Subject to change

Patrick Indri

September 1, 2008

Contents

Project notes

TODO:

- placeTypeType and address type;
- check UML cardinality;
- data models for discovery;
- more details on the preservation section;
- sort names;

- possible expansion to deal with images;
- standard for archiving: WAW 96khz, 24bit or FLAC, with MP3 download;

Introduction

Aim: Investigation of (long term) audio file archiving for music + prototyped data model design and implementation.

Data resource: audio files, probably different formats, encodings, metadata content (anche dentro il file id3). **Concepts Contents**

Discovery Access Interoperability

Aim of the project

On the lack of a metadata standard for audio files: EBUcore

- EBUcore (based on Dublin Core).

Data Model: design

What should the data model be able to represent?

A model for audio data resources should be able to represent songs (and their possible different versions) and their groupings (in albums, compilations or other releases). Moreover, it should model authoring for artists (singers and lyricists) and other professional figures such as music producers. The model should handle technical metadata as well. Additionally, relations between different classes (e.g., a work is part of another work, two artists cooperated for a release) should be handeld as well.

A more detailed description of the model entities/classes

The proposed model consists of the following four main classes:

- The smallest object (in terms of granularity) is a **recording**, which represents a uniquely identifiable version of a song, that is the actual audio data. ISRC is a standard identifier for *sound recordings* and can be associated to each recording. A recording has title, ISRC, duration, artist credit and a link to the physical data resource (the audio file). Additional technical metadata for the data resource can be provided.

- Different recordings of the same song refer to the same **work**, which embeds all the different versions of a song. It represents the intellectual creation and can be put in relation to different recordings, different artists or other

works. ISWC is a standard identifier for *musical works/compositions* and can be associated to each work. A work has name, ISWC and links to correlated works/artists/records.

- Commercial issues of works are grouped into a **release**. Each release contains a track-list for the recordings it contains and has a title, an artist, information about the label and a form of identification (e.g., a catalogue number provided by the label).

- The paternity of works belongs to **artists**, which represent single musicians, groups of musicians or other professional figures (e.g., producers, lyricists, sound engineers). ISNI is a standard identifier for public entities of contribution to media content and can be associated to each artist. Artists have name, type and ISNI.

Additionally, the following secondary classes are defined:

- **File**. Base class for all the files. A file has name, size and a locator (that is, a link to the actual resource).
  - **Audio File**. Adds file details for audio files, has a hash and a bitrate.
- **Format**. Base class for format metadata. A format has ID, version and name.
  - **Audio Format**. Adds format metadata details for audio files, has codec, compression flag and lossless flag.
- **Label**. Holds information about imprints and record companies that take part in the production and distribution of a release. Label can uniquely identified by their label code.
- **Place**. Specifies the place where music is performed, recorded, mixed, etc.

UML model

In the following model square endpoints denote aggregation and arrow endpoints denote inheritance.
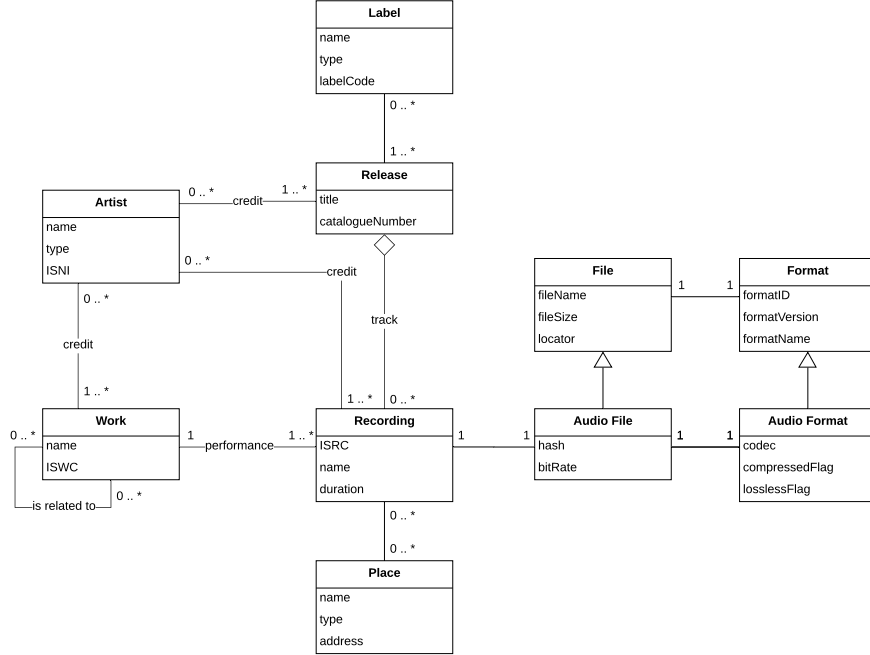
The proposed design assumes each recording, work and release to be associated with a credited artist (i.e., their participation to the *credit* relationship is total). This assumption can be satisfied defining special purpose artists to deal with unknown authors, traditional songs, etc.

Data Model: implementation

Choice of implementation

Concerning available implementation choices for the proposed model, two macro-alternatives have been taken into account: relational databases (RDB) and XML databases.

Relational databases offer a trivial way to enforce constraints between entities using primary keys and foreign keys. For instance, this would allow to properly model the constraints between releases and artists (i.e., a release cannot be credited to an artist whose name is not present in the "artist" relation). Additionally, indexing is easily supported and RDBs have a widespread use. Moreover, RDBs handle many-to-many and many-to-one relationships with ease (consider

the artist-release and the work-recording relationships). However, relational databases manifest a reduced flexibility due to their strict integrity constraints and, once instantiated, are moderately difficult to expand.

On the other hand, XML databases (document-based databases in general) are meant to be a more flexible means to store data, integrity not being their first concern. For a given XML schema, an XML documents can usually contain partial/incomplete information. This is appealing for the task at hand since not all the metadata for a given release may be available. However, many-to-many and many-to-one relationships are harder to handle.

For this particular project, flexibility has been privileged: the model implementation will consists in a XML schema definition (XSD). It should be noted that a PostgreSQL implementation for a similar base model is provided by MusicBrainz, a collaborative music metadata project.

XSD

An implementation of the proposed model can result in XSDs with varying degrees of flexibility. EBUcore XSD provides extreme flexibility and can handle both audio and visual broadcasting resources. Such great flexibility (a necessity, given the wide range of resources EBUcore is able to descrive) is deemed not to provide enough structure for this particular project. At the opposite side of the spectrum a strictly hierarchical XSD, where each document must contain "releases" made of "recordings", might lead to a overly strict model implementation (in this case, a RDB implementation would provably serve the aim of this project better).

Drawing from the considerations of the previous paragraph, the proposed XSD should:

- Provide, where possible, a refinement of the Dublin Core elements, aiming for interoperability;
- Find a balance between a hierarchic structure and the possibility ti accept "partial" entries (e.g., a document that contains only an artist must be handled by the XSD);
- Handle relationships between different elements with reasonable detail.

The resulting XML schema will now be discussed.

On relationships

The UML model presents several many-to-many and many-to-one relationships which have been implemented with different approaches:

- Recording-Place and Release-Label (many-to-many). This relationship has been simplified by defining a Place/Label to be one of the elements of a Recording/Release. This reduces expressiveness: Places and Labels cannot be stand-alone entities. This solution is acceptable since Places and Labels are secondary entities.

5

- Recording-Audio File (one-to-one). Each Recording is forced to have exactly one Audio File element (using `minOccurs="1"` and `maxOccurs="1"`). The other constraint is enforced by the existence and uniqueness of a `fileID` attribute.
- Audio File-Audio Format (one-to-many). Each Audio File is forced to have exactly one an Audio Format. Audio Format is not a stand-alone entity.
- Work-Recording (one-to-many). Works have the `hasPerformance` relation element, which links a Work to at least one Recording ISRC. On the other hand, Recordings have the `isPerformanceOf` relation element, which links a Recording to exactly one Work ISWC.
- Work-Work (many-to-many). Works have the `hasRelatedWork` relation element.
- Release-Recording (one-to-many, composition). Releases have the `hasTrack` relation element, which links a Release to at least one Recording ISRC.
- Work-Artist, Recording-Artist and Release-Artist (many-to-many). This relationships have been modelled using the `hasArtist` relation element.

All the constraints (cardinality and participation) have been modelled.

On references (and between different documents)

Relations implemented via REFID and ID. Key can be used (multi field). Between documents is a problem.

XML example

```xml
<work>
  <ISWC>
    ISWC_T-000.000.000-A
  </ISWC>
  <title lang="en">
    <dc:title>Test Work</dc:title>
  </title>
  <hasArtist label="William Wilson" description="Singer"></hasArtist>
  <hasPerformance label="Test Recording" description="Studio Version">
    <relationIdentifier>
      <ISRC>
        ISRC_AAAAA0000000
      </ISRC>
    </relationIdentifier>
  </hasPerformance>
</work>
```

Other

Cloud Storage OAIS: sketch model.

Preservation

Software and tools used