

# OPEN DATA MANAGEMENT & CLOUD EXAM PROJECT

AUDIO MUSIC FILE ARCHIVING

PATRICK INDRI

MAY 10, 2020

The presentation is organised as follows:

1. Introduction

Aim and project description

2. Model design and implementation

UML, XSD and XML

3. Interfaces and services

Search filter, data annotation and storage

4. Preservation and interoperability

Metadata preservation, semantic interoperability and audio file formats

5. Final considerations

## Aim of the project

Investigation of audio file archiving for music.

In particular:

- ▶ UML metadata model;
- ▶ XSD implementation and XML sample document;
- ▶ discussion of data discovery/access and interoperability;
- ▶ discussion of (long term) archiving and data preservation.

**Data resource:** not an actual dataset but music files in general.

Great variability:

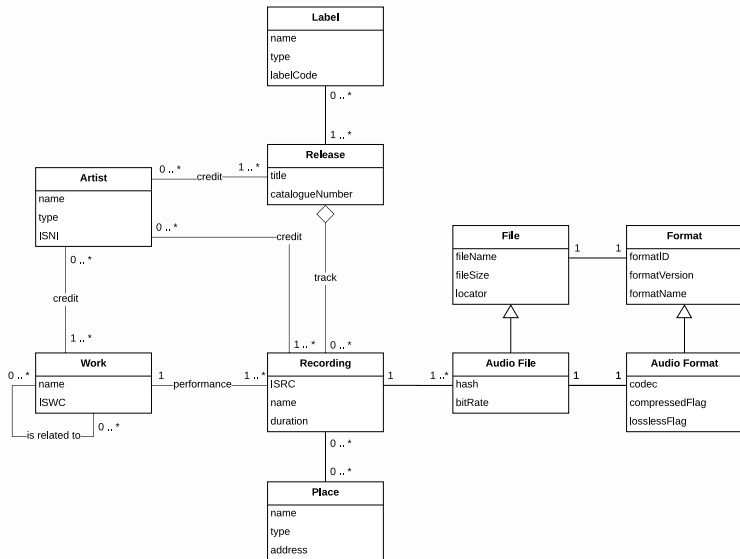
- ▶ File formats;
- ▶ Encodings;
- ▶ Metadata containers and contents.

There is no widely used and standardised metadata model for music audio files.

- ▶ Dublin Core: simple (15 terms), focus on descriptive metadata;
- ▶ EbuCore: detailed DC extension, fine grain technical and administrative metadata for broadcasting;
- ▶ METS: handles the structural/hierarchical metadata of a digital library. Open flexibility (no vocabulary).

What should the data model represent?

- ▶ Songs and their different versions;
- ▶ Groups of songs (i.e., commercial releases);
- ▶ Artists;
- ▶ Basic technical metadata (file formats);
- ▶ Relationships between songs, releases and artists.



Choice of implementation:

- ▶ RDB: easy to enforce constraints (primary/foreign keys), widely used, easy to model relationships, rigid structure;
- ▶ **XSD**: flexible, easily handle partial data, more difficult relationship handling.

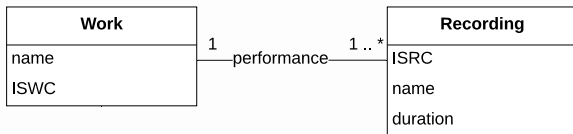
The proposed XSD implementation should:

- ▶ Refine Dublin Core;
- ▶ Balance integrity constraints and partial data;
- ▶ Model relationships with detail.

The resulting XSD can be retrieved [here](#).



# XSD RELATIONSHIPS



```
<xs:complexType name="workType">
  ...
  <xs:element name="hasPerformance" type="odmc:relationType"
    minOccurs="1" maxOccurs="unbounded"/>
  ...
</xs:complexType>

<xs:complexType name="recordingType">
  ...
  <xs:element name="isPerformanceOf" type="odmc:relationType"
    minOccurs="1" maxOccurs="1"/>
  ...
</xs:complexType>
```

# XSD RELATIONSHIPS - CONT'D

Relationships are implemented using the KEY/KEYREF syntax.

```
<xs:key name="relationId">
  <xs:selector xpath="./odmc:odmcItems/odmc:work/odmc:ISWC"/>
  <xs:field xpath="@id"/>
</xs:key>
<xs:keyref name="relationIdref" refer="odmc:relationId">
  <xs:selector xpath="../odmc:relationIdentifier/*"/>
  <xs:field xpath="@idref"/>
</xs:keyref>
```

The XSD structure can be interactively navigated using a web browser using [this](#) SVG file.

# XML EXAMPLE

Example of an XML document, valid against the proposed XSD.

```
<work>
  <ISWC id="ISWC_T-000.000.000-A"></ISWC>
  <title lang="en">
    <dc:title>Test Work</dc:title>
  </title>
  <hasArtist label="Will Wilson" description="Singer">
  </hasArtist>
  <hasPerformance label="Test Rec." description="Studio Ver.">
    <relationIdentifier>
      <ISRC idref="ISRC_AAAAA0000000"></ISRC>
    </relationIdentifier>
  </hasPerformance>
</work>
```

# DIFFICULTIES AND POSSIBLE EXPANSIONS

## Difficulties:

- ▶ Choosing the right degree of flexibility;
- ▶ Handling large XML documents.

## Possible expansions:

- ▶ Include sort names;
- ▶ Include pictures for artists and releases;
- ▶ Model inter-document constraints.

# DATA DISCOVERY: SEARCH/FILTER SERVICE

The most fundamental service would be a search/filter service.

- ▶ Free text queries on the various categories (artist, release, work, recording);
- ▶ Queries can be implemented using XQuery;
- ▶ Information retrieval techniques to improve results: edit distance and k-gram distance for spelling correction;
- ▶ Popularity rank for the results.

Once the resource has been identified: download (lossy and lossless file) and online preview.

**Crucial point:** XML indexing, trade-off between memory usage and performance.

Metadata can be embed in audio files. **Is it advisable?** A minimal amount, the *catastrophic* metadata is necessary. Most metadata should be saved in external files.

Viable metadata containers:

- ▶ ID3: designed for Mp3, highly structured;
- ▶ XMP: ISO standard for JPEG images, defined for other formats including Mp3 and WAW;
- ▶ BWF <bext> chunk: XML administrative metadata for BWF;
- ▶ Vorbis comments: unstructured metadata for Vorbis and FLAC.

**Conclusion:** data annotation heavily depends on file format choice.

# STORAGE AND CLOUD SOLUTIONS

Storing **XML** files:

- ▶ XML-native databases are not scalable;
- ▶ XML-enabled databases are well established, scalable and can be queried with SQL.

**Crucial point:** the proposed XSD itself is not scalable.

**Audio** files can be stored in BLOBs in the database: not flexible.  
Alternatively, store pointers to the resource:

- ▶ Up to *few* TB: remote filesystem;
- ▶ Large archives: distributed DB;
- ▶ Cloud solutions: Amazon S3 offers versioning, orphan files handling, redundancy and access control.

# DATA PRESERVATION

A primary concern for an audio archive. The proposed XSD prevents data duplication and limits orphan data.

Most technical and descriptive metadata is **stable** and, in principle, it is not essential to preserve it. Catastrophic metadata is **ephemeral** (e.g., the ISRC for a recording) and must be preserved.

Audio file hashes can be used to check file integrity.

Cloud solutions for **long term preservation**: AWS S3 Glacier and Glacier Deep.

**Crucial point:** an OAIS implementation could not be *open access* for copyrighted files.



The proposed model strives for semantic interoperability.

- ▶ Syntactic interoperability: XML is an open format;
- ▶ Semantic interoperability: XSD equips data with meaning, Dublin Core support;
- ▶ Standardised and persistent identifiers for the main classes;

A report is associated with the project: it provides ontological information on the model.

# AUDIO FILE FORMATS

One of the most crucial design choices: obsolete and proprietary file formats make data handling troublesome (*digital dark age*).

(Quasi) open file formats should be used:

- ▶ BFW: uncompressed extension of Microsoft WAV, open format;
- ▶ FLAC: smaller file size, compressed lossless format, good metadata support, open format;
- ▶ MP3: most popular audio file format, very small file size, lossy compression, not a viable alternative for archiving (good for download/preview).

**Crucial point:** the file format should be well established.

Audio file data management is difficult because:

- ▶ Many impactful design choices;
- ▶ Lack of metadata standard can make interoperability difficult;
- ▶ Practical issues: storage and indexing;
- ▶ Cloud solutions to deal with large volumes and concurrent access.