Title	Catalogue : 自律分散環境におけるグラフ構造を用いたファイル間関係表現
Sub Title	
Author	宮下, 山斗(Miyashita, Yamato)
Publisher	慶應義塾大学デジタルメディア・コンテンツ統合研究センター
Publication year	2015
Jtitle	慶應義塾大学DMC紀要 (DMC Review Keio University). Vol.2, No.1 (2015. 3) ,p.39- 49
JaLC DOI	
Abstract	一般的な検索手法であるテキストベースの検索では着目ファイルがいかなる文脈に属しどのようなファイルと関連しているかを知るためには, ユーザが予め適切なキーワードを知っておく必要がある。また, テキストベースの検索システムを構築するためにはウェブ空間を全探索しインデックスを作成する必要があり, 膨大なストレージコストと演算コストを要する。 そこで本論文では, 自然言語を用いずにファイルをグループ化しファイル間の関係を記述するCatalogueとその自律分散管理システムについて述べる。 本研究ではまず, 自然言語によらない数値識別子で分類することで狭義の一貫性をもったグループを作成可能にする。 また, 有向グラフを用いてグループ内のファイルの関係を記述することで柔軟な関係表現を可能にする。 この関係表現をCatalogueと呼ぶ。さらに, Catalogueをファイルの実体と分離して作成・管理することで多様なCatalogueが生成可能となる。 そして, 着目ファイルとそのファイルを含むCatalogueの対応関係を管理することで全探索を回避して分類情報を検索可能にする。 最後に, あるファイルのプロパティを別ファイルとして保存しCatalogueで対応付けることで任意のユーザがプロパティを作成可能にする。 このように, Catalogueの汎用的なグラフ表現を用いてファイルの所有者や種類, 分野を問わず関連付けることが可能となる。 本研究では, 自律分散管理された有向グラフを用いて仮想ミュージアムやLive映像配信といったCatalogueを用いたアプリケーションが動作することを確認した。
Notes	
Genre	Departmental Bulletin Paper
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=KO32002001-00000002-0039

Catalogue: 自律分散環境における グラフ構造を用いたファイル間関係表現

宮下　山斗†

† 慶應義塾大学理工学研究科　〒 223–8522 神奈川県横浜市港北区日吉 3 丁目 14-1
E-mail: †mine@inl.ics.keio.ac.jp

あらまし　一般的な検索手法であるテキストベースの検索では着目ファイルがいかなる文脈に属しどのようなファイルと関連しているかを知るためには，ユーザが予め適切なキーワードを知っておく必要がある．また，テキストベースの検索システムを構築するためにはウェブ空間を全探索しインデックスを作成する必要があり，膨大なストレージコストと演算コストを要する．そこで本論文では，自然言語を用いずにファイルをグループ化しファイル間の関係を記述する Catalogue とその自律分散管理システムについて述べる．本研究ではまず，自然言語によらない数値識別子で分類することで狭義の一貫性をもったグループを作成可能にする．また，有向グラフを用いてグループ内のファイルの関係を記述することで柔軟な関係表現を可能にする．この関係表現を *Catalogue* と呼ぶ．さらに，Catalogue を ファイルの実体と分離して作成・管理することで多様な Catalogue が生成可能となる．そして，着目ファイルとそのファイルを含む Catalogue の対応関係を管理することで全探索を回避して分類情報を検索可能にする．最後に，あるファイルのプロパティを別ファイルとして保存し Catalogue で対応付けることで任意のユーザがプロパティを作成可能にする．このように，Catalogue の汎用的なグラフ表現を用いてファイルの所有者や種類，分野を問わず関連付けることが可能となる．本研究では，自律分散管理された有向グラフを用いて仮想ミュージアムや Live 映像配信といった Catalogue を用いたアプリケーションが動作することを確認した．

キーワード　グラフデータベース，局所探索，ソーシャルキュレーション

1. はじめに

近年インターネット上のファイルは急激に増加しており，それらから所望のファイルを検索するために Yahoo！や Google といったテキストベースの検索エンジンが広く用いられている．テキストベースの検索エンジンでは，ユーザがあるファイルがもつ文脈に精通していない場合でも，適切なキーワードを予め知っておく必要がある．例えば，あるユーザが Isaac Newton にどのような側面があるのかを知ろうとしているとする．そのユーザが検索エンジンで “Isaac Newton” というキーワードを入力して検索すると，Wikipedia (英語版) が検索結果として提示されると予想される．その Newton の記事には，彼に関する様々な情報が記述されているが，彼が “猫好き” だったということについては触れられていない (※ 日本語 Wikipedia の Newton の記事では述べられている)．そのため，その記事を読んだユーザは彼の “猫好き” という側面に気がつくことは困難であると考えられる．このように，検索エンジンでは検索対象がどのように分類され他のファイルと関係しているのかを発見するのは困難である．

また，一般的にテキストベースの検索エンジンはネットワーク全体を予めクローリングしておく必要がある．さらにファイルをキーワード毎にインデックス化し，ファイルを Popularity-based アルゴリズムでランキング化する必要がある [1] [2]．このインデックス化およびランキング化のアプローチでは Web 空間の全探索を必要とし，膨大なストレージコストおよび演算コストを要する．上記の例では，Newton の多様な側面を各ユーザが全探索することなく取得可能であるべきである．

上記の課題を解決するにあたりファイル間の関係を記述するための Catalogue を提案する．Catalogue ではファイル間の関係を表現する際に，ファイルのグループ化と有向グラフによる関連付けを行う．一般的にファイルの集合があった場合，それらの間には複数の関係が潜在的に存在すると考えられる．さらに，ユーザがあるファイルをその特徴や分類に基づいてグループ化しそれらを参照可能にすることによって，閲覧者がそのファイルに対する見識を深められると考えられる．その際，ファイルのグループのみが提示され，自然言語による関係の説明がなされない場合であっても，ユーザは容易に分類意図を認識することが可能であると考えられる．例えば，画像ファイルのグループ {Isaac Newton, Aristotle, Albert Einstein} が与えられた場合，ユーザはそのグループが物理学者のグループであると解釈することが予想される．またグループ化に加えて，有向グラフを用いてファイル同士を関連付けることによって，より詳細な関係を示すことが可能である．例えば {Aristotle → Isaac Newton → Albert Einstein} のようなグループが与えられた場合，ユーザはこの順序は誕生順を表していると判断すると予想される．

本研究では，このような Catalogue をグローバル環境で自律分散的に保存・共有するための Catalogue システムを設計・構築した．Catalogue の汎用的なグラフ表現はファイルの所有者や種類，分野を問わず関連付けることができる．これらの Catalogue を共有することにより，ユーザはファイルの未知の側面を発見することが可能となる．例えばユーザ A が {Isaac

Newton, Abraham Lincoln, Winston Churchill} という猫好き人物の画像ファイルからなる Catalogue を作成し，ユーザ B が物理学者の Catalogue を作成したとする．本システムではユーザ A が Newton の画像ファイルに着目する際に，その画像ファイルを含むような上記の 2 つの Catalogue が参照可能となる．Newton の画像ファイルを含む Catalogue 数が増加すると，ユーザは多様な Catalogue 同士を比較可能となる．このような Catalogue 同士の比較によって着目ファイルのより詳細な情報が得られる．例えば関連ファイルの違いや，どのようなトポロジの Catalogue に含まれやすいのかといったこと，あるいは Catalogue をいつだれが作成したのかといった違いについて知ることができる．

本研究では，異なるユーザによる様々な目的で作成された Catalogue を用いて，多様なアプリケーションが動作することを想定している．アプリケーションの例としては仮想ミュージアムや Live 映像配信システムなどが挙げられる．仮想ミュージアムでは，各ミュージアムのキュレータが様々なミュージアムのデジタル収蔵品を多様な背景に基づいて収集し，各背景に基いて Catalogue 化してそれを共有する．ユーザがある絵画に着目している場合，その絵画を含んでいるような Catalogue を全て取得することが可能であり，それらの Catalogue を通してその絵画の多様な側面を知ることが可能となる．Live 映像配信では視聴者によって見たい映像が異なる可能性がある．そこで Catalogue によって動画のフレーム順序を記述し，保存された映像ファイルを組み合わせるだけで視聴者に最適化された映像を生成することを可能にする．

2. 関連研究

従来の検索エンジンでは Web ページ内の自然言語がインデックス化され，その自然言語を識別子とするようなファイルのグループが形成される．しかしながら，ユーザ間で自然言語の解釈・利用方法が異なるため，異なる意味のファイルが同じグループに属したり，同じ意味のファイルが別のグループとして扱われる可能性があり，一貫性のあるグループを形成することが困難である．また，ハイパーリンクを用いたファイル間の関係の記述では，ある HyperText Markup Language (HTML) ファイルを始点とするようなリンクはその HTML ファイルにしか記述できず，表現可能なトポロジは単一のエッジもしくは Star 型のみである．HTML ファイル内のテキスト情報で詳細な関係を記述することはできるが，その場合には自然言語処理が別途必要となる．また，文書内の自然言語のインデックス化や，ハイパーリンクによって記述された関係では，クローラによる Web 空間の全探索が必要となる．

インターネット上のファイルに対してユーザが自然言語のタグを付与して整理する Folksonomy [3] では，Web ページのインデックスと同様に自然言語を識別子としてグループを作成するため，一貫性のあるグループを形成することが困難である．また，タグを用いたファイルの整理では，ファイル間の対等性しか表せない．さらに，一般的にタグはあるサービス内で中央集中的に管理されているため，整理情報の資産性が保証されない．また，サービス間をまたぐようなファイルの関係を発見するためには，サービス間でタグを意味的に統一する必要がある．

セマンティックウェブでは，Resource Description Framework (RDF) 内の有向辺に自然言語で意味を定義した Uniform Resource Identifier (URI) を付与し Web 上のリソース間の関係を記述する [4]．しかしながら，エッジに付与された URI をもとにグループを形成すると，その URI の解釈や用法がユーザによって異なる可能性があるため，グループの一貫性を保証することは困難である．また，RDF ファイル内の複数のエッジは異なる意味グループ を表す可能性がありそれらを区別できないため，RDF のグラフ表現では単一のエッジもしくは HTML ファイルを中心とした Star 型のトポロジしか表現できない．さらに，RDF のエッジによる関係を得るためには，ハイパーリンクと同様に全探索する必要が生じる．

Pregel [5] や Trinity [6]，GPS [7] といった分散グラフ処理システムでは，グラフデータは解析者のもとで集中管理されているため，整理情報の資産性を保証することは困難である．また，特定のグラフアルゴリズムに最適化するために，グラフデータは密結合なサーバ群に保存される．

3. 設計

3.1 ファイルの整理における要求事項

Catalogue システムにおける要求事項として以下の 5 点が挙げられる．

- 各分類の差異の識別

従来の自然言語を用いたタグなどの分類では，検索の際にそのタグに一致するファイルがすべて取得されてしまう．ユーザ間のわずかな分類意図の違いを識別可能であることが重要である．

- 柔軟な分類表現

着目ファイルに関する多様な特徴を得るために，柔軟な分類表現が必要である．例えば，グループ内で階層構造や順序関係，対等関係等を表現できれば，単にファイルを収集してラベル付けされたグループよりも多くの情報が得られる．

- 任意のユーザによる自由な分類

HTML のようにファイル所有者による画一的な関連付けをするのではなく，着目ファイルの所有者でなくてもそれらを自由に分類可能にすることによって，ファイルの様々な側面を記述し取得することが可能になると考えられる．

- 全探索を回避した分類の参照

着目ファイルがどのような分類をされているかを知る上で，従来の検索エンジンのように情報ネットワークを全探索するのではなく，そのオブジェクトの識別子から直接的に分類情報を取得できるようにする必要がある．このようにグループ一覧を効率的に取得することでそれらの比較を可能にし，着目ファイルの特徴が取得可能になると考えられる．

- ユーザによる自由なプロパティ付加

従来のファイルシステムでは，ファイルの作成日やサイズといったプロパティは画一的に付加される．ファイルをグローバ

ルに整理していくうえで，ファイルの所有者に依存せずにタイトルやディスクリプションといったプロパティを自由に付加していく仕組みが必要である．ただし，プロパティはファイルの検索やグループ化に用いるのではなく，あくまでアプリケーション内でのファイルの並び替えや補助的な説明等に用いるものとする．

上記の要求事項を解決するにあたり，以下のようなアプローチに従ってシステムを設計した．アプローチの詳細については 3.2 節で述べる．

- 自然言語を用いないグローバルなグループ識別子

ファイルの分類意図の違いを識別可能にするために，自然言語のラベルを用いずに，各グループに対してグローバルな識別子を付与する．

- 有向グラフを用いた柔軟な分類表現

多様な関係を表現可能にするために有向グラフを使用する．この有向グラフによるファイルの整理表現を Catalogue と呼ぶ．

- Catalogue とファイルの管理構造を分離

任意のユーザがファイルを分類可能にするために，Catalogue とファイルの実体を分離して管理する．

- 閲覧ファイルからの逆引き機能

閲覧ファイルがどのような Catalogue に属しているかという逆引きの対応関係を自律分散的に管理する．本研究では，ある Catalogue に含まれる有向グラフとファイルの識別子を取得する操作を正引きと呼ぶ．

- プロパティをファイルとして保存

プロパティを任意のユーザが動的に記述できるように，プロパティをファイルとして保存しそれらの関係を Catalogue で記述する．

本研究では，“オブジェクト (*Object*)” はファイル (*File*) もしくは Catalogue のいずれかであるものと定義する．Catalogue はファイル同士だけでなく，ファイルと Catalogue の関係や Catalogue 同士の関係を記述することができる．また，着目している Catalogue からその Catalogue を含むような Catalogue を参照可能である．Catalogue は有向グラフを用いてオブジェクト同士の関係を記述し，有向グラフのサイズに制限はないものとする．Catalogue の構造の詳細は 3.3.2 項で述べる．

図 1 はオブジェクトの構成概要および，ファイルの所有者 (*File Owners*), Catalogue 作成者 (*Cataloguers*), クライアント (*Clients*) の 3 種類のユーザの関係を表している．ファイルの所有者はファイルを作成しインターネットで共有可能にする．各々の共有されたファイルはファイルの所有者によって管理され，グローバルユニークな ID で識別されるものとする．また，Catalogue 作成者は複数のオブジェクトを選択して Catalogue を作成し，ファイルと同様にインターネットで共有可能にする．Catalogue もファイルと同様に Catalogue 作成者自身によって管理され，グローバルユニークな ID で識別される．オブジェクトの ID の詳細については 3.3.1 項で述べる．クライアントはオブジェクト (ファイルもしくは Catalogue) をそれぞれ ID を指定することで取得することが可能である．また，クライアントは着目しているオブジェクトを含むような Catalogue の

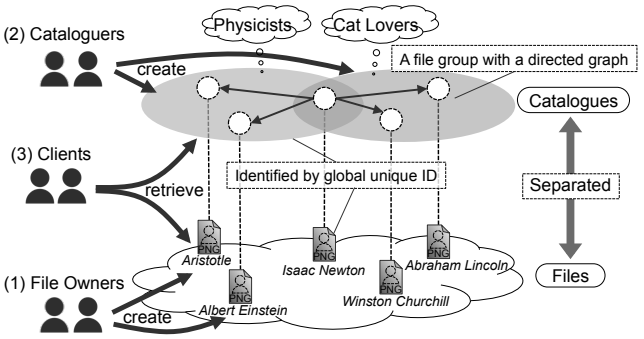

図 1　オブジェクトおよびユーザの関係の概要．

一覧を参照可能であり，着目オブジェクトに接続する有向グラフの集合も取得可能である．Catalogue システムの動作概要については 3.5 節で述べる．

3.2　アプローチ

3.2.1　自然言語を用いないグローバルなグループ識別子

Collaborative Tagging [8] のような既存のグループ化手法ではタグと呼ばれるテキストラベルを用いてファイルを分類する．例えば YouTube や Flickr, Del.icio.us. などのサービスでは，投稿された動画や画像ファイルなどに対してタグを付与することで，それらのファイルの特徴を明示し，分類して検索可能にする．このようなタグを用いたファイルの分類では，自然言語の曖昧性 [9] によって本来異なるグループ同士の混同が生じる可能性がある．そのため，複数の意味をもつキーワードでファイルを検索すると，検索結果としてファイルの集合が未分類なまま返される可能性がある．例えば，Flickr で “apple” というキーワードで検索すると “果実のリンゴ” という意味での画像のグループと，“Apple 社” という意味での画像のグループが混同された状態で出力される．これは “apple” という異なる意味を持つテキストラベルがグループの識別子として用いられているためである．さらに，果物のリンゴのなかにも多様な品種が存在し，味や色，大きさ，価格などは様々である．専門家が植物学的な側面で分類し菓子職人が味に基いて分類する場合であっても，同一のテキストラベルが用いられれば同じグループとなる．このように混同されたグループを複数に切り分けることは容易ではない．

また，自然言語の意味は時間の経過とともに変わる可能性があり，単語の意味が中世と現在で異なるものが存在する [10] [11]．例えば，“dangerous” という形容詞は 15 世紀では “difficult” のような意味で用いられていた [12]．意味の混同したグループ同士の比較は，一貫性のあるグループ同士の比較よりも特徴の理解が薄れると考えられる．セマンティックウェブでは語彙で単語の意味が定義されているが，ユーザ間で完全に同一の意味・概念を共有することは困難である．したがって，単語の意味の一貫性を保ち続け新しい概念を採用していくという過程は困難であると考えられる．

そこで，本研究では Catalogue を識別する際に自然言語を使用せず 128-bit の ID でグローバルに識別する．一貫性のある数値識別子を用いることにより，着目オブジェクトを含むよ

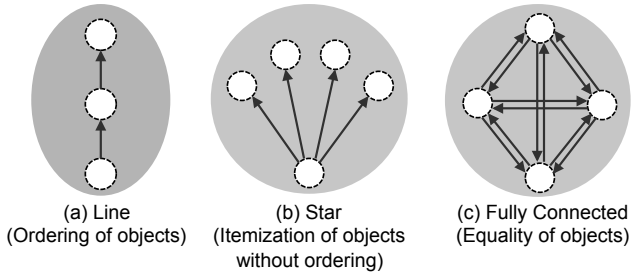

図 2　3 つの典型的な Catalogue の例.

うなカタログをそれぞれ区別して列挙可能となる．識別子には Catalogue の所有者やバージョンの情報が含まれており，時間を経てもその情報を保持しておくことができる．Catalogue の識別子の詳細については 3.3.1 項で述べる．

3.2.2　有向グラフを用いた柔軟な分類表現

数値識別子だけでは Catalogue 内のオブジェクト間の柔軟な関係，例えば順序関係や対等関係，階層構造などを表現することは困難である．インターネット上に Isaac Newton の肖像画のファイルが複数あった場合，ファイルを肖像画が描かれた年代順に並べたり，共通の画家の関係を表せるような仕組みが必要である．

本研究では Catalogue における柔軟な関係表現のために有向グラフを用いる．グラフ理論では，無向グラフおよび有向グラフの 2 種類のグラフが存在する．無向グラフは対等性を表すのには適しているが，順序や階層構造を表すためには付加情報が必要なため適していない．一方で，有向グラフの場合は簡潔に順序関係等を表現でき，対等関係は双方向のグラフを用いることで表現可能である．そこで本研究では，Catalogue を有向グラフで構成する．各エッジの始点および終点ノードはオブジェクトの識別子からなる．このような Catalogue 同士を組み合わせることによって多様な表現が可能になる．

図 2 は Catalogue のトポロジの例 (*Line*, *Star*, *Fully Connected*) を表している．*Line* トポロジの Catalogue は着目オブジェクトの順序関係等を表すことができる．例えば，オブジェクトの作成日やある地点からの距離，動画のフレームの順序などを記述できる．*Star* トポロジの Catalogue はあるオブジェクトを起点としてそれに関係するオブジェクトの一覧を表示できる．これは HTML ファイルのハイパーリンクに類似した関係表現である．*Fully Connected* トポロジの Catalogue では，各オブジェクトが双方向の有向グラフで接続されている．双方向の有向グラフはオブジェクト同士の対等関係を表しており，これは Collaborative Tagging のタグ表現と類似している．Catalogue のトポロジの選択は各アプリケーションが自由に選択し，トポロジの意味の解釈はユーザに委ねられる．

また，ファイル同士だけでなく Catalogue 同士の関係を記述することで階層構造を表現することも可能である．例えば，図 3 では “哺乳類 (*Mammal*)” と “爬虫類 (*Reptile*)” という Catalogue を生物学的に分類した Catalogue を表している．階層的な分類では，一般的にあるグループがいくつかの特徴的なサブグループへ分割され，そのように細分化されたグループは

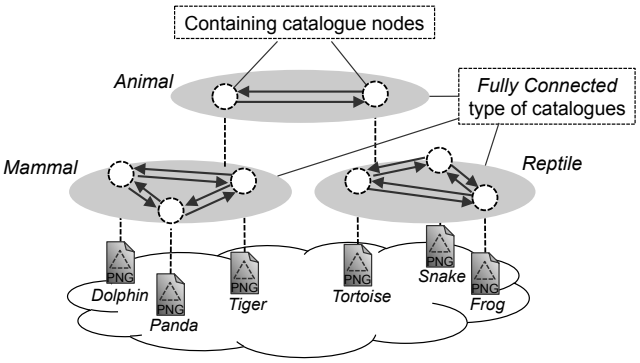

図 3　Catalogue を包含することによる階層的な Catalogue 表現.

より多くの知見をもたらすと考えられる．3.2.1 項で述べたように，ここで重要なのは Catalogue 作成者が Catalogue に対して自然言語のタグを付与する必要はないという点である．タグが付与されない場合，あるユーザが “動物 (*Animal*)” という意味で作成した Catalogue を “脊椎動物 (*Vertebrate*)” と解釈可能であり，その Catalogue を “無脊椎動物 (*Invertebrate*)” と関連付けることも可能である．このようにファイルと同様に Catalogue の解釈は自由であり，自然言語によってグループの意味を定めるのではなく，他のオブジェクトとの関係によってその意味が定まっていくことを想定している．

3.2.3　Catalogue とファイルの管理構造を分離

任意のユーザがオブジェクトを分類可能にするために，Catalogue とオブジェクトを分離して管理する．HTML のハイパーリンクは関連付けの点において Catalogue の有向グラフと類似しているが，ハイパーリンクは HTML ファイル内に埋め込まれているという点で異なる．このファイルの実体とリンクの一体化には以下の 3 つの問題が存在する．まず，HTML では所有者のみがハイパーリンクで他のファイルとの関係を記述可能である．また，ハイパーリンクは HTML ファイル自身を始点とする必要がある．さらに，HTML ファイル内で自然言語を用いてハイパーリンク間の順序関係や対等関係等を説明することは可能であるが，それらの関係を識別するためには別途，自然言語の解析が必要となる．

そこで，本研究では Catalogue をファイルから分離して作成・管理することにより多様な Catalogue を生成可能にする．多様で膨大な数の Catalogue が生成されれば，着目オブジェクトの新しい側面を知ることが容易になると考えられる．また，Catalogue の種類や数が増えることにより，エッジの重複を計算することで関係の人気度や重要度といった統計情報を得ることが可能になる．

本研究では Catalogue は作成者の資産であるものとし，作成者自身が Catalogue を管理できるようにする．Catalogue とファイルの分離によって Catalogue 作成者は柔軟な管理が可能になると考えられる．例えば，Catalogue 作成者は Catalogue をどのサーバに保存するかや，クライアントからの Catalogue に対するアクセス制限を設けることができる．Catalogue 管理の詳細については 3.5 節で述べる．

3.2.4 閲覧ファイルからの逆引き機能

逆引きはあるオブジェクトを含むような Catalogue を全て検索するというオペレーションである．逆引きによって着目オブジェクトを含むような Catalogue の一覧を取得し，それらを比較することが可能になる．そのため，逆引きはユーザがあるオブジェクトの特徴を得る際に有効である．本研究では，あるオブジェクトを含むような Catalogue を *Parent Catalogue* と呼び，ある Parent Catalogue に含まれるオブジェクトのことを *Child Object* と呼ぶ．

Parent Catalogue は Child Object の識別子の情報をもつため，ある Parent Catalogue からその Child Object を見つけることは容易である．一方で，あるオブジェクトの Parent Catalogue をすべて発見することは容易ではない．なぜなら 3.2.3 項で述べたように，Catalogue システムにおいて Parent Catalogue は Child Catalogue と分離して作成・管理されるためである．

そこで，本研究では Child Object とその Parent Catalogue の対応関係を，Child Object が保存されているドメインで分散的に管理する．これにより，クライアントが着目オブジェクトにアクセスする際に，そのオブジェクトの Parent Catalogue の一覧を得ることができる．また，Parent Catalogue との対応関係に加えて Child Obejct に接続するエッジも分散保存することで，着目オブジェクトから直接，関連オブジェクトを発見することが可能となる．この分散管理された有向グラフを用いれば，オブジェクトの所有者が自身のオブジェクトに関するグラフ解析をすることが可能となる．

オブジェクトの所有者が Catalogue システムに参加し，上記の対応関係を管理することによって，彼らのオブジェクトがより多くの Catalogue に含まれると，逆引きを経て彼らのオブジェクトに対するアクセスが高まると推定される．逆にあるオブジェクトが少数の Parent Catalogue にしか含まれない場合，そのオブジェクトの知名度は低くアクセス数は少なくなると考えられる．そのため，逆引きのためのリソース確保が軽減されると考えられる．

3.2.5 プロパティをファイルとして保存

タイトルやディスクリプションといった文脈に応じて変化するメタデータは，Exif や IPTC のようにファイルに直接埋め込むのではなく各ユーザが自由に付加できるようにする必要がある．そこで本研究では図 4 に示すように，プロパティをオブジェクトとは別のファイルとして保存し Catalogue を用いて紐付ける．このプロパティを含むような Catalogue を *Property Catalogue* と呼ぶ．本研究では Catalogue はファイルと分離して作成・管理されるため，あるファイルに対して複数のプロパティを付加することが可能となる．また，Catalogue はグローバルユニークに識別され，ネスト構造で記述できるため，他のユーザが作成した Property Catalogue を再利用することが可能となる．

本研究では，ファイルに紐づけられたプロパティはそのファイルを検索したりグループ化するために用いるのではなく，

表 1　重要度・普遍性に応じたプロパティの分類．

Importance	Property		How to manage	
Additional	Property of catalogue	Time, geo, description, thumbnail, manual, ...	Associating with property file	Catalogue Region
		Version (branch), history, ...	Expressed by catalogue	
Basic		Created date, last accessed, ...	Catalogue Server, Graph Manager	
Essential		Organization, owner, version (no branch), whether normal catalogue, ...	Global Catalogue ID	
Additional	Property of file	Time, geo, description, thumbnail, ...	Associating with property file	
		Version (branch), history, ...	Expressed by catalogue	
Basic		Created date, file size, last accessed, extensions, ...	File Manager	File Region
Essential		Organization, owner, version (no branch), ...	Global File ID	

Catalogue では提示されていないオブジェクトの側面をアプリケーションで提示する際等に用いられる．図 4 の例では Isaac Newton の画像ファイルに対して “物理学者” と “猫好き” というタイトルが付加されており，アプリケーションでオブジェクトと共に提示することで特徴を理解しやすくできる．その他にも，時間情報や位置情報を付与することによりオブジェクトの配置方法や可視化方法を変えてユーザに対して新たな側面を提示するために用いられる．

プロパティファイルをオブジェクトから分離することによって，どのような属性のプロパティが記述されているのかを知るためには，プロパティファイルを取得しなければならないという問題が生じる．そこで，各エッジに対して Edge Attribute を付与することによって，そのエッジの終点プロパティファイルがどのような属性のプロパティをもつか記述する．Property Catalogue における Edge Attribute の取り扱いについては 3.4.1 項で述べる．

また，プロパティを別ファイルとして保存することによってプロパティを参照するためのコストは大きくなるため，プロパティの重要度や普遍性に応じて異なる管理方法をとる．表 1 は重要度と普遍性に応じたプロパティの分類を表している．オブジェクトの所有者やバージョンといった *Essential* と呼ばれるプロパティは普遍的でありかつ参照頻度が高くなると考えられるため，Global Unique ID (GUID) に埋め込むことによって管理する．また，オブジェクトの作成日や最終アクセス時刻，ファイルの拡張子といった *Basic* と呼ばれるプロパティは普遍的ではあるが，Essential なプロパティよりも参照頻度が低いと考えられるため，各オブジェクトの管理サーバ (3.5 節で述べる) で保存される．さらに，*Additional* なプロパティは 2 種類の管理方法に分けられる．1 つ目は前述したようなプロパティファイルとオブジェクトを Catalogue を用いて紐付ける方法で，プロパティの例としてはオブジェクトに関する時間情報や地理的情報，サムネールなどが挙げられる．2 つ目は Catalogue のグラフ構造を用いてオブジェクトのプロパティを提示する方法であり，例えばブランチを含むようなバージョン情報や履歴情報などが挙げられる．

3.3 Catalogue による関係表現

Catalogue システムではファイルと Catalogue はそれぞれ *Global File ID (GFID)* および *Global Catalogue ID (GCID)*

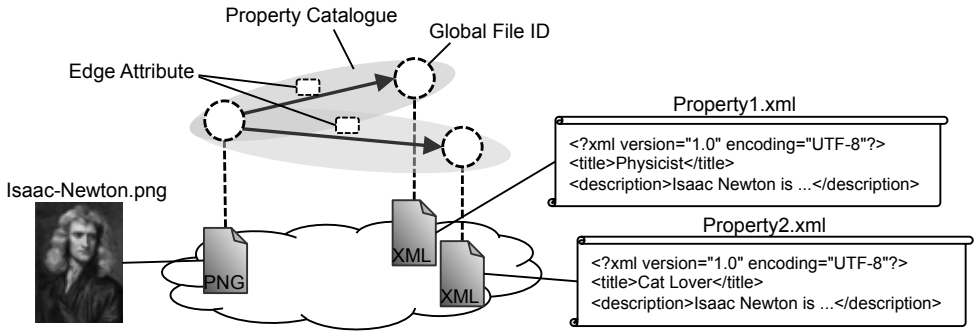

図 4　Catalogue によるオブジェクトとプロパティファイルの紐づけ.

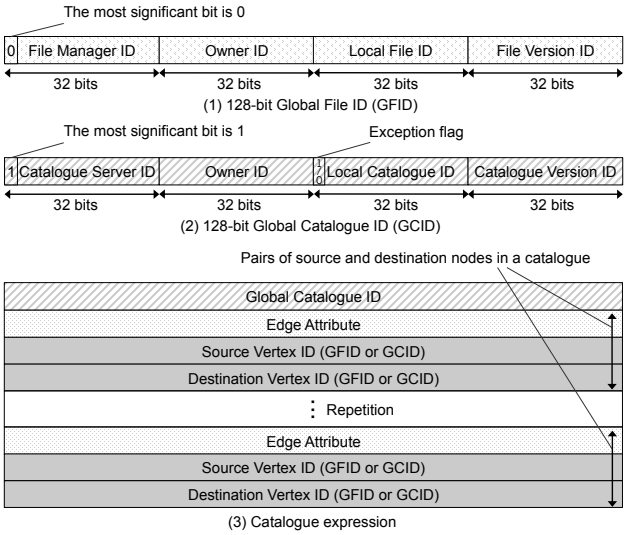

図 5　グローバルユニーク識別子と Catalogue の構成.

によってグローバルに識別される．以下では，これらのグローバル識別子とそれらから構成される Catalogue 構造について述べる．Catalogue の種別とそれらの Catalogue 内での Edge Attribute の取り扱いについて述べる．

3.3.1　各種識別子

図 5 (1), (2) はそれぞれ GFID および GCID の構造を表している．GFID, GCID はともに 128-bit の識別子であり，最上位ビットはそれぞれ GFID であるか GCID であるかを識別するために用いられ，0 は GFID であること，1 は GCID であることを示す．GFID と GCID の構造は同じコンセプトに基づいているため，以下では GCID の詳細についてのみ述べる．

GCID における最上位ビットを含む 32-bit の領域を *Catalogue Server ID* と呼ぶ．Catalogue Server ID はその Catalogue を管理する Catalogue Server の識別子である．本研究では，Catalogue Server ID から IP アドレスへの変換機能があることを前提としている．Catalogue Server の詳細については 3.5.1.2 で述べる．

2 つ目の 32-bit の領域は *Owner ID* と呼ばれる．Owner ID は各 Catalogue Server における Catalogue 作成者を識別するために用いられる．Owner ID を用いれば，クライアントは逆引きによって得られた Catalogue の一覧から Catalogue 作成者に基づいて Catalogue を選択することが低コストで実現可能となる．

3 つ目の 32-bit の領域は *Local Catalogue ID* と呼ばれ，ある Catalogue 作成者が作成した Catalogue を識別するために用いられる．Local Catalogue ID の最上位ビットは *Exception Flag* と呼ばれ，その Catalogue が *Normal Catalogue* であれば 0 を，*Exception Catalogue* であれば 1 を設定する．Normal Catalogue と Exception Catalogue の詳細については 3.3.3 項で述べる．

最後の 32-bit の領域は *Catalogue Version ID* と呼ばれ，Catalogue のバージョンを識別するために用いられる．Catalogue Version ID を用いれば，クライアントは Catalogue Server にアクセスすることなく低コストで Catalogue のバージョンが異なることを認識できる．

3.3.2　Catalogue の構成

図 5 (3) は Catalogue の構成を表している．各々の Catalogue は GCID が割り当てられ，Edge Attribute および始点・終点ノードの識別子を 1 組以上もつ．ノード識別子は GFID もしくは GCID のいずれかであるものとする．このようにエッジのノードに Catalogue も配置可能であることから，Catalogue を用いて Catalogue 同士の関係を記述することにより階層構造を表現することが可能となる．また，Catalogue は有向辺の繰り返しによって記述されており，そのサイズに制限はないものとする．

3.3.3　Catalogue の種別

図 6 は Catalogue, ファイルおよびオブジェクトの種別を示している．本研究では画像や動画といった通常のファイルのことを *Normal File* と呼び，それらの関係を記述した Catalogue を *Normal Catalogue* と呼ぶ．Normal Catalogue は Exception Flag が 0 に設定される．また，Normal File と Normal Catalogue を共に *Normal Object* と呼び，Normal Catalogue 同士の関係を記述した Catalogue も Normal Catalogue であるものとする．

一方，プロパティファイルや 3.4.3 項で述べるマニュアルファイルは *Exception File* と呼ばれる．本研究では，Normal File と Exception File は Global File ID によって区別しない．例えば，プロパティファイルは解釈次第では通常のファイルとして扱うことが可能であるものとする．3.2.5 項で述べた Property Catalogue のように，Normal Object 同士の関係で

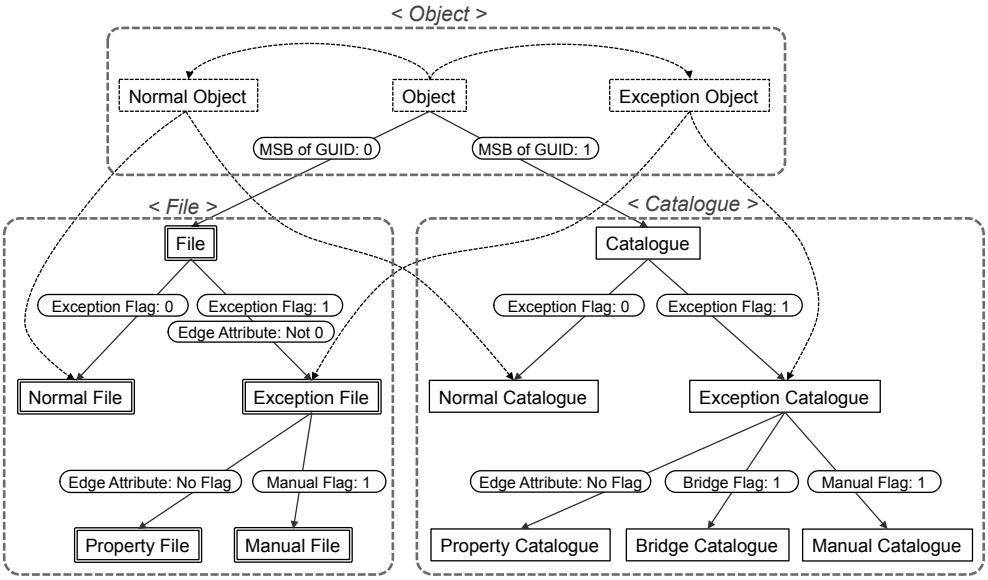

図 6　Catalogue, ファイルおよびオブジェクトの種別.

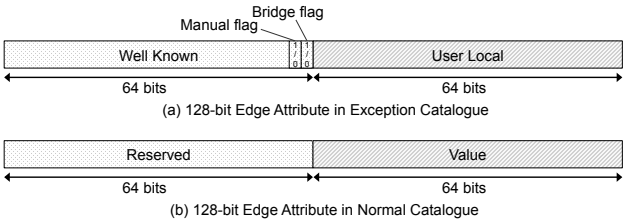

図 7　Exception Catalogue と Normal Catalogue における Edge Attribute の構成.

はなく Exception File との関係を記述している Catalogue を *Exception Catalogue* と呼び，Exception Flag を 1 に設定する．また，Exception Catalogue と他のオブジェクトとの関係を記述した Catalogue も Exception Catalogue であるものとする．ただし，例外として 3.4.3 項で述べる Manual Catalogue 同士の関係を記述した Catalogue は，Exception Catalogue 同士の関係を記述しているが Normal Catalogue であるものとする．

3.4　特殊な Catalogue と Edge Attribute の振る舞い

Exception Catalogue には図 6 に示すように，*Property Catalogue*, *Bridge Catalogue*, *Manual Catalogue* の 3 種類が存在する．また，Edge Attribute は Catalogue 内の各エッジに付与される 128-bit のデータであり図 7 に示すような構造をもつ．以下では 3 つの Catalogue の詳細と，それらの Catalogue における Edge Attribute の役割について述べる．

3.4.1　Property Catalogue

3.2.5 項で述べたように，あるオブジェクトに対して任意のユーザがプロパティを付加可能にするために，本研究では Property Catalogue を用いてオブジェクトとプロパティファイルを紐付ける．このように，プロパティファイルをオブジェクトから分離することによって，どのような属性のプロパティが記述されているのかを知るためにはプロパティファイルを取得しなければならないという問題が生じる．そこで Property Catalogue では Edge Attribute を用いてエッジの終点プロパティファイルがどのような属性であるかを示す．Property Catalogue 内で使用される Edge Attribute は図 7 (a) に示すように，Well Known と User Local と呼ばれるそれぞれ 64-bit の 2 つの領域からなる．

1 つ目の Well Known 領域は Catalogue システム全体で予め定義されたプロパティを識別するために用いられる．同時に複数の属性のプロパティをプロパティファイルに記述できるように，Well Known 領域内では各属性毎にビットフラグを用いる．ただし，下位 2 ビットの Bridge Flag と Manual Flag は後述される Bridge Catalogue と Manual Catalogue をそれぞれ識別するために用いられ，Property Catalogue 内では 0 に設定される．

2 つ目の User Local 領域はアプリケーション作成者がプロパティの種類を独自に識別するために用いられる自由な領域である．例えば，動画視聴サイトで動画のジャンルのようなアプリケーション固有のプロパティを用いる際に，アプリケーション内で識別可能にするために用いられる．

3.4.2　Bridge Catalogue

オブジェクトとプロパティファイルを紐付ける際には，図 8 のように Normal Catalogue と Property Catalogue を (a) 一体化させるか (b) 分離するかの 2 種類が考えられる．(※ 以降の Catalogue の例では便宜上，Catalogue 内に自身の GCID を始点とするエッジを含んでいる．)

図 8 (a) のような Normal Catalogue と Property Catalogue を一体化させる方法では，Normal Catalogue と Property Catalogue を対応付ける必要がないため記述方法は簡潔になる．しかしながら，この方法ではある Catalogue で記述されているオブジェクト同士の関係を別のユーザが再利用しようとした際に，そのユーザがプロパティファイルだけを変更したいような場合，異なる ID の Catalogue を作成する必要がある．

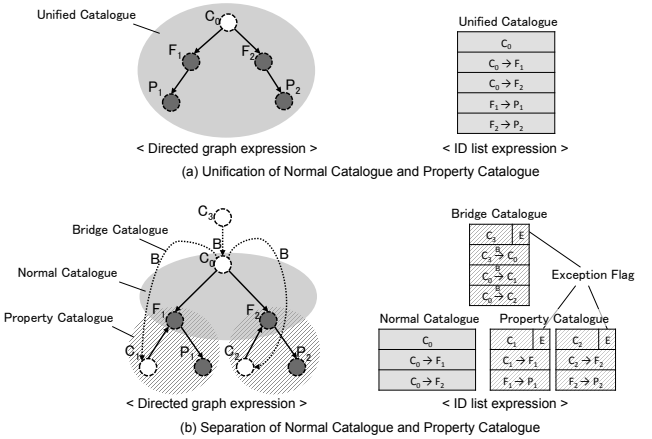

図 8　Bridge Catalogue による Normal Catalogue と Property Catalogue の対応付け.

一方で，Normal Catalogue と Property Catalogue を分離する方法ではこれらを対応付ける必要がある．本研究では図 8 (b) に示すような *Bridge Catalogue* と呼ばれる Catalogue を用いる．Bridge Catalogue を用いることで Catalogue の記述方法はやや煩雑になるが，Normal Catalogue および Property Catalogue をそれぞれ再利用することが可能となり，プロパティファイルだけを変更したい場合であっても Bridge Catalogue を変更することで Normal Catalogue を再利用することができる．このような Catalogue の再利用によってもたらされる Catalogue の被参照情報は有益な統計情報となると考えられる．

図 8 のように Bridge Catalogue は Normal Catalogue の GCID と各 Property Catalogue の GCID を有向辺で対応付ける．Bridge Catalogue の Exception Flag は 1 に設定され，Bridge Catalogue 内の全 Edge Attribute の Bridge Flag は 1 に設定される．

3.4.3　Manual Catalogue

即時性が要求されるアプリケーションでは Edge Attribute を用いてオブジェクトの “特徴の属性” を明示するだけでなく，“特徴の値” も明示したい場合がある．例えば動画配信システムにおいて動画フレームの順序を Catalogue で記述しているとする．図 9 のようにユーザの受信環境に合せて 4K, 2K, HD のような複数の解像度の画像ファイルを用意し，同一フレームで解像度の異なる画像ファイルのグループを Catalogue で記述したとする．もし図 9 (a) のようにプロパティファイルだけでフレームの解像度を識別する場合，受信者はある解像度の動画を再生するために各フレーム毎にプロパティファイルを読み込む必要がある．これは Property Catalogue 内の Edge Attribute はプロパティファイルに記述されている “特徴の属性” しか示していないためである．そのため，エッジによって “特徴の値” を明示する必要がある．

本研究では，図 9 (b) に示すように Normal Catalogue 内で Edge Attribute に値を設定し，その値の意味をマニュアルファイルと呼ばれる XML ファイルに記述することで，各エッジの終点オブジェクトの “特徴の値” を明示する．Normal Catalogue 内での Edge Attribute の振る舞いは Exception Catalogue 内

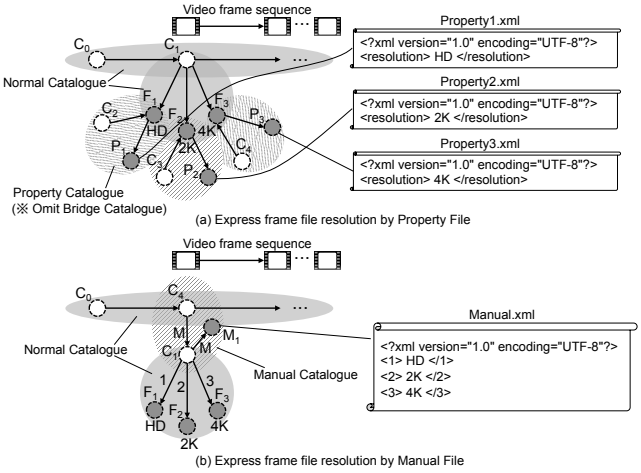

図 9　マニュアルファイルと Edge Attribute を用いたオブジェクトの特徴説明.

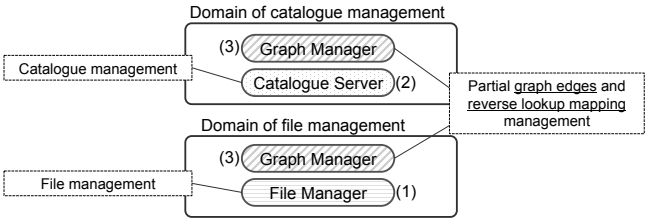

図 10　Catalogue システムにおける 3 つのモジュール.

での振る舞いとは異なり，図 7 (b) に示すように，Reserved と Value と呼ばれるそれぞれ 64-bit の 2 つの領域で構成される．Normal Catalogue 内で Edge Attribute を用いる際は，基本的に Value 領域に値が設定され，マニュアルファイルでその意味を記述するものとする．マニュアルファイルは説明対象 Catalogue に紐づけられて *Manual Catalogue* と呼ばれる Catalogue によって管理される．

3.5　Catalogue システム

3.5.1　構成モジュール

Catalogue システムは図 10 に示すように *File Manager*, *Catalogue Server*, *Graph Manager* の 3 つのモジュールから構成される．ファイルの管理ドメインでは，File Manager によってファイルが管理され，Graph Manager によってそのファイルに接続するようなエッジ及び，そのファイルと Parent Catalogue の対応関係が管理される．また，Catalogue の管理ドメインでは，Catalogue Server によって Catalogue が管理され，ファイルの場合と同様に，Graph Manager によってその Catalogue に接続するようなエッジ及び，その Catalogue と Parent Catalogue の対応関係が管理される．

3.5.1.1　File Manager

File Manager はファイルの管理ドメインに配置され，主にファイルの書き込み・読み込みの処理を制御する．基本的な機能としては，ファイルの所有者が GFID を指定してファイルを書き込むとそのファイルが作成され，クライアントがその GFID を指定してファイルを要求するとファイルが返却される．クライアントが File Manager を発見する際には，GFID

内の File Manager ID が File Manager の IP アドレスに変換され，適切な認証・認可を経てファイルにアクセスされるものとする．また Catalogue システムでは，当研究室で開発された Content Espresso [13] をストレージのモデルとしているが，本質的にはストレージシステムの構成には依存せず，あくまで File Manager というモジュールを介してファイルへアクセス可能であればよいものとする．

3.5.1.2 Catalogue Server

Catalogue Server は Catalogue の管理ドメインに一つ配置され，Catalogue の書き込み・読み込みといった処理を制御する．Catalogue Server は Catalogue Server ID で識別される．本研究では，Catalogue Server ID の割り当てができれば Catalogue 作成者は自由に Catalogue Server を設置し公開可能であるものとする．このように Catalogue 作成者が自身の Catalogue Server を運用することによって，自分の Catalogue に対してアクセス制御を設定することができ，Catalogue の資産性が確保可能となる．また，Catalogue 作成者が Catalogue Server を運用することが技術的に困難な場合は，Gmail や Hotmail といったメールのホスティングサービスと同様に，他の信頼できるサービス・プロバイダに Catalogue の管理を委託することも可能である．その際，3.3.1 項で述べたように Owner ID によって Catalogue 作成者が識別される．

3.5.1.3 Graph Manager

Graph Manager は逆引きの対応関係と，Catalogue の部分グラフを管理する．Graph Manager はファイルの管理ドメインと Catalogue の管理ドメインの両方に配置される．Graph Manager は 2 種類の情報， 1) Child Object と Parent Catalogue の逆引きの対応関係と，2) ドメイン内で保有するオブジェクトに接続するエッジを管理する．

Catalogue 作成者が新規に Catalogue を作成すると，その Catalogue における逆引きの対応関係と部分グラフが Graph Manager に送信される．逆引きの対応関係は Child Object のグローバルユニーク ID とその Parent Catalogue の GCID からなる．この対応関係によってクライアントは着目オブジェクトを含むようなすべての Parent Catalogue を参照可能となる．また，ドメイン内のオブジェクトを始点もしくは終点にもつようなエッジが管理されるため，クライアントがある着目オブジェクトからそれに関係するオブジェクトを参照可能となる．本研究では，Graph Manager の IP アドレスは各オブジェクトの Server ID，すなわち Catalogue Server ID もしくは File Manager ID，から変換されるものとする．

3.5.2 動作概要

図 11 は Catalogue の書き込みと読み込み処理の流れを表している．まずはじめに Catalogue 作成者が GCID X の Catalogue を作成する (図 11 (i))．作成された Catalogue はその Catalogue 作成者が設置した Catalogue Server に保存される．その際，3.5.1.3 で述べた規則に従って，逆引きの対応関係と Catalogue 内の部分グラフを Graph Manager へ送信する (図 11 (ii))．例えば，(GFID A → GFID B) というエッジは GFID A と GFID B のファイルを管理するドメインの Graph Manager へそれぞれ送信される．

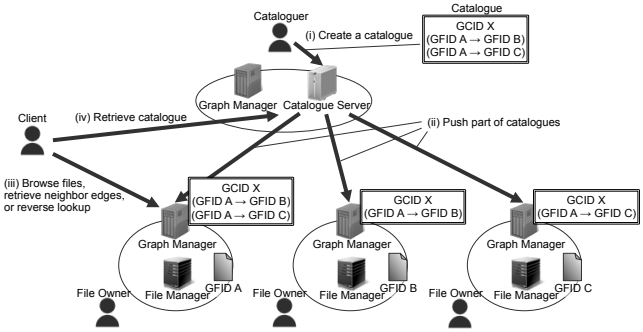

図 11　Catalogue 書き込み・読み込み処理の流れ．

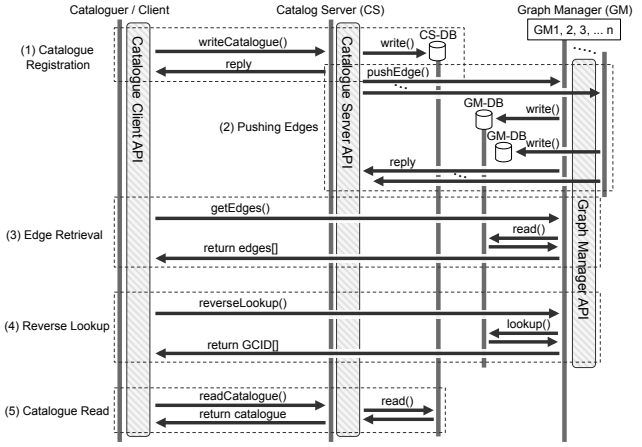

図 12　3 つの API 間におけるメッセージシーケンス．

また，クライアントは着目ファイルに接続しているエッジを検索したり，着目ファイルを含むような Catalogue を逆引きできる (図 11 (iii))．クライントは逆引きで得た GCID を指定して Catalogue Server に要求することによって，オリジナルの Catalogue を取得することも可能である (図 11 (iv))．

3.5.3 メッセージシーケンス

図 12 は Catalogue の書き込みと読み込み処理におけるメッセージシーケンスを表している．Catalogue の書き込み・読み込みをする際に各モジュール用に，Catalogue Client API, Catalogue Server API, Graph Manager API の 3 つを設計した．各処理の詳細について以下で述べる．

3.5.3.1 Catalogue Registration Procedure

Catalogue の書き込みでは，まずクライアントが `writeCatalogue()` 関数を用いて，GCID と始点・終点のノード識別子の対を Catalogue Serve へ送信する．その後，Catalogue Server は `write()` 関数で CS-DB に Catalogue を保存し，直ちにクライアントに対して完了を通知する．

3.5.3.2 Pushing Edges Procedure

Catalogue が Catalogue Server に書き込まれると，Catalogue Server は逆引きの対応関係と Catalogue 内のエッジを `pushEdge()` 関数を用いて Graph Manager へ送信する．各 Graph Manager は逆引きの対応関係とエッジを GM-DB へ保存し，Catalogue Server へ完了を通知する．

3.5.3.3 Edge Retrieval Procedure

クライアントが着目オブジェクトに接続するエッジを取得する際には，`getEdges()` 関数を用いてオブジェクトの ID とエッジの向き (Incoming / Outgoing) を指定して Graph Manager にオブジェクトを要求する．その後，Graph Manager は `read()` 関数を用いて GM-DB から指定されたオブジェクトに接続するエッジを検索しその結果を返却する．その際 Catalogue 毎にエッジは区別され，エッジとともにそのエッジをもつ Catalogue の GCID も伴って返却する．

3.5.3.4 Reverse Lookup Procedure

クライントが逆引きをする際には，`reverseLookup()` 関数を用いて着目オブジェクトの ID を指定して Graph Manager に GCID のリストを要求する．その後，Graph Manager は指定されたオブジェクトの Parent Catalogue を GM-DB から `lookup()` 関数を用いて検索し，その結果を返却する．

3.5.3.5 Catalogue Read Procedure

クライアントが Catalogue を読み込む際には，`readCatalogue()` 関数を用いて GCID を指定して Catalogue Server に Catalogue を要求する．その後，Catalogue Server は `read()` 関数を用いて CS-DB から Catalogue を検索しクライアントへ返却する．

4. Catalogue システムの応用

4.1 仮想ミュージアム

従来のミュージアムでは，各ミュージアムのキュレータによって収蔵品が画一的に関連付けられて展示されてきた．絵画などの収蔵品をデジタル化してインターネット上で公開したとしても，自然言語を用いたタグによって整理されることが一般的である．

Catalogue システムを用いた仮想ミュージアムでは，任意のユーザが Catalogue によってファイルをキュレーション可能であり，キュレーション情報は作成した各自が所有可能である．また，逆引き機能によってクライアントは閲覧ファイルからそのファイルを含むような Catalogue を網羅的に取得することが可能となる．

図 13 は当研究室で開発された Campus Museum と呼ばれる仮想ミュージアムアプリケーションである．図 13 (a) は Campus Museum のメインページであり，中央の画像はユーザが着目しているファイルである．着目ファイルの下にはディスクリプションが表示されており，これは閲覧中の Catalogue に応じて変化する．メインページの下にはユーザが着目している Catalogue 内の画像が表示され，同一グループ内のファイル同士を比較できる．メインページを下にスクロールすると図 13 (b) のように着目ファイルから逆引きされた Catalogue の一覧が表示される．これによって，着目ファイルを含む Catalogue 同士を比較することができる．また，図 13 (c) のようにプロパティを用いて位置情報に基づいて可視化することで，有向グラフによるファイルの接続だけでは表現できない側面を提示することが可能となる．

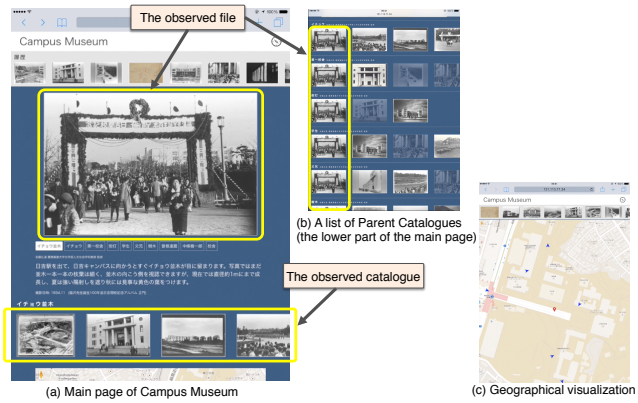

図 13 Campus Museum.

4.2 グローバルな Live 映像配信

Live 映像配信では視聴者によって見たい映像が異なる可能性がある．例えばサッカー中継において，視聴者がある選手の成績情報を知りたい場合もあれば，サッカーのルールを知りたい場合もある．また，フィールドを俯瞰して見たい場合や，選手の脚さばきを拡大して見たい場合もある．また，スマートフォンのようなモバイル端末で低画質の動画のみ視聴可能な場合もあれば，自宅の光回線を介して高画質な動画を視聴可能な場合もある．

従来の Live 映像配信の手法では視聴者毎にこのような最適な映像を配信することは困難である．Live 映像配信の手法には (1) 放送波による配信と，(2) インターネットを介した配信が考えられる．(1) 放送波による配信では，同時に不特定多数の視聴者に同一の映像を配信することには適しているが，片方向通信であるため視聴者毎に異なる映像を配信することには適していない．また，伝送帯域の制限から 4K などの高精細映像を配信するためには映像を圧縮する必要があるため品質が劣化する．(2) インターネットを介した配信では，Content Delivery Network (CDN) [14] による配信が一般的である．CDN では映像データをユーザの近傍に複製配置する必要がある．そのため，視聴者毎に異なる多様な種類の映像を配信するためには膨大なストレージコストを要する．

そこで，当研究室で開発された Content Espresso [13] と Catalogue を用いることにより，視聴者毎に最適化された映像をインターネットを介して配信することを実現する．Content Espresso ではファイルに FEC 符号を付与しチャンクに分割して，グローバルに分散保存し，UDP でクライアントに送信することで大容量で低コストな映像配信を可能にする．

また，Catalogue によって動画のフレーム順序を記述し，保存された映像ファイルを組み合わせるだけで視聴者に最適化された映像を生成することを可能にする．動画を構成する各フレームファイルは Global File ID でグローバルユニークに識別される．Catalogue を用いてこれらのフレームファイルの関係を記述することで，一つのフレームファイルを再利用して複数の動画を生成できる．また，動画を表す Catalogue は Global Catalogue ID でグローバルユニークに識別されるため，これ

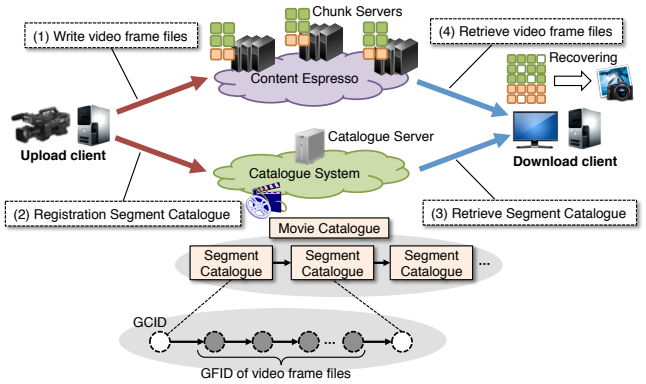

図 14　Live 映像配信システムの概要.

らを組み合わせて再利用することで新しい動画を生成できる．さらに，動画以外のオブジェクトとの関係をサービスをまたいで記述し整理することも可能である．

図 14 は Live 映像配信システムの概要を示している．Live 映像配信をするためには，まずアップロードクライアントが (1) Content Espresso にフレームファイルを保存し，(2) Catalogue システムに単位時間当たりの動画の順序を表す *Movie Catalogue* を保存する．Movie Catalogue は *Segment Catalogue* の順序関係から成る．Segment Catalogue は単位時間当たりに生成されるフレームの順序を記述した Catalogue であり，末尾に次の Segment Catalogue の GCID が配置される．末尾に次の Segment Catalogue を配置することで，視聴者は Movie Catalogue を毎回取得しその中身をパースする必要がなくる．視聴者は (3) Catalogue システムから Movie Catalogue を受信し有向グラフを辿ることで，Movie Catalogue 内の末尾の Segment Catalogue を検出し Catalogue システムから取得する．次に，(4) 取得した Segment Catalogue に記載されているフレームファイルを順に Content Espresso から取得する．以降は Segment Catalogue の末尾に記載された Segment Catalogue を取得し，Movie Catalogue を再度要求することを回避する．

5.　まとめ

テキストベースの検索では，着目ファイルが如何なる文脈に属しどのようなファイルと関連しているかを知るためには，ユーザが予め適切なキーワードを知っておく必要がある．また，情報を検索するためには，ウェブ空間を全探索してインデックス化する必要があり，これには膨大なストレージと演算コストを要する．本論文では，このような情報探索における課題を解決するために *Catalogue* と呼ばれるファイルの整理表現を提案した．

本研究ではまず，1) 自然言語のタグを用いずに数値識別子によって識別された *Catalogue* でファイルをグループ化し，また，2) 有向グラフを用いてグループ内のファイル同士の柔軟な関係を記述する．3) Catalogue はファイルの実体とは分離して作成・管理し，4) 着目ファイルからそのファイルを含む Catalogue を検索可能にする．さらに，5) プロパティを別ファイルとして保存し Catalogue を用いて対象ファイルと対応付ける．

以上をふまえて，Catalogue をグローバルに自律分散保存し共有するための Catalogue システムを設計・実装した．Catalogue の汎用的なグラフ表現を用いて，仮想ミュージアムや Live 映像配信といったアプリケーションが動作することを確認した．

文　献

[1] S. Brin and L. Page. The Anatomy of a Large-scale Hypertextual Web Search Engine. *Comput. Netw. ISDN Syst.*, 30:107–117, 1998.

[2] J. Cho and S. Roy. Impact of Search Engines on Page Popularity. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 20–29, 2004.

[3] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu. Exploring Folksonomy for Personalized Search. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 155–162, 2008.

[4] Resource Description Framework (RDF) Model and Syntax Specification. `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/`.

[5] G. Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: A System for Large-scale Graph Processing. In *Proceedings of the 2010 ACM SIGMOD*, pages 135–146, 2010.

[6] B. Shao, H. Wang, and Y. Li. Trinity: A Distributed Graph Engine on a Memory Cloud. In *Proceedings of the 2013 ACM SIGMOD*, pages 505–516, 2013.

[7] S. Salihoglu and J. Widom. GPS: A Graph Processing System. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, pages 22:1–22:12, 2013.

[8] A. Nanopoulos. Item Recommendation in Collaborative Tagging Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 41(4):760–771, 2011.

[9] Kilian Q. Weinberger, M. Slaney, and R. Van Zwol. Resolving Tag Ambiguity. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 111–120, 2008.

[10] E. Sagi, S. Kaufmann, and B. Clark. Semantic Density Analysis: Comparing Word Meaning Across Time and Phonetic Space. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, GEMS '09, pages 104–111, 2009.

[11] H. Halpin, V. Robu, and H. Shepherd. The Complex Dynamics of Collaborative Tagging. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 211–220, 2007.

[12] Oxford English Dictionary. Discover the story of English. More than 600,000 words, over a thousand years. `http://www.oed.com`.

[13] D. Ando, M. Kitamura, F. Teraoka, and K. Kaneko. Content Espresso: A System for Large File Sharing Using Globally Dispersed Storage. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, pages 337–340, 2013.

[14] A. Sherman, Philip A. Lisiecki, A. Berkheimer, and J. Wein. ACMS: The Akamai Configuration Management System. In *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, pages 245–258, 2005.