *Val* of *values*.

DEFINITION 1 (Property graph). — *A property graph $\mathcal{G}$ is a tuple $(V, E, \rho, \lambda, \sigma)$ where (1) $V \in \mathcal{V}$ is a finite set of nodes; (2) $E \in \mathcal{E}$ is a finite set of edges; (3) $\rho : E \to (V \times V)$ is a total function assigning to each edge an ordered pair of nodes (where $\rho(e) = (n_1, n_2)$ indicates that e in an edge going from n_1 to n_2); (4) $\lambda_V : V \to \mathcal{P}(Lab)$ is a partial function assigning a set of labels to the nodes of V; and $\lambda_E : V \to Lab$ is a total function assigning a label to each edge of E; (Without ambiguity, λ refers either to λ_V or to λ_E according to the domain of the assigned element.) (5) $\sigma : (V \cup E) \times Prop \to Val$ is a partial function assigning property-value pairs to nodes of V and edges of E (where $\sigma(n, p) = v$ (resp. $\sigma(e, p) = v$) indicates that the node n (resp. edge e) has a property p with the value v).*

Classical notions come with the definition of a property graph $\mathcal{G} = (V, E, \rho, \lambda, \sigma)$.

DEFINITION 2 (Path). — *A path p in $\mathcal{G}$ is a sequence $n_1\, l_1\, n_2\, l_2\, n_3 ...n_{k-1}\, l_{k-1}, n_k$ where $\{n_1, ..., n_k\} \subseteq V$ and $\{l_1, ..., l_{k-1}\} \subseteq Lab$ and $k \geq 1$ and each (n_i, l_i, n_{i+1}) s.t. $i \in \{1, ..., k-1\}$ is an edge of $\mathcal{G}$.*[1] *Path p is said to connect n_1 to n_k, and its* size *is $|p| = k - 1$. Any non empty path that is a subsequence of p, is a* subpath *of p. Path p is a* cycle *iff $k \geq 2$ and $n_1 = n_k$. Path p is* cyclic *iff one of its subpaths is a cycle, otherwise p is said to be* acyclic.

DEFINITION 3 (Paths). — *Let n_1 and n_2 be two nodes of V. We denote by $Paths(n_1, n_2)$ the set of acyclic paths that connect n_1 to n_2.*

3.2. The domain of musical facts

Several fact domains can be envisaged to describe a musical object. Due to space limitation, we will focus here on the main one, *sounds*. A sound can be characterized

1. (n_i, l_i, n_{i+1}) is an edge of $\mathcal{G}$ means that there is $e \in E$ such that $\rho(e) = (n_i, n_{i+1})$ and $\lambda(e) = l_i$.

by many properties, including intensity, timbre and frequency. In the language of music notation, a finite set of frequencies, or *pitches*, is used to refer to the sounds usable in a musical piece. We follow the designation of the International Standards Organization (ISO) for enumerating the pitches. In this designation, each pitch is referred to by a pitch class P (a letter A, B, C, D, E, F, or G), an index I in $[1, 7]$, and an optional accidental a in $\{\sharp, \flat\}$ (*sharp* and *flat* accidentals). We will thus model a sound as a triple $\{class : P,\ octave : I,\ accidental : a\}$, and represent it as symbol of the form $P[a]I$. Addition of other relevant properties (e.g., intensity) is trivial.

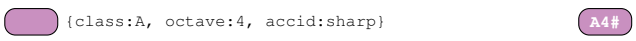
Graphically (i.e., in music scores), frequency levels are materialized by groups of horizontal lines (called staves) and pitches are represented by black or white heads vertically positioned on staves. The first pitch in the score of Fig. 2 is a $C5$, followed by a $D5$, an $E5$, etc. Music is also made of silences (or *rests*), and we thus add the *rest symbol* r to the domain. The German anthem starts with a rest, graphically represented by a small rectangle.

Finally, in conventional music notation, sounds can be "tied" (graphically represented as curves over the heads, such as in the first measure of Fig. 2). We add the *continuation symbol* _ to our domain to represent ties. We obtain the domain of musical facts.

DEFINITION 4 (Domain of (atomic) musical facts). — *The domain $\mathcal{F}_{\mathcal{M}}$ of musical facts consists of:*

1. *the set of sound facts $P[a]I$, $P \in \{A, B, C, D, E, F, G\}, a \in \{\sharp, \flat\}, I \in [1, 7]$,*
2. *the rest fact, noted* `r`*,*
3. *the continuation fact, noted* _ .

Facts are represented as nodes with properties. Fig. 3.(a) is a fact node that models an instance of an A4♯ note. The node has three properties, namely `class` (this property has the value `A` for the node), `octave` (having the value `4`) and `accid` (having the value `sharp`, which models a ♯ alteration). In the following, such a node will be depicted by a compact representation embedding the core property values in the node itself (Fig. 3.(b)).



(a) fact node with its property values (b) compact representation

Figure 3. Fact node: instance of an $A4\sharp$ note

We can derive some important notions from musical facts. An *interval* is a distance between two sounds, physically characterized by the ratio of their respective frequencies. A ratio of 1 denotes a *unison*, a ratio of 2 an *octave*. The octave is the fundamental interval that structures symbolic music representation. In Western music notation, an octave range is divided in 12 *semi-tones*. This defines a scale, called *chromatic*, with 12 *steps*, corresponding each to exactly one semi-tone. A *chromatic interval* is the number of semi-tones between two pitches.

3.3. Temporal organization of facts

A music piece is a temporal organization of sounds inside a bounded time range. Musical facts cannot be assigned to any timestamp but fall on a set of positions that defines a discrete partitioning of this range. More precisely, this partition results from a recursive decomposition of temporal intervals, yielding a rhythmic organization which is inherently hierarchical.

In Western music notation, a music piece is divided in *measures* (graphically represented as vertical bars on Fig. 2), and a measure contains one or more *beats*. Beats can in turn be divided into equal units (i.e., sub-beats). Further recursive divisions often occur, generating a hierarchy of pulses called *metrical structure*. The *time signature*, a rational number (in our example $4/4$ denoted by **C**), determines the preferred decomposition. A $4/4$ measure consists of 4 beats, and each beat is one quarter (graphically, a black note ♩) long. Still in the context of a $4/4$ time signature, the preferred decomposition of a measure, is into 4 sub-intervals (some other partitions are possible, although less likely), beats are preferably partitioned in two quavers (graphically, a ♪), themselves (generally) partitioned in semi-quavers (𝅘𝅥𝅯), etc. For other meters (e.g., $3/4$, $6/8$), temporal decomposition follows different patterns. However, in all cases, the metrical structure can be represented as a *rhythmic tree* based on the following principles: i) the time range of the piece in divided in equal-sized measures, ii) each measure is recursively divided according to metric rules, and iii) each leaf of the tree corresponds to a musical fact that occurs during the sub-interval defined by the position of the leaf.

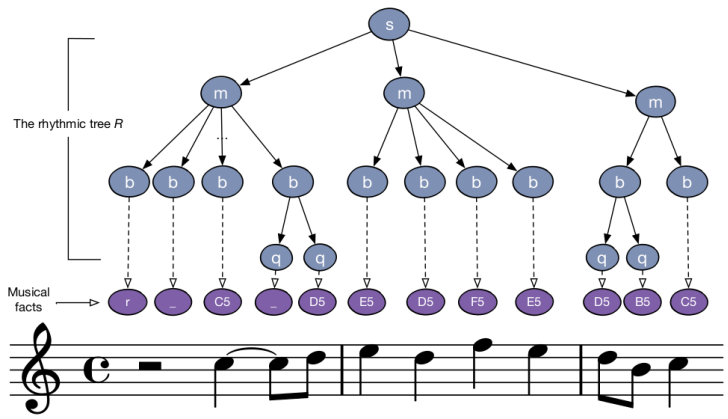


Figure 4. Modelling the German anthem, with its rhythmic tree and associated facts

Fig. 4 illustrates our structure: a rhythmic tree that temporally organizes the musical facts. The part with blue nodes represents the rhythmic tree. The root (s) corresponds to the whole music piece. The level under the root is made of *measure nodes*. Each measure is itself decomposed in beats, quavers, etc., according to the required temporal decomposition. The first measure for instance is decomposed in four beats, the latest being itself decomposed in two quavers.

Each leaf of the rhythmic tree is linked to a musical fact (the purple nodes). The first beat of the first measure is a *rest* fact, the second is a *continuation* fact (extending the previous rest), the third is a *sound* fact (a $C5$). The fourth beat is divided in two facts: a continuation of the $C5$, and a $D5$ fact. This structure summarizes our modelling, and the basis for building a graph representing music content. Note that we have only been discussing so far of *monodic* music i.e., a single flow of sounds. Respecting a well-established vocabulary in the field of music notation, we will call such a structure a *voice*.

DEFINITION 5 (Voice). — *A* voice *is a pair* $(R, \mathcal{M})$*, where* R *is a rhythmic tree and,* $\mathcal{M}$ *is mapping from the leaves of* R *to the set of musical facts* $\mathcal{F}_{\mathcal{M}}$*.*

Given a voice, we can easily infer several properties that will serve as a basis for the querying process. Let us start with the durations and temporal positions of the facts, determined from the rhythmic tree thanks to the following definition.

DEFINITION 6 (Temporal partition). — *Let* $I = [\alpha, \beta[$ *be a time range and* R *a rhythmic tree. The* temporal partitioning *of* I *with respect to* R *assigns an interval* $itv_I(N)$ *to each node* N *of* R *as follows.*

1. If N *is the root of* R*,* $itv_I(N) = I$

2. If N *is of the form* $N(N_1, \cdots, N_n)$*,* $itv_I(N) = [\alpha_N, \beta_N[$ *is partitioned in* n *sub-intervals of equal size* $s = \frac{\beta_N - \alpha_N}{n}$ *each:* $itv_I(N_i) = [\alpha_N + (i-1) \times s, \alpha_N + i \times s[$

Given a time range I, the leaf level of a rhythmic tree R defines a partitioning of I as a set of non-overlapping temporal intervals. Now, consider a voice $(R, \mathcal{M})$. Each leaf l of R covers an interval $itv_R(l)$, whose duration is naturally that of its associated fact $\mathcal{M}(l)$. Together, $itv_R(l)$ and $\mathcal{M}(l)$ constitute a *musical event*.

DEFINITION 7 ((Musical) event). — *Let* I *be a time range, and* $V = (R, \mathcal{M})$ *a voice. If* l *is a leaf of* R*, then the pair* $(itv_I(l), \mathcal{M}(l))$ *is a* musical event.

We adopt the following convention to represent temporal values: the duration of a measure is 1, it extends over interval $[0, 1[$, and the music piece range is $I = [0, n[$, n being the number of measures. Both the duration and interval of a node result from the recursive division represented by the tree. If the meter is $4/4$, the duration of a half note for instance is $\frac{1}{2}$, the duration of a beat is $\frac{1}{4}$, etc. Turning back to Fig. 4, the first event is (I_1, r), with $I_1 = [0, \frac{1}{4}[$ (the first beat of the first measure). The second event is $(I_2, _)$, with $I_2 = [\frac{1}{4}, \frac{2}{4}[$ (we recall that it represents a continuation of the rest for one beat). The third event is $([\frac{2}{4}, \frac{3}{4}[, C5)$, the fourth is $([\frac{3}{4}, \frac{7}{8}[, _)$, etc.

3.4. Polyphonic Music

The representation of polyphonic music simply consists of a set of voices sharing a same number of measures. Fig. 5 gives an illustration (the same theme, with a bass part added). In terms of modelling, we have two choices. Either we simply model a polyphonic piece as a set of voices $\{V_1, V_2, \cdots, V_n\}$, with the constraint that all the upper level (measures) of their respective rhythmic trees are similar. Or we "factorize"

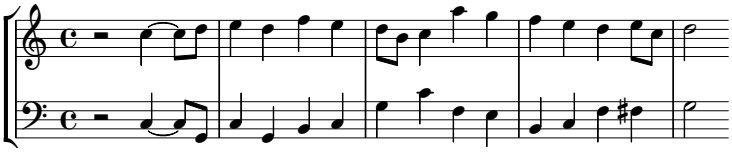


Figure 5. German anthem, with two voices

these upper levels, representing the common sequence of measures, and we manage individual subtrees for each measure and each voice.

3.5. Graph based representation of voices

We now turn to the graph-based modelling of a voice. The graph is almost directly obtained from the voice structure thanks to the following transformations:

1. An edge is added between adjacent measures: this allows to navigate from one measure to the other.

2. Intermediate levels between the level of the measures and the leaf levels are removed: measures are directly connected to leaves.

3. Since the previous simplification removes the structural information that determines the temporal features (intervals and durations) of the leaves (see Def. 7), we add these features explicitly in the graph: an edge is added between two adjacent leaves, with a duration property.

4. The content of a voice may be reached either from the root of the rhythmic tree (the rhythmic tree perspective) or from its first event (the time series perspective). Any relevant information that qualifies a voice as a whole may be attached to the voice node e.g., the instrument or the name of the voice.

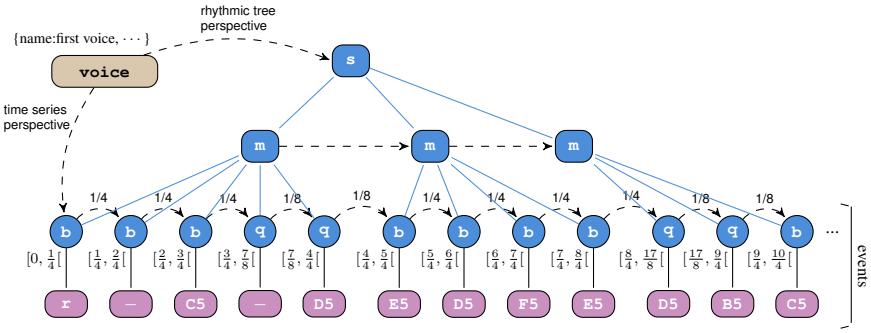


Figure 6. Graph-based representation of a voice

One obtains, from the voice of Fig. 4, the graph of Fig. 6. It can be seen as a compacted representation of the voice structure, with pre-computed important values (e.g., durations), and additional edges that help navigating the graph.

4. Querying a graph-based music score content

Let us now consider the problem of querying the graph-based modelling of an encoded score. We first recall the basic notion of property graph pattern. A *graph pattern* is classically defined as a graph where variables and conditions can occur (Barceló, 2013; Angles *et al.*, 2017). Intuitively, it defines a shape that has to be found in data.

DEFINITION 8 (Graph pattern syntax). — *Let Var_{nodes} and Var_{lab} be distinct sets of node variables and label variables respectively. A* graph pattern query *is a tuple of the form $(V, E, \rho, \lambda, \sigma, Cond_n, Cond_e)$ where variables can occur on nodes and on edge labels,*[2] *and $Cond_n$ (resp. $Cond_e$) denotes Boolean conditions over property values of the elements of V (resp. E) (e.g., if f_1 is a node variable, then a condition could be $f_1.class = E \wedge f_1.octave = 4$, which indicates in our context that the node mapping in f_1 must be a $E4$ fact note).*

For the sake of querying a music content, we extend the notion of graph pattern query to the relevant musical vocabulary, which allows using, in the pattern conditions, the notions of interval between two notes and distance between two events.

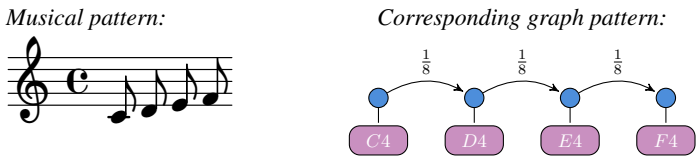


Figure 7. Sequence of C4-D4-E4-F4

In its simplest form, a graph pattern denotes a path. Fig. 7 and 8 are some examples of such derived patterns. In these patterns, nodes and edges are free variables, and conditions may be attached to the pattern. The pattern of Fig. 7 aims at retrieving the sequences of eighth notes $C4$-$D4$-$E4$-$F4$ that appear in data. The pattern contains conditions over the pitch class and octave of the notes, and over the duration of the events.

The pattern of Fig. 8 aims at retrieving the occurrences of two notes following a $C4$ one. `Fact1` and `Fact2` are node variables. A condition is attached with `Fact1` that must be a E note. Another condition is attached to the pattern concerning the adequacy of the octaves of `Fact1` and `Fact2`. There is no other condition over the notes (for instance, there is no condition over the duration of each of the three notes).

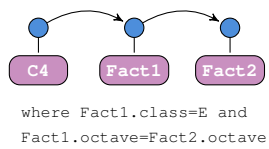
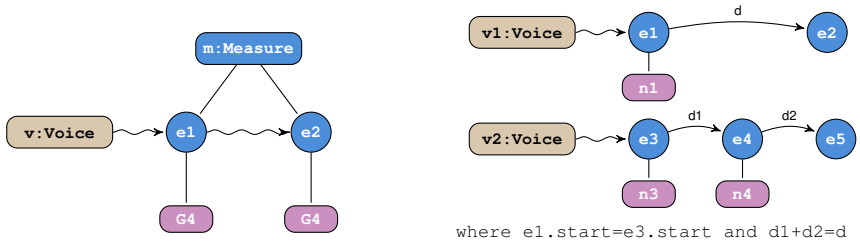


Figure 8. Notes following a $C4$

2. This means that $(V, E, \rho, \lambda, \sigma)$ is a property graph such that $V \in \mathcal{V} \cup Var_{nodes}$ and $\lambda : (V \cup E) \rightarrow Lab \cup Var_{lab}$.

A query may also have the form of a more complex graph pattern. For instance, the complex pattern P_{twoG4} of Fig. 9.(a) aims at retrieving two $G4$ notes that belong to the same measure in the same voice (but the notes are not necessary adjacent). The wavy relationship between the events denotes an arbitrary path. Another complex pattern is P_{poly} (Fig. 9.(b)), which aims at retrieving the occurrence of two notes (`n3` and `n4` on the figure) played during the time of another one (`n1` on the figure) in another voice.



(a) Pattern P_{twoG4} (b) Polyphonic pattern P_{poly}

Figure 9. More complex graph patterns

Formally, the interpretation (evaluation) $[\![P]\!]_{\mathcal{G}}$ of a graph pattern query P over a graph database $\mathcal{G}$ consists in finding all the subgraphs of $\mathcal{G}$ that are homomorphic to P (Barceló, 2013).

DEFINITION 9 (Graph pattern interpretation). — *The answer $[\![P]\!]_{\mathcal{G}}$ of a pattern query P over a graph $\mathcal{G}$ is the set of subgraphs $\{g \in \mathcal{P}(\mathcal{G}) \mid g$ "matches" $P\}$. A subgraph g "matches" P iff there is a homomorphism h from the nodes and labels variables of P to $\mathcal{G}$ and each node (resp. label) $h(v)$ satisfies its associated conditions $Cond_n$ (resp. $Cond_e$).*

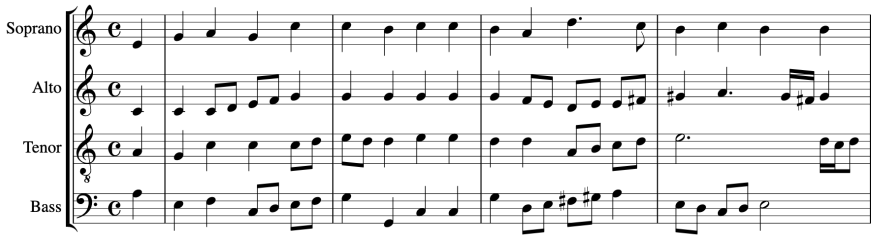


Figure 10. J.S. Bach cantate BWV111*, 6th movement*

In order to illustrate the interpretation of a graph pattern query, we consider the polyphonic music score of Fig. 10, denoted by BWV111 in the following. The encoding of this music score is available (MEI format) in the J.S. Bach collection of the NEUMA platform (Neuma, 2022). The music score contains four voices: *Soprano*, *Alto*, *Tenor* and *Bass*. Five measures appear on Fig. 10, numbered from 0 to 4.

Fig. 11 presents the graph-based representation of BWV111, restricted to the beginning of the *Soprano* and *Alto* voices. This graph respects the model proposed in

Section 3. The rhythmic tree, common to all the voices, appears in blue. Fact nodes appear in purple. The `Soprano` and `Alto` voice nodes appear on the left. With each voice is associated the time series of events that compose the voice content. The edges of the time series embed, in property values, the *duration* of each event, expressed here in the time unit of the *beat*. The events embed their time range in additional properties named start and end (by lack of space, these values do not appear in the figure but can easily be inferred from the duration value carried by the edges of the time series). This graph is denoted by $\mathcal{G}_{\text{BWV111}}$ in the following.

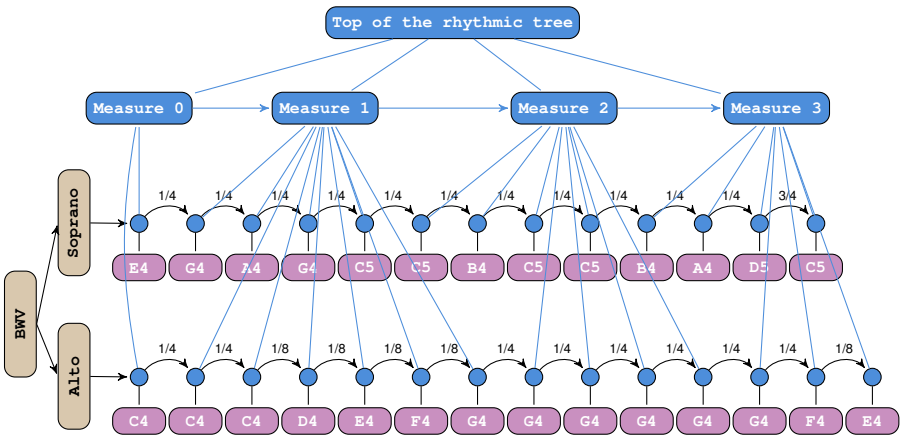


Figure 11. ($\mathcal{G}_{\text{BWV111}}$) Graph-based modelling of BWV111

Let us also consider the graph pattern P_{twoG4} of Fig. 9.(a). This pattern matches into several subgraphs of $\mathcal{G}_{\text{BWV111}}$, including a subgraph that contains the first and third notes of the measure 1 for the voice *Soprano*, and another subgraph that contains the first and second notes of the measure 2 for the voice *Alto*. These two corresponding subgraphs of $\mathcal{G}_{\text{BWV111}}$ belong to $[\![P_{twoG4}]\!]_{\mathcal{G}_{\text{BWV111}}}$. Note that other answers that match P_{twoG4} can be found in $\mathcal{G}_{\text{BWV111}}$.

Expressiveness, limitations and cost. In terms of expressiveness of the model, the graph-based model itself -from a static point of view- implements the hierarchical modelling of the rhythm and the time series oriented perspective of (Fournier-S'niehotta *et al*., 2018). It also captures the core structure of a tree-based representation of a music content (XML-based formats). Some rhythmic features like those considered in the model of (Jin, Raphael, 2015) are also captured, limited to musical events (those that "produce a sound") and their coincidence (series of such events in a voice).

The problem of querying a graph has been studied in the literature (Barceló, 2013; Angles *et al*., 2017). In terms of query expressiveness, as a tree is special case of a graph, the navigational queries (e.g., XPath-like) that applied over a tree-based rep-

resentation can be applied to the graph-based structure (Libkin *et al.*, 2013).[3] Graph patterns queries, whose evaluation is based on graph pattern matching, are a very expressive formalism, not surprisingly more expensive in terms of evaluation cost. Such a cost depends on the form of the query pattern (Wood, 2012; Barceló, 2013; Barceló *et al.*, 2014; Angles *et al.*, 2017), going from $O(|\mathcal{G}|.|P|)$, where $|\mathcal{G}|$ is the size of the data graph and $|P|$ is the size of the pattern, for a pattern having the "simple" form of a regular path query (a path connecting two nodes where the edge is labelled by a regular expression), to an NP-complete problem for the most general case of a complex pattern that may contain a cycle.

5. Implementation

We implemented a proof of concept of the proposed framework, in the Neo4j graph database management system (Neo4j, 2022), for querying a music score through the Cypher (Cypher, 2022) query language. The implementation is based on a software tool called MUSYPHER, developed for our project.[4] MUSYPHER makes possible to process a MEI (XML-based) file in order to translate its music content into a Neo4j graph database that respects the graph-based representation proposed in Section 3.

In order to illustrate the implementation, we consider again the music score of Fig. 10 (p. 11), whose digitized content was initially available in the MEI format in the J.S. Bach collection of the NEUMA platform (Neuma, 2022). This music score was translated in its graph-based representation (see Fig. 11) and loaded in a Neo4J database.

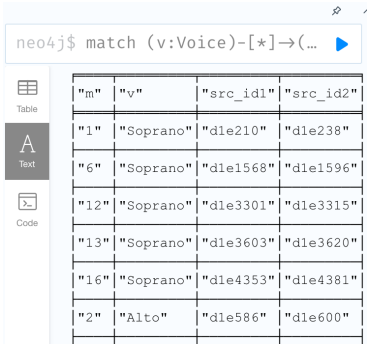
We now consider the querying of such data "in practice". Let us return to the graph pattern P_{twoG4} of Fig. 9.(a). In the Cypher language, a query based on P_{twoG4} could be expressed by Query Q_{twoG4} given in Fig. 12. The shape of the graph pattern is declared in the MATCH clause (lines 1 to 6). In the Cypher formalism, a graph pattern is defined in the manner of the ASCII art, where the graphic symbol ( ) denotes a node, which may contain information of the form variable:Type, and the symbol –[expr]–> defines the form of a connection (expr is either an edge label, or a regular expression denoting a path). Expressions of the form {attr1:value1, attr2:value2, ...} are additional conditions over properties. The RETURN clause (lines 7 to 9) contains the elements of interest that should be rendered as a result. Here, we retrieve for each answer subgraph: the number of the measure, the name of the voice, and the initial MEI identifier of the elements[5] that was considered as a relevant property in our context, to be transferred into the graph.

3. Of course, the literature proposes a flavour of navigational query languages.

4. MUSYPHER is a Java application, based on a DOM parsing of the initial XML encoded score. The current version of the tool is available at https://www-shaman.irisa.fr/musypher

5. Knowing the MEI identifiers, we can highlight the elements in a rendering tool like Verovio (Pugin *et al.*, 2014).

```
MATCH
  (v:Voice)–[*]–>(e1:Event)–[*]–>(e2:Event),
  (m:Measure)––(e1),
  (m:Measure)––(e2), // the same measure m
  (e1)––(note1{class:'g',octave:4}),
  (e2)––(note2{class:'g',octave:4})
RETURN
  m.number AS m, v.name AS v,
  e1.id AS src_id1, e2.id AS src_id2
```



"m"	"v"	"src_id1"	"src_id2"
"1"	"Soprano"	"d1e210"	"d1e238"
"6"	"Soprano"	"d1e1568"	"d1e1596"
"12"	"Soprano"	"d1e3301"	"d1e3315"
"13"	"Soprano"	"d1e3603"	"d1e3620"
"16"	"Soprano"	"d1e4353"	"d1e4381"
"2"	"Alto"	"d1e586"	"d1e600"

Figure 12. Query Q_{twoG4} and the result of its evaluation by Neo4j over $\mathcal{G}_{\text{BWV111}}$

A part of the result of Q_{twoG4}, evaluated by the Neo4j query evaluator engine over $\mathcal{G}_{\text{BWV111}}$, is given in the Neo4j desktop screenshot placed at the right of Fig 12. Among the answers of the table, the first and sixth answers correspond to the answer subgraphs previously discussed according to the interpretation of P_{twoG4} over $\mathcal{G}_{\text{BWV111}}$ (see section 4, p. 12).

6. Conclusion and perspectives

In this paper, we proposed a graph-based data model for modelling the musical content of digital scores. This model captures the two perspectives of a music score content: (i) as a metrical structure that starts from a rhythmic organization of the temporal range, and associates a fact with each leaf of the rhythmic tree, and (ii) as a time series of events, which is close to the intuitive understanding of a music score as a flow of sounds. We discussed the querying of such data through graph pattern queries. We also presented a proof-of-concept of the approach that allows uploading music scores in a Neo4j database (using a tool called MUSYPHER), and expressing searches and analyses through graph pattern queries with the query language Cypher.

This work opens many perspectives. Some of them concern the extension of the model. First, we considered only the domain of sound facts, but other domain can be added to enrich the description of music information, following the same principle: such information is a mapping from a rhythmic organization of time to facts. The domain of *syllables* for instance allows to model lyrics in vocal music; one could add domains of semantic facts (timbre, texture, intensity) to model in general *annotations* that qualify temporal fragments of a music piece. A second kind of extension could address more sophisticated musical facts, such as harmonic sounds, ornaments, performance directives, etc.

Concerning the querying process, we plan to conduct a more detailed theoretical study of the cost of querying, as advanced results of the literature concerning the cost of graph pattern queries exhibited several classes of tractable queries (e.g., patterns

that do not contain a cycle (Barceló, 2013)). These classes should be mapped into musical patterns.

Finally, this representation could be combined with previous results for managing data quality in graph databases (Pivert *et al.*, 2020; Rigaux, Thion, 2017), in order to detect data quality problems in the music score content (Foscarin *et al.*, 2021). And finally, in terms of implementation, the MUSYPHER tool is still under development. A current short-term development consists in integrating MUSYPHER in the NEUMA platform, in the form of a module allowing the graph-based querying of the available collections.

Acknowledgements

The authors thank Clément Van Straaten for his work on the MUSYPHER *tool.*

References

Angles R. (2012). A comparison of current graph database models. In *Proc. of the intl. conf. on data engineering (icde) workshops*, p. 171-177.

Angles R., Arenas M., Barceló P., Hogan A., Reutter J. L., Vrgoc D. (2017). Foundations of modern query languages for graph databases. *ACM Comput. Surv.*, Vol. 50, No. 5, pp. 68:1–68:40.

Angles R., Gutierrez C. (2008). Survey of graph database models. *ACM Computing Surveys (CSUR)*, Vol. 40, No. 1, pp. 1–39.

Barceló P. (2013). Querying graph databases. In *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, pp. 175–188.

Barceló P., Libkin L., Reutter J. L. (2014). Querying regular graph patterns. *J. ACM*, Vol. 61, No. 1, pp. 8:1–8:54.

BnF. (2022). *Bibliothèque numérique de la BnF*. https://gallica.bnf.fr/. (Accessed Feb. 2022)

Bonifati A., Fletcher G. H. L., Voigt H., Yakovets N. (2018). *Querying graphs*. Morgan & Claypool Publishers.

Foscarin F., Rigaux P., Thion V. (2021). Data Quality Assessment in Digital Score Libraries. The GioQoso Project. *Intl. Journal on Digital Libraries*, Vol. 22, No. 2, pp. 159-173.

Fournier-S'niehotta R., Rigaux P., Travers N. (2018). Modeling Music as Synchronized Time Series: Application to Music Score Collections. *Information Systems*, Vol. 73, pp. 35–49.

Ganseman J., Scheunders P., D'haes W. (2008). Using XQuery on MusicXML databases for musicological analysis. In *Proc. of the intl. conf. on music information retrieval, (ismir)*, pp. 433–438.

Good M. (2001). The virtual score: Representation, retrieval, restoration. In, p. 113-124. W. B. Hewlett and E. Selfridge-Field, MIT Press.

Haydn J. (1797). *Das lied der deutschen.* (Lyrics by August Heinrich Hoffmann von Fallersleben)

Humdrum. (2022). *Representing Music Using **kern.* https://www.humdrum.org/guide/ch02/. (Accessed Feb. 2022)

IMSLP. (2022). *Intl. Music Score Library Project.* https://imslp.org. (Accessed Feb. 2022)

Jin R., Raphael C. (2015). Graph-based rhythm interpretation. In *Proc. of the intl. society for music information retrieval conf. (ismir)*, pp. 343–349.

KernScores. (2022). http://kern.ccarh.org/. (Accessed Feb. 2022)

Libkin L., Martens W., Vrgoč D. (2013). Querying Graph Databases with XPath. In *Proceedings of the Intl. Conf. on Database Theory ICDT*, pp. 129–140.

Music Encoding Initiative. (2022). http://www.music-encoding.org. (Accessed Feb. 2022)

MusicXML. (2022). http://www.musicxml.org. (Accessed Feb. 2022)

Neo Technology. (2022). *The Neo4j Manual.* https://neo4j.com/developer. (Accessed Feb. 2022)

Neo4j web site. (2022). www.neo4j.org. (Accessed Feb. 2022)

NEUMA. (2022). http://neuma.huma-num.fr. (Accessed Feb. 2022)

Pivert O., Scholly E., Smits G., Thion V. (2020). Fuzzy quality-aware queries to graph databases. *Information Sciences*, Vol. 521, pp. 160–173.

Pugin L., Zitellini R., Roland P. (2014). Verovio: A library for Engraving MEI Music Notation into SVG. In *Proc. of the Intl. Society for Music Information Retrieval (ISMIR)*, pp. 107–112.

Rigaux P., Abrouk L., Audéon H., Cullot N., Davy-Rigaux C., Faget Z. *et al*. (2012). The design and implementation of neuma, a collaborative digital scores library - requirements, architecture, and models. *Intl. Journal on Digital Libraries*, Vol. 12, No. 2-3, pp. 73–88.

Rigaux P., Thion V. (2017). Quality Awareness over Graph Pattern Queries. In *Proc. of the Intl. Database Eng. & Applications Symp. (IDEAS).*

Rolland P. (2002). The Music Encoding Initiative (MEI). In *Proc. of the Intl. Conf. on Musical Applications Using XML*, p. 55-59.

Webber J., Robinson I., Eifrem E. (2013). *Graph databases*. O'Reilly Media.

Wood P. T. (2012). Query languages for graph databases. *SIGMOD Rec.*, Vol. 41, No. 1, pp. 50-60.

Zhu T., Fournier-S'niehotta R., Rigaux P., Travers N. (2022). A Framework for Content-based Search in Large Music Collections. *Big Data and Cognitive Computing*, Vol. 23, No. 6.