

MANIPULATING AUDIO INTO A DBMS

RANIA LUTFI, JOSE MARTINEZ AND MARC GELGON

*Institut de Recherche en Informatique de Nantes (IRIN / CID / BaDRI)
Ecole polytechnique de l'université de Nantes
Site de la Chantrerie – Bât. IRESTE – B.P. 50609 – 44306 Nantes Cedex 3
France
E-mail: {firstname.surname}@irin.univ-nantes.fr*

To be fully usable, digital multimedia contents should be supported by a set of tools to query them, and more generally to manipulate them. This is one of the major goals of an audio database management system (DBMS). Existing work related to audio documents, e.g., radio or television archives, generally tackles the signal processing aspects, often leaving the DBMS question open, or focusing on a dedicated audio DBMS. In this paper, we lay the foundations for integrating audio into a general purpose DBMS, in the form of an audio abstract data type. We introduce its properties and associated operators. The practical interest and some technical difficulties are underlined. In particular, through a few query examples, we examine the problem of the complexity of the querying expressions (and of time complexity).

1 Introduction

The increasing production of digital audio-visual content should be supported by a set of technologies permitting an adequate usage. This paper focuses on effective access to audio content, which implies some significant difficulties calling for contributions in the fields of databases and of content analysis, because audio has little, if any, structure.

In point of fact, multimedia data is always queried *via meta-data*¹⁵, the only difference with the usual terminology being that part of this meta-data is extracted from the content itself and represents it, with inherent estimates and inaccuracies due to the used techniques. MPEG-7 proposes a classification of meta-data starting from descriptions of the contents (coding, duration, used focal, place and date of recording ...), physicals attributes, perceptual attributes, transcriptions, syntactical structures, and annotations^{13,14}.

End-user access to the contents of multimedia documents is considerably facilitated by the availability of meta-data directly matching the (i) search, (ii) comparison, and/or (iii) visualization criteria. Consequently, it is useful for this purpose to possess, e.g., the temporal partitions of audio documents into types of contents (speech, music, noise ...), or into different speakers. A preliminary classification of documents, into the same musical genre for instance, makes

navigation easier. Finally, the forms of visualization appear limited for audio. Yet, a textual transcription, even incomplete, of the spoken contents is to be the most useful to the end-user.

In the case of audio, extracting meta-data from the raw signal is a difficult and widely investigated research field^{4,11} where numerous questions remain open, with major applicative relevance. Indeed, as the stake seems considerable, numerous laboratories and media companies have undertaken research work towards making their audio or audio-visual contents easier to query, or to browse through^{8,16}. Algorithms were demonstrated, which provide encouraging results in terms of quality of the obtained results, and current hardware enables their generally heavy computation in a reasonable amount of time.

However, these works do not address the database issue. Yet, provision of elaborated services that include access to the audio content requires information systems that correctly exploit the nature of the audio documents and their meta-data. They should completely integrate these resources into a system allowing storage, indexing, and general querying services.

To our knowledge, audio data integration into a DBMS has been essentially conducted through meta-data modeling^{9,10,12,13,14}. More particularly, commercial DBMSs offer minimal integration. DB2 V6 can store and retrieve image, audio, and video data but only images can be actually queried, with QBIC⁷. Audio characteristics are limited to format descriptions (duration, size, channels, filename ...)^a. The same is true for ORACLE 8i InterMedia. Text, geographical data, and images can be queried (with Virage³ for images), but neither audio, nor video.

However, in an actual multimedia DBMS, audio should be not only queried by content, but also manipulated and modified at will. Signal processing techniques are certainly outside the scope of a DBMS. In contrast, the concepts of queries and views already call for audio restructuring, in a way similar to string processing. Therefore, in this paper we deal with the integration of audio data into a DBMS as an abstract data type along with its main properties.

The paper is organized as follows: First, section 2 outlines the problem and the state-of-the-art regarding the extraction of meta-data from audio documents, focusing on spoken content. In spite of the difficulties, analysis methods provide sufficient results to be exploited into a DBMS, i.e., in a non-specialized context. Then, section 3 describes a proposal towards representing, querying, and manipulating audio data, at a very high level of abstraction. The proposal described in this paper is restricted to queries (comparative¹⁹ and browsing¹⁷ aspects are not considered here). Next, section 4 introduces some examples of queries that require audio data restructuring in addition to meta-data querying. Finally, the conclusion lists additional points that seem essential during the development of a DBMS for integrating totally audio data.

^a See <http://itsuite.it.bton.ac.uk/db2/dmba6/dmba6mst.html>.

2 Extraction of Meta-data from Audio Content

Algorithms for extracting meta-data produce three kinds of information, which we shall name *features*, *descriptors*, and *structures*. *Features* are low-level information, obtained by selection and transformation of the raw signal. They translate it into a much more compact structure. Besides, they usually exhibit good properties for pattern recognition. *Descriptors* already correspond to qualitative, interpretation-level information. For instance, they include labeling of a given piece of audio data as being music or speech, or spoken by a given person. Finally, *structures* are organizational information: temporal extents exhibiting an homogeneous property, in the case of audio. The segmentation process may remain somewhat independent from the descriptors. The remainder of this section aims at giving the reader a few general landmarks in the issue of extracting meta-data by audio content analysis.

2.1 Some Features

Local features are computed on a short temporal window of the signal (in the order of 25 milliseconds), where the signal is supposed stationary. As an illustration, let us briefly describe some usual features for speech analysis, namely *pitch*, *cepstral coefficients*, and *zero-crossing rate*:

- Production of sound by human beings can be modeled as the filtering (convolution) of a source (vocal cords) by the vocal track. *Pitch* analysis aims notably to find, from the produced sound, the fundamental frequency of the source. Reliable determination of this frequency is an arduous task, yet it characterizes well the speaker. It can also discriminate effectively between man, woman, child, and noise.
- Another very frequently used attribute is the set of *cepstral coefficients* (MFCC for *Mel-scaled Frequency Cepstral Coefficients*)¹¹, which are also related to this source/filter deconvolution and, thanks to their robustness, are very useful for transcription as well as for speaker recognition.
- In contrast, *zero-crossing rate* (ZCR) is a simple statistics measuring the number of zero-level crossings of the signal on the temporal axis per time unit. It reflects the energy distribution of this signal along the frequency axis. Its temporal variations is a good cue to distinguishing between speech (which alternates high and low ZCR) and music (characterized by a less varying ZCR). Let us note that it is an example of a feature that one can apply not only to local windows, but also to global segments, including the recording in its entirety.

2.2 Constructing Descriptors from Features

The assignment of descriptors to document segments, given the acoustic features computed within these segments, reduces mostly to pattern recognition problems,

often dealt with using statistical or neuronal approaches, or techniques based on information theory. Difficulties in this task are numerous and, moreover, combined, as exemplified below.

One often faces ill-posed inverse problems (as recognition of a spoken word or segmentation into different speakers), for which the use of contextual knowledge is useful. On the one hand, one could introduce *a priori* knowledge on the likelihood of the realization of words and word sequences, supplied by a language model, learned beforehand. As a result, though their acoustic realizations are similar, “recognize speech” will be clearly preferred to the meaningless “wreck a nice beach.” On the other hand, a classical technique consists in introducing a Bayesian prior encouraging simplicity of the segmentation, say into speakers. The search for the optimal trade-off between these two objectives is generally conducted with hidden Markov models and the associated estimation algorithms.

One frequently encounters another issue when dealing with documents that contain mixtures of entities that one wishes to extract and describe (e.g., a radio program containing several speakers and music), and neither the temporal segmentation, nor the model of the speakers are known in advance. This is the well-known interwoven problem of associating data (i.e., features computed in local windows) to models (e.g., speaker-homogeneous segments), and estimating these models (e.g., acoustic speaker models), given that features from a single local window is hardly enough data for a reliable model estimate. In this case, the three problems of (i) learning the models¹¹, (ii) assigning the data to these models, and (iii) determining the number of models are interdependent, and finding a globally optimal solution to the joint problem is usually very involved.

If this last example is the most general and difficult task, more or less strong assumptions can be made on the contents, which make the task of meta-data extraction easier. Restriction from the complete vocabulary of a language to a reduced vocabulary (say, a dozen of words), knowledge of the acoustic models of the speakers, or of how many speakers there are, or of which language they use, naturally eases the analysis task. Besides, the acoustic quality of the recording and the degree of articulation of the speech strongly affects the quality of the results. For the transcription problem (audio to text), an error rate of 20% for recordings made in a controlled and articulated environment, but of 85% for ordinary advertisements, has been reported¹⁸.

Further, indexing large volumes of audio documents motivated analysis at a higher level, including determining the topic of a spoken conversation, detecting a conversation pertaining to a topic supplied as a query, topic tracking, and so on.

3 Audio Recordings

The previous section showed briefly that the analysis of audio documents leads to descriptions that are complex, semi-structured, and may contain fuzzy information. Low-level attributes seem well-mastered²¹, whereas semantic information is a further endeavor. Nevertheless, even if the results provided by automatic analysis schemes are not error-free, we can rely on them to introduce the main concepts that a DBMS must possess to exploit sound recordings.

In this section, we introduce three concepts: L , the base recordings; T , the complex recordings; and very shortly M , the meta-data. Loosely speaking, L corresponds to audio files, and audio-recognition techniques presented in section 2 are applied on its instances. Structural meta-data consist of a temporal segmentation. Each segment is characterized by a complex set of descriptors and features. However, we simplify them into a set of predicates that allow querying. Next, L instances are translated into T instances, the data type that is made visible to the database user. In particular, it is possible to query complex recordings through the meta-data that have been associated to their underlying logical recordings. Moreover, in order to deal with complex queries and views, i.e., audio restructuring rather than mere selections, we introduce *temporal projections* and *concatenations*.

3.1 Logical Recordings

The core concept is that of sound recording as a coherent logical unit. Notice that this notion remains vague. For instance, the whole soundtrack of a movie can be chosen as a logical recording unit; another choice could have been that each scene be a logical unit. We do not discuss here the limits, but state only the existence of this concept, which we denote L (for *logical*^b). The rationale is that this logical unit is the one that is really indexed into the database.

Loosely speaking, indexing an audio document consists in segmenting it along the temporal axis, according to some homogeneity criterion. Segments obtained at a given level can be further refined, according to the available tools at the time indexing occurs. In this way, we obtain a hierarchical segmentation, the levels of which are generally strongly typed, hence limited in depth (m_3 and m_4 are part of the refinement of m_2 in Figure 1). There may be some overlapping, should parts of segments be simultaneously classified into several categories, e.g., a segment containing superimposed voice and music (as m_1 and m_3 in Figure 1).

^b We prefer “logical” to “physical” because the implementation requires an additional level to deal with large files that are truncated into several pieces.

Figure 1 illustrates several levels of understanding of a logical recording, a French radio ^c program named “2000 ans d’Histoire” ^d. A rough segmentation separates the musical zones (credits at the beginning and at the end, and musical breaks) from the actual content, in which a guest talks, possibly along with pre-recorded interviews (this corresponds to the *syntactical structure*).

Still at a very unrefined level, we can differentiate between male and female voices. But, according to the knowledge already accumulated in the database, persons (as well as music) can be identified (resp. *descriptors*). This identification can remain anonymous if no name has yet been associated (which is the case of the presenter, matched to other occurrences found in other broadcasts of this daily program). At a finer level, a system could index all or some of the pronounced words (resp. *transcription*).

On the example, we can note the presence of structured attributes too (as of the database meaning). If some can be relatively general, as the name of famous people ^e, most will remain close to a specific application, as the first date of broadcasting of a radio recording or the titles of persons (resp. *annotations*). Therefore, it is advisable to establish a clear distinction between meta-indexing, which is bound to the audio signal, and the additional, often semantic information. Ideally, we want to build a common database sub-schema for audio data that any audio-aware database application can rely on, the application being able to remove ambiguities, and even errors in the audio part.

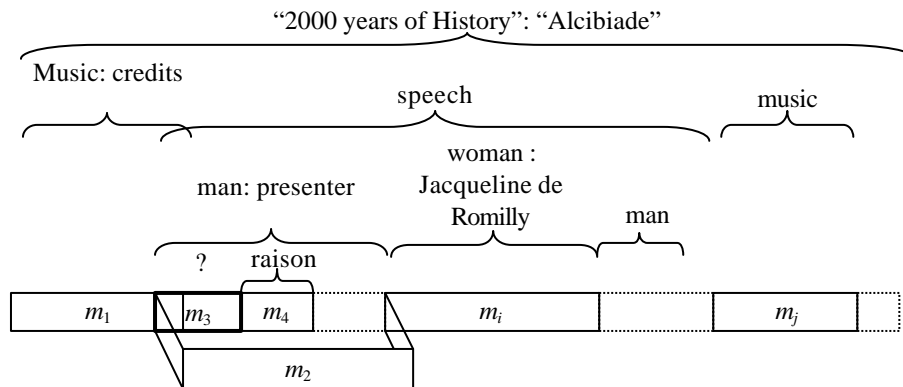


Figure 1. An example of temporal segmentation

^c France Inter.

^d “2000 years of History.”

^e Jacqueline de Romilly is a Hellenistic authority, and a member of the *Académie française*.

Indexing audio data generates the three classes of meta-data introduced in section 2: structure, descriptors, and features. Formally, the segmentation of the audio track is a function of L that associates to each base recording a set of meta-data for a given part of the track:

$$\begin{aligned} \text{segments : } L &\rightarrow 2^{\mathbb{R}^+ \times \mathbb{R}^+ \times M} \\ l &\rightarrow \{ ([s, e[, m) \} \end{aligned} \quad (1)$$

where the time intervals $[s, e[$ describe the temporal structure of the recording, whereas M represents the set of meta-data information that can be associated to a temporal segment: descriptors (music, speech, man/woman ...), and features (ZCR ...).

In practice, time intervals are never empty. By convention, they are always closed to the left and opened to the right, as underlined by the notation, in order to avoid point intervals when using intersections (instant modeling seems irrelevant in practice).

Descriptors and features lead to an arbitrary set of functions and predicates because they depend on the available audio analysis tools. MPEG-7 itself allows the definition of one's own descriptions. These can become M . For instance, we shall retain, in accordance with section 2, *music(m)*, *speech(m)*, *male(m)*, *samevoice(m₁, m₂)*^g ... Then, the example of Figure 1 can be translated formally into *segments(FI_2000Ans_Alci) = {([0, 12[, m₁), ([10, 1 000[, m₂), ([10, 35[, m₃), ([11, 13[, m₄), ...}* where satisfied predicates are *music(m₁)*, *speech(m₂)* (and in turn *speech(m₃)*, *speech(m₄)* ...), *male(m₃)*, *word(m₄)*, etc., and where some transcriptions could be *transcription(m₄) = "raison"*, etc.^h

3.2 Complex Recordings

Complex recordings must be built over basic ones. In point of fact, if base recordings support indexing, and allow for querying based on mere selections, more complex queries and views often require the derivation of new audio data values. The solution consists essentially in defining a new data type, T (for *track*), supplying operators to manipulate it, and defining the derived segmentation function for querying.

^f s stands for *start*, whereas e stands for *end*.

^g The last binary predicate is an example of incorporating indirectly comparison techniques into a query language.

^h Technically, *music* and *speech* can be implemented as classes into an object-oriented model, whereas *male* can become an attribute of class *speech*.

3.2.1 Definition

A priori, \mathbb{T} is polymorph, being bound to any algebraic expression producing recordings. Nevertheless, it is possible to propose a normal form as a concatenation of excerpts:

$$\mathbb{T} = (\mathbb{L} \times |\mathbb{R}^+ \times |\mathbb{R}^{+*}|^{\mathbb{N}} \quad (2)$$

the values of which are denoted $\langle l_i, [s_i, e_i] \rangle_{i=0 \text{ to } n}$ with $e_i > s_i$ and $\forall i > 0, l_{i-1} = l_i \Rightarrow e_{i-1} < s_i$, i.e., intervals are never empty, and consecutive elements are either related to different base recordings, or to non contiguous parts of the same base recording.

The duration of a complex recording is naturally computed as the sum of the durations of its excerpts: $|t| = \sum_{i=0 \text{ to } n} (e_i - s_i)$.

Let us dwell upon the difference between a *segment* and an *excerpt*. Both represent contiguous parts of a recording, but segments are related to base recordings and the associated information is meta-data, whereas excerpts are related to complex recordings and the associated information is a base recording. In the sequel, we shall investigate segments of excerpts (See Figure 2).

3.2.2 Operators

Manipulating \mathbb{T} instances is achieved thanks to two operators, which are directly related to the normal form: *concatenation* and *temporal projection*.

Concatenation is a binary operator, denoted “.”, and used in infix notation:

$$_ . _ : \mathbb{T} \times \mathbb{T} \rightarrow \mathbb{T} \quad (3)$$

such that:

- $\forall 0 \leq i < n, (t.t')_i = t_i$ and
- either $(t.t')_n = (l, [s, e]) \wedge \forall 1 \leq i \leq n', (t.t')_{n+i} = t'_i$ if $t_n = (l, [s, k])$ and $t'_0 = (l, [k, e])$,
- or $(t.t')_n = t_n \wedge \forall 0 \leq i \leq n', (t.t')_{n+i+1} = t'_i$

i.e., it is a usual concatenation except when the last segment of the first sequence can be merged with the first segment of the second sequenceⁱ.

The temporal projection, denoted $\Pi_{[s, e]}(t)$ by analogy with the relational algebra, is defined as follows:

$$\mathbb{T} \times |\mathbb{R}^+ \times |\mathbb{R}^{+*}| \rightarrow \mathbb{T} \quad (4)$$

ⁱ Note that indice-based accesses are used just to supply a denotational specification of the operation; they are not authorised as –visible– \mathbb{T} operators. This kind of specification loosely imposes an implementation, though a largely acceptable one here, which is the normal form introduced in definition (2). Much more abstract specifications, and freedom of implementation, can be obtained through axiomatic specifications. The main axiom, that mixes concatenation and temporal projection would be: $\Pi_{[0, |t_1|]}(t_1.t_2) = t_1 \wedge \Pi_{[|t_1|, |t_2|]}(t_1.t_2) = t_2$.

$$\Pi: \mathbb{T} \times \mathbb{R}^+ \times \mathbb{R}^{+*} \rightarrow \mathbb{T} \quad (4)$$

$$\Pi: \begin{cases} \text{with:} \\ \Delta_0 = 0 \\ i-1 \\ \Delta_i = \sum_{j=0}^{i-1} e_j - s_j \end{cases} (t, s, e) \rightarrow \langle \langle l_i, [\max\{s_i, s - \Delta_i\}, \min\{e_i, e - \Delta_i\}] \rangle \mid [s_i + \Delta_i, e_i + \Delta_i] \cap [s, e] \neq \emptyset \rangle$$

where the semantics of the used notation is the transposition to sequences of the similar notation for sets: elements are selected and possibly concatenated in the increasing order of the indices.

Generally, queries are expressed with set-oriented languages. Therefore, we generalize this restriction operator to a set of intervals I , we denote it $\Pi_I(t)$, and we define it as follows:

$$\mathbb{T} \times 2^{\mathbb{R}^+ \times \mathbb{R}^{+*}} \rightarrow \mathbb{T}$$

$$(t, I) \rightarrow \cdot \Pi_{[s, e]}(t)$$

$$\forall [s, e] \in \cup_{i \in I}$$

where (i) the union of intervals returns a set of pair-wise disjoint intervals, and (ii) the concatenations are done in the increasing order of the intervals ^{*j*}. (The concatenation operator could have been generalized in the same way, but we do not use it in the sequel.)

3.2.3 Properties of the Operators

Unfortunately, from the query optimizer point of view, (\mathbb{T}, \cdot, Π) has poor algebraic properties. First, (\mathbb{T}, Π) offers the properties of neutral element(s) ($\Pi_{[0, e]}(t) = t$, $\forall e \geq |t|$ where $|t| = \sum_{i=0 \text{ to } n} (e_i - s_i)$) and of reducibility ($\Pi_{[s_2, e_2]}(\Pi_{[s_1, e_1]}(t)) = \Pi_{[s_1 + s_2, e_1 + s_2]}(t)$). Next, (\mathbb{T}, \cdot) is a monoid: neutral element $(t \cdot \diamond = \diamond \cdot t = t)$, and associativity $(t_1 \cdot (t_2 \cdot t_3) = (t_1 \cdot t_2) \cdot t_3)$. Finally, for (\mathbb{T}, \cdot, Π) we have a form of distributivity ($\Pi_{[s, e]}(t_1 \cdot t_2) = t_1 \cdot \Pi_{[s, e]}(t_2)$ if $s \geq |t_1|$, $\Pi_{[s, e]}(t_1) \cdot t_2$ if $e \leq |t_2|$, and $\Pi_{[s, |t_1|]}(t_1) \cdot \Pi_{[0, e - s]}(t_2)$ in the general case), which allows to write down all the expressions under the proposed normal form of definition (2).

Moreover, the expressive power of these operators is not that high; they can be rewritten using the relational algebra. To demonstrate it, it is necessary to translate \mathbb{T} into a relation, $\mathbb{T}(id, i, l, s, e)$ where id is an arbitrary identifier to distinguish the instance values of \mathbb{T} , i is an indice to translate explicitly the notion of order in a

^{*j*} This double condition can be satisfied by firstly sorting the intervals, and then merging consecutive and non-disjoint intervals. The complexity is in $O(|I| \cdot \log_2 |I|)$, but it can remain linear if the implementation maintains the intervals in increasing order.

sequence, both forming the key of the relation, and l , s , and e are the actual information.

Concatenation is the simplest operator to simulate. This consists in obtaining the maximal indice n of the first recording, say id_1 , and to compute the union with the information of the second recording, say id_2 , by adding $n + 1$ to the values of the corresponding indices. Formally, we can write:

$$\pi_{i, l, s, e}(\sigma_{id=id_1}(\mathbf{T})) \cup \pi_{i+n+1, l, s, e}(\mathbf{ID}_2 \times \rho_{i \rightarrow n}(\max(\pi_i(\mathbf{ID}_2))))^k$$

where the second member of the Cartesian product is the singleton $\{n\}$, and \mathbf{ID}_2 is an abbreviation for $\sigma_{id=id_2}(\mathbf{T})$. (If the result has to be stored, it is necessary to generate a new identifier.)

The temporal projection on $[start, end[$ introduces two difficulties: the calculation of the duration of a prefix of the recording, and the renumbering of the indices of the result. Both problems are similar. Omitting the second, simpler problem, we have to use an extended version of the relational algebra^l in order to obtain the Δ 's values:

$$\mathbf{T}' = \pi_{id, i, \Delta}(\sum_{\pi_{e-s}}(\sigma_{id=id' \wedge pred < i \rho_{id, i \rightarrow id', pred}(\mathbf{T}))), l, s, e(\mathbf{T}))$$

and then applying the definition (4):

$$\pi_{id, i, l, \max(s, start - \Delta), \min(e, end - \Delta)}(\sigma_{start < e + \Delta \wedge end > s + \Delta}(\mathbf{T}'))$$

In spite of these shortcomings, \mathbf{T} is far from unuseful. The obvious advantage of an "atomic" data type \mathbf{T} (like strings in SQL), and of its dedicated operators, is to avoid (i) these tricky manipulations, (ii) the creation of a relation shared by all the applications using audio recordings, as well as (iii) the management of transient recordings in this stored relation.

3.2.4 Extending the Meta-index

\mathbf{T} is the audio data type used by the DBMS users. Therefore, the indexing facilities of \mathbf{L} have to be extended to \mathbf{T} . For that purpose, each instance of \mathbf{L} can be implicitly translated into a sequence consisting of only itself. Formally, we write: $l \in \mathbf{L} \Leftrightarrow \langle l, [0, |l|] \rangle \in \mathbf{T}$. One can verify immediately that $\forall l \in \mathbf{L}, |l| = |\langle l, [0, |l|] \rangle|$. The difficulty lies in the *segments* function.

Indexing complex recordings is based on the meta-data of the logical recordings that constitute them. But, the notion of excerpt introduces a difficulty: its time interval $[s, e[$ on a logical recording l can have a *strict* intersection with several segments of l (See Figure 2).

^k or $\pi_{i, l, s, e}(\sigma_{id=id_1}(\mathbf{T})) \cup \pi_{i+n+1, l, s, e}(\mathbf{ID}_2 \times \rho_{i \rightarrow n}(\pi_i(\mathbf{ID}_2 \setminus (\pi_i(\mathbf{ID}_2 \bowtie_{i=succ-1} \rho_{i \rightarrow succ}(\mathbf{ID}_2))))))$ in order to avoid aggregation operators. ($\rho_{a_1, \dots, a_n \rightarrow b_1, \dots, b_n}$ is the renaming operator, mandatory in the named relational algebra^l, but often implicit.)

^l Aggregate functions can be totally avoided if \mathbf{T}' is stored in place of \mathbf{T} , the concatenation operation is modified accordingly, and we do not impose indices to form a prefix of \mathbf{N} .

Several solutions are possible. The simplest one consists in ignoring the truncated segments. Conversely, the second solution consists in assigning the information of the whole segment to a strict part of it. Of course, this variant remains imperfect because the information about the sub-segment is only an estimate. We can control the quality of the assignment by defining a threshold, $0 < \tau < 1$: the more important the overlap, the more relevant the meta-data.

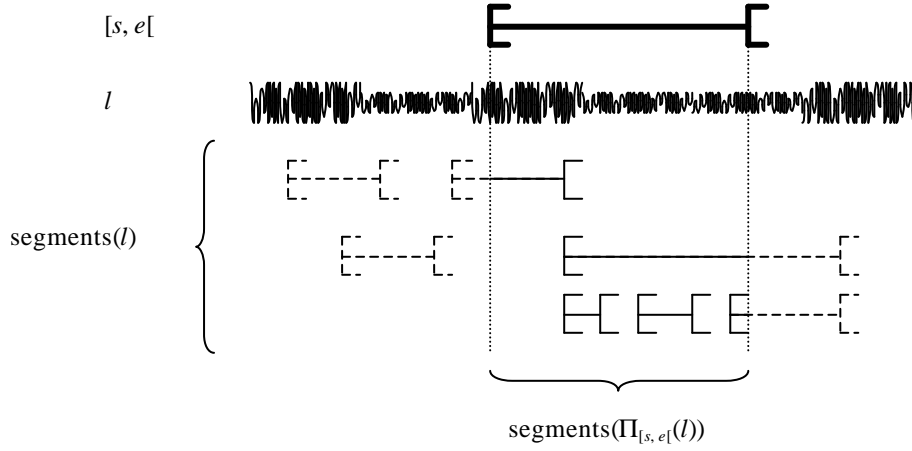


Figure 2. Segments of an excerpt

We extend the definition of the *segments* function from the type L (1) to the type T in the following way^m:

$$\begin{aligned}
 \mathbf{T} &\rightarrow 2^{\mathbb{R}^+ \times \mathbb{R}^{+*} \times M} \\
 &\{ ([s + \Delta_i, e + \Delta_i], m) \mid \\
 &([s, e], m) \in \text{segments}(l_i) \wedge \\
 &[s, e] \subseteq [s_i, e_i] \} \\
 \text{segments: } t &\rightarrow \bigcup_{i=0}^n \{ ([\max\{s, s_i\} + \Delta_i, \min\{e, e_i\} + \Delta_i], m) \mid \\
 &([s, e], m) \in \text{segments}(l_i) \wedge \\
 &[s, e] \cap [s_i, e_i] \subset [s_i, e_i] \wedge \\
 &(\min\{e_i, e\} - \max\{s_i, s\}) / (e - s) \geq \tau \} \tag{6}
 \end{aligned}$$

We leave to the reader the task of verifying that this definition achieves a generalization, i.e., $\forall l \in \mathbf{L}, \text{segments}(l) = \text{segments}(\langle l, [0, |l|] \rangle)$.

^m It suffices to remove the inequality containing τ to obtain the definition for the second solution, and the whole second term of the inner union in order to obtain the first solution.

4 Querying a Database with Audio

Querying an audio database requires both the predicates on the meta-data M and the operators on T seen above. Let us dwell upon the fact that intervals, and more generally sets of intervals, are so present in the modeling of the recordings (See function (5) especially) that we need set operators on them too: union, intersection, and difference. Besides, one can use Allen's binary predicates: before, meet, overlap, start, during, finish and their inverse, as well as equal², the definitions of which are illustrated in Figure 3¹¹.

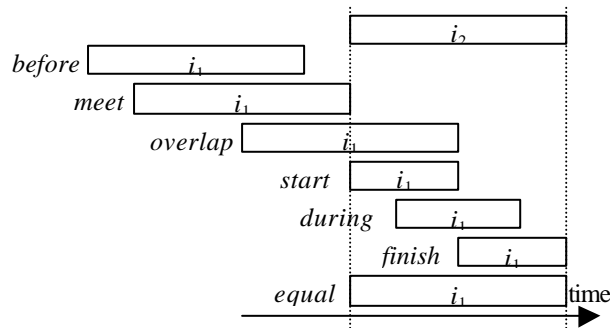


Figure 3. Allen's temporal relationships between two intervals

We are eventually ready to answering queries that require to build recordings. Let us take as an example a single relation on a radio archive, the schema of which is `Broadcasts(program: string, radio: string, date: Date, hour: Time, track: T)`. Note carefully that this relation is *not* in first normal form, since T contains complex information. We could have normalized it by creating a relation for segments. In contrast, we prefer to rely on an extended form of the relational algebra.

As a first example, let us solve the query consisting in finding “the list of all the programs of France Inter on March 21, 2001, with their time of broadcasting and the first two recorded minutes, but music.” For the structured part, we can write:

$$\pi_{\text{program, hour, introduction}}(\sigma_{\text{radio} = \text{"France Inter"} \wedge \text{date} = 21/03/2001}(\text{Broadcasts}))$$

where the construction of the *introduction* value requires all of the proposed extensions:

$$\Pi_{[0, 2 \times 60]}(\Pi_{\{[0, | \text{track} |]\} \setminus \pi_{[s, e]}(\sigma_{\text{music}(i)}(\text{segments}(\text{track})))}(\text{track}))$$

since we can split it up into:

¹¹ Notice that this kind of interval is different from the `Interval` data type provided by SQL.

- (i) determining all the time intervals of the musical parts: $\pi_{[s, e]}(\sigma_{\text{music}(i)}(\text{segments}(\text{track})))$, by using the predicates on the meta-data,
- (ii) subtracting them from the interval corresponding to the complete track, by using the operations on the intervals: $\{[0, |\text{track}|]\} \setminus \pi_{[s, e]}(\sigma_{\text{music}(i)}(\text{segments}(\text{track})))$,
- (iii) using the obtained set intervals to execute temporal projections and concatenations, i.e., the operators on T via (5), the generalized temporal projection,
- (iv) and ending up by a last temporal projection to obtain the “first two minutes, but music.”

As a second example, assuming that the first speaker is the presenter, let us select “programs presented by a woman.” The spoken segments, for a given recording, are:

$$S = \sigma_{\text{speech}(i)}(\text{segments}(\text{track}))$$

and the first segment spoken by a woman is the singleton^o or the empty set given by:

$$F_1 = \sigma_{\text{female}(i)}(S \setminus \pi_{[s, e]}(\rho_{[s, e], i \rightarrow [s', e'], i}(S)))$$

which finally gives:

$$\pi_{\text{program, radio}}(\sigma_{F_1 \neq \emptyset}(\text{Broadcasts}))$$

These typical examples raise the problem of the complexity of the query expression. From this point of view, OQL (*Object Query Language*), the standard language for object-oriented DBMSs⁵, is preferable to SQL^p. Indeed, it accepts straightforwardly inter-weaved expressions. It also incorporates the generic ‘list’ type constructor, thereby allowing indice-based expressions^q. Nevertheless, OQL remains insufficient for computing directly, i.e., without providing the proposed operators, the “first two minutes” in the first query, or the “ten seconds before a laugh.”

Temporal databases²⁰ do not offer a directly transposable solution, in spite of the shared concepts (intervals, sets and sequences of intervals, most notably). The essential difference is that audio is *intrinsically* a temporal element. Let us point out that the temporal projection that we defined in definition (4) has not the semantics of temporal DBMSs. Its effects on the associated segments (See Figure 2) are closer to this semantics.

^o In point of fact, we can obtain a set of segments when many such begin at the very same time, usually at different levels of description (See Figure 2), but also, though rarely, if different persons are synchronously speaking.

^p More generally, we advocate an implementation on top of an object-oriented DBMS rather than an object-relational one⁶.

^q For instance, it is possible to avoid the F_1 pseudo-boolean in the second query by sorting S on the beginning of the intervals. Then, we can write just $\pi_{\text{program, radio}}(\sigma_{\text{female}(\text{segments}(\text{track})\uparrow[s]\downarrow[0], i)}(\text{Broadcasts}))$. We can expect the optimizer to be able to find the minimal value s without actually sorting the data, and even being informed that the data is kept sorted as suggested previously, which leads to a complexity in only $O(1)$ for the condition part.

Also, the problem of efficiency becomes more serious as audio analysis grows richer, i.e., associates an increasing number of segments to a recording. The use of a generalization of quota queries (which return only the n first answers), and of lazy evaluation is highly desirable.

5 Conclusion and Future Work

Having outlined a few important points of audio signal analysis for the purpose of high-level interpretation, we proposed a way to model audio data in the form of a data type, associated properties, and operators to manipulate the values. Briefly, audio analysis is applied on audio files and the extracted meta-data is stored into the database. In contrast, inside the database, audio content is accessed to and manipulated through a more complex encapsulating data type.

With respect to meta-data, and still at a very high level of abstraction, audio meta-data is based on a temporal segmentation, each segment being queryable with a variable number of functions and predicates, according to the capabilities of the signal analysis tools at hand. This level can incorporate the efforts of several proposals, and more particularly of MPEG-7.

Then, a minimal algebra enables the effective manipulation of complex audio objects, roughly speaking similar to character strings. Thanks to these operators, we can mix queries that use the structured part of the database (i.e., classes, relations ...) as well as the audio components. Also, they extend the capabilities of the standard query languages without being too costly at run-time.

We are currently implementing and validating these ideas, based on structures and descriptors obtained from MFCC features. From the software engineering point of view, the proposed extensions are not that easy to implement and deserve to be offered to all the audio-aware applications. Let us note that current DBMS architectures are able to integrate such modules, e.g., Informix datablades, DB2 extenders ... Also, the performance issue is important and we have already envisaged alternative solutions in the object-oriented design schema that follows the study of this paper.

Among the possible extensions, let us mention the handling of uncertainty by probabilistic modeling, the extraction of high-level interpretation descriptors (for instance, recognition of dialogues, monologues, debates, musical broadcasts, comedy programs ...) and, naturally, the integration of the two complementary ways to search for multimedia contents: comparative search, and navigation into the audio database.

References

- 1 Abiteboul, S., Hull, R. and Vianu, V., *Foundations of Databases*. (Addison-Wesley, 1995).
- 2 Allen, J. F., Maintaining Knowledge About Temporal Intervals. *Communications of the ACM* **26**, 11 (1983) pp. 832-845.
- 3 Bach, J. R., Fuller, C., Gupta, A., Hampapur, A., Horowitz, B., Humphrey, R., Jain, R. C. and Shu, C., Virage Image Search Engine: An Open Framework for Image Management. *Proc. of the IS&T/SPIE Int'l Symposium on Electronic Imaging: Science and Technology, Storage & Retrieval for Image and Video Databases IV* (1996) pp. 76-87.
- 4 Campbell, J. P., Speaker Recognition: A Tutorial. *Proceedings of the IEEE* **85**, 9 (1997) pp. 1437-1462.
- 5 Cattell, R. G. G., Barry, D. K., Bartels, D., Berler, M., Eastman, J., Gamerman, S., Jordan, D., Springer, Strickland, H. and Wade, D., *The Object Database Standard: ODMG 2.0*. (Morgan Kaufmann, 1997).
- 6 Date, C. J. and Darwen, H., *Foundation for Object/Relational Databases: The Third Manifesto*. (Addison-Wesley, 1998).
- 7 Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D. and Yanker, P., Query by Image and Video Content: The QBIC System. *IEEE Computer* **9** (1995) pp. 23-32.
- 8 Gauvain, J.-L., Lamel, L. and Adda, G., Partitioning and Transcription of Broadcast News Data. *Int'l Conf. on Spoken Language Processing*, Sydney, Australia, **5** (1998) pp. 1335-1338.
- 9 Herrera, P. and Serra, X., A Proposal for the Description of Audio in the Context of MPEG-7. *Proc. of the 1st European Workshop on Content-based Multimedia Indexing*, Toulouse, France (1999) pp. 81-88.
- 10 Hu, M. J. and Jian, Y., Multimedia Description Framework (MDF) for Content Description of Audio/Video Documents. *Proc. of the 4th ACM Conf. on Digital Libraries*, Berkeley, California (1999) pp. 67-75.
- 11 Jelinek, F., *Statistical Methods for Speech Recognition*. (MIT Press, 2000).
- 12 Kahn, L. and McLeod, D., Audio Structuring and Personalized Retrieval Using Ontologies. *Proc. of IEEE Advances in Digital Libraries*, Washington, D.C. (2000) pp. 116-126.
- 13 Nack, F. and Lindsay, A., Everything you Wanted to Know about MPEG-7: Part 1. *IEEE Multimedia* **3** (1999) pp. 65-77.
- 14 Nack, F. and Lindsay, A., Everything you Wanted to Know about MPEG-7: Part 2. *IEEE Multimedia* **4** (1999) pp. 64-73.
- 15 Sheth, A. and Klas, W. (Eds.), *Multimedia Data Management: Using Metadata to Integrate and Apply Digital Media*. (McGraw-Hill, 1998).

- 16 Turk, A., Johnson, S. E., Jourlin, P., Spärck-Jones, P. and Woodland, P. C., The Cambridge University Multimedia Document Retrieval Demo System. *Int'l Conf. on Information Retrieval*, Athens, Greece (2000) pp. 394.
- 17 Tzanetakis, G. and Cook, P., MARSYAS: A Framework for Audio Analysis. In *Organised Sound* (Cambridge University Press, 2000).
- 18 Wactlar, H., Hauptmann, A. and Witbrock, M., Informedia: News-on-demand Experiments in Speech Recognition. *Proc. of the ARPA Speech Recognition Workshop*, Arden House, Harriman, New-York (1996).
- 19 Wold, E., Blum, T., Keislar, D. and Wheaton, J., Content-Based Classification, Search, and Retrieval of Audio. *IEEE Multimedia* **3**, 3 (1996) pp. 27-36.
- 20 Wu, Y., Jajodia, S. and Wang, X. S., Temporal Database Bibliography Update (7th bibliography). <http://isse.gmu.edu/~csis/tdb/bib97/bib97.html> (1998).
- 21 Zhang, T. and Kuo, C.-C. J., Heuristic Approach for Generic Audio Data Segmentation and Annotation. *Proc. of the 7th ACM Int'l Multimedia Conf.*, Orlando, Florida (1999) pp. 67-76.