ProjectLA

バックエンドの技術解説

RDFを使った三つ組みデータの格納と検索

2013/02/15

クラウド・テクノロジー研究部会 リーダー

荒本道隆

（アドソル日進株式会社）

何故、RDFか？

- 断片的なデータを相互につなぎたい
  - RDFは主語・述語・目的語の三つ組構造で表現
  - 目的語と主語に同じ値を設定して、それぞれをつなぐ
- 属性を事前に決定できない
  - RDFはスキーマレスなので、柔軟に対応できる
  - RDFは多様な情報を関係性で表現できる
- 大量のデータを蓄積・分析したい
  - RDFは構造が単純なので、コンピュータ・パワーが活きる

バックエンドの概要

- 格納はRDF形式
- 検索はSPARQL（RDFクエリ言語）を使用

RDFとは

- RDF：Resource Description Framework
  - 2008年にW3C勧告となった
  - 代表的な使用例
    - RSS（RDF site summary）0.9, RSS1.0
    - **DBpedia**
- 主語、述語、目的語の3つの要素で表現
  - 主語（Subject）：URI
    - http://aitc.jp/LA/rdfs/1
  - 述語（Predicate）：URI
    - http://aitc.jp/LA/model/ski#宿泊先
  - 目的語（Object）：値
    - 「２月スキー旅行」
    - 「ペンション・アルプ」

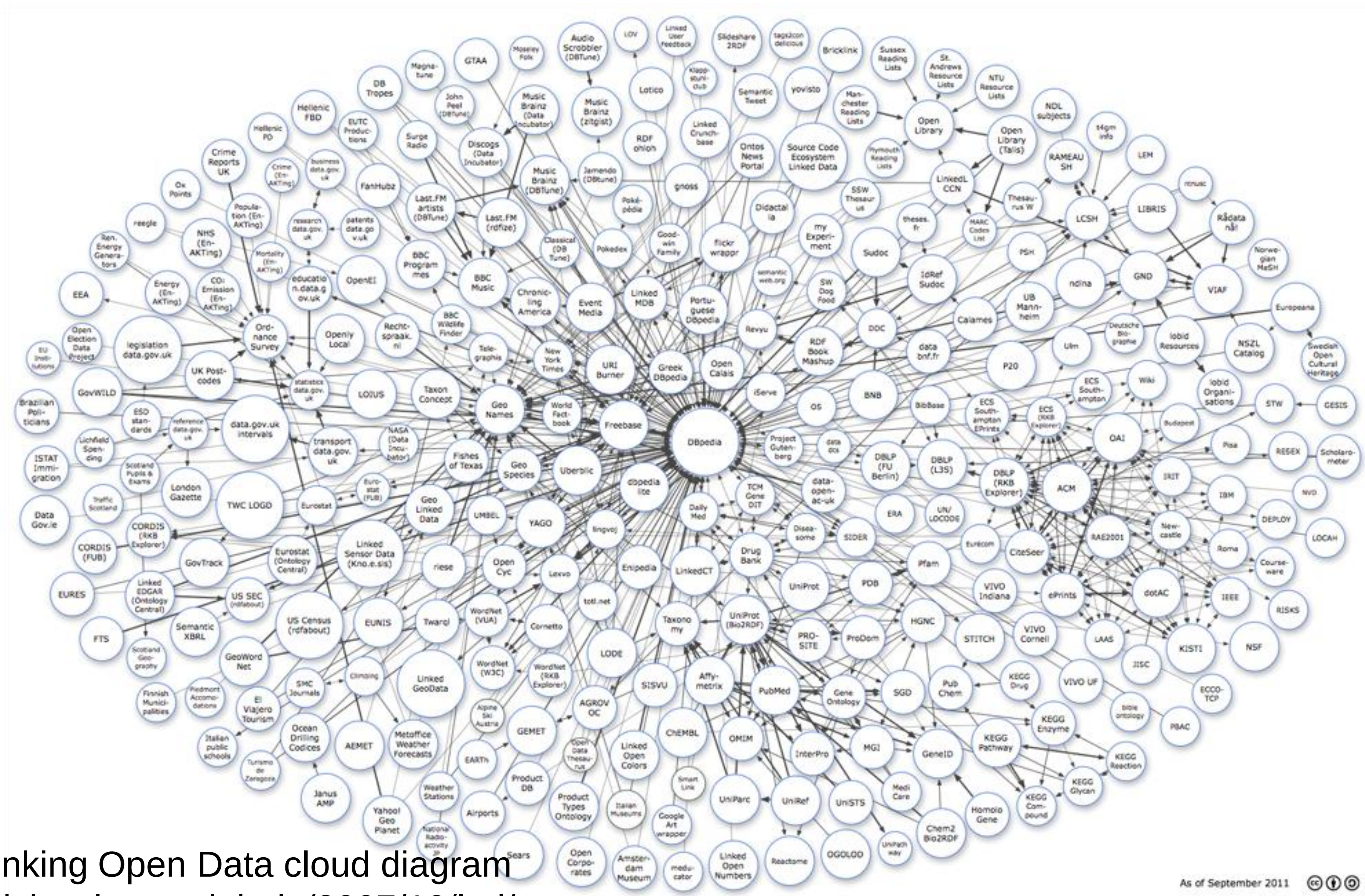
RDFとNoSQLの比較

- RDFとCassandraの比較
  - 主語（Subject）←KEY
  - 述語（Predicate）←Column
    - **どちらもスキーマレス**
    - **どちらも主語（Key）＋述語（Column）でユニーク**
  - 目的語（Object）←Value

  - 主語の名前空間←Column Family
    - 名前空間を変えることで、異なるデータを混在
      - http://aitc.jp/LA/users/1
      - http://aitc.jp/LA/plans/1

RDFで複雑なデータを表現

- 目的語（Object）と主語（Subject）に同じ値を入れる

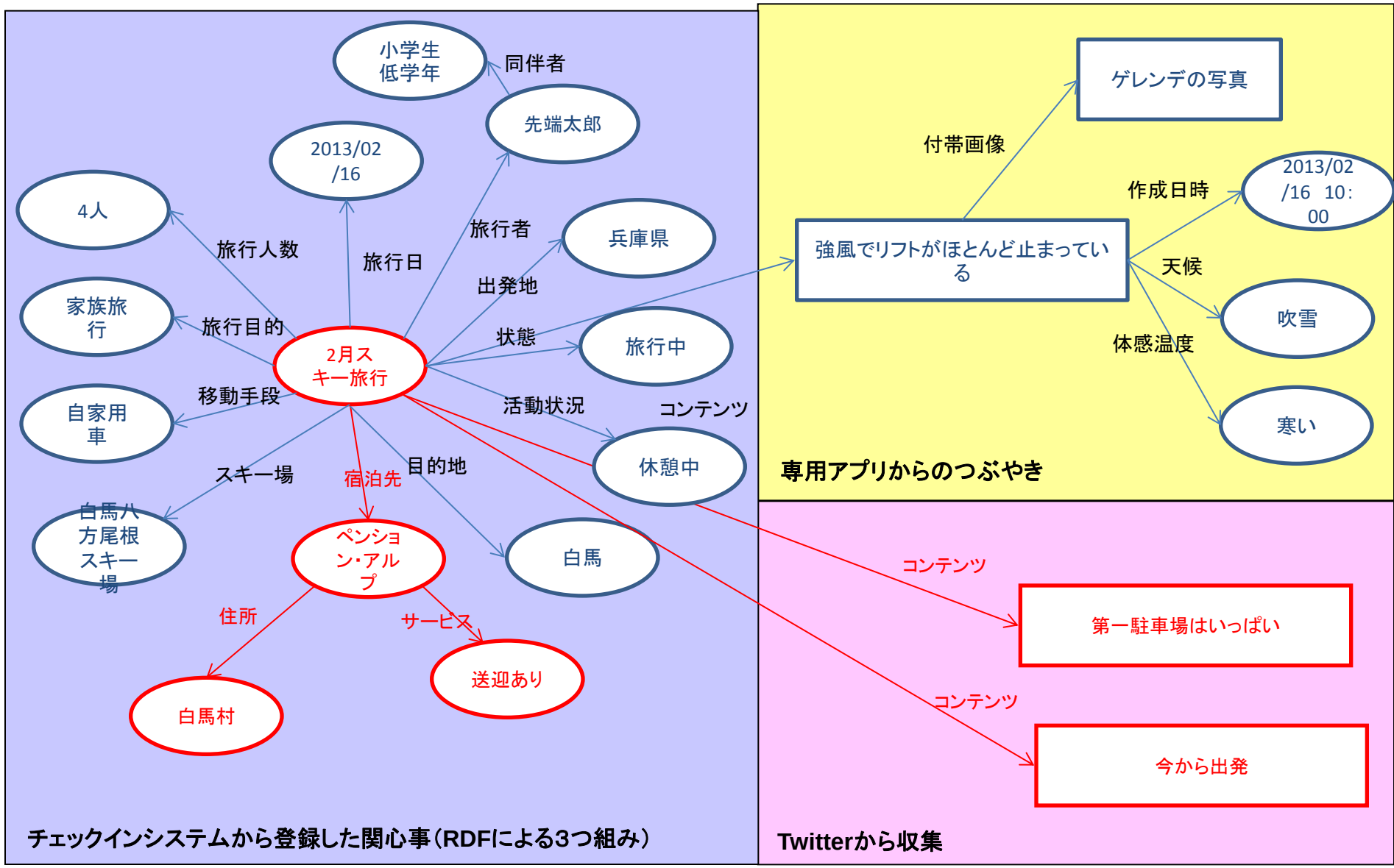


The Linking Open Data cloud diagram
http://richard.cyganiak.de/2007/10/lod/

RDFの実装：Jena

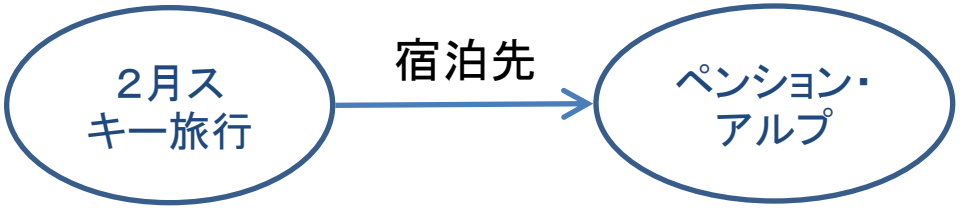
- Apacheのプロジェクトの１つ
  - http://jena.apache.org/
- Java製フレームワーク
  - Semantic Webアプリケーションを構築
  - RDFデータを読み、書き、処理するためのAPI
    - ファイルシステム
    - RDB（Oracle, MS-SQLServer, DB2, PostgreSQL, MySQL, Derby, H2, HSQLDB）
  - RDFとOWLを使ったルールベースの推論エンジン
    - OWL：Web Ontology Language
  - SPARQLのクエリーエンジン
  - RDFデータを公開するためのサーバ

関心事参加者とコンテンツ



RDFへの変換

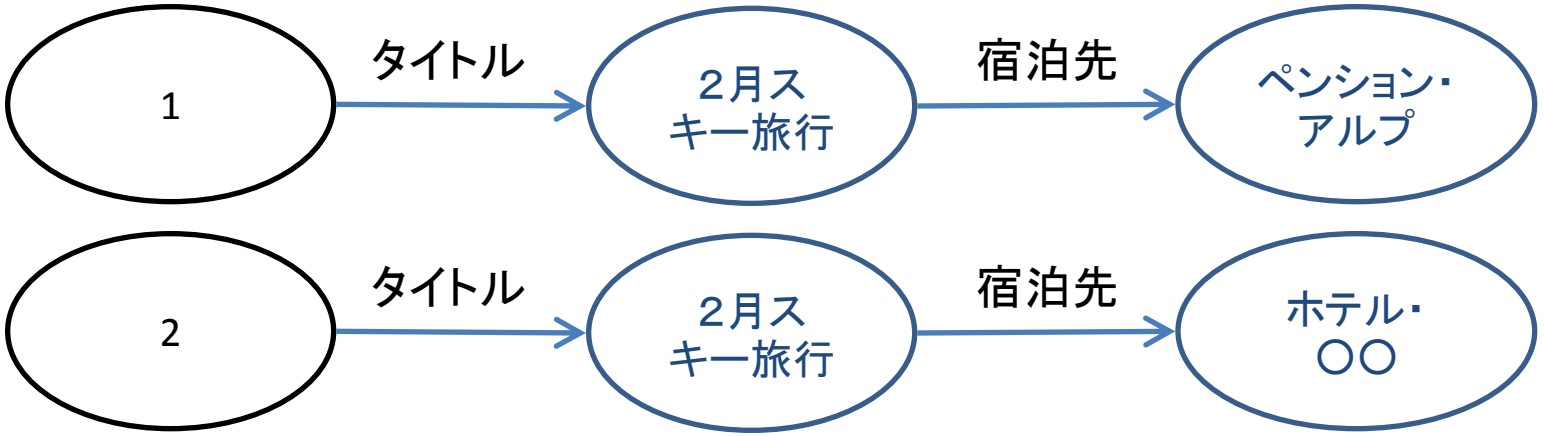
- この部分をRDFにする場合



- 識別できるようにIDを追加

IDを振っただけだと

- これをそのままRDFにすると問題がある



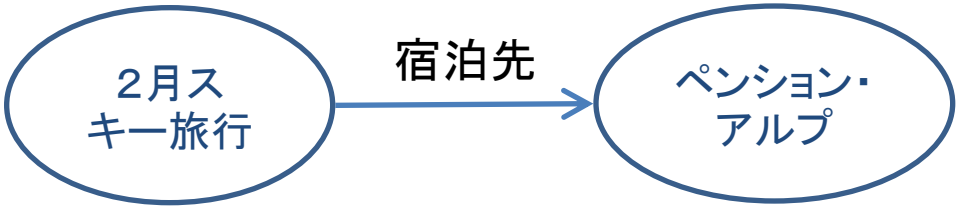
- データがこうなる

主語	述語	目的語
1	タイトル	2月スキー旅行
2月スキー旅行	宿泊先	ペンション・アルプ

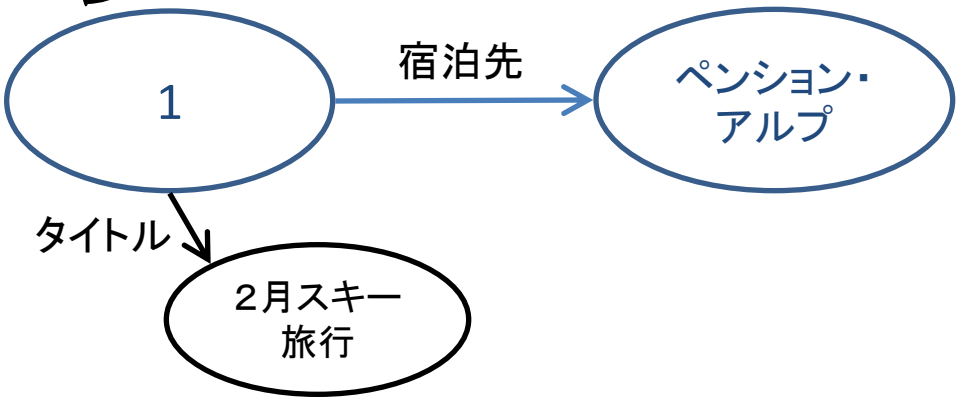
  - 同じタイトルを付けられると区別がつかない

区別可能にするには

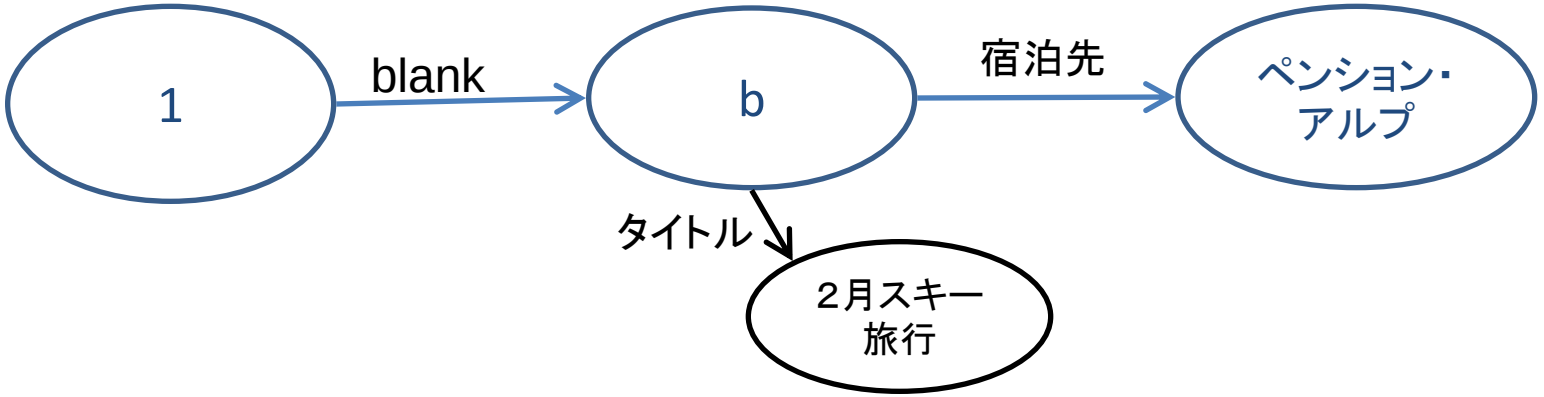
- 元



- IDを追加するパターン

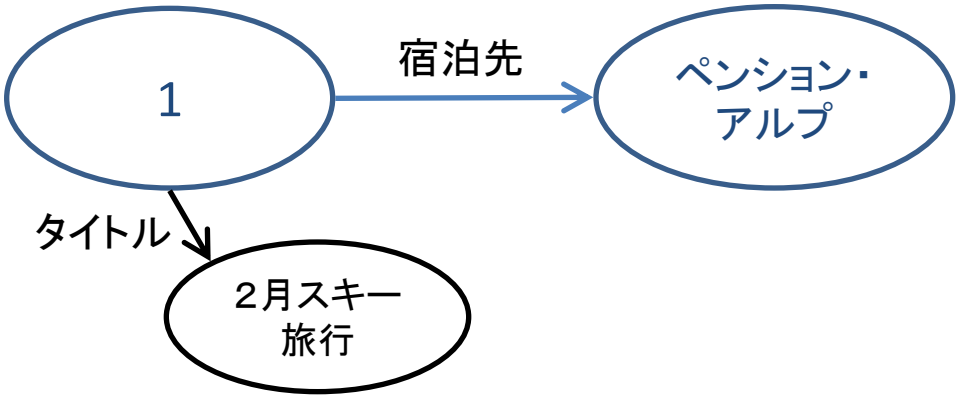


- IDと空白ノードを追加するパターン



IDを追加するパターン

- モデル

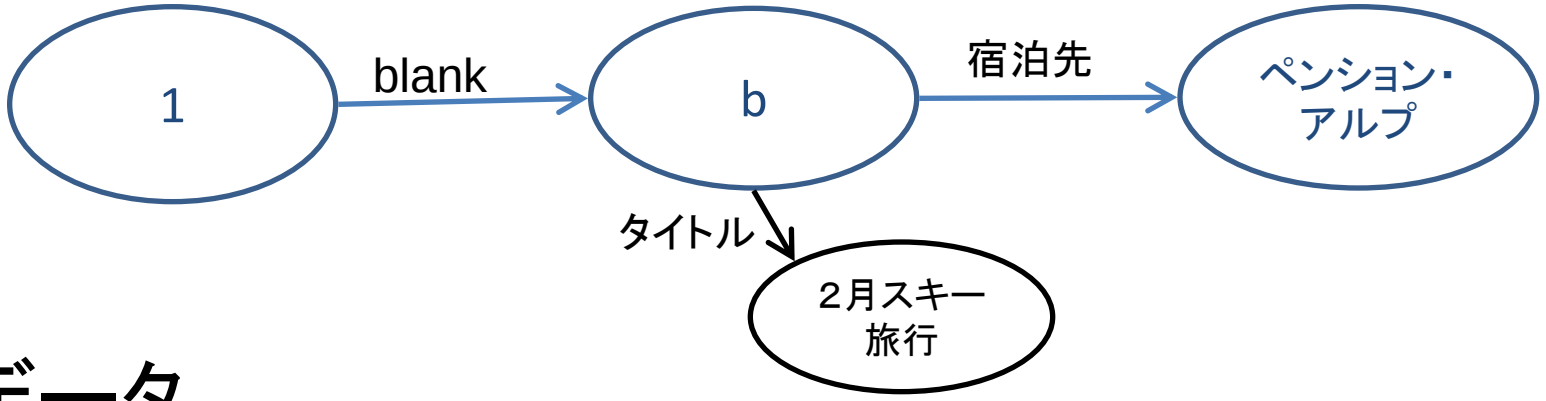


- データ

主語	述語	目的語
1	タイトル	2月スキー旅行
1	宿泊先	ペンション・アルプ

IDと空白ノードを追加するパターン

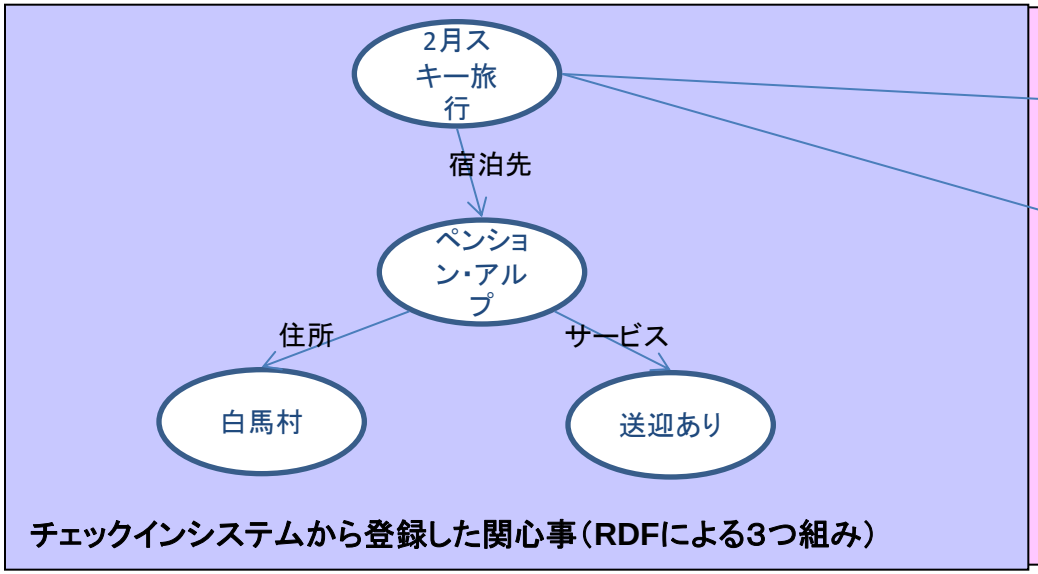
- モデル

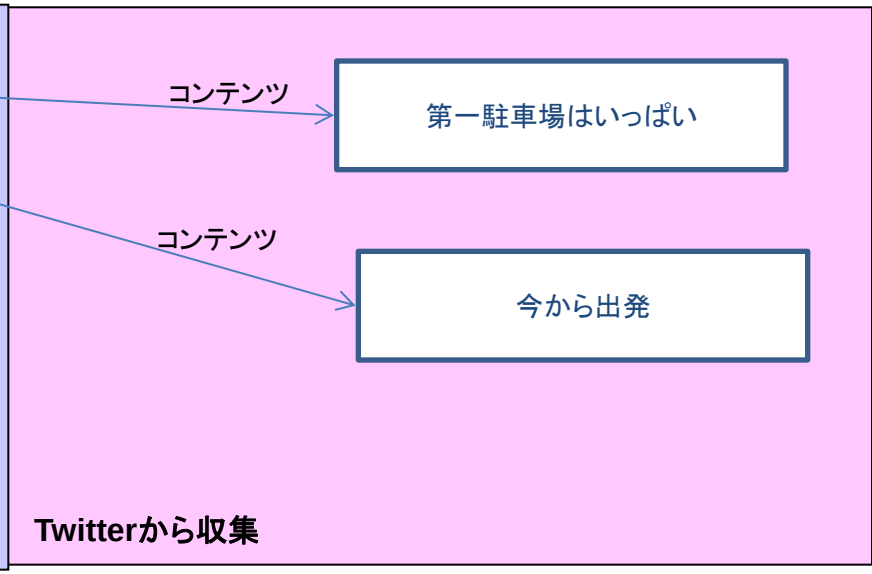


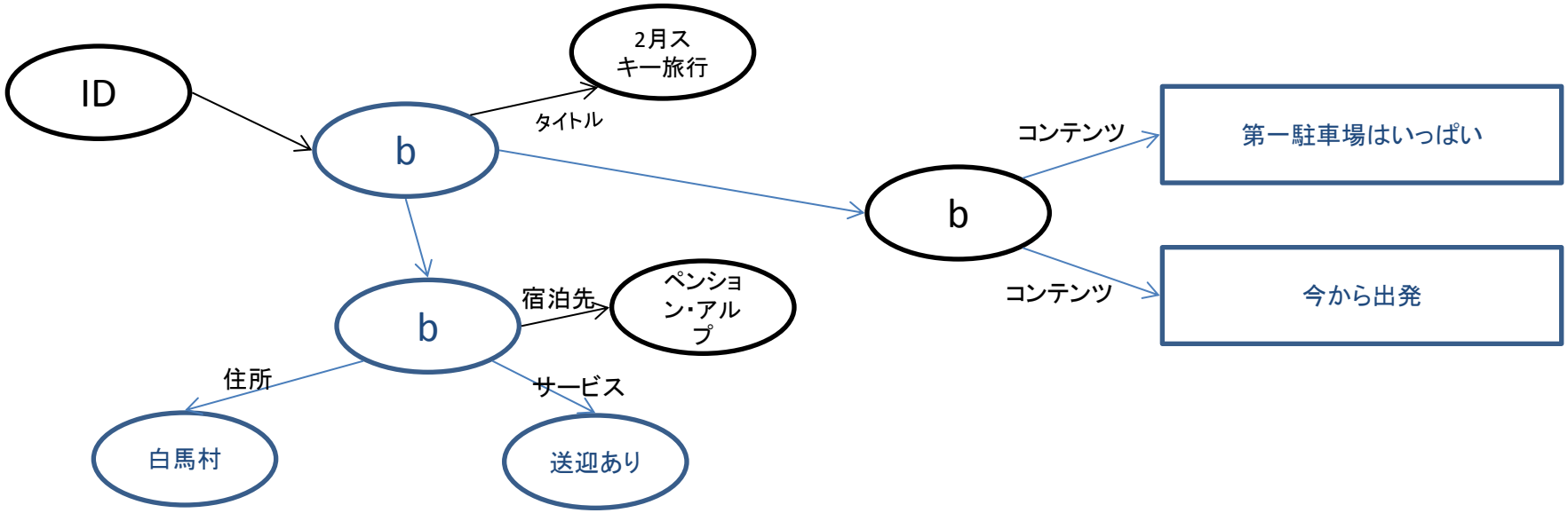
- データ

主語	述語	目的語
1	node	b1
b1	タイトル	２月スキー旅行
b1	宿泊先	ペンション・アルプ

モデルをRDF化し易いよう変換







RDF化に必要なこと

- IDをURIにする
  - http://aitc.jp/LA/users/1
  - http://aitc.jp/LA/plans/1
- 述語の書き方を決める
  - http://aitc.jp/LA/model#blank
  - http://aitc.jp/LA/model#タイトル
  - http://aitc.jp/LA/model/ski#宿泊先
  - http://aitc.jp/LA/model/ski#住所

Jenaを使ってRDFを作成する

```
// ファイルシステム上にデータセットを作成する
Dataset dataset = TDBFactory.createDataset("MyDatabases/Dataset1");
dataset.begin(ReadWrite.WRITE);

// モデルを作成する
Model model = dataset.getDefaultModel();

// IDを作成する
Resource b0 = model.createResource("http://aitc.jp/LA/rdfs/1");

// 空白ノードを作成し、ラベルを追加する
Resource b1 = model.createResource();
b0.addProperty(model.createProperty("http://aitc.jp/LA/model#", "blank" ), b1);
b1.addProperty(model.createProperty("http://aitc.jp/LA/model#", "タイトル"), "2月スキー旅行");
Resource b2 = model.createResource();
b1.addProperty(model.createProperty("http://aitc.jp/LA/model#", "blank" ), b2);
b2.addProperty(model.createProperty("http://aitc.jp/LA/model/ski#", "宿泊先"), "ペンション・アルプ");
b2.addProperty(model.createProperty("http://aitc.jp/LA/model/ski#", "住所"), "白馬村");
b2.addProperty(model.createProperty("http://aitc.jp/LA/model/ski#", "サービス"), "送迎あり");
b1.addProperty(model.createProperty("http://aitc.jp/LA/model#", "blank" ),
        model.createResource()
            .addProperty(model.createProperty("http://aitc.jp/LA/sns#", "twitter" ), "第一駐車場はいっぱい")
            .addProperty(model.createProperty("http://aitc.jp/LA/sns#", "twitter" ), "今から出発")
);
System.out.println("----------------------------------");
model.write(System.out, "RDF/XML-ABBREV");
System.out.println("----------------------------------");
model.write(System.out, "N-TRIPLE");
System.out.println("----------------------------------");

model.close();

dataset.commit();
dataset.end();
```

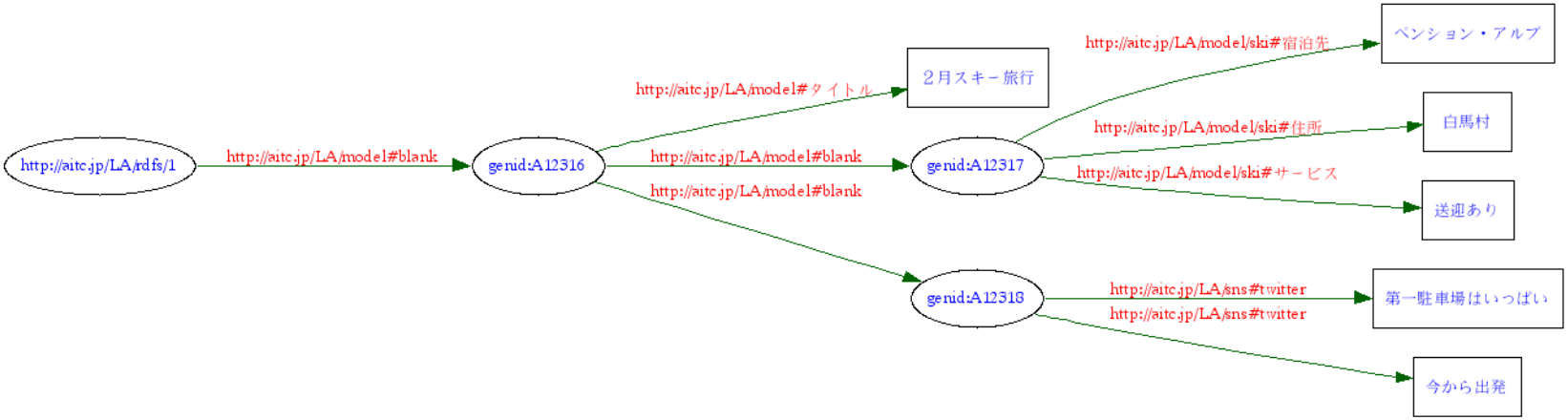
作成したRDF

```
<rdf:RDF
    xmlns:j.0="http://aitc.jp/LA/sns#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.1="http://aitc.jp/LA/model/ski#"
    xmlns:j.2="http://aitc.jp/LA/model#">
  <rdf:Description rdf:about="http://aitc.jp/LA/rdfs/1">
    <j.2:blank rdf:parseType="Resource">
      <j.2:タイトル>2月スキー旅行</j.2:タイトル>
      <j.2:blank rdf:parseType="Resource">
        <j.1:宿泊先>ペンション・アルプ</j.1:宿泊先>
        <j.1:住所>白馬村</j.1:住所>
        <j.1:サービス>送迎あり</j.1:サービス>
      </j.2:blank>
      <j.2:blank rdf:parseType="Resource">
        <j.0:twitter>第一駐車場はいっぱい</j.0:twitter>
        <j.0:twitter>今から出発</j.0:twitter>
      </j.2:blank>
    </j.2:blank>
  </rdf:Description>
</rdf:RDF>
```

作成したRDFの検証方法

- W3Cの検証サイトに、出来たRDFを入れてみる
  - http://www.w3.org/RDF/Validator/

Number	Subject	Predicate	Object
1	http://aitc.jp/LA/rdfs/1	http://aitc.jp/LA/model#blank	genid:A12316
2	genid:A12316	http://aitc.jp/LA/model#タイトル	"2月スキー旅行"
3	genid:A12316	http://aitc.jp/LA/model#blank	genid:A12317
4	genid:A12317	http://aitc.jp/LA/model/ski#宿泊先	"ペンション・アルプ"
5	genid:A12317	http://aitc.jp/LA/model/ski#住所	"白馬村"
6	genid:A12317	http://aitc.jp/LA/model/ski#サービス	"送迎あり"
7	genid:A12316	http://aitc.jp/LA/model#blank	genid:A12318
8	genid:A12318	http://aitc.jp/LA/sns#twitter	"第一駐車場はいっぱい"
9	genid:A12318	http://aitc.jp/LA/sns#twitter	"今から出発"

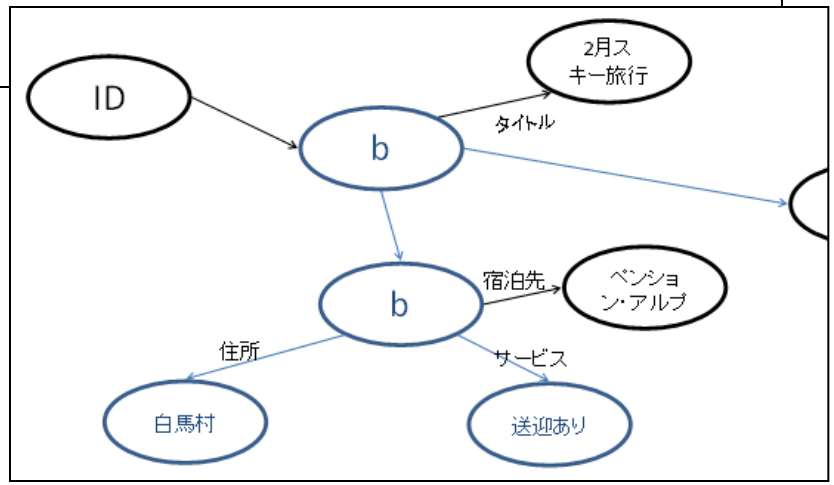


SPARQLを使った検索

- SPARQL
  - RDFのクエリ言語
  - 2008年にW3C勧告となった
  - http://www.w3.org/TR/rdf-sparql-query/
  - http://www.asahi-net.or.jp/~ax2s-kmtn/internet/rdf/rdf-sparql-query.html
  - SELECT * WHERE { ?s <http://aitc.jp/LA/model#タイトル> ?o }
  - SELECT * WHERE {?s ?p ?o . FILTER (regex(?o, “白馬"))}
  - SELECT ?o (count(?s) as ?z)
    WHERE {?s <http://aitc.jp/LA/model/ski#宿泊先> ?o} GROUP BY ?o

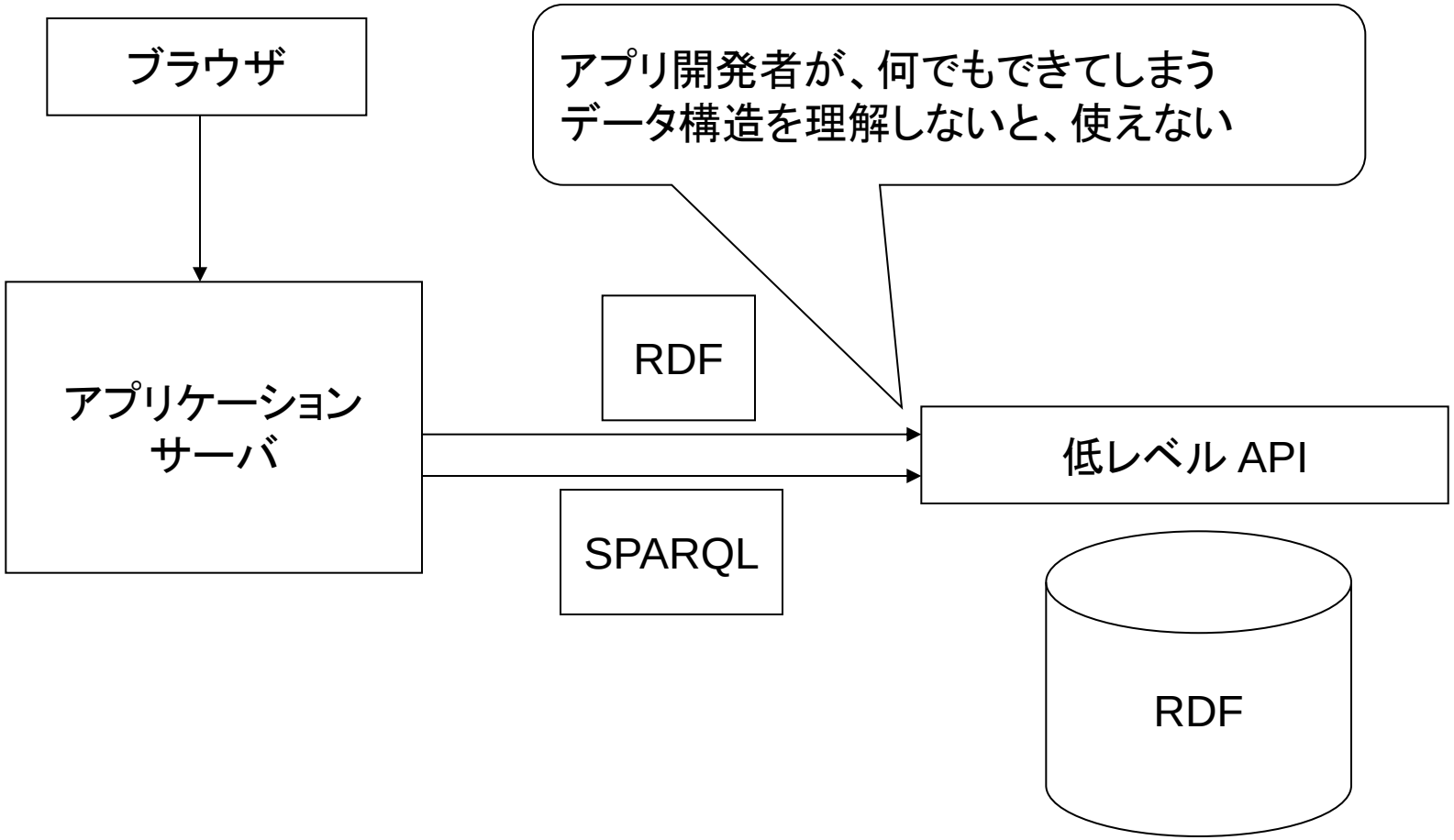
SPARQLを使った検索

```
SELECT ?value WHERE {
  <http://aitc.jp/LA/rdfs/1>  <http://aitc.jp/LA/model#blank>  ?x  .
  ?x  <http://aitc.jp/LA/model#タイトル> ?value  .
}

SELECT ?value WHERE {
  ?x  <http://aitc.jp/LA/model/ski#住所>  "白馬村"  .
  ?x  <http://aitc.jp/LA/model/ski#宿泊先>  ?value  .
}

SELECT ?value WHERE {
  ?x  <http://aitc.jp/LA/model/ski#住所>  "白馬村"  .
  ?y  <http://aitc.jp/LA/model#blank>  ?x  .
  ?y  <http://aitc.jp/LA/model#タイトル>  ?value  .
}
```

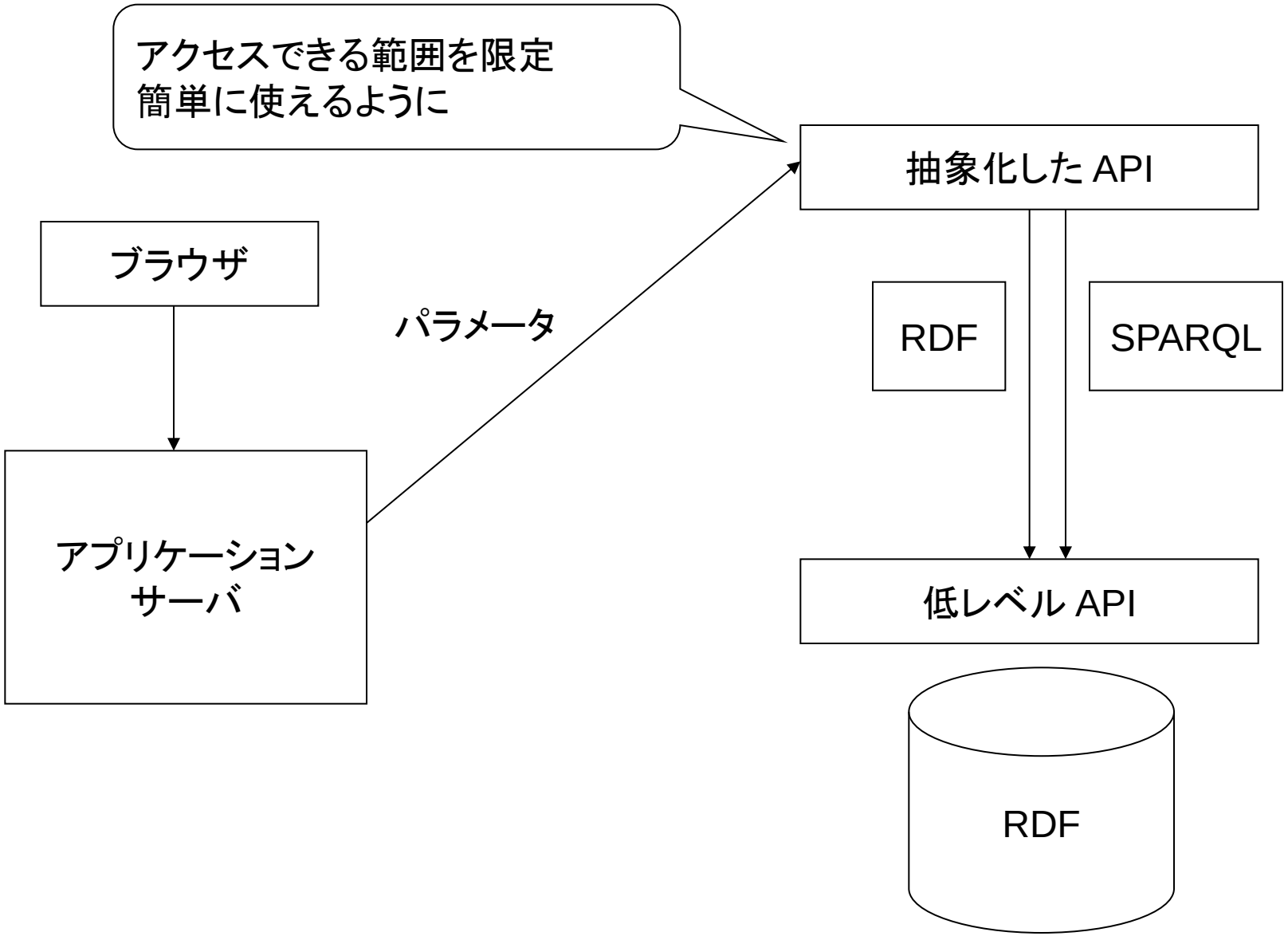


システム構成

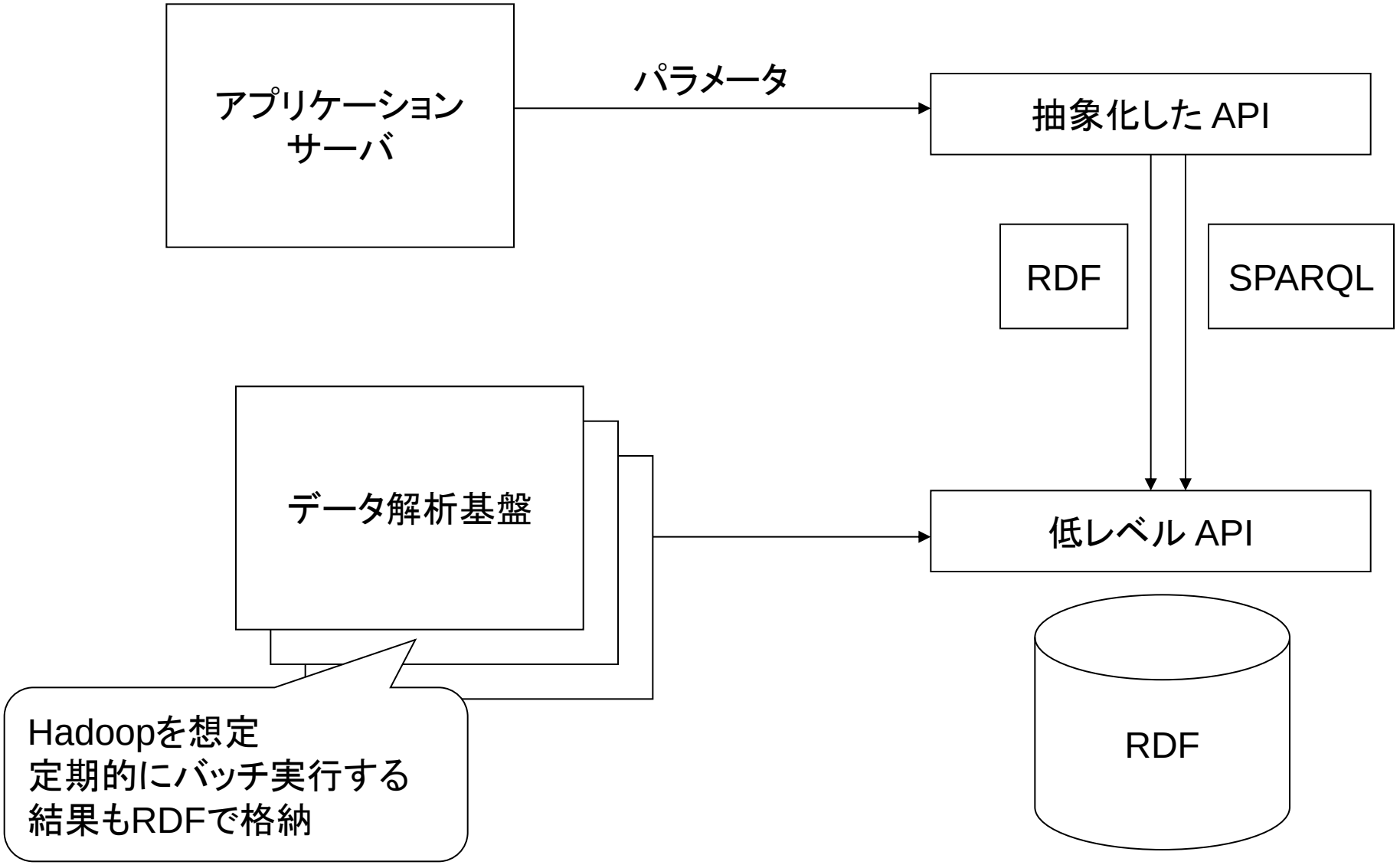
システム構成－初期



システム構成ー現在



システム構成－今後の予定



バックエンドのまとめ

- RDFは、テーブルレス＆スキーマレス
  - 小さな粒度で自由に格納できる
- RDFでの名前空間の使い分け
  - 同じ値でも、名前空間が違えば別物
- 性能はそれなりに出る
  - 各要素にインデックス
- SPARQLは書いてみると意外に簡単
  - RDFを図化したものを参照しつつ