情報処理学会論文誌　コンシューマ・デバイス & システム　Vol.9　No.2　1–10 (May 2019)

コンシューマ・システム論文

データフィードバックモデルを導入したユーザ参加型LODプラットフォームの構築

浦川 真[1,a)]　荒井 研一[1]　小林 透[1]

受付日 2018年10月9日, 採録日 2019年2月20日

概要：Linked Open Data（LOD）は，Web 上でデータの公開および共有を可能とする手法の 1 つであり，ソフトウェアが処理できる形式でデータを記述できる．LOD で情報を記述することで，多くのアプリケーションが利用できるデータとなるため，データの利活用が進むことが期待されている．そのため，行政が保有する情報のオープンデータ化をはじめ，市民が地域情報を作成し公開するケースも多くなっている．しかし，LOD では，生成したデータがどのように利用されたのか分からない．そこで，LOD におけるデータ利用実績を，データ制作者側が把握できるプラットフォームを構築することを目的として，LOD を構成するデータの組合せをハッシュ値によりインスタンス化することで効率的かつ確実にデータ生成者などの情報を記述できるデータ構造や，外部のアプリケーションから利用実績などをフィードバックするためのデータ処理手法を提案する．さらに，提案するデータ構造やデータ処理の有用性を検証するため，LOD プラットフォームの構築から，データプラットフォーム上でデータ入力できるアプリケーションの開発，市民とのデータ入力ワークショップまで行ったので，本論文にて報告する．

キーワード：LOD，セマンティック Web，構造化データ，フィードバック，市民，地図アプリケーション

Development of a Consumer Generated LOD Platform based on Integrating all the Data Usage

Makoto Urakawa[1,a)]　Kenichi Arai[1]　Toru Kobayashi[1]

Received: October 9, 2018, Accepted: February 20, 2019

Abstract: The Linked Open Data (LOD) is one of the ways to enable data to be published and shared on the web. Since data described in LOD can also be processed by a software, data can be used by various software applications such as sightseeing applications. In addition to converting governments' data to LOD, citizens try to generate open data about areas where they live in. However, it is difficult to track how many times each data has been used and which applications have used it. That is why one of challenges is to keep on motivating them to input data. The authors propose the data structure which can express information of a person who owns or generates each data in LOD. All the data owners can directly know all the usage of data they produce. The authors developed an application where citizens can input any information about the cities where they live. This paper shows results about effectiveness they can experimented on the application.

Keywords: LOD, semantic Web, structured data, feedback, citizen, map application

1. はじめに

2017 年 5 月に，内閣官房情報通信技術（IT）総合戦略室から，官民データ活用推進基本法 [1] に基づいて「世界最先端 IT 国家創造宣言・官民データ活用推進基本計画」[2] が策定された．本計画において，地方公共団体が保有するデータのオープンデータ化の推進に取り組むことが掲げられている．このオープンデータとは，ティム・バーナーズリーが提唱した LOD（Linked Open Data）[3] を指し，Web 上でデータの公開および共有を可能とする手法の 1 つ

1　長崎大学
Nagasaki University, Nagasaki 852–8521, Japan
a)　bb52218201@ms.nagasaki-u.ac.jp

© 2019 Information Processing Society of Japan

である．LOD では，データは構造化され，ソフトウェアが処理できる形式で記述できる．そのため，Web 上のリソースのメタデータを記述する枠組みである RDF（Resource Description Framework）[4] によりデータ表現を行うことになる．RDF で Web 上のデータの構造や関係を記述することで，多くのアプリケーションが利用できるデータとなるため，データの利活用が進むことにつながる．

オープンデータ化に関して，行政のデータだけでなく，民間企業でも取り組みが進んでいる [5]．そのほか，市民と行政が協業し，地域情報を作成し公開するケースも多くなっている [6]．しかし，LOD では，生成したデータがどのように利用されたのか分からない．政府 CIO ポータル [7] で公開されている，平成 28 年 12 月に実施された「オープンデータの取組に関する自治体アンケート結果」では，「オープンデータの効果・メリット・ニーズが不明確」「オープンデータを担当する人的リソースがない」といった課題が高く認識されている．

そこで，本研究では，データ入力者がオープンデータの利用実績を把握できるようにするため，「主語–述語–目的語」の 3 つ組で表現する RDF 形式のデータ（以下，トリプルデータ）をハッシュ値によりインスタンス化することで，作成者や作成日時などの情報を効率的に保持できるデータ構造を提案する．本データ構造を用いることにより，たとえば，入力者 A が「コインロッカー（主語）–緯度（述語）–32.74472（目的語）」というトリプルデータを登録した場合に，入力者 A がこのトリプルデータを登録したこと自体も RDF 形式で表現できる．結果として，作成したトリプルデータが，外部アプリケーションから利用された際に，入力者 A にその利用実績などをフィードバックすることが可能になる．本論文では，データ構造の提案だけでなく，データを生成するため，および利用実績をフィードバックするためのデータ処理手法をモデル化したうえで具体的に示す．

著者らは，本研究が提案するデータモデルの有用性を示すため，本データ構造を API 経由で登録および参照できる LOD プラットフォームを構築した．また，Web アプリケーションを開発し，地域情報を入力するワークショップを行った．

本論文では，2 章で関連研究について，3 章および 4 章で提案するデータ構造やデータ処理について，5 章で構築したプラットフォームおよび Web アプリケーションについて示す．その上で，6 章で考察し，7 章でまとめる．

2.　関連研究

オープンデータが地域活性を担う市民を顕在化させる可能性を持っていることを示唆する研究 [8] や，入力者をいかに巻き込んでいくかが重要であることを示唆する研究 [9], [10] がされている．一方，データモデルに関する技術的な研究として，トリプルデータをインスタンス化する手法はいくつか研究されている．その手法には，標準化された RDF Statement [11] や，述語を ID 化する Singleton Properties [12]，トリプルを複数まとめて定義する Named Graphs [13] などがある．RDF Statement は，新しいインスタンスを生成し，インスタンス表現したいトリプルデータを構成する，主語，述語，目的語に対して，特定の語彙を用いて関係を生成する．そのため，トリプル数が 3 倍になる．Singleton Properties は，インスタンス化したいトリプルデータの述語を別の ID で表現し，その ID をインスタンスとして利用する方法である．この場合，必要なトリプル数は 2 倍におさえられるが，元のトリプルデータの主語と目的語の関係であり重要な役割である述語を変更する必要がある．Named Graphs は，複数のトリプルをまとめるために考案されているため，グラフを特定する処理が必要になる．Singleton Properties が最も表現力が高い [14] が，元々のデータに変更を加える必要があるため，また，Named Graphs は複数のトリプルをまとめることを目的としているため，本論文が提案するデータモデルは，その目的や方法が類似する RDF Statement と比較することで評価する．

3.　データモデルと生成処理

LOD におけるトリプルデータに対して，作成者情報などを付与するためには，トリプルデータ自体をインスタンス化する必要がある（**図 1**）．図 1 では，「コインロッカー（主語）」の「緯度（述語）」が「32.74472（目的語）」であるというトリプルデータをインスタンスとし，主語もしくは目的語として利用できる例を示している．この例では，「入力者 A（主語）」が「インスタンス X（目的語）」を「登録（述語）」し，「インスタンス X（主語）」の「登録日時（述語）」は「2018 年 8 月 1 日」である例を示す．インスタンス X は，「コインロッカー–緯度–32.74472」というトリプルデータを表す．

データをインスタンス化する方法として，上述したとおり rdf:Statement が W3C にて勧告されていてる．この rdf:Statement クラスのインスタンスと他のインスタンスとを，主語，述語，目的語という関係性で関連付けするためのプロパティ語彙も定義されている．インス

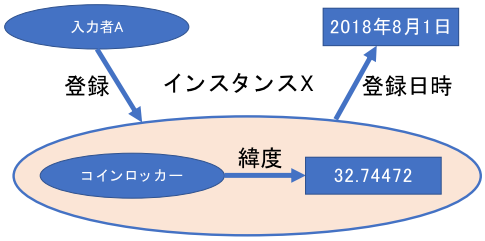


図 1　トリプルデータのインスタンス化イメージ
Fig. 1　Instantiation image of triple data.

© 2019 Information Processing Society of Japan

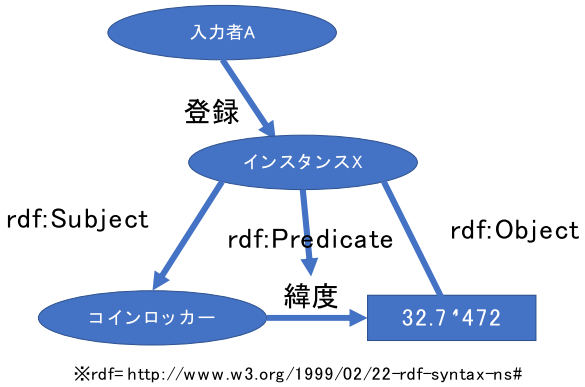


図 2　RDF Statement によるインスタンス化
Fig. 2　Instantiated data by RDF Statement.

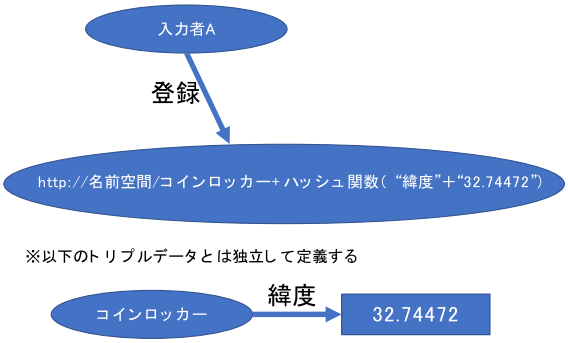


図 3　提案するインスタンス化モデル
Fig. 3　Proposed instantiation model.

タンス化したいトリプルデータの主語は rdf:Subject，述語は rdf:Predicate，目的語は rdf:Object で表現可能である．rdf:Statement を適用した例を**図 2** に示す．図 2 にあるように，「インスタンス X（主語）」–「rdf:Subject（述語）」–「コインロッカー（目的語）」，「インスタンス X（主語）」–「rdf:Predicate（述語）」–「緯度（目的語）」，「インスタンス X（主語）」–「rdf:Object（述語）」–「32.74472（目的語）」の 3 つのデータを追加して表現していることになる．なお，rdf:は，http://www.w3.org/1999/02/22-rdf-syntax-ns#の接頭辞を表す．

本論文では，インスタンスを URI で一意に表現する RDF の考え方にのっとり，かつ効率的にトリプルデータのインスタンス化を実現するため，ハッシュ値によるリソース表現を提案する．主語に，述語と目的語のハッシュ値を連結することで，もともと表現されていた主語インスタンスの URI と一意性を維持したまま，述語と目的語を文字列として連結することで効率的なインスタンス表現が可能となる．図 1 を本論文が提案するデータ構造で表現した例を**図 3** に示す．

図 3 に示すとおり，もともと生成したトリプルデータには直接影響を与えることなく，インスタンス化が可能である．そのため，外部アプリケーションから受け取るフィードバックデータを，元のトリプルデータに直接付与する必要がない．

3.1　インスタンス化処理

ハッシュ関数を用いたインスタンス化を具体的な例を用いて示す．トリプルデータのインスタンス化において，インスタンス化したデータ自体を LOD として公開するためには，LOD の原則である URI により一意に特定でき，http でのアクセスを可能とする必要がある．そこで，本論文では，インスタンス化したいトリプルデータの主語の URI を拡張することで，URI 記述と http アクセスを実現する．これにより，トリプルデータ用のインスタンスのための URI 発行を，トリプルデータの登録処理と付随させて実行することができる．また，SPARQL クエリ言語 [15] には，SHA1 をはじめとする標準的はハッシュ関数が用意されているため，既存のクエリ言語によりインスタンス化処理が可能である．

以下に，あるデータ入力ユーザが，街のスポット（観光地や公園など）情報を，トリプルデータとして登録する際の処理の流れを示す．なお，例では，長崎市にある丸山公園をスポットとして登録する流れである．

(1)　スポットの ID と関連情報を表すトリプルの生成

各スポットを一意に特定できる ID を生成し URI で記述する．たとえば，「http://nagasakipj.or.jp#SPOT_001」と表現する．なお，http://nagasakipj.or.jp（以下，接頭辞として ngsk:で表現）は本研究用に一時的に設定した名前空間である．このスポットに対して，たとえば，「名前」として「丸山公園」を登録することを想定し，述語として「ngsk:name」，目的語として「丸山公園」を登録する．この場合，トリプルデータは，「ngsk:SPOT_001–ngsk:name–丸山公園」となる．ここまでは，LOD アプリケーションで必要な一般的なデータ生成処理となる．

(2)　トリプルをインスタンス化する

図 3 の考え方に基づき，上記トリプルデータをインスタンス化する．そのため，まず，述語と目的語を連携した文字列をハッシュ値にする．具体的には，「http://nagasakipj.or.jp#name 丸山公園」と連結した後に，SHA1 でハッシュ化すると，「6b1d32a3d4c4694da845940e503c3f28e14b75e7」という文字列になる．そこで，スポットを示すインスタンス URI（ngsk:SPOT_001）に，ハッシュ値を連結させインスタンス化すると，「ngsk:SPOT_0016b1d32a3d4c4694da845940e503c3f28e14b75e7」と記述できる．なお，これ以降，インスタンス化されたトリプルを，トリプルインスタンスと呼ぶ．この例では SHA1 でハッシュ値を計算したが，MD5 といった他のハッシュ関数でも処理可能である．

(3)　ユーザが登録したことを示すトリプルを生成する

「ngsk:SPOT_001–ngsk:name–丸山公園」というトリプルデータは，上述のとおりインスタンス化されたため，登録者情報や登録日時などを付与することができる．ここでは，あるデータ入力ユーザが登録したことを示すトリプルデータを生成する．具体的には，「ngsk:USER_A–ngsk:produce–

© 2019 Information Processing Society of Japan

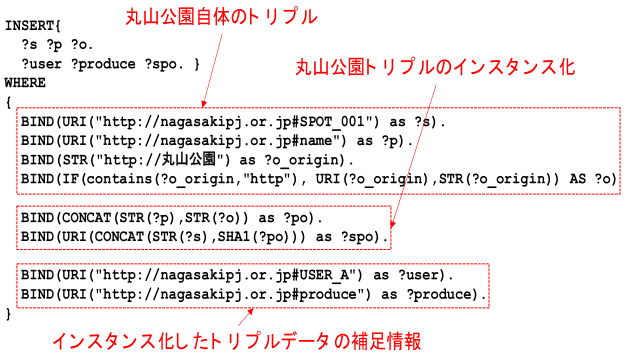


図 4　SPARQL クエリ
Fig. 4　SPARQL query.

```
INSERT{
  ?s ?p ?o.
  ?x ?y ?spo. }
WHERE
{
  BIND(URI( S ) as ?s).
  BIND(URI( P ) as ?p).
  BIND(STR( O ) as ?o_origin).
  BIND(IF(contains(?o_origin,"http"),
URI(?o_origin),STR(?o_origin)) AS ?o)

  BIND(CONCAT(STR(?p),STR(?o)) as ?po).
  BIND(URI(CONCAT(STR(?s),SHA1(?po))) as ?spo).

  BIND(URI( X ) as ?x).
  BIND(URI( Y  ) as ?y).
}
```

図 5　トリプルデータのインスタンス化クエリのモデル
Fig. 5　Data model for instantiation.

ngsk:SPOT_0016b1d32a3d4c4694da845940e503c3f28e14b75e7」と表現される．ngsk:USER_A は，登録したユーザを一意に指すインスタンスですでに存在するものとし，http://nagasakipj.or.jp#produce は，登録したことを意味する述語とする．なお，これらのデータ構造および処理は，汎用化して SPARQL クエリにて表現できることは後述する．

(4)　**RDF ストアに登録するクエリを生成する**

丸山公園自体のトリプルデータと，このトリプルデータをデータ入力ユーザ A が作成したことを示すトリプルデータを RDF ストアに登録するための SPARQL クエリを**図 4** に示す．図 4 中の，「丸山公園自体のトリプル」が上記手順の (1) を，「丸山公園トリプルのインスタンス化」が上記手順の (2) を，「トリプルデータの補足情報」が上記手順の (3) を指す．上述したデータ登録の流れにおいて，丸山公園を登録したことを示すトリプルデータは，丸山公園自体のトリプルデータから，標準的なハッシュ関数により機械的に生成することが可能である．つまり，一般化した SPARQL 表現により，トリプルインスタンスも登録可能である．図 4 の SPARQL クエリを一般化したクエリを**図 5** に，生成するトリプルデータのイメージを**図 6** に示す．

図 5 中の「S」「P」「O」は，生成するトリプルデータの

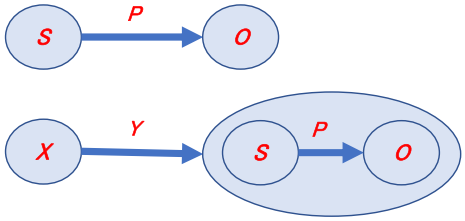


図 6　生成されるトリプルデータモデル
Fig. 6　Image of generated triple data.

主語，述語，目的語をそれぞれパラメータとして表現した値を意味する．また，「X」「Y」は，トリプルインスタンスに対して，付加したい主語，述語をパラメータ化した値である．たとえば，上述の例では，ngsk:USER_A が X に，ngsk:produce が Y にあたる．図 5 に示す，パラメータである，S，P，O，X，Y を，API 変数にして受け取れるようにすることで，データ登録アプリケーションは，SPARQL 文やハッシュ化などの処理を考えることなく，データ登録アプリケーションが開発可能にある．

4.　トリプルデータへのフィードバック処理

3 章で示したとおり，インスタンス化されたトリプルデータに対して，登録者情報や登録日時などの情報を付加できるようになる．本章では，LOD 形式で表現されたトリプルインスタンスを利用する外部アプリケーションから，トリプルデータの表示回数やコメントなどを，データ入力ユーザにフィードバックするデータ処理について詳述する．

4.1　データモデル

フィードバックにあたり，どのアプリケーションからのフィードバックなのかを指定したうえで，表示回数などの利用実績を利用者が参照できる必要がある．また，フィードバックする利用実績は，データの表示回数やデータクリック数など，アプリケーションごとに柔軟に設定できる必要がある．そのため，トリプルインスタンスとそのデータを利用するアプリケーションの関係を一意に特定できるデータ構造を根幹にして，その構造に，利用回数といった実績を柔軟に設定できるデータ構造とする（**図 7**）．

図中の「関係インスタンス」が，「ngsk:SPOT_001–ngsk:name–丸山公園」というトリプルデータや，そのトリプルインスタンスが LOD として公開される場合に，参照元である外部アプリケーション単位で，表示回数などを区別するために生成される関係インスタンスである．図 7 の，ngsk:feedback および ngsk:app という述語で構成された関係インスタンスを根幹に，外部アプリケーションごとに利用実績やコメントなどを柔軟に設定できる構造となっている．図 7 では，「ngsk:関係インスタンス 1–viewcount–13」がフィードバックデータにあたる．APP01 での表示回数（viewcount）は 13，APP02 でのクリック回数（clickcount）

© 2019 Information Processing Society of Japan

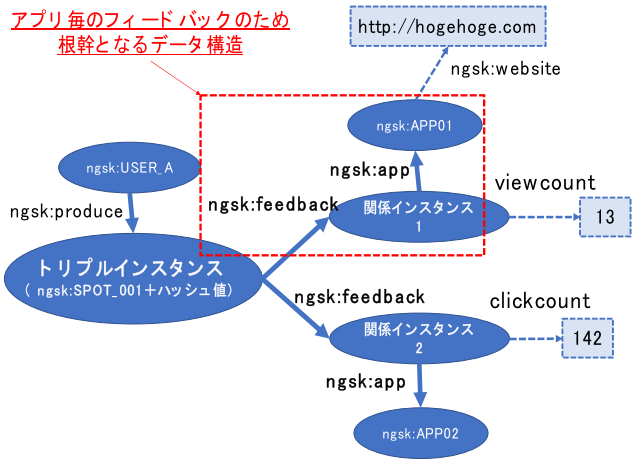


図 7　フィードバックを可能とするデータモデル
Fig. 7　Data model for feed back.

は 142 というように，外部アプリケーションごとに独立してデータ化できる．なお，アプリケーションを識別するインスタンスである図中の ngsk:APP01 や ngsk:APP02 は，LOD の利用時に登録する方法や，データアクセス時の参照元情報から生成する方法が考えられる．以下に，外部アプリケーションが LOD データを利用した際のフィードバックデータの追加処理と，データ入力者が自身の利用実績を確認するための処理を示す．

4.2　フィードバックデータの追加

本項では，図 7 に示したデータモデルを実際に利用する際の処理を示す．まず，外部アプリケーションからインスタンストリプルへのアクセスがあった際に，インスタンストリプルとアプリケーションの関係インスタンスの生成や，表示回数などの初期化を行い，図 7 の根幹部分の構造を作る．なお，インスタンストリプルとアプリケーションの関係インスタンスは連結することで一意に表現可能である．外部アプリケーションから利用実績を更新する場合は，すでに取得済みのインスタンストリプルと自身を示すアプリケーションインスタンスから関係インスタンスを特定できるため，特定できた関係インスタンスの利用実績の更新クエリを発行するのみである．**図 8** に，初回アクセス時と利用実績を更新する際のパラメータを汎用化した SPARQL クエリと，**図 9** に，生成できるフィードバックデータのイメージを示す．

パラメータ F に，フィードバックしたい値を外部アプリケーションごとに柔軟に設定できる．

4.3　フィードバックデータの確認

ここでは，図 7 に示したデータモデルから，あるデータ入力ユーザが登録したデータに対して，各アプリケーションの利用実績を参照するクエリを**図 10** に示す．図中の変数 X に，管理済のユーザ ID を入れることで，各データ入力ユーザが入力したデータと，そのデータがどのアプリでどのような利用実績が登録されているかを一覧できる．**表 1** に，その結果の一例を示す．このようなフィードバックデータを，データ入力ユーザが入力した情報の実績として確認することが可能である．

3 章，4 章で示したデータモデルおよびデータ処理モデルを**図 11** に示す．データ入力ユーザ X が入力したトリプルデータ（S-P-O）と，トリプルインスタンス（SPO）は，

【初回アクセス時】

```
prefix ngsk: <http://nagasakipj.or.jp#>
INSERT{
?hash ngsk:feedback ?hash_app.
?hash_app ngsk: app ?app.
?hash_app <F> "0".
}
WHERE
{
BIND(URI(SPO) as ?hash).
BIND(URI(CONCAT(SPO,APP)) as ?hash_app).
BIND(URI(APP) as ?app).
}
```

【利用実績更新時】

```
DELETE {?s <F> ?count.}
INSERT{?s <F> ?countUP. }
WHERE{
< SPO > ngsk:feedback ?s.
?s ngsk:app < APP >.
?s <F> ?count.
BIND(xsd:integer(?count)+1 as ?countUP).
}
```

図 8　フィードバッククエリモデル
Fig. 8　Data model for feed back.

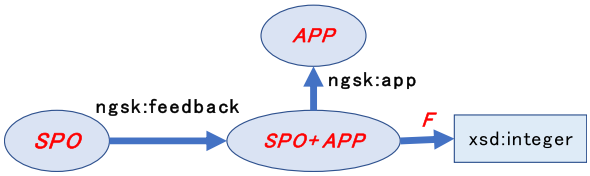


図 9　フィードバッククエリが生成するトリプルデータ
Fig. 9　Tripe data generated by feedback query.

```
prefix ngsk: <http://nagasakipj.or.jp#>
SELECT ?s ?app ?usage_property ?usage_data
WHERE
{
  {SELECT ?hash
    WHERE
    {<X> ngsk:produce ?hash.}
  }
?s ?p ?o.
BIND(CONCAT(STR(?p),STR(?o)) as ?po).
BIND(CONCAT(STR(?s),STR(SHA1(?po)))
as ?spo).
FILTER(STR(?hash)=?spo).
?hash ngsk:feedback ?relation.
?relation ngsk:app ?app.
?app ?usage_property ?usage_data.
}
```

図 10　フィードバックデータ確認用クエリ
Fig. 10　Query for confirmation of feedback data.

© 2019 Information Processing Society of Japan

表 1　結果
Table 1　Result.

?s	?app	?usage_property	?usage_data
ngsk:SPOT_001	ngsk:APP01	ngsk:viewcount	5
ngsk:SPOT_001	ngsk:APP02	ngsk:clickcount	15

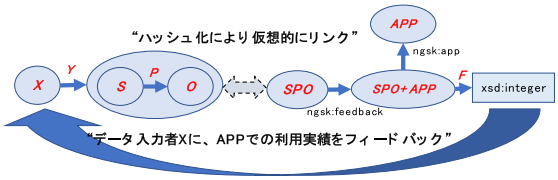


図 11　フィードバックのためのデータモデル・データ処理
Fig. 11　Data processing to feedback on the data model.

ハッシュ化により間接的にリンクされている状態にあるため，外部アプリケーション APP での利用実績をデータ入力ユーザ X にフィードバックすることが可能である．

5.　LOD プラットフォーム

本項では，上述のデータモデルおよび一般化したクエリモデルを実際にシステムに導入できることを示すために構築した LOD プラットフォームについて紹介する．本プラットフォームを利用するアプリケーションとして，観光に関連する街の情報を登録できるアプリケーションをプロトタイプとして開発した．

5.1　データアーキテクチャ

本研究が提案するデータモデルにより，LOD としてトリプルデータを公開するだけでなく，そのトリプルデータをインスタンス化して所有者情報を付加することや外部アプリケーションでの利用実績などをフィードバックできる．そのため，LOD としてデータを公開し，他のデータセットと連携するだけでなく，LOD データのメタ情報を公開する構造が必要となる．本データモデルを LOD として利用する際のデータアーキテクチャを**図 12** に示す．図 12 に示すとおり，LOD として公開するオープンデータレイヤーと，トリプルデータのメタ情報を公開するためのメタデータレイヤーで構成される．LOD データのメタ情報（所有者など）を確認するために必要となるハッシュ関数やその SPARQL クエリや，フィードバックをするための SPARQL クエリは，事前に公開しておく必要がある．これは，データは LOD として公開する際に必要となるエンドポイントへのアクセス方法やデータ構造などを，LOD 利用者に通知しておく作業と類似したステップである．

なお，図中のトリプルデータのメタ情報の取得は，図 10 に示したようなトリプルデータとハッシュ化されたインスタンスを FILTER により抽出するクエリにより可能である．また，図中のトリプルデータに対するフィードバックは，図 8 のクエリにより実現できる．

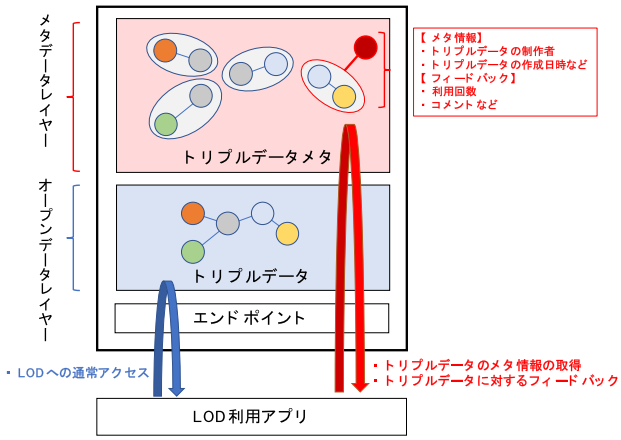


図 12　データアーキテクチャ
Fig. 12　Data architecture.

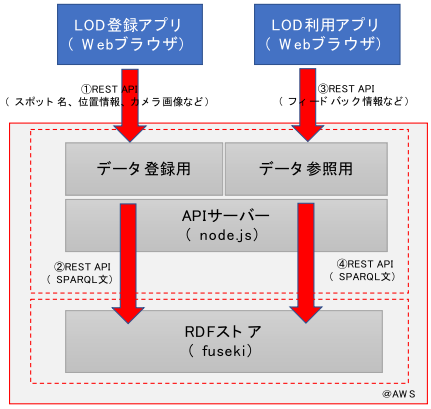


図 13　システムアーキテクチャ
Fig. 13　System architecture.

5.2　システムアーキテクチャ

本項では，上述したデータモデルやデータアーキテクチャを利用するための，データ生成・参照アプリケーションとそのシステムアーキテクチャについて示す．

Web アプリケーション側で SPARQL クエリを生成して，RDF ストアにクエリを送信することも可能ではあるが，クエリが見えてしまうというセキュリティ観点や，LOD 利用アプリの効率的な開発を実現するため，データ処理機能と，データ蓄積機能を API 化してクラウド上に実装する．また，データ処理機能は，データ登録アプリケーション用と，データ利用アプリケーション用に分けて構成する．つまり，図 5 に示すクエリと，図 8 に示したクエリを利用するアプリケーションを分けることを意味する．**図 13** に，システムアーキテクチャを示す．データ登録用 API では，クライアントである Web アプリケーションから，スポット名や緯度・経度情報やカメラ画像を引数として POST コマンドにより受け付ける．その際，クライアントアプリケーションからは，図 5 に示すパラメータを引数として API サーバに送信する．API サーバでは，図 4 の SPARQL 文を生成し，後段にある RDF ストアに送信する．データ参

© 2019 Information Processing Society of Japan

図 14　登録アプリケーション・スポット登録画面
Fig. 14　The screenshot of the registration application.

照用 API も同様に，図 8 のパラメータを引数として POST コマンドにより受付け，SPARQL 文を生成し RDF ストアに送信する．

データ登録アプリケーションは，さらに，図 10 に示すフィードバックデータ確認クエリにより，データ入力ユーザが登録したデータに対するフィードバックを確認可能である．

5.3　データ生成クライアントの試作

登録する街の情報は，観光地や公園といったスポット情報であり，その内容は位置情報やジャンルなどが考えられる．また登録するデータは，テキストデータだけとは限らず，画像や動画なども考えられる．効率的なデータ入力のためには，標準的に実装されている Web ブラウザで入力でき，かつ，モバイルデバイスの状態（ロケーション情報など）を活用することが重要である．本項では，上述のデータ登録用 API を利用した Web アプリケーションを紹介する．**図 14** に，スポット情報を登録する画面を示す．

Web ブラウザから，カメラ起動や，GPS データの取得などを可能としている．また，入力データを登録用 API 経由で RDF ストアに登録することができる．さらに，図 14 に示すデータ生成クライアントアプリケーションでは，5.4 節で述べるデータ参照アプリケーションからのフィードバックデータを確認する画面もある．

5.4　データ参照アプリケーションの試作

本項では，LOD を利用する外部アプリケーションから，トリプルデータに直接フィードバックできるかを検証するため，外部アプリケーションを想定したデータ参照アプリケーションも開発したので本項で紹介する．図 13 に示すシステムアーキテクチャの LOD 利用アプリに該当する．

図 15　参照アプリケーションでのフィードバック例
Fig. 15　Example of feedback in reference application.

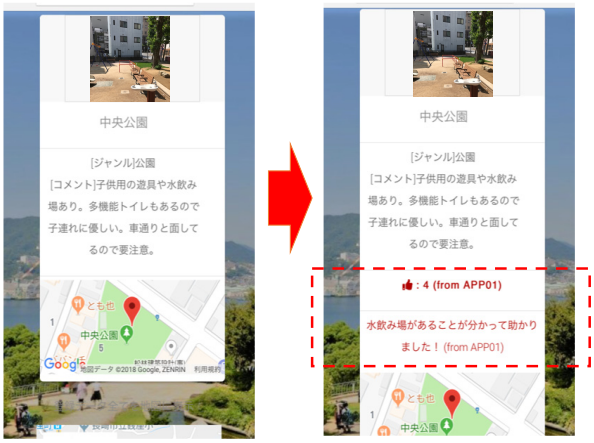


図 16　登録済スポット情報確認画面
Fig. 16　Registered spot information confirmation screen.

上述したとおり，フィードバック用のデータを連携するための述語は，柔軟に設定できる．**図 15** に示すデータ参照アプリケーションでは，登録された街の情報を一覧できるだけでなく，「いいね」を押すことやコメントを入力可能である．押された「いいね」の回数や，コメントは，図 8 に示したクエリにより，図 13 のデータ参照用 API を経由して RDF ストアにフィードバックデータとして登録することができる．

5.5　データフィードバックの確認

図 16 に，データ登録アプリケーションでの，登録済スポットデータの確認画面を示す．外部アプリケーションからのフィードバックがある場合は，その内容を確認することができる．図 16 が示すように，図 15 の参照用アプリケーションにて「いいね」が押下され，また，コメント登録された後，その内容がフィードバックされていることが分かる．

6.　評価

提案するデータ構造に基づき開発した Web アプリケー

© 2019 Information Processing Society of Japan

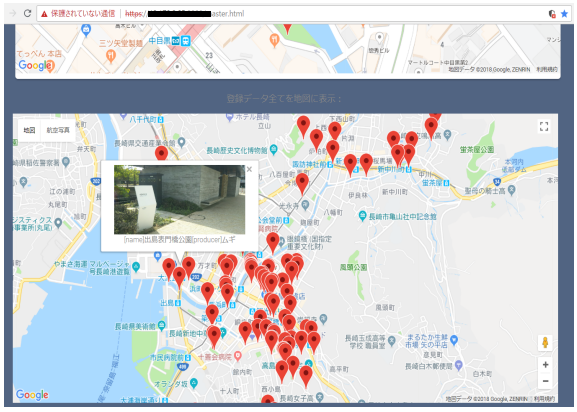


図 17　登録されたスポット一覧
Fig. 17　Registered spot list.

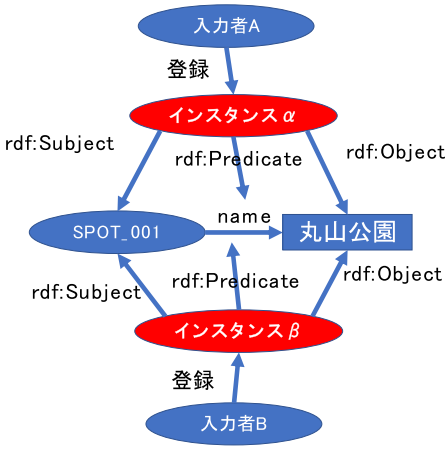


図 18　Statement における重複表現
Fig. 18　Duplicate expression by RDF Statement.

ションを使い，長崎市民に街の情報を入力してもらうワークショップを実施した．本章では，ワークショップの結果から，データモデルや，フィードバック型 LOD の有用性について評価する．

6.1　ワークショップについて

長崎都市・景観研究所[*1]の協力のもと，2018 年 5 月 27 日に長崎市にて，街を歩きながら街の情報（公園，トイレ，休憩スポットなど）を入力するワークショップを実施した．参加者は長崎市在住の 7 名で，2 時間かけて行った．その結果，49 件のスポットデータが登録された．**図 17** にスポット一覧を図示する．なお，5.2 節に示したデータ生成クライアントアプリケーションへのログインに関しては，プライバシーの観点から，参加者が一意に識別できるニックネームなどの名前を適宜決めてワークショップを実施した．

Web ブラウザからアクセスでき，簡単な操作で情報登録できるため，登録をスムーズにすることができたというコメントを得た．また，フィードバックを受けることで街づくりに貢献できている達成感が得られるとの感想を得ることができ，入力作業へのフィードバックが重要であることが分かった．その一方で，たとえば同じ公園を，違う人が重複して登録するといった課題もあった．この課題は，1 つ 1 つのスポットデータに対して，誰がいつ入力したのかを明確にして，重複なくインスタンス化できる必要性を示唆している．

6.2　データモデルの有用性

1 つ 1 つのデータに対して，誰がいつ入力したのかを担保できるか，データモデルという観点で評価する．**図 18** に示すとおり，RDF Statement の場合は同じトリプルに対して複数のインスタンス化が可能となる．図 18 に示すインスタンス α とインスタンス β を生成した後，同一のトリプル（図中の SPOT_001–name–丸山公園）の主語，述語，目的語に対して，RDF Statement を構成する rdf:Subject, rdf:Predicate, rdf:Object をそれぞれ設定することができる．そのため，ある 1 つのスポットのインスタンスを表すデータを複数保持することができてしまう．

一方，本論文で提案するデータモデルは，トリプルを構成する，主語，述語，目的語をもとにインスタンスを生成するため，3.1 節の (2) の SPARQL クエリにより 1 つのインスタンスしか生成できない．また，このデータモデルを用い，3.1 節の (1)〜(3) の処理をトランザクション化した図 4 の SPARQL クエリを用いることにより，複数ユーザが入力したと表現できないようにすることが可能である．そのため，フィードバックを入力ユーザに確実にするためには，提案するデータモデルが有用である．

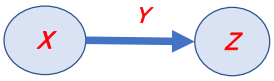
さらに，アプリケーション開発を通して，データ管理の面でも提案するデータモデルは有用であることが分かった．その理由として，元になったトリプルの主語 URI をそのまま採用し拡張するため，運用者や開発者が見ても理解しやすい構造になっている．また，インスタンス化する元のトリプルデータとは直接つなぐ必要がないため，別のデータベースとして設計することができるため，相互に影響を与えない．

データ処理に関しては，Statement の場合は，1 つのトリプルデータをインスタンス化する場合に，1 つのインスタンスと 3 つのトリプルを要するため，提案したデータモデルに対してデータ量が 4 倍になる．また，図 5 に示すデータ登録クエリにも 3 行追加する必要がある．提案するデータモデルでは，インスタンス 1 つのみで表現できるため，クエリに簡潔に一般化表現可能である．ただし，図 10 に示すように，提案したデータモデルでは，トリプルデータから，そのハッシュ化されたインスタンスを特定するためには，SPARQL クエリにハッシュ化処理や FILTER 処理を組み込む必要がある．提案手法と既存手法それぞれにおい

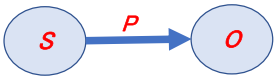
*1　http://nullproject.blog31.fc2.com/

© 2019 Information Processing Society of Japan

提案モデルとトリプルデータ特定クエリ

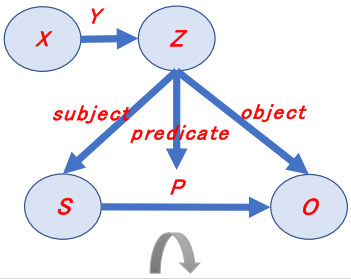


※ZはS-P-Oをハッシュ化したデータ




```
select ?s ?p ?o
where
{
<X> <Y> ?z.
?s ?p ?o.
BIND(CONCAT(STR(?p),STR(?o)) as ?po).
BIND(URI(CONCAT(STR(?s),SHA1(?po)))
as ?spo).
FILTER(?z=?spo).
}
```

Statementデータモデルとトリプルデータ特定クエリ




```
select ?s ?p ?o
where
{
<X> <Y> ?z.
?z <subject> ?s.
?z <predicate> ?p.
?z <object> ?o.
}
```

図 19　データ処理性能検証のためのデータとクエリ（上：提案手法，下：既存手法）

Fig. 19　Data and queries for data processing performance verification.

て，インスタンス化に必要となるデータ数と，インスタンス化されたデータから，インスタンス化元のトリプルデータを抽出するための検索性能を，トリプルデータ数ごとに比較した（図 **19**）．検索応答の検証は，RDF ストアである Apache Jena Fuseki（Version 3.7.0.）を Windows10Home 64 bit（Core i5-6267U RAM8.00 GB）で動作させて実施した．**表 2** に示すとおり，既存手法である Statement に比べて，インスタンス化するためのデータ数が 4 分の 1 になる一方，検索応答が長くなる結果となった．データ処理性能については改善の余地はあるが，RDF サーバのインデックス化などによる改善も可能である．本論文の目的である，データ入力者がオープンデータの利用実績を把握できるようにするためには，1 つ 1 つのデータに対して，誰がいつ入力したのかを担保できるかが最も重要である．そのため，本論文で提案したデータモデルは有用である．

表 2　データ処理性能に関する評価結果

Table 2　Evaluation result on data processing performance.

トリプル数（図中のS-P-O数）	生成されるデータ数		検索応答時間（秒）※10回の平均	
	提案手法	既存手法	提案手法	既存手法
100	100	400	0.7	0.5
300	300	1200	2.3	0.7
500	500	2000	5.3	0.8
1000	1000	4000	17.4	1.6
1500	1500	6000	39.4	1.8
2000	2000	8000	68.2	2.6
2500	2500	10000	105.0	3.2
3000	3000	12000	142.4	5.0

7.　まとめ

本論文では，オープンデータの発展に必要な継続的なデータ生成を実現するため，トリプルデータへの利用状況の付与および入力者へのフィードバックに着目し，トリプルデータをハッシュ値によりインスタンス化するデータモデルや，データ生成およびフィードバックを可能とするデータ処理を提案した．さらに，データモデルの有用性を検証するため，登録用と参照用に開発したアプリケーションを用いて，実際に長崎市にてデータ入力ワークショップを実施した．登録データの一意性を確保することや確実なデータ管理などの観点から，本論文が提案するデータモデルが有用であることが確認できた．

本論文は，LOD の発展に貢献できるものであるため，W3C などに対してデータモデルの標準化を提案していくとともに，長崎市の街づくりにつながる市民によるデータ入力を継続し，API やデータストアの公開を検討する．なお，本研究では，観光に関する街の情報を対象としたが，防災や防犯など，幅広く展開することが可能である．また，長崎だけでなく他の街にも応用可能である．本検証においては，ニックネームのみでログイン可能としたが，ユーザ情報の暗号化やユーザ認証の強化を行い，外部ポイントシステムとの連携といったデータプラットフォームの整備および拡張に取り組む．

謝辞　本研究は長崎都市・景観研究所の協力のもと検証することができた．関係者には謹んで感謝の意を表する．本研究の一部は，総務省・戦略的情報通信研究開発推進事業（SCOPE）の助成を受けて実施したものである．

参考文献

[1]　官民データ活用推進基本法，入手先 ⟨https://www.kantei.go.jp/jp/singi/it2/hourei/detakatsuyo_honbun.html⟩（参

© 2019 Information Processing Society of Japan

照 2018-07-01）.
[2] 世界最先端 IT 国家創造宣言･官民データ活用推進基本計画，入手先 ⟨https://www.kantei.go.jp/jp/singi/it2/kettei/pdf/20170530/siryou1.pdf⟩（参照 2018-07-01）.
[3] Linked Data - Design Issues, available from ⟨https://www.w3.org/DesignIssues/LinkedData.html⟩ (accessed 2018-07-01).
[4] RDF 1.1 Semantics, available from ⟨https://www.w3.org/TR/rdf11-mt/⟩ (accessed 2018-07-01).
[5] 乙守信行，湯本正典：Linked Data とメディア—メディアが Linked Data を活用する理由，情報処理，Vol.52, No.3, pp.293–299 (2011).
[6] ちばレポ，入手先 ⟨https://chibarepo.secure.force.com/CBC_VF_WebReportMap⟩（参照 2018-08-01）.
[7] 政府 CIO ポータル，入手先 ⟨https://cio.go.jp/policy-opendata⟩（参照 2018-07-01）.
[8] 本田正美：地域活性を担うアクターの顕在化，pp.251–254 (2015).
[9] 本田正美：オープンデータ化された「ちばレポ」の登録者情報を基にした行政アプリケーションの浸透過程の推定，情報知識学会誌，Vol.26, No.2, pp.187–194 (2016).
[10] 田畑恒平：スマートデバイス向けアプリケーションとゲーミフィケーションによる地域活性化の可能性，江戸川大学紀要 26 (2016).
[11] RDF Schema 1.1 Statement, available from ⟨https://www.w3.org/TR/rdf-schema/#ch_statement⟩ (accessed 2018-07-01).
[12] Vinh, N., Bodenreider, O. and Sheth, A.: Don't like RDF reification?: Making statements about statements using singleton property, *Proc. 23rd International Conference on World Wide Web*, ACM (2014).
[13] Carroll, J.J., Bizer, C., Hayes, P. and Stickler, P.: Named graphs, Web Semantics: Science, *Services and Agents on the World Wide Web*, Vol.3, No.4, pp.247–267 (2005).
[14] Hernández, D., Hogan, A. and Krötzsch, M.: Reifying RDF: What works well with wikidata?, SSWS@ISWC, Vol.1457, pp.32–47 (2015).
[15] RDF 1.1 Query Language, available from ⟨https://www.w3.org/TR/sparql11-query/⟩ (accessed 2018-07-01).

浦川 真　（学生会員）

2003 年横浜国立大学工学部卒業．2005 年同大学大学院環境情報学府修士課程修了．同年日本放送協会（NHK）に入局しセマンティック Web 技術を活用した映像メディアの高度化に従事．2018 年からアクセンチュア株式会社にてメディアコンサルタントとして従事．同年から長崎大学大学院工学研究科にてメディア・データ処理技術の研究に従事．修士（情報学）．

荒井 研一　（正会員）

2004 年信州大学工学部情報工学科卒業．2006 年同大学大学院工学系研究科博士前期課程修了．2010 年同大学大学院総合工学系研究科博士課程修了．博士（工学）．以来，情報セキュリティに関する研究に従事．2011 年東京理科大学理工学部・嘱託助教．2015 年より長崎大学大学院工学研究科情報工学コース・助教．電子情報通信学会，日本応用数理学会各会員．

小林 透　（正会員）

1985 年東北大学工学部精密機械卒業．1987 年同大学大学院工学研究科修士課程修了．同年 NTT 入社．以来，ソフトウェア生産技術，情報セキュリティ，データマイニング，Web 技術等の研究開発に従事．2013 年から長崎大学大学院工学研究科教授．IEEE，電子情報通信学会各シニア会員，博士（工学）．

© 2019 Information Processing Society of Japan