プライベート情報のLOD公開

2015年1月18日

日本総合システム株式会社
中川雅三

概要

- 目的：個人や組織のプライベートな情報を適切に公開して、役立てる。
  - 高齢者サポート、家電製品の進化、ビジネスツール…
  - 興隆しつつあるIoTを生かす一つの方法でもある。
  - プライベートな情報のためのLOD（以下LOD+と略）をつくる。

- アイデア１：Linked Dataアーキテクチャを拡張したLOD+を提案する。
  - ユビキタスなSPARQLエンドポイントでIoTや様々な機器、人々をつなぐ。
  - 互換性を保ちながら、SPARQLエンドポイントへ機能追加する。
    - 公開範囲の制御
    - 動的な変化への対応
    - 語彙の進化のサポート

- アイデア２：LOD+を実装するLOD Gatewayと呼ぶアーキテクチャを提案する。
  - 既存の実装をそのまま利用しながら、機能拡張をLOD Gatewayで行う。
  - これまでのSPARQLクエリを変更せずに、アクセス制御などを実現できる。

目的

- 個人や組織のプライベートな情報を適切に公開して、役立てる。
  - 資源に限りある地球上で安心で持続可能な社会を実現してゆくためには、人間の能力の限界をICTで破ってゆくことが現実的な方法である。
  - そのためには今は活用できていない様々な情報を、相手や範囲を選んで公開してゆく必要がある。
    - 限定公開により情報利用の裾野を広げることで、無制限に公開する元々のLODをも豊かにする。
  - 情報には、人の一生、あるいは世代を超えて使い続ける寿命をもたせることも必要である。

活用例- 高齢者の生活

- 自立を支えあう人々と情報を共有する

- たとえば:
  - 人や機器に機微な情報を利用してもらう
    - 小売業者には冷蔵庫の在庫がみえる
    - 介護関係者には行動量などのバイタル情報が見える
    - 消防署員には、誰が災害で屋内に残されているのかがわかる
    - 屋内の家電やロボットが情報共有し、生活者をサポートする。
  - 互いにプライバシーをコントロールしながらコミュニケートする
    - 近所の人が、住人の家族や支援者と緊急連絡をとれる。相手がどの程度信頼できそうかがわかる。
    - 宅配業者が介護担当者と連絡をとれる。
    - 外出できない人が、趣味の集まりに参加する。
    - プライベートな問題を衆知を使って解決する。

活用例- 家電

- メーカー・サービス業者・ユーザがそれぞれ利益を得ながら持続可能社会を実現してゆく

- たとえば
  - メーカーにかかわらず自分が選んだ業者に保守を頼む。
  - 家電やロボットを買い換えたとき、設定を自動的にひきつぐ。
  - 機械やサービス業者がユーザーの安全や健康のための情報を取得する
  - 様々な業者が、自社・他社や他業界の製品・サービス利用状況を調査する
    - 製品の改良や新製品・サービスの開発につかう
  - ユーザーが身元を明かさなくても、メーカーからユーザーへ連絡をとれる。

活用例- 会議

- 煩雑なことは機器に任せ、人間は本質的な作業に集中する

- たとえば
  - ある場所に集まるだけで汎用製品が連携する。

    特定製品では実現できているが、汎用製品でそれを可能にする。
    - 会議室にある機器を共同利用する
    - 出席者のPCへ情報を配布する
    - 出席者の携帯エージェントと連絡をとりあう
      - 次回のスケジュールを登録する、など。
    - 遠隔からバーチャルに参加する

LODアーキテクチャの採用

- Linked Open Dataアーキテクチャは、こうした目的に最適である。
  - 利用しやすい形式・アクセス方法である。
    - RDF：単純明快な構造
    - SPARQL：標準化された柔軟なアクセス方法
  - 現在・未来に発生するあらゆるデータ構造に対応できる
    - 人の一生、あるいは世代にわたって情報を継承・追加してゆける。
  - 世界中の関連情報と相互にリンクできる。
  - 「公開」を前提としている。

現在のLODアーキテクチャに足りないこと

- 安全の確保が難しい
  - 機密性:
    - 信用できる相手に、必要なデータを提供したい。
    - 企業は個人情報の保持リスクを回避したい。
  - 可用性:
    - 災害や障害で孤立した環境でも、重要なサービスを動作させたい。
    - 省資源・安定性のために広域ネットワークへ流す情報を限定したい。
    - 秘密を守るために、公開可能なデータまで秘匿してしまうのを防ぎたい。
  - 完全性:
    - サービス業者が廃業したり、業者を代えたりしても、自分のデータを保存・利用したい
    - 廃れた形式の情報も使えるようにしたい。
    - リアルタイム性を保証したい。

アイデア1：LOD+の仕様

2つのアプローチでLOD+を実現する

- LOD+サーバのユビキタス化
  - プライベートな情報が発生する現場にLOD+サーバを置く

- SPARQLエンドポイントの機能拡張
  - データごとの公開制御
    - 個別RDFのデータの公開可否を制御する。
  - 時刻による公開制御
    - 鮮度が必要な情報、古いデータしか公開できない情報などを適切に扱う。
  - IRIの匿名化
    - プライベートな物のユニークIDや、関係者を特定できるIRIを匿名化する。
  - データの翻訳
    - 旧式のデータや異なる文化のデータを新しい構造のデータへ翻訳する。
    - データを処理して機微な情報をとりのぞく

LODサーバのユビキタス化

- LODデータをクラウドではなく、現場（個人宅やオフィス）に置く
  - プライバシーを持ち主が制御できる
    - 企業にとっても個人情報漏洩リスクが無くなるメリットがある。
  - 人の一生、世代にわたって利用できる
    - 企業が持つデータは、サービス撤退や廃業のときに失われることがある。
  - サービス提供者を選択できるようになる
    - データをもつ業者に囲い込まれず、最適なサービスを選択できる。
  - 効率的に処理できる
    - リアルタイム性を確保できる
    - 無駄なトラフィックの発生を防げる
  - 孤立しても動作する
    - 広域災害やネットワーク故障のときにもローカルな動作を続けられる
- 永続データは暗号化してクラウドへバックアップする

目的による公開制御

SPARQLエンドポイントの機能拡張

- 「すべてを無制限公開か非公開か」という二者択一から、個別データの柔軟な公開制御へ。
  - 個別のデータごとにきめ細かく公開を制御すれば、前述したような様々な活用が可能となる。
    - 匿名利用者へ公開
      - 無制限の公開
      - いまここにいる者への公開
      - 第三者が信用を保証する者への公開
    - 認証した者への公開
- 一律非公開としてきたデータセットの中には無制限に公開してもよいものが含まれている。
  - 非公開のための機能を作ることで、実はオープンデータが増える。

時刻による公開制御

SPARQLエンドポイントの機能拡張

- 現在の情報のみを公開
  - 鮮度を保証する情報
    - 例:
      - 今ここにいる人々にとっては、現在のセンサ値が必要。
      - 温度計が壊れていたら、最後に取得した温度ではなく「欠測」という値を返す。

- 過去の情報のみを公開
  - 現在値を公開できない情報
    - 例:
      - 防犯上、現在のセンサ値は秘密
      - センサ値の過去ログを家電メーカーが利用する

IRIの匿名化

SPARQLエンドポイントの機能拡張

- プライベートなユニークIDを公開せずに利用する
  - 例
    - 連絡先をしらせずに連絡をとる
      - 近所の人が独居高齢者の家族と連絡をとりたいときに、LOD+はPROXYのURLをしらせる。
    - 外部のサービス業者へプライベートなIRIを公開したくない。
      - 個人宅の内部構成を推測できてしまうような家電のIRIを、LOD+はハッシュして家電メーカーへ渡す。

データの翻訳

SPARQLエンドポイントの機能拡張

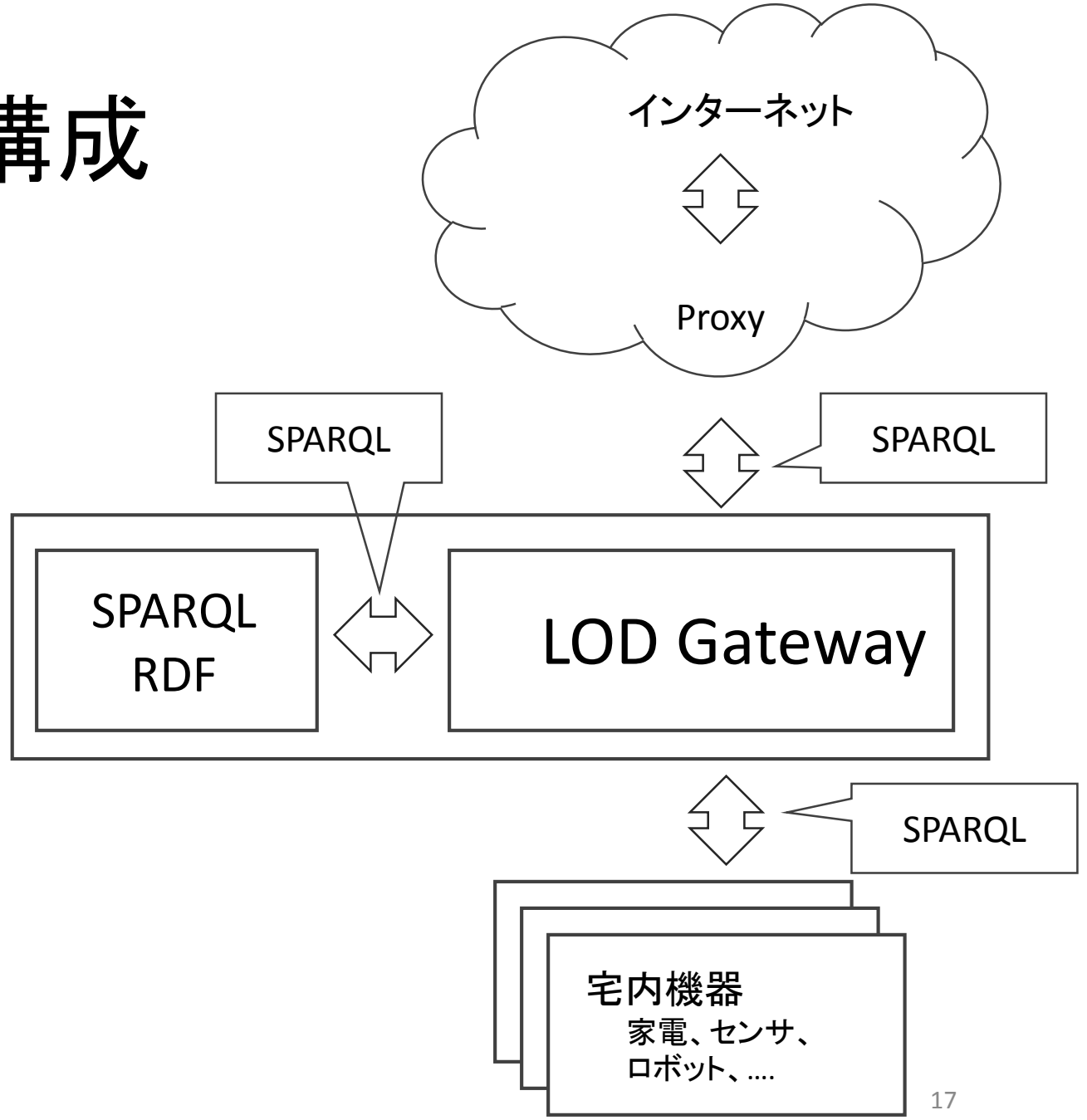
- 長年の利用で変化してゆく語彙を翻訳する
  - 蓄積された古いデータを利用する。
  - 世代の異なる機器が相互に連携する。
  - 引っ越した人が、これまでのデータを異なる地域で利用する。

- データを処理して機微な情報をとりのぞく
  - 家電の利用情報を統計情報へ変換し、家族の生活パターンをみえなくして公開する。

アイデア2：LOD Gatewayによる実装

ユビキタスサーバの構成

- 既存のRDF+SPARQLエンジンへLOD　Gatewayとよぶソフトウエアを追加する
- LOD Gatewayのインタフェース:
  - 宅内機器から接続するSPARQLエンドポイント
  - 専用PROXY経由でインターネットから接続するSPARQLエンドポイント
- クラウド上Proxyの機能:
  - 認証
  - DOSなどの攻撃からの防御
  - プライベートURLの隠蔽



LOD Gatewayの機能

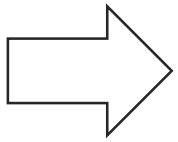
- LOD Gatewayの動作
  - 情報利用者は通常のSPARQLリクエストをLOD Gatewayへ送信する
  - LOD GatewayはSPARQLリクエストを変換して、SPARQLエンジンへ実行させる
  - RDFストアには、アクセス制御のためのRDFなど必要な情報を追加しておく

- リクエストと応答の変換
  - アクセス権限のチェック
    - OPTIONAL文でアクセス制御情報を読み出し、FILTER文で許可されている情報だけを抽出する。
    - 鮮度や公開可能時刻のチェックも行う。
  - 内部IRIの匿名化
    - 応答に含まれるIRIのうち、自空間に属するものを匿名化する。
    - 応答に含まれるURLのうち、指定されたものをPROXYのURLへ変換する。
  - 概念の翻訳

リクエストの変換例

- アクセス権限によるフィルタリング

```
SELECT  ?s ?p ?o
WHERE { ?s ?p ?o}
```



```
SELECT ?s ?p ?o
WHERE
{
   { ?s ?p ?o }
   OPTIONAL {
            ?s acc:level ?s_level .
   }
   OPTIONAL {
            ?s acc:attr-level ?def .
            ?def acc:predicate ?p .
            ?def acc:value ?p_level .
   }
   OPTIONAL {
            ?o acc:level ?o_level
   }
   OPTIONAL {
             acc:hide ?p   ?p_hide .
   }
   FILTER (
            (coalesce(?s_level, 1) < 3) &&
            (coalesce(?p_hide, ?p_level, 1) < 3) &&
            (coalesce(?o_level, 1) < 3)
   )
}
```

ロードマップ

- 最初はクラウド上にLOD Gatewayを置く選択肢もある。
  - いますぐに実装をはじめられる。

- LOD Gatewayで実装経験を積む
  - SPARQLエンジンとRDFストアには既存のものを使うことができる。

- LOD Gatewayの機能の一部をSPARQLエンジンへ組み込む
  - LOD Gatewayを使う実装方式の欠点を緩和するため
    - SPARQLのパージングが2回というオーバーヘッドがある。
    - SPARQL変換では実現しにくい機能がある
      - 例：プロパティ・パスを含むクエリでは、マッチング列途中の要素の権限チェックが難しい