Clustered Nodes Of Metadata For Time-Code Based Media

An AV Asset Management Solution in Neo4j

by Laurence Cook, M.S.L.I.S

503.312.3681
connect@metacirque.com
@metacirque

2014-02-17, rev 02-21

Audio-Visual Assets: Problems of Volume, Variety & Distribution

For any given audio-visual feature length moving image, the production, post-production, and distributed files for consumer access involve petascale volumes of storage, and multitudes of clips and other elements, often from different sources, spread over multiple locations and data systems.

Problems confronting feature films aside, the volume of audio-visual content (video, images, and music, along with their associated metadata) is one of the fastest growing components of Internet.

The development of more efficient workflows, designed to cultivate faster and more accurate storage & retrieval, are required. As well as, intuitive visualizations and touch-screen interfaces.

A Neo4j open-source data model can address content management requirements for:

- Content Creators: Production, Post-Production, Distribution
- Media Archives: Cost effective content curation & licensing
- New Media: Tracking and re-purposing of assets over time
- Consumers: More refined & fluid access to AV resources

A Fountainhead Of Pain Points

There are many pain points impacting workflows of the audio-visual domain:

- **Cameras (different devices, with different quality, settings, and metadata)**
- **Media Servers (ingest, storage & retrieval of heterogeneous media fragments)**
- **Archival Footage (tracking derivatives/provenance to/from film, tape, and digital)**
- **Digital Images (multiple generations; scanned & retouched from physical originals)**
- **Non-Linear Editing Applications (masters & derivates; metadata e.g., Chapters & EDLs)**
- **Access Copies (tracking, cultivation & purge of assets for various distribution channels)**

A Use Case:

A documentary filmmaker integrates archival footage & still images, fresh digital interviews, B-Roll establishment shots, some minor animation, licensed music, as well as production company and standard industry logos, titles and credits into her work. Her film may be created over many years, with different crews and equipment, through both production and post-production.

She (as well as her producers and distributors) want to track provenance of all footage to provide proper attribution credit & control distributed licensed versions, as well as maintain an ability to access, re-edit and repurpose each original source media fragment.

Some Requirements

The solution must:

- Be open source, extensible and highly scalable
- Address use cases and specifications for Media Fragments per W3C
- Able to integrate metadata schema such as PBCore, EBUCore and SMPTECore
- Able to leverage controlled vocabulary & UMIDs, but agnostic as to specific regimes

The solution should seek compatibilty with standards & requirements of the following entities:

- Library of Congress
- Advanced Media Workflow Association
- Association of Moving Image Archivists
- Society of Motion Picture and Television Engineers
- National Digital Information Infrastructure and Preservation Program

metaCirque AV Data Model Overview

The model uses Neo4j 2.0 Labels to create, maintain and access four classes of nodes:

- **Instance**: A media resource's technical attributes, designed to carry information.
- **Content**: The information stored with an Instance, including some metadata.
- **Essence**: Audio and/or video signal information encoded in one or more tracks.
- **TimeCode**: A frame-rate metered sequence, synchronizing essence playback over a duration

An Instance's contained media fragments flow within a hierarchy of content object relationships, while provenance and derivatives are connected by relationships which include target metadata.

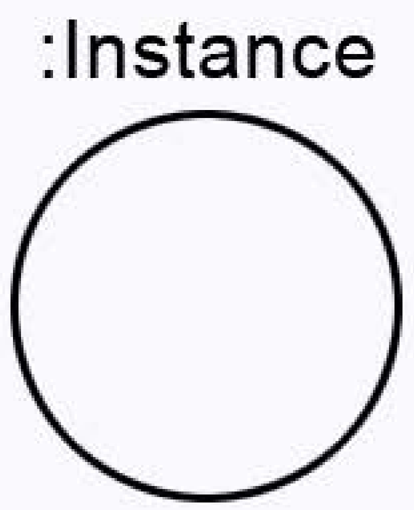
```
a-[r:CONTAINS{NID: "Ve1.4"}]->b

a-[r:TIME_SYNC{NID: "Ve1.1", TimeIn: "00:00:00;00", TimeOut: "00:01:01;04"}]->b

a-[r:PROVENANCE{UMID: "mcDemo_06", NID: "Ve1.2", Type: "Direct Source", TimeIn: "00:04:35;00", TimeOut: "00:07:20;15"}]->b
```

Neo4j is open-source and fully ACID. It facilitates faster and more complex search & retrieval, with a potential for more intuitive visualizations, over relational databases and key-value stores.

The model is extensible, and very scalable. It is also schema and UMID structure agnostic, placing identification emphasis upon a media fragment's context, rather than requiring adoption of a universal namespace. As such, it can provide interoperability for diverse cataloging regimes and media stores.

Instance (Technical Properties & Use Rights)

An Instance is a single occurrence (physical or digital) that carries information, and is classified as a FRBR, Group 1, Item level entity, with a unique identifier (i.e., UMID).



An Instance node contains properties that describe the technical attributes of a media resource, as set forth in metadata schema such as PBCore, EBUCore, and SMPTECore.

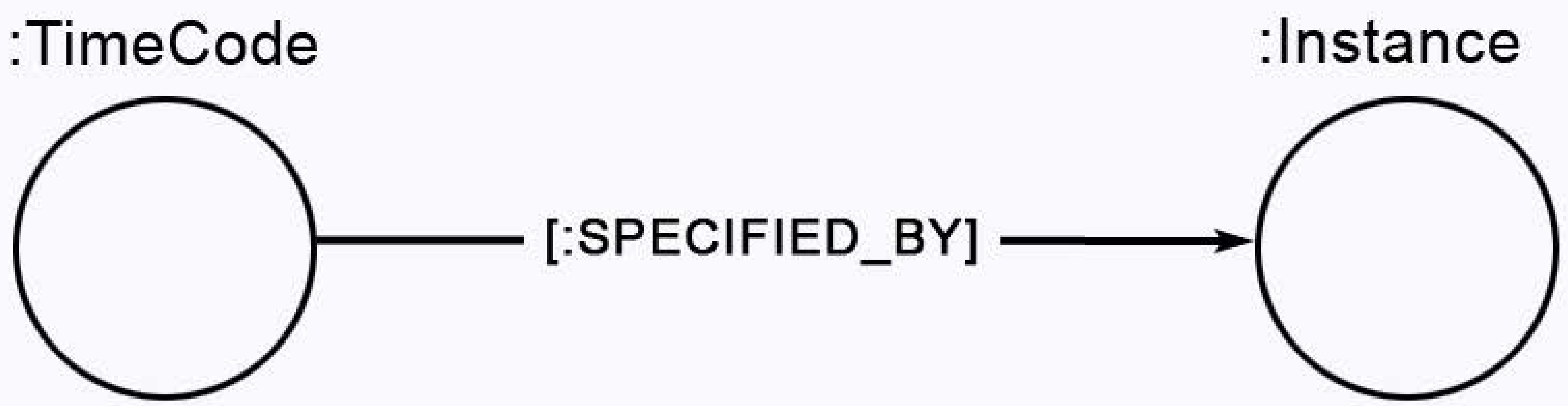
Depending on whether an Instance is a physical tape or digital file, its' metadata can address a mix of properties including equipment, location, format, date(s), duration, dimensions, aspect ratio, file size, codec, frames per second, data rate, bitrate, language, color, tracks, and access rights, etc.

Each copy of an item is a unique instance, regardless of apparent equivalence to other instances.

The right to hold and/or access an instance can be encoded as a property within the Instance.

TimeCode (synchronizes media fragments)

A TimeCode controls the linear order in which things are scheduled to happen. Within a media file it is a common reference of finite duration to which essence elements are synchronized for playback.



TimeCode is specified by the Instance. In addition to UMID & NID, a node has two properties, TimeStart and TimeDuration, to synchronize an Instance's video and/or audio essences.

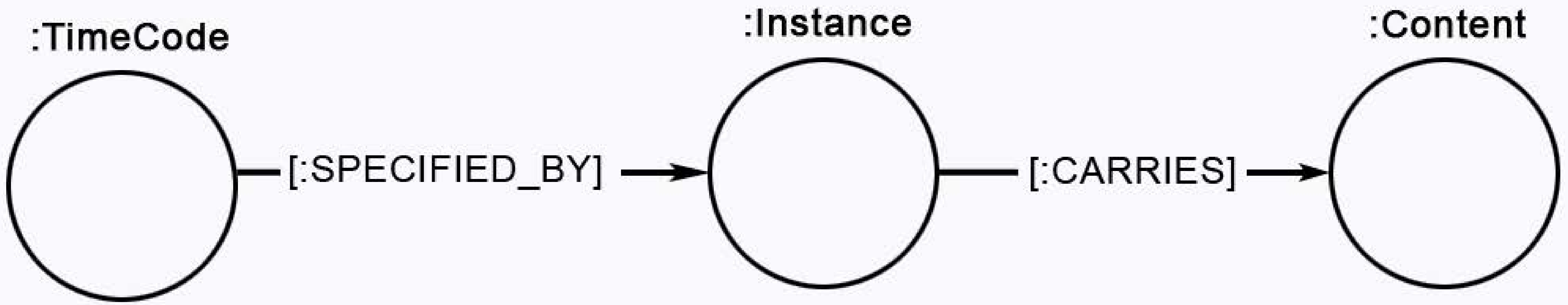
TimeCode provides a path to essence synchronization data, both for fragments within an Instance structure, as well as for harvesting Instance data for collection level & workflow analytics.

For container instances (e.g., MXF), the instance wrapper can have multiple distinct time tracks, and associated essences. To represent this, the model can expand to multiple TimeCode nodes.

All nodes associated with an Instance, share the same UMID, along with having a distinct NID.

Content (the reason for creating a file)

Content is information carried by an Instance, both in terms of audio-visual essences & metadata.

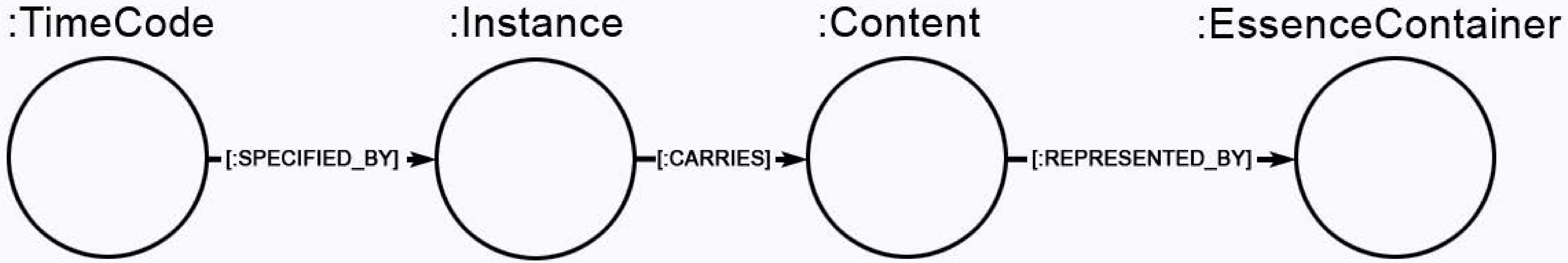


The Content node contains properties describing information carried by a physical resource or digital file instance. Within the model, Content is equivalent to catalogue descriptive information, and may include ownership information for a media asset (i.e., the Content is owned, while an Instance, such as a digital file, is held and/or accessed under licensed rights).

Bifurcating a Content node's descriptive and ownership metadata from its' associated Instance node's technical and rights holder metadata, facilitates locating equivalent and/or similar content across multiple entities, while separately controlling curation, access and use of an Instance. It reduces the number of relationships per node (speeding queries) and provides better visualization.

EssenceContainer

Content carried by an Instance is represented through essence tracks. An EssenceContainer is an abstract transition node, containing one or more essence tracks of a media file.



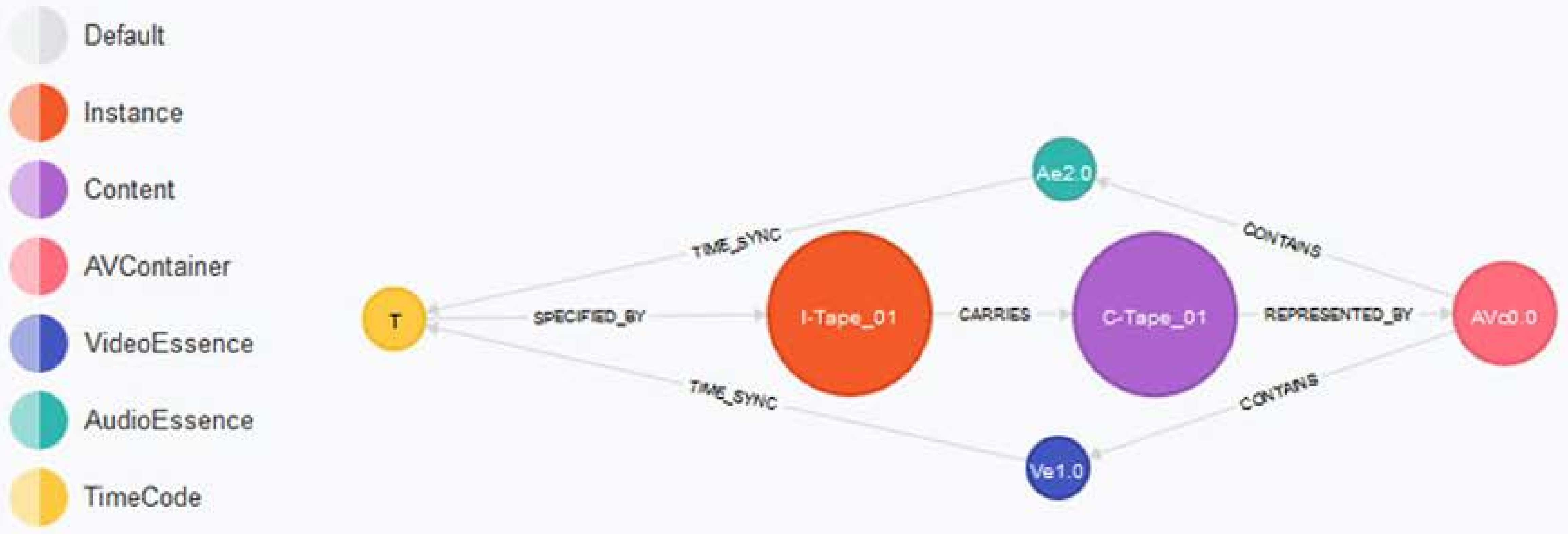
There is one EssenceContainer for each TimeCode in an audio-visual instance.

A common video instance contains 2 tracks: 1 audio & 1 video. However, some formats (e.g., MXF) can contain multiple TimeCodes and their associated essence tracks. In such cases, the model can scale-up with multiple EssenceContainers, each containing distinct essences synchronized to its' corresponding TimeCode, as specified by the Instance.

This multi-track scalability can represent diversely rendered media objects, while also having the potential to adapt to NLE workflows to help control media bins and multiple editing tracks.

Audio & Video Essence Tracks

An EssenceContainer contains one or more Audio and/or Video Essence tracks. Each track, in turn, can contain one or more AudioEssence or VideoEssence elements or media fragments, each synchronized to the TimeCode. Below is a simple representation of a Digital8 magnetic tape.

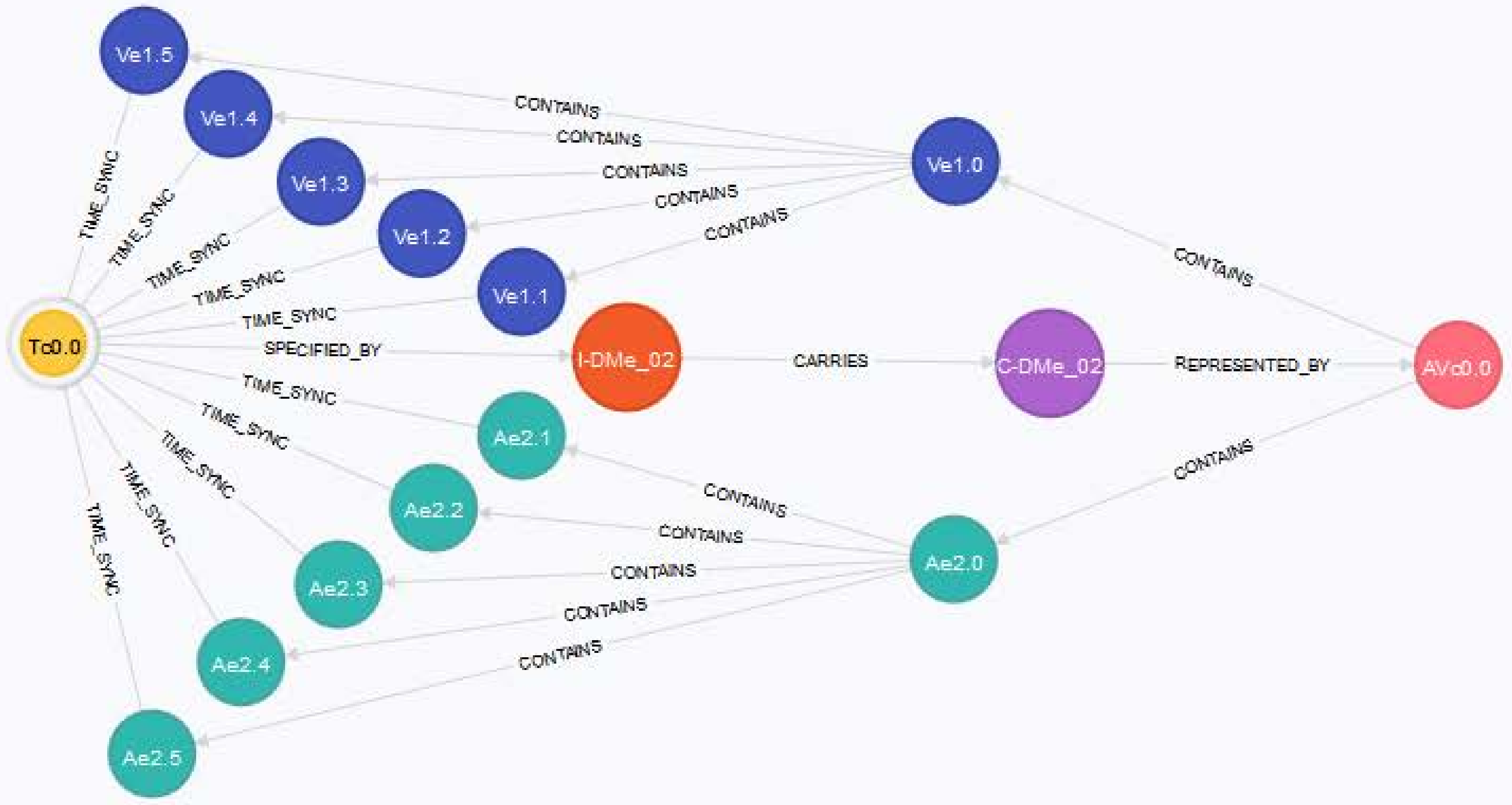


The [:TIME_SYNC] relation properties, TimeIn & TimeOut, are the points at which an essence, or essence element, synchronizes to the TimeCode. Essence TimeIn and/or TimeOut points can overlap to express dissolves, montage or other structural elements of a moving image.

An essence track can have mulitple fragments, which in turn can contain other fragments.

Multitude Audio & Video Essences

Each Essence track can contain one or more AudioEssence or VideoEssence elements otherwise known as Media Fragments, each synchronized to the TimeCode.



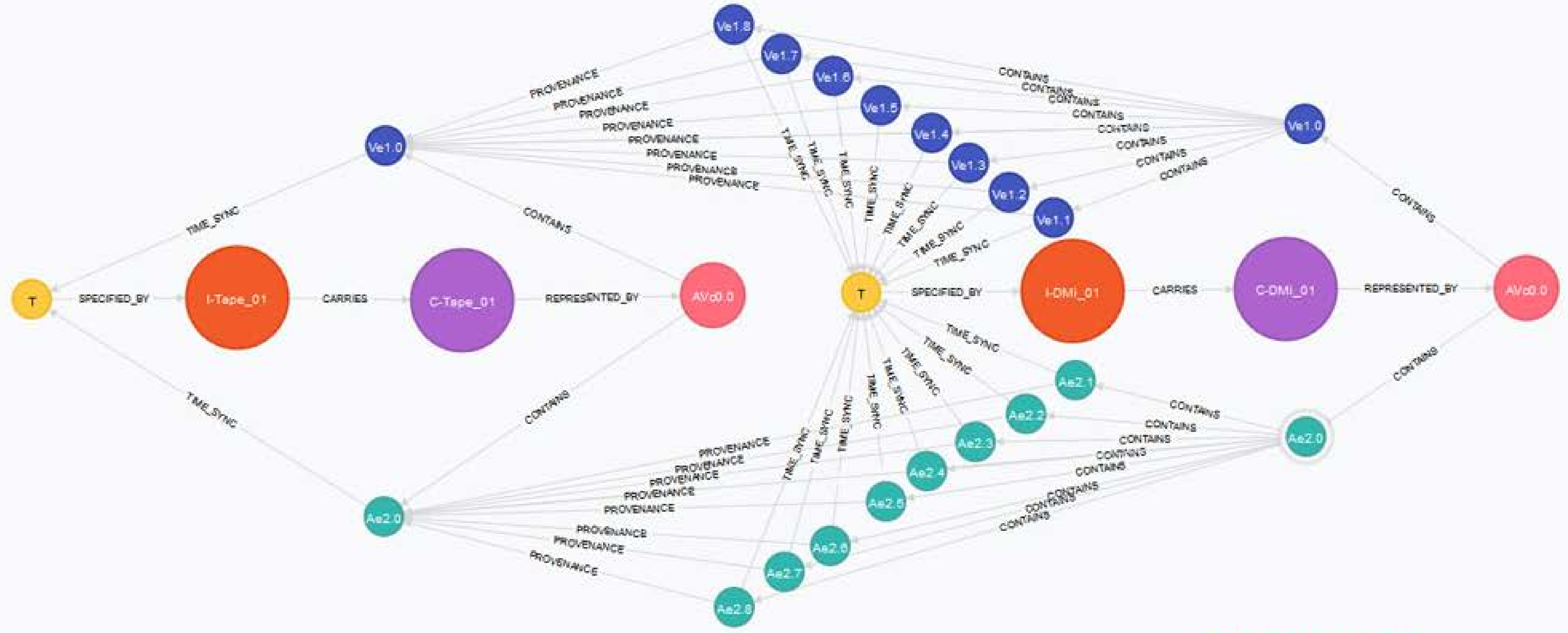
The granularity scope of fragments within an Essence can be as large as an entire essence track, or as small as a frame, or portion of a frame. They can denote clip, scene, act, or other abstract classifications, as well as points of view, or overlaps in time (e.g., dissolves & montage).

Each fragment can have relationships regarding descriptive metadata, generation and provenance with other fragments, within its' Instance cluster, or with respect to fragments in other clusters.

Provenance (tracing fragment sources)

Provenance is the origin & lineage of content. In the model, there are two types of provenance:

- Direct Source (immediate source used to create a subject essence, or any part thereof)
- Primary Source (the original or best source, perhaps many generations back)

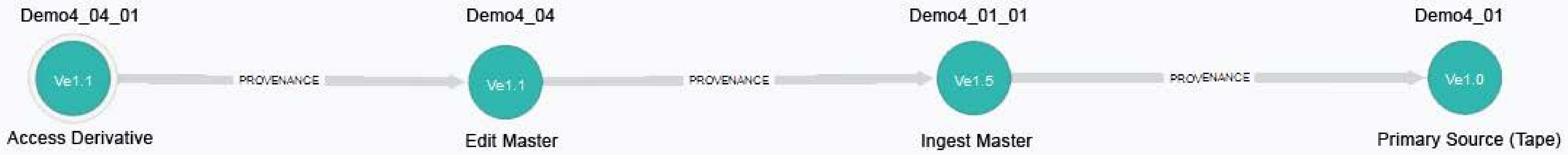


Above is a screenshot from the metaCirque demo database indicating Provenance between the primary source (left), and its derivative clips within an ingested digital master instance (right).

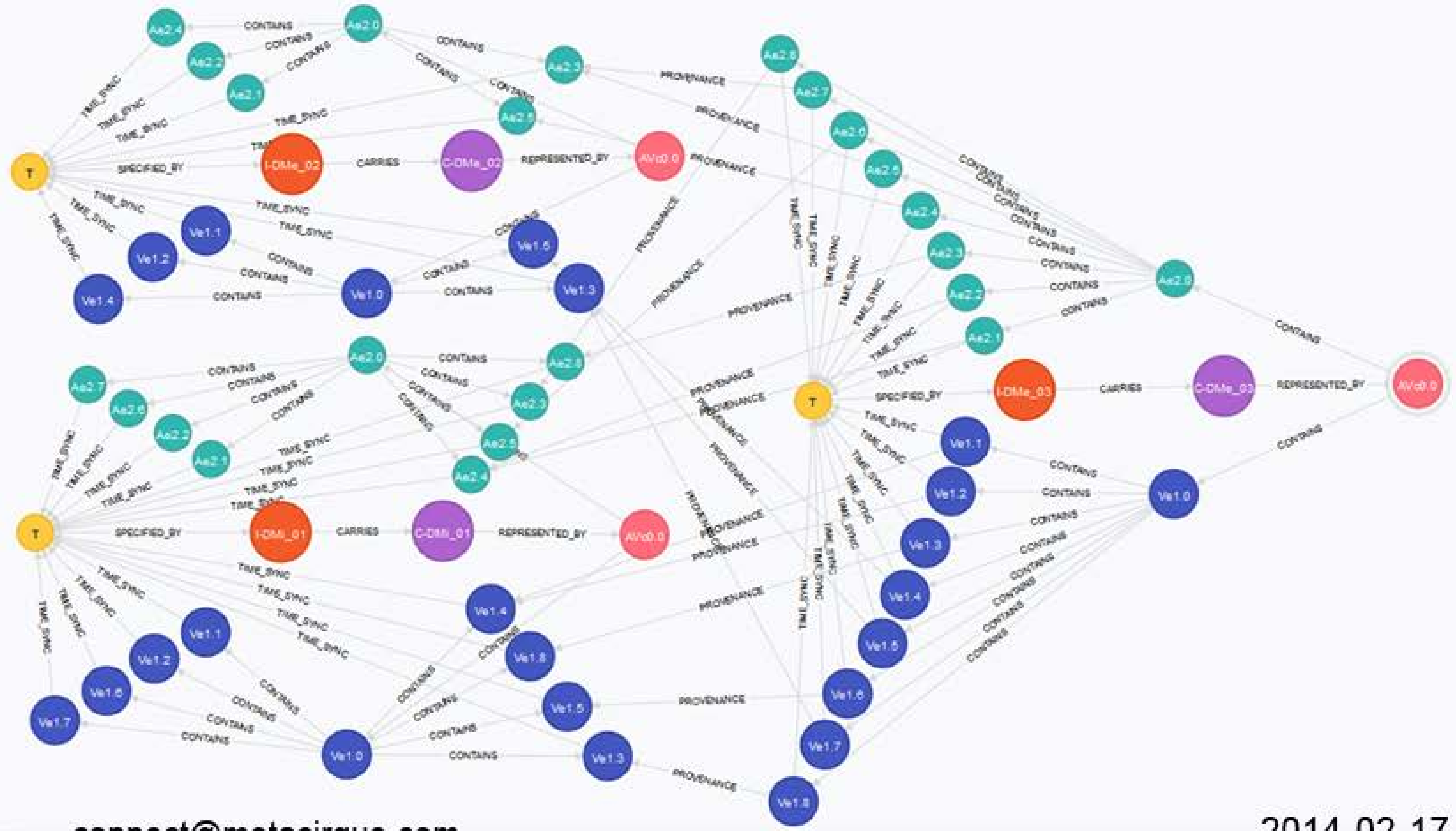
In this case, the tape (left) is both the Primary and Direct source of the digital master.

Provenance (discrete & complex arrays)

Provenance includes tracing lineage of a fragment back multiple generations to its primary source:

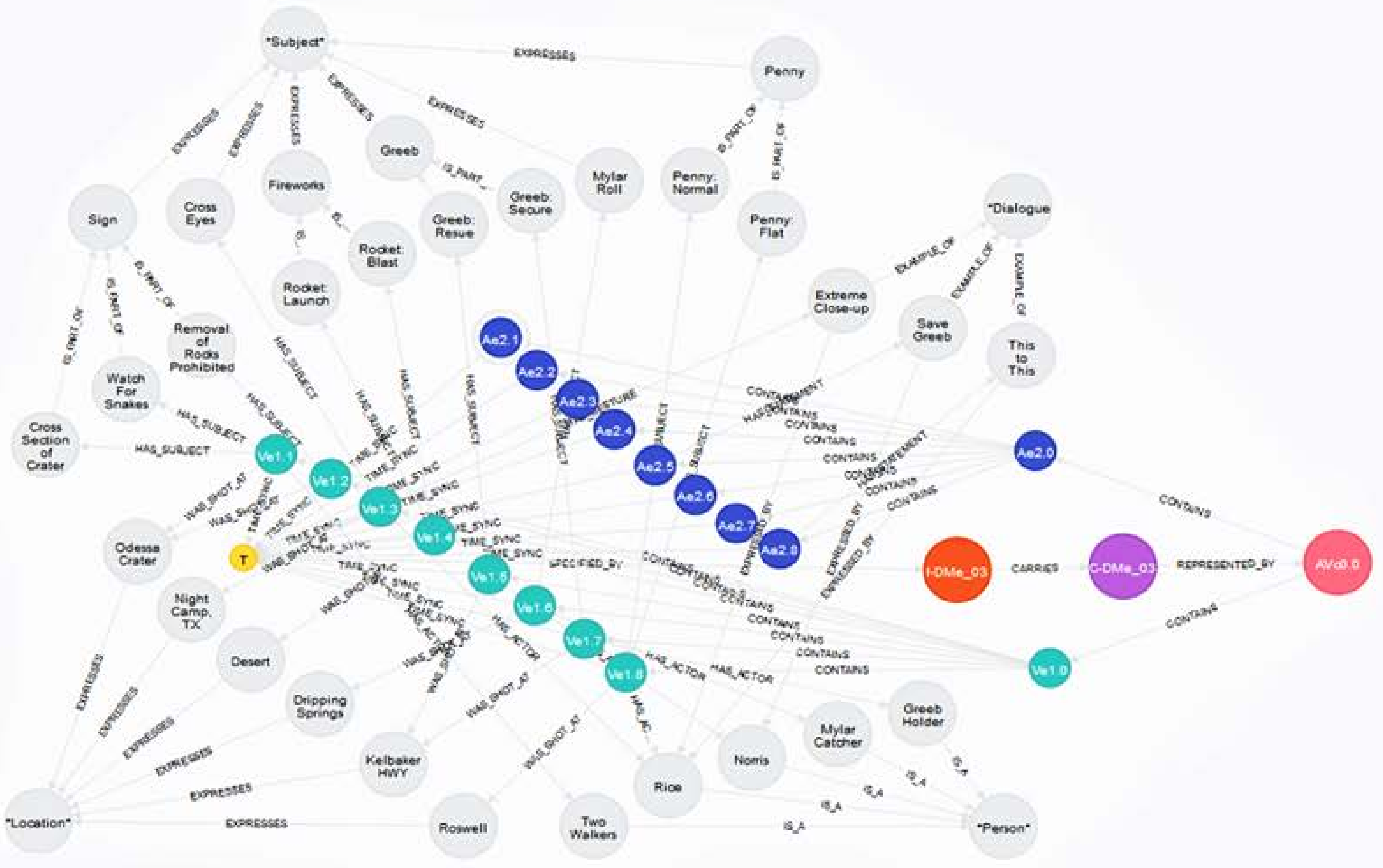


Or can show direct relationships of fragments edited from multiple files into a new digital master:



Descriptive Metadata (meaning from structure)

The model is fully extensible to integrate with a descriptive metadata schema at node level.



Above is a screenshot from the demo database of an adhoc descriptive schema on one Instance cluster, which includes sample Subject, Location, Person, and Dialogue descriptive metadata. Node tags express context appropriate identification (e.g., NID, class and/or descriptive names).

Sample Code

Here is sample code with minimal adhoc metadata for an instance cluster representing a Digital-8 Tape:

```
CREATE (:Instance {UMID:"mcDemo_01", NID: "I-Tape_01", InstanceCreated: "2003", InstanceSourceDevice: "Sony DCR-TRV340",
InstanceStandard: "Digital8", InstanceGeneration: "Primary Source (D8 Tape)", InstanceLocation: "Physical Archive", InstanceDuration: "01:00:00;00"})

CREATE (:Content {UMID:"mcDemo_01", NID: "C-Tape_01", ContentTitle: "SoupQuinox", ContentDate: "2003", ContentDescription: "Roadtrip to Texas",
ContentOwner: "Studio LarZ"})

CREATE (:AVContainer {UMID:"mcDemo_01", NID: "AVc0.0", ElementTitle: "Essence Container", TimeIn: "00:00:00;00", TimeDuration: "01:00:00;00"})

CREATE (:VideoEssence {UMID:"mcDemo_01", NID: "Ve1.0", ElementTitle: "Video Essence", TimeDuration: "01:00:00;00"})

CREATE (:AudioEssence {UMID:"mcDemo_01", NID: "Ae2.0", ElementTitle: "Audio Essence", TimeDuration: "01:00:00;00"})

CREATE (:TimeCode {UMID:"mcDemo_01", NID: "Tc0.0", TimeCodeTitle: "Time", TimeStart: "00:00:00;00", TimeDuration: "01:00:00;00"})

MATCH a, b WHERE (a:Instance AND a.UMID="mcDemo_01") AND (b:Content AND b.UMID="mcDemo_01") CREATE a-[r:CARRIES]->b RETURN r

MATCH a, b WHERE (a:TimeCode AND a.UMID="mcDemo_01") AND (b:Instance AND b.UMID="mcDemo_01") CREATE a-[r:SPECIFIED_BY]->b RETURN r

MATCH a, b WHERE (a:Content AND a.UMID="mcDemo_01") AND (b:AVContainer AND b.UMID="mcDemo_01")
CREATE a-[r:REPRESENTED_BY]->b RETURN r

MATCH a, b WHERE (a:AVContainer AND a.UMID="mcDemo_01") AND (b.NID="Ve1.0" AND b.UMID="mcDemo_01")
CREATE a-[r:CONTAINS{NID: "Ve1.0"}]->b RETURN r

MATCH a, b WHERE (a:AVContainer AND a.UMID="mcDemo_01") AND (b.NID="Ae2.0" AND b.UMID="mcDemo_01")
CREATE a-[r:CONTAINS{NID: "Ae2.0"}]->b RETURN r

MATCH a, b WHERE (a.NID="Ve1.0" AND a.UMID="mcDemo_01") AND (b:TimeCode AND b.UMID="mcDemo_01")
CREATE a-[r:TIME_SYNC{NID: "Ve1.0", TimeIn: "00:00:00;00", TimeOut: "01:00:00;00"}]->b RETURN r

MATCH a, b WHERE (a.NID="Ae2.0" AND a.UMID="mcDemo_01") AND (b:TimeCode AND b.UMID="mcDemo_01")
CREATE a-[r:TIME_SYNC{NID: "Ae2.0", TimeIn: "00:00:00;00", TimeOut: "01:00:00;00"}]->b RETURN r
```

Demo Database

Available for download through Github:

https://github.com/metacirque/Neo4j-AV-Data-Model

Note: The Neo4j Browser is designed to cluster nodes based upon relationship weights. As excellent & natural as that is, the metaCirque database requries development of new a visualization & interface that can emphasize instance node array structures, akin to those provided in this presentation. This is currently in development. Any input is appreciated.

Questions or Comments?