Vol.2012-DD-87 No.2
2012/12/22

EPUBCFIを用いた読書情報管理システムの提案

片岡 えり[1,a)]　天笠 俊之[2,3,b)]　北川 博之[2,c)]

概要：本論文では，EPUB に対する読書情報の管理を目的として，EPUBCFI を用いた読書情報管理システムを提案する．EPUBCFI とは，EPUB 出版物の任意のコンテンンツを参照するための標準であり，1) CFI によってコンテンツ内の任意の点が唯一に識別される，2) CFI 同士を比較することによって，コンテンツ内での順序を判別することができるといった特徴を持つ．本研究では，EPUBCFI を用いて，EPUB 出版物に関連するブックマークやアノテーションなどのメタデータを管理するとともに，それをシステムの利用者間で共有する機能を持った読書情報管理システムを構築する．これにより利用者は，出版物に付箋を貼ったり書き込みといった行為を電子書籍上でできるようになる．さらにそれらを多人数で共有したり、検索やマイニングを行うことにより，現在では不可能なリッチな読書体験の共有を実現することが期待される．このための EPUB での読書情報管理手法，さらにはソーシャルリーディングを行うための手法について議論する．

Eri Kataoka[1,a)]　Toshiyuki Amagasa[2,3,b)]　Hiroyuki Kitagawa[2,c)]

1. はじめに

ここ数年，ネットワーク上で様々な情報を公開，共有するソーシャルネットワーキングサービスが数多く登場し，広く利用ている．ソーシャルリーディングサービスは，新たなソーシャルサービスの一つであり，インターネット上で読書情報を多人数で共有する機能を提供する．ここでいう読書情報とは，読書に関連する様々な情報のことを指しており，例えば，いつどこで誰がどのようのな本を読んだのか，読んだ本に関する評価やレビュー，本の引用などが挙げられる．これらの情報を共有し，レビューや評価などにコメントをつけたり評価の集計などを行うことで，利用者は読書体験を複数人で共有したり，新たな知見を得たりすることができる．既存のソーシャルリーディングサービスとしてはブクログ[*1]，booklook[*2]などがある．

その一方，スマートフォンやタブレット端末，電子書籍リーダーの普及に伴って電子書籍が現在急速に浸透しつつある．電子書籍には様々なフォーマットが規定されており，例えば EPUB[1]，PDF[2]，KF8[3]，XMDF[4]，などがある，その中でも EPUB は International Digital Publishing Folum（IDPF）[5] によって策定され，電子書籍の標準フォーマットとなっている．EPUB は Web の標準規格を用いて構成されており仕様が公開されているため，誰でも自由に電子書籍をつくることが可能となっている．また，EPUB は画面サイズに応じて自動的に最適化されるリフロー型のフォーマットであるため，様々な端末で読まれる電子書籍において有用である．2011 年には EPUB3.0[6] がリリースされ，日本語の縦書やルビへの対応など多くの機能が追加された．

また，EPUB3.0 ではコンテンツの識別子として EPUBCFI[7] が策定されている．EPUBCFI を用いることで，電子書籍内の任意の一点を唯一に識別することが可能となる．また，CFI 同士を比較することで，コンテンツ内における順序関係を特定することができ，またテキストだけでなく画像，音声，動画に対しても有効であるといった特徴があり，さまざまな応用が考えられる．

今後，電子出版物の普及に伴い，電子書籍を対象にしたソーシャルリーディングの取り組みが高度化することが考えられる．そこで本研究では，電子書籍リーダを利用した読書情報の共有を目的としたシステムを提案する．具体的には，EPUB に対応した電子書籍リーダを対象として，読

1 筑波大学情報学群情報科学類
2 筑波大学システム情報系情報工学域
3 宇宙航空研究開発機構宇宙科学研究所所学際科学研究系
a) erikata@kde.cs.tsukuba.ac.jp
b) amagasa@cs.tsukuba.ac.jp
c) kitagawa@cs.tsukuba.ac.jp

*1 http://booklog.jp/
*2 http://booklook.jp/

書情報の取得と管理の手法を検討する．取得した読書情報はEPUBCFIを識別子としてデータベースで管理する．またソーシャルメディアと同様，ユーザ間で読書情報を共有する仕組みも提供する．これにより，EPUBでソーシャルリーディングを行うことが可能となる．さらには読書情報を管理・共有することで読書の高度化だけでなく，教育などへの応用も考えられる．

本稿の構成は以下のとおりである．まず2節では読書情報管理システムに関する基本事項として，電子書籍フォーマットのひとつであるEPUBと部分文書識別子であるEPUBCFIについて述べる．3節では，提案するシステムについての概要・設計について述べる．4節ではクライアントサイドの実装について述べる，最後に5節でまとめと今後の課題について述べる．

2. 基本事項

2節では，読書情報管理システムに関する基本事項について述べる．2.1節ではEPUBについて，2.2節ではEPUBCFIについて述べる．

2.1 EPUB

EPUB[1]はIDPF[5]が定めた電子出版物のフォーマットである．EPUBはWebで用いられている標準規格，例えばXML[8]，HTML5[9]，CSS[10]などを利用しているため，誰でも自由にEPUB形式の電子出版物を作ることが可能となっている．

EPUBの特徴の一つとしてリフロー機能が挙げられる．リフローとは文書が画面サイズに応じて自動的に最適化される方式のことである．ページが可変であるという想定のもとでコンテンツが作成されるため，例えば文字サイズの変更や端末の向きの変更などがあった場合や画面サイズの異なる端末で表示した場合に，最適なレイアウトに自動的に変更される．これにより，読み手の最適な環境で読むことが可能となる．

EPUBの現在のバージョンはEPUB3.0[6]であり，2011年10月にIDPFによって提唱された．EPUB3.0は複数のファイルをEPUB Containerとしてひとまとめにし，ZIP形式に圧縮することで作成できる．EPUB3.0の構成ファイルを**図 1**に示す．

mimetypeはEPUB出版物の必須要素で，EPUB Conatinerが ZIP形式に圧縮されたepubファイルであることを示しているファイルである．**META-INF**はEPUB Containerに関するメタデータが記された XMLファイルが格納されているディレクトリである．META-INFに含まれるファイルのうち，パッケージ文書のファイル名と所在を記述する`container.xml`という XMLファイルが必須要素となっている．**パッケージ文書**はEPUB出版物の必須構成要素であり，XMLで記述され，opfという拡張

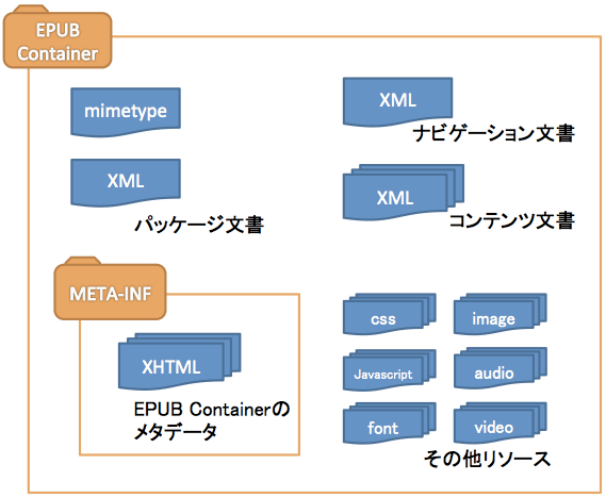

図 1 EPUB3.0の構成ファイル．

```
<?xml version="1.0" encoding="UTF-8"?>
<package xmlns="http://www.idpf.org/2007/opf" version="3.0" unique-identifier="uid">
  <metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
    <dc:identifier id="uid">samples.samplebook</dc:identifier>
    <dc:title>本のタイトル</dc:title>
    <dc:creator>筑波 太郎</dc:creator>
    <dc:language>ja</dc:language>
    <meta property="dcterms:modified">2012-11-22T16:38:35Z</meta>
  </metadata>
  <manifest>
    <item href="title.xhtml" id="ttl" media-type="application/xhtml+xml"/>
    <item href="nav.xhtml" id="nav" media-type="application/xhtml+xml" properties="nav"/>
    <item href="1.xhtml" id="term1" media-type="application/xhtml+xml"/>
    <item href="2.xhtml" id="term2" media-type="application/xhtml+xml"/>
    <item href="style.css" media-type="text/css" id="css"/>
    <item href="logo.jpg" media-type="image/jpeg" id="logo"/>
    <item href="cover.jpg" media-type="image/jpeg" id="cover" properties="cover-image"/>
  </manifest>
  <spine>
      <itemref idref="ttl"/>
      <itemref idref="nav" linear="no"/>
      <itemref idref="term1" />
      <itemref idref="term2" />
  </spine>
</package>
```

図 2 パッケージ文書の例．

子が付けられている．例を**図 2**示す．図2の3行目から9行目にあたる`<metadata>`という要素の中には書誌情報（出版物のメタデータ）が書かれている．10行目から18行目の`<manifest>`という要素の中には出版物に含まれる全ての文書の一覧，そしてそれらに必要なリソースが指定されている．また，19行目から24行目にあたる`<spine>`というタグの中にはEPUB出版物に含まれるコンテンツのデフォルトの読み順が定義されている．**ナビゲーション文書**はEPUB出版物の必須要素で，目次の役割を果たしているXMLファイルであり，`<ol>`タグや`<li>`タグなどを用いて書かれる．**コンテンツ文書**はEPUB出版物の中身にあたるファイルで，HTML5もしくはSVGで記述されている．例を**図 3**に示す．その他のリソースとしてCSSやJavaScript，画像，フォント，音声，動画などのファイルを含めることができる．

2.2 EPUBCFI

EPUBは2.1節で述べたように Webで用いられている標準規格を使用しており，リフロー機能が特徴として挙げ

```
<?xml version="1.0" encoding="utf-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8"/>
        <title>タイトル</title>
        <link rel="stylesheet" type="text/css" href="style.css"/>
    </head>
    <body>
      <h1 class="title">本のタイトル</h1>
      <p>この本はコンテンツ文書の例です。</p>
      <div class="legalnotice">
        <p id="text">あいう<span>えおかき</span>くけこさしすせそたちつてと</p>
        <p>なにぬねのはひふへほまみむめもやゆよわおん</p>
      </div>
    </body>
</html>
```

図 3 コンテンツ文書の例.

られる．しかしながらページレイアウトがユーザひとりひとりに最適化されることにより，紙の本における「ページ」という概念がなくなり，文書内の位置を指定することが困難となった．

EPUBCFI[7] は Canonical Fragment Identifier と呼ばれる部分文書識別子のことで，IDPF[5] によって 2011 年 10 月に規定された．これは EPUB 出版物のある部分を指し示す Fragment Identifier であり，仕様により，出版物の中のある一点に対する CFI は一つしかないことが保証されている．また，CFI のみを比較することで文書中における順序が判別できる．さらにテキストだけでなく画像や音声，動画についても詳細な場所，時間について参照することができる．

EPUBCFI は，XML や XHTML のノードをルート要素ノードをはじめとして親から子へたどることで表す．それぞれの要素については正の偶数のインデックスが割り当てられ，最初の要素の前や要素間，最後の要素の後ろに位置する非要素ノードには正の奇数のインデックスが割り当てられ，テキスト以外のノードは無視される．親から子へのステップ参照は “ / ” を用いて表す．要素に ID が含まれる場合は要素インデックスの後に ID を角括弧で囲んで含めなければならない．また EPUBCFI は，パッケージ文書からコンテンツ文書を参照することで文書の位置を特定している．パッケージ文書からコンテンツ文書への参照は読み順を示す`<spine>`というタグに書かれた参照から行われる．参照関係は “ ! ” を要素インデックスの後に置くことで表す．文字オフセットは “ : ”，時間オフセットは “ ~ ”，空間オフセットは “ @ ” を用いて表す．

例えば，図 2 で示したパッケージ文書と図 3 で示したコンテンツ文書の一ファイルを用いて作られた EPUB ファイルがあるとする．この EPUB ファイルの中の一点が

```
#epubcfi(/6/8!/4/6/2[text]/3:5)
```

のような EPUBCFI で示されている時，文書のどの一点を示しているか説明する．ここでは図 3 のコンテンツ文書が図 2 内の`<metadata>`内に示されファイル一覧のうち，14

```
<?xml version="1.0" encoding="UTF-8"?>
<package xmlns="http://www.idpf.org/2007/opf" version="3.0" unique-identifier="uid">
 2 <metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
     <dc:identifier id="uid">samples.samplebook</dc:identifier>
     <dc:title>本のタイトル</dc:title>
     <dc:creator>筑波 太郎</dc:creator>
     <dc:language>ja</dc:language>
     <meta property="dcterms:modified">2012-11-22T16:38:35Z</meta>
   </metadata>
 4 <manifest>
     <item href="title.xhtml" id="ttl" media-type="application/xhtml+xml"/>
     <item href="nav.xhtml" id="nav" media-type="application/xhtml+xml" properties="nav"/>
     <item href="1.xhtml" id="term1" media-type="application/xhtml+xml"/>
     <item href="2.xhtml" id="term2" media-type="application/xhtml+xml"/>
     <item href="style.css" media-type="text/css" id="css"/>
     <item href="logo.jpg" media-type="image/jpeg" id="logo"/>
     <item href="cover.jpg" media-type="image/jpeg" id="cover" properties="cover-image"/>
   </manifest>
 6 <spine>
     2 <itemref idref="ttl"/>
     4 <itemref idref="nav" linear="no"/>
     6 <itemref idref="term1" />
     8 <itemref idref="term2" />
   </spine>
</package>
```

図 4 パッケージ文書 CFI 参照例.

```
<?xml version="1.0" encoding="utf-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
 2 <head>
     <meta charset="utf-8"/>
     <title>タイトル</title>
     <link rel="stylesheet" type="text/css" href="style.css"/>
   </head>
 4 <body>
   2 <h1 class="title">本のタイトル</h1>
   4 <p>この本はコンテンツ文書の例です。</p>
   6 <div class="legalnotice">
     2 <p id="text"> 1 あいう 2 <span>えおかき</span> 3 くけこさしすせそたちつてと</p>
       <p>なにぬねのはひふへほまみむめもやゆよわおん</p>
     </div>
   </body>
</html>
```

図 5 コンテンツ文書 CFI 参照例.

行目の `2.xhml` というファイルの内容を示していたとする．

まず，パッケージ文書内を参照する．**図 4** にパッケージ文書の参照例を示した．ルートから`<spine>`（図 4 内外枠），`<itemref idref="term2" />`（図 4 内中枠）の順に参照する．次に，パッケージ文書から参照されたコンテンツ文書内を参照する．**図 5** にコンテンツ文書の参照例を示した．ルートから`<body>`（図 5 内外枠），`<div class="legalnotice">`（図 5 内赤枠），`<p id="text">`（図 5 内黄緑枠）の順に参照する．`<p id="text">`内では非要素である「くけこさしすせそたちつてと」（図 5 内中枠）が参照され，その要素内の 5 番目の箇所が EPUBCFI で示された一点である．これにより，`#epubcfi(/6/8!/4/6/2[text]/3:5)` は図 3 で示したコンテンツ文書の XML ファイルの 12 行目「し」と「す」の間一点を指し示していることがわかる．

3. EPUBCFI を用いた読書情報管理システム

3 節では，EPUBCFI を用いた読書情報管理システムの概要と設計について述べる．3.1 節では提案する読書情報管理システムの概要，3.2 節では EPUB クライアントから取得するデータ，3.3 節ではデータベースの設計，3.4 節で

は想定する Web API について述べる．

3.1 読書情報管理システムの概要

本研究では，EPUBCFI を利用した電子書籍向けの新たなソーシャルリーディングシステムを提案する．EPUBCFI を用いることで EPUB コンテンツ内の任意の場所を一意に識別することができるため，例えば現実世界における付箋を貼ったり書き込みをするような行為を，電子書籍上で実現するのに利用できる．さらにそれらを多人数で共有したり，検索・マイニングを行うことにより，現在では不可能なリッチな読書体験の共有を実現することが期待される．このための読書情報管理システムの新たな手法の提案，さらには新しいソーシャルリーディングの手法の提案を行う．

提案する読書情報管理システムの前提として，EPUB 出版物はユーザによって購読され，EPUB クライアントを利用して読まれることを仮定する．また，EPUB クライアントを使ってアノテーションの作成や検索を行うだけでなく，Web ブラウザを用い Web インタフェースを通じて各種アノテーション検索や他ユーザのフォローといったソーシャルなアクティビティを行うこともできる．システムの概要を**図 6** に示す．

システムでは，EPUB3.0 出版物のファイルが利用できる場合は，それをデータベースに登録するとともに，内容に関する索引を生成する．これはコンテンツに対する検索を行なうために利用されるが，ここでは詳しく述べない．EPUB クライアントからは，EPUBCFI によって識別される読書関連情報，すなわちアノテーションやブックマークが送信される．さらに，EPUB クライアントの操作状況をログデータとして取得する．取得したアノテーションとログは，それぞれ Web API を通じてアノテーションデータベース，ログデータベースに格納される．データベースに格納されたデータは，必要に応じて Web API から利用される．これにより EPUB クライアントからの読書情報の共有が可能となる．収集した各種データについては，各種マイニングを行なうことによって，有益な情報を発見することも考えられる．

本研究では図 6 で示した読書情報管理システムのうち以下の項目について扱う．

- EPUB クライアントからのデータ取得
- Web API 設計
- 取得したデータのデータベースへの格納
- 取得したデータの共有

3.2 EPUB クライアントからのデータ取得

本研究では，EPUB クライアントから以下のデータを取得する．

- ブックマーク情報
- アノテーション情報

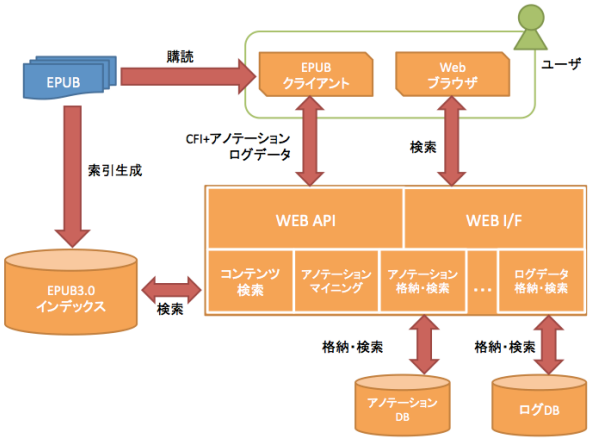

図 6 読書情報管理システムの概要．

- EPUB クライアントの操作ログ

ブックマーク情報には，ユーザが明示的に作成するしおりに加えて，どこまで読み進んだかといった読書の進捗情報も含む．さらに，利用者が EPUB クライアントを通じて作成するアノテーションを取得する．ブックマークとアノテーションは，それぞれ EPUBCFI によって識別され，アノテーションデータベースに保存される．操作ログとは EPUB クライアントを操作する際に記録されるデータのことで，例えばページをめくったタイミングやタイムスタンプなどの情報のことである．取得したデータはログデータベースに保存する．これらから，ユーザの読書行為やコンテキストの分析が可能になると考えられるが，本稿ではログ情報の詳細については述べない．

3.3 データベース設計

取得されたデータはデータベースに格納する．ここではデータベースの設計について述べる．データベーススキーマを**図 7** に示す．また具体的な定義は以下の通りである．

- User (user_id, user_name, ...)
- Group (group_id, group_name, ...)
- Belong (user_id , group_id)
- FollowUser (user_id, user_id)
- FollowGroup (user_id, group_id)
- Document (doc_id, isbn, title, author, ...)
- Has (user_id , doc_id)
- Bookmark (user_id , doc_id, bookmark_cfi)
- Annotation (user_id , doc_id, annotation_cfi, annotation)

User はユーザの情報に関するデータベースであり，ID とユーザの情報を持つ．またユーザはフォローを行うことができる．これはフォローしたユーザやグループのの情報を共有することができるものとする．例えば，あるユーザ A がユーザ B（グループ B）をフォローすると，A だけではなく，ユーザ B（グループ B に所属するメンバー）が作成したアノテーションやブックマークも検索や表示をする

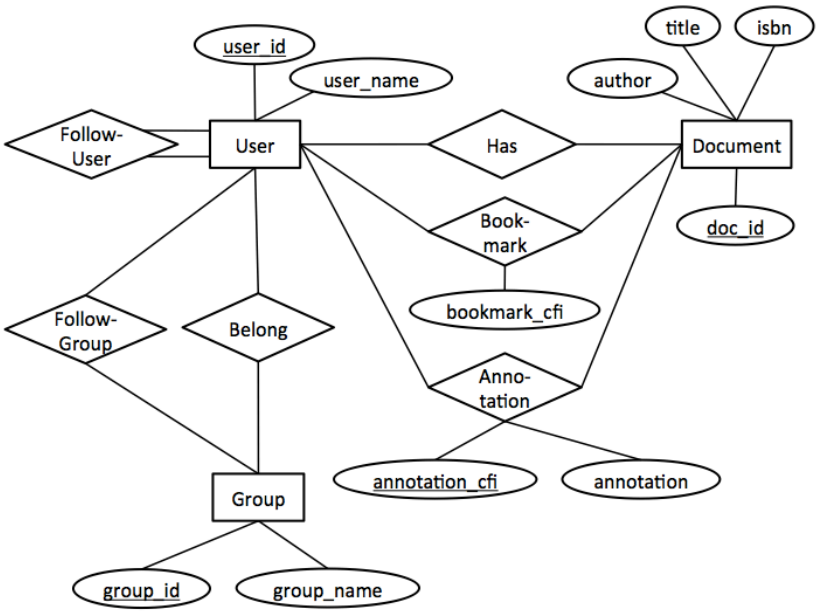

図 7 データベーススキーマ.

表 1 User テーブルの例.

user_id	user_name	···
taro	筑波 太郎	...
hanako	鈴木 花子	...
ichiro	加藤 一郎	...
yamada	山田 正子	...

表 2 Document テーブルの例.

doc_id	isbn	title	author	...
1	4101010137	こころ	夏目漱石	...
2	4101025037	蜘蛛の糸	芥川龍之介	...
3	4101006067	走れメロス	太宰治	...
4	4003101049	草枕	夏目漱石	...

表 3 Group テーブルの例.

group_id	group_name	···
natsume	夏目漱石を読む	···
rinko	輪講	···

表 4 Belong テーブルの例.

user_id	group_id
taro	natsume
taro	rinko
hanako	rinko
ichiro	natsume
ichiro	rinko

表 5 Has テーブルの例.

user_id	doc_id
taro	1
taro	2
taro	4
hanako	2
hanako	3
ichiro	1
ichiro	2
ichiro	4
yamada	3

表 6 FollowUser テーブルの例.

user_id	user_id
taro	hanako
taro	ichiro
hanako	taro
ichiro	hanako
ichiro	yamada

表 7 FollowGroup テーブルの例.

user_id	group_id
taro	natsume
taro	rinko
hanako	rinko
ichiro	natsume
ichiro	rinko
yamada	rinko

ことができる．**Document** は登録された EPUB に関するデータベースであり，ID と書誌情報を持つ．**Group** はグループの情報に関するデータベースであり，ID とグループ情報を持つ．**Belong** は本とグループの所属関係を示す．**Has** はユーザと EPUB の所有関係を示す．**Bookmark** はユーザと EPUB のブックマークにおける関連を示す．固有の属性として場所を示す EPUBCFI を持つ．**Annotation** はユーザと EPUB のアノテーションにおける関係を示す．固有の属性として場所を示す EPUBCFI とアノテーションの内容を持つ．それぞれの格納例は，**表 1** から**表 9** に示した通りである．

表 8 Bookmark テーブルの例.

user_id	doc_id	bookmark_cfi
taro	1	#epubcfi(/6/8!/4/2/2[chapter1]/3:5)
hanako	3	#epubcfi(/6/8!/4/10/8[text]/10:25)

表 9 Annotation テーブルの例.

user_id	doc_id	annotation_cfi	annotation
taro	1	#epubcfi(/6/10!/2[chap2]/3:5)	わからない
hanako	2	#epubcfi(/6/4!/4/8/20:2)	質問する
hanako	3	#epubcfi(/6/8!/4/10/52/1:1)	お気に入り

3.4 Web API の設計

サーバには，EPUB クライアントや Web クライアントからアクセスするための Web API を用意する．API は主に，1) ユーザに関する API，2) グループに関する API，3) EPUB 出版物に関する API，4) ユーザ（グループ）のフォローに関する API，5) ブックマークに関する API，6) アノテーションに関する API から構成される．一覧は**表 10** に示した通りである．

表 1，**表 2**，**表 3** で示されたデータベースにいくつかの問合せを想定した例を示す．データの追加は以下の Web API で行なわれる．

```
createUser("erikata", "片岡 えり", ...)
```

という API を用いると，表 1 に user_id=erikata, user_name="片岡 えり"といった情報が追加される．また，「片岡えり」があるドキュメントを登録し，そのドキュメントにブックマークを作成すると，以下の API が呼び出される．

```
addDocumentToUser("erikata", 1)
saveDocumentBookmark("erikata", 1, #epubcfi(/6/8
!/4/2/2[chapter1]/3:5)
```

これにより，**表 5** に user_id=erikata, doc_id=1 が保存され，**表 8** に user_id=erikata, doc_id=1, bookmark_cfi=#epubcfi(/6/8!/4/2/2[chapter1]/3:5) が保存される．

ブックマークの取得は，データベースへの問合せも含め以下のように実現される．

```
getDocumentBookmark("hanako", 1)
```

という API を用いたとする．これは user_id=hanako のユーザがフォローしているユーザもしくはグループに所属するユーザが持っているブックマークで，doc_id=1 のドキュメントに対して問合せを行っている．この問合せは以下のような SQL 問合せで実現可能である．

```
SELECT bm.bookmark_cfi
FROM Bookmark as bm, Belong as blg,
     FollowUser as fu, FollowGroup as fg
WHERE bm.doc_id = 1
      AND fu.user_id = 'hanako'
      AND fg.user_id = 'hanako'
      AND blg.group_id=fg.group_id
      AND (bm.user_id = fu.user_id
               OR bm.user_id = blg.user_id)
```

また，

```
getDocumentBookmark("hanako", 1, #epubcfi(/6/
8!/2[chapter1]/3:5)
```

という API を用いると，あるブックマーク周辺のブックマークが取得可能である．これは以下のような SQL 問合せで実現できる．

```
SELECT bm.bookmark_cfi
FROM Bookmark as bm, Belong as blg,
     FollowUser as fu, FollowGroup as fg
WHERE bm.doc_id = 1
      AND fu.user_id = 'hanako'
      AND fg.user_id = 'hanako'
      AND blg.group_id = fg.group_id
      AND (bm.user_id = fu.user_id
               OR bm.user_id = blg.user_id)
      AND bm.bookmark_cfi =
         NEAR #epubcfi(/6/8!/2[chapter1]/3:5)
```

この SQL では user_id=hanako のユーザがフォローしているユーザもしくはグループに所属するユーザが持っているブックマークで，doc_id=1 のドキュメントに対して問合せを行っている．bookmark_cfi に #epubcfi(/6/8!/4/2/2[chapter1]/3:5) が指定されているが，これは指定された EPUBCFI に近いブックマークのうち特定の件数を取得すると定義し，SQL では

```
NEAR epubcfi 文字列
```

という形で表している．このような EPUBCFI を指定した問合せ手法についても検討が必要であると考えられる．

4. 実装

4 節では，EPUB クライアントにおける実装について述べる．4.1 節では EPUB クライアント，4.2 節では EPUBCFI のライブラリ，4.3 節では EPUBCFI の取得と保存の方針について述べる．

4.1 EPUB クライアント

本研究におけるソーシャルサービスの実装には Readium[11] を用いる．Readium は IDPF[5] と支援企業によるプロジェクトで，EPUB 出版のためのシステムとレンダリングエンジンのオープンソースリファレンスである．ソースコードは github 上で公開されており，最新のリリースを Google Chrome の拡張機能として使うことができる．Readium は jQuery[12], Backbone.js[13] などの様々なオープンソースリファレンスを用いて開発が行われている．本研究では，Readium を拡張し機能を追加する形でソーシャルサービスを開発する．

4.2 EPUBCFI ライブラリ

Readium では，EPUB3.0 の CFI ライブラリ[*1]が組み込まれている．これは EPUBCFI の解釈と解析に対応しており，JavaScript で書かれている．EPUBCFI の解釈は，EPUBCFI の文字列と以下の項目を用いることで解釈を行い，参照された電子書籍内の一点を特定する．

[*1] https://github.com/readium/EPUBCFI

ユーザに関する処理	Web API
ユーザ登録	createUser(user_id, user_name, ...)
ユーザ削除	deleteUser(user_id)
グループに関する処理	Web API
グループを作成	createGroup(group_id, group_name, ...)
グループを削除	deleteGroup(group_id)
ユーザをグループに追加	addGroupMember(user_id, group_id)
ユーザをグループから削除	removeGroupMember(user_id, group_id)
ドキュメントに関する処理	Web API
ドキュメントを登録	addDocument(doc_id, isbn, title, author, ...)
ドキュメントを削除	removeDocument(doc_id)
ユーザにドキュメントを登録	addDocumentToUser(user_id, doc_id)
ユーザからドキュメントを削除	removeDocumentFromUser(user_id, doc_id)
フォローに関する処理	Web API
ユーザをフォロー	followUser(user_id, user_id)
ユーザのフォロー解除	unfollowUser(user_id, user_id)
グループをフォロー	followGroup(user_id, group_id)
グループのフォロー解除	unfollowGroup(user_id, group_id)
ブックマークに関する処理	Web API
ブックマークを保存	saveDocumentBookmark(user_id, doc_id, bookmark_cfi)
ブックマークを削除	removeDocumentBookmark(user_id, doc_id, bookmark_cfi)
ブックマークを取得	getDocumentBookmark(user_id, doc_id, [bookmark_cfi])
アノテーションに関する処理	Web API
アノテーションを保存	saveDocumentAnnotation(user_id, doc_id, annotation_cfi, annotation)
アノテーションを削除	removeDocumentAnnotation(user_id, doc_id, annotation_cfi)
アノテーションを取得	getDocumentAnnotation(user_id, doc_id, [annotation_cfi])

表 10 Web API 一覧.

- EPUB のパッケージドキュメント
- コンテンツ文書への参照
- コンテンツ文書

EPUBCFI の解析は以下の項目を用いることで EPUB 内の場所を特定する.

- EPUB のパッケージドキュメント
- 生成したい EPUBCFI のあるテキストノード
- テキストノードのオフセット
- 現在のコンテンツ文書の IDREF

4.3 EPUBCFI の取得・保存

本研究では 4.2 節で述べた EPUBCFI ライブラリを用いて EPUBCFI の取得を行う．取得位置は Readium 上の文書をクリックした時の一点とする．ライブラリを用いて取得した EPUBCFI はアノテーションと共にローカルストレージに保存する．ローカルストレージに蓄積されたアノテーションなどの情報は，上記 API を用いて適切なタイミングでサーバに格納される．

表示の際には，レンダリングされる対象コンテンツおよび周辺に含まれるアノテーションを，EPUBCFI をキーとして検索し，コンテンツとともに重畳表示する．

5. まとめ

本稿では，EPUB に対する読書情報の管理・共有を目的とした読書情報管理システムについて述べた．アノテーションやブックマークの識別子として EPUBCFI を用い，これらを管理するためのデータベースの構成を述べる，また Web API の設計についても議論した．読書情報管理システムをによって，電子書籍上での新しい読書情報の共有が可能となることが期待される．今後の課題としては，図 6 で示したシステムの実装を進めるとともに，EPUBCFI をキーとした効率的な検索を可能にする索引法の検討，EPUB クライアントから取得する情報の検討などが挙げられる．

謝辞

本研究を進めるにあたり，村田真博士（国際大学 GLOCOM フェロー）に貴重な情報を頂いた．ここに記して謝意を表す.

参考文献

[1] IDPF：EPUB, http://idpf.org/epub/.
[2] Adobe：Portable Document Format, http://www.adobe.com/jp/products/acrobat/adobepdf.html.
[3] Amazon：Kindle Format 8 Overview, http://www.

amazon.com/gp/feature.html?ie=UTF8&docId=1000729511.
[4] SHARP：XMDF, http://www.xmdf.jp/.
[5] IDPF：IDPF, http://idpf.org/.
[6] IDPF：EPUB3, http://idpf.org/epub30/ (2011.10.11).
[7] IDPF：EPUB Canonical Fragment Identifier, http://idpf.org/epub/linking/cfi/epub-cfi.html (2011.10.11).
[8] W3C：XML, http://www.w3.org/TR/xml/(2008.11.26).
[9] W3C：HTML5, http://www.w3.org/TR/html5/(2012.5.29).
[10] W3C：Cascading Style Sheets, http://www.w3.org/Style/CSS/.
[11] International Digital Publishing Forum：Readium, http://readium.org/.
[12] The jQuery Foundation：jQuery, http://jquery.com/.
[13] DocumentCloud：Backbone.js, http://backbonejs.org/.