2015 年度

2016 年 2 月 26 日

多階層オミクス情報 RDF を利用した SPARQL 検索の高速化ならびに Stanza の開発

機能設計書

田中 聡
Trans-IT

目次

1. はじめに

本文書は 2015 年度「多階層オミクス情報 RDF を利用した SPARQL 検索の高速化ならびに Stanza の開発」の機能設計について記述したものである。

2. 概要

本開発では大きく分けて次の 4 つから構成される。

(a) 多階層オミクスデータの RDF 化プログラムの開発
(b) リレーショナル・データベース(RDB) に蓄積されている
　　多階層オミクスデータの SPARQL 検索を可能にするための
　　マッピングファイルの作成
(c) ゲノム座標情報を簡便に検索するための SPARQL ライブラリの開発
(d) SPARQL 検索結果を表示する為の Stanza の開発

3. 機能設計

本章ではそれぞれの機能設計の詳細を記述する。
今回作成する機能は RDF (Resource Description Framework) を扱うが、その際、前提となる構造を下記の図で示す。

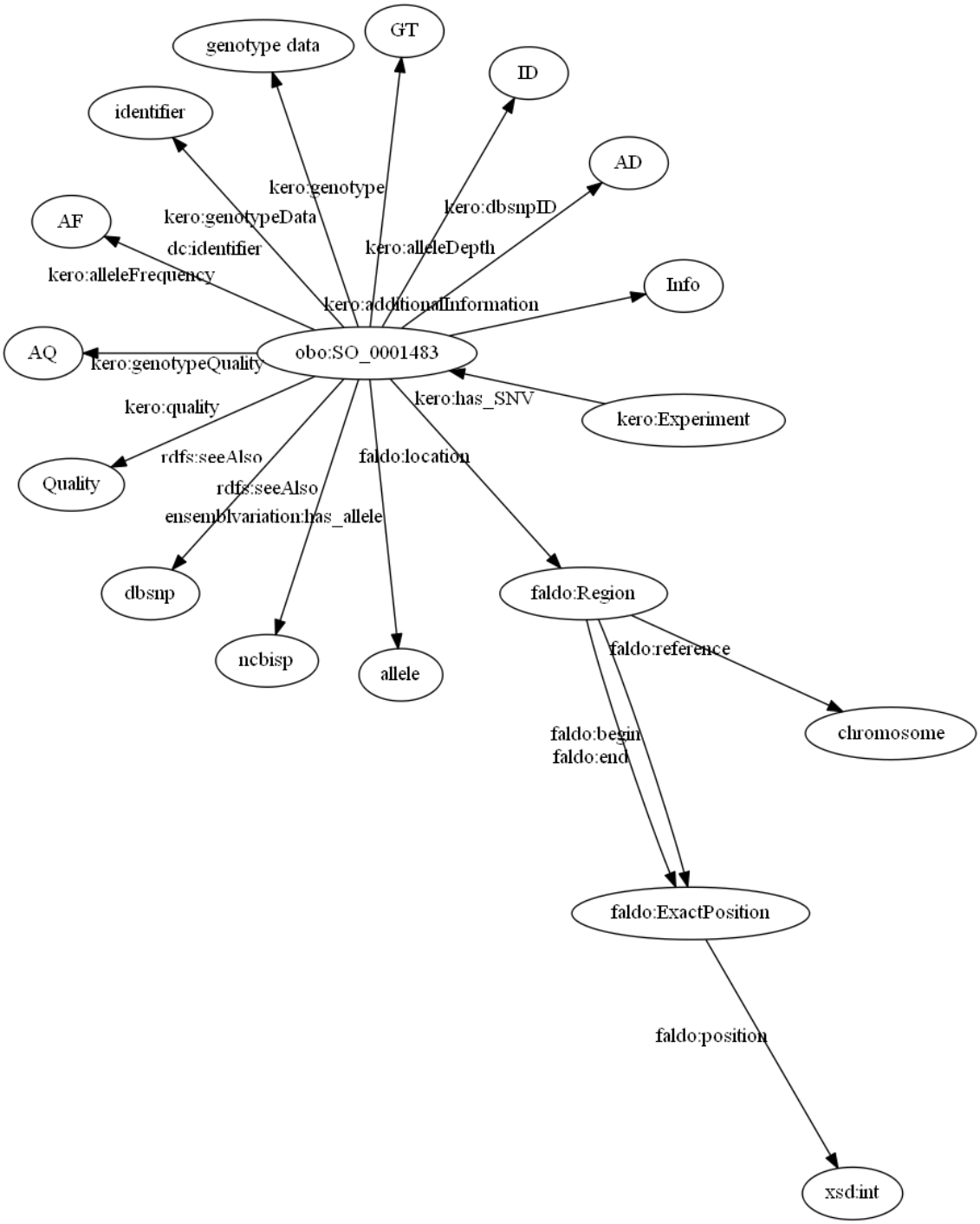

図 1 RDF スキーマ

3.1. 多階層オミクスデータの RDF 化プログラムの開発

今回、VCF ファイルを RDF データに変換するプログラムを作成する。
その際、前提となるスキーマは図 1 で示したものになるが、しかし、これまでの経緯を見るとスキーマは若干の変化をしている。 例えば上記の has_SNV はプログラムやデータに

は hasSNV（アンダースコアなし）と書かれているものもある。

そういった若干の変化にも簡単に対応できる様に、テンプレートファイルを用意し、そのテンプレートに沿って変換処理を行なう。

入力

VCF ファイル（変換元データ、必須）

Template ファイル（変換情報、任意）

出力

RDF ファイル（TTL 形式）

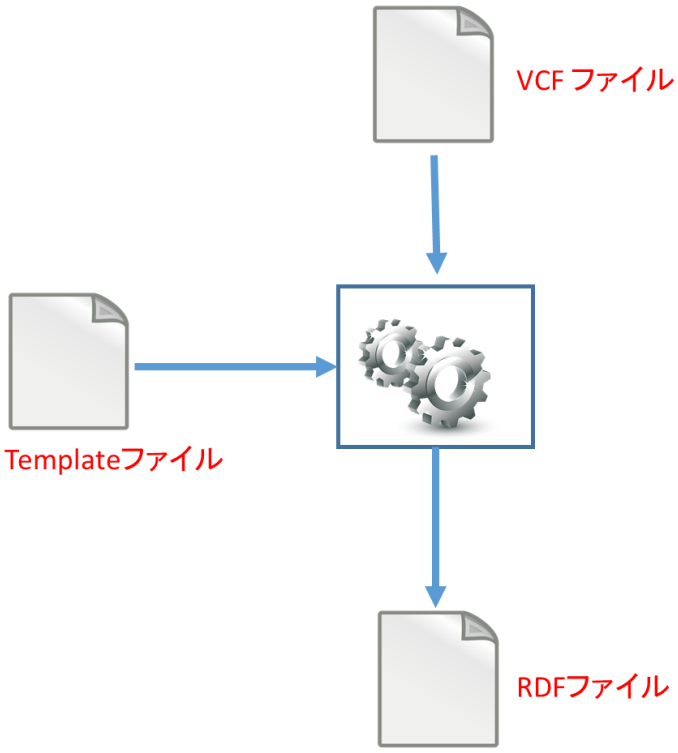

図 2　VCF 変換ツールの入出力

表 1　VCF ファイル構造

Chrom	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	s_18
c1.fa	10234	rs145599635	C	T	72.66	.	AC=1;AF=0.50;AN=2;BaseQRankSum=-0.777;DB;DP=53;Dels=0.02;FS=1.862;HRun=1;HaplotypeScore=43.1232;MQ=26.10;MQ0=1;MQRankSum=3.530;QD=1.37;ReadPosRankSum=2.915;SB=-42.24	GT:AD:DP:GQ:PL	0/1:42,10:52:61.94:103,0,62
c1.fa	10235	.	T	A	106.33	.	AC=1;AF=0.50;AN=2;BaseQRankSum=0.773;DP=52;Dels=0.02;FS=1.922;HRun=2;HaplotypeScore=39.1992;MQ=26.04;MQ0=1;MQRankSum=3.142;QD=2.04;ReadPosRankSum=3.043;SB=-42.24	GT:AD:DP:GQ:PL	0/1:40,9:51:21.93:136,0,22
c1.fa	14907	rs79585140	A	G	101.53	.	AC=1;AF=0.50;AN=2;BaseQRankSum=0.724;DB;DP=82;Dels=0.00;FS=5.251;HRun=1;HaplotypeScore=0.0000;MQ=17.43;MQ0=47;MQRankSum=2.257;QD=1.24;ReadPosRankSum=1.406;SB=-64.97	GT:AD:DP:GQ:PL	0/1:74,8:82:99:132,0,125
・・・									

プログラムはコマンドラインとし、入力となる VCF ファイルおよび Template ファイル、出力する RDF ファイル（TTL）はコマンドラインの引数で指定するものとする。

vcf2rdf –in [入力ファイル名] –out [出力ファイル名] –sample [サンプル名]
　　（ -template [テンプレートファイル名]）

コマンド入力例

vcf2rdf –in data.vcf –out data.ttl –sample TSE000086 –template template.ttl

vcf2rdf –in data.vcf –sample TSE000086 –out data.ttl

3.2. RDB に蓄積されている多階層オミクスデータの SPARQL 検索を可能にするためのマッピングファイルの作成

今回、対象となる RDB のスキーマは chromosome 毎にテーブルが分かれていて、それぞれが同じ column を持つ。この様なデータから効率的に mapping ファイルを作成する目的と前章の VCF データ変換プログラムと同様にスキーマの変化に対応する為に、この mapping ファイルもプログラムにより作成する。

入力

設定ファイル（DB 接続情報およびテーブル情報等、必須）

テンプレートファイル（変換情報、必須）

出力

mapping ファイル

今回、SPARQL ラッパーとして D2RQ および ontop に対応するが、この両者の

mapping ファイルの違いは template ファイルを変える事によって対応する。

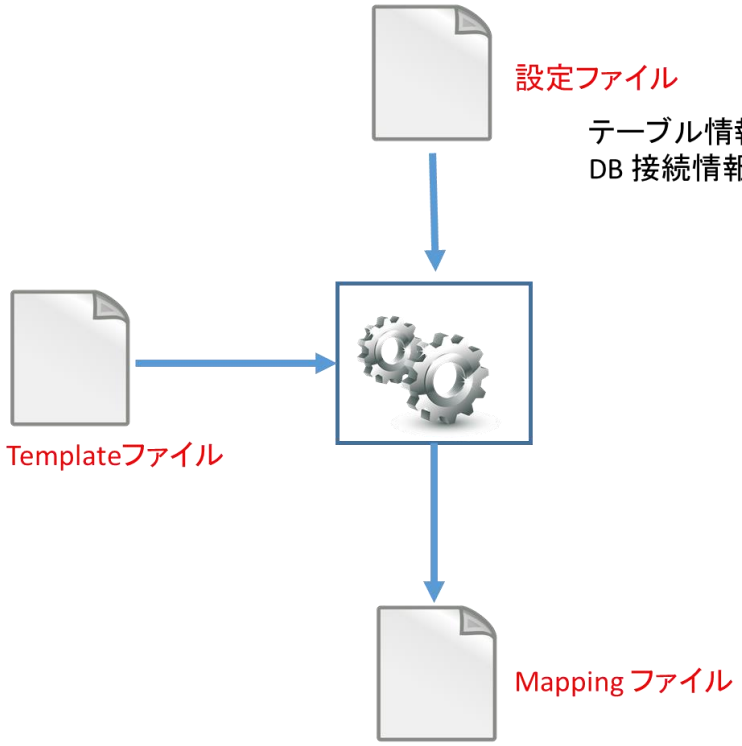

図 3　Mapping ファイル作成プログラムの入出力

今回もプログラムはコマンドラインとして実装し、入力する設定ファイル、テンプレートファイルおよび出力のマッピングファイルは引数として指定する。

create_map –props [設定ファイル] –template [Template ファイル] –out [mapping ファイル]

コマンド入力例

create_map –props db.properties –template d2rq.ttl.template –out d2rq.ttl

3.3.　ゲノム座標情報を簡便に検索するための SPARQL ライブラリの開発

今回は SPARQL ライブラリの開発を JavaScript を使って行なう。

まずは、座標情報を持つオブジェクトを下記の様に定義する。

表 2 区間情報オブジェクトのプロパティ

プロパティ名	説明	例
name	区間情報名	A549H3K27ac_peak32
label	ラベル情報	A549H3K27ac_peak32
chromosome	染色体情報	chr1
type	forwad か reverse か	forward
begin	区間開始位置	1240248
end	区間終了位置	1241893

また、ゲノム座標を計算する機能として、以下の関数を提供する。

表 3 ゲノム座標計算機能の関数

関数名	説明	引数	戻り値
setEndpoint	エンドポイントをセットする。	endpoint: エンドポイント	なし
exec	SPARQL 文を実行する。	sparql: SPARQL 文字列	実行結果オブジェクト (jquery から提供されているもの)
executeQuery	SPARQL 文を実行し、オブジェクト配列を取得する。	sparql: SPARQL 文字列	結果オブジェクト配列（オブジェクトが持つプロパティは sparql 文に依存する。)
getRegion	区間情報を取得する。	name: 区間名	区間オブジェクト
getInclusionRelation	2 つの区間オブジェクトの関係を取得する。	name1: 区間名 1 name2: 区間名 2	2 つの区間の関係を示す数値
isOverlapping	指定した2つの区間が重なっているかを判定する。	name1: 区間名 1 name2: 区間名 2	重なっていれば true, そうでなければ false
isIncluding	指定した2つの区間が包	name1: 区間名 1	包含関係にあれ

	含関係にあるかを判定する。	name2: 区間名 2	ば true, そうでなければ false
getRegionsInRange	指定した区間に存在する区間情報を取得する。	chrom: 対象染色体 start: 指定区間開始位置 end: 指定区間終了位置 including: 全て指定区間に収まっている必要があれば true	区間オブジェクト配列
getUpstreamRegions	指定した区間の上流に位置する区間情報を取得する。	region: 検索基準区間名 length: 検索範囲の長さ including: 全て指定区間に収まっている必要があれば true	区間オブジェクト配列

各々の関数の呼び出し関係を図にすると以下の様になる。

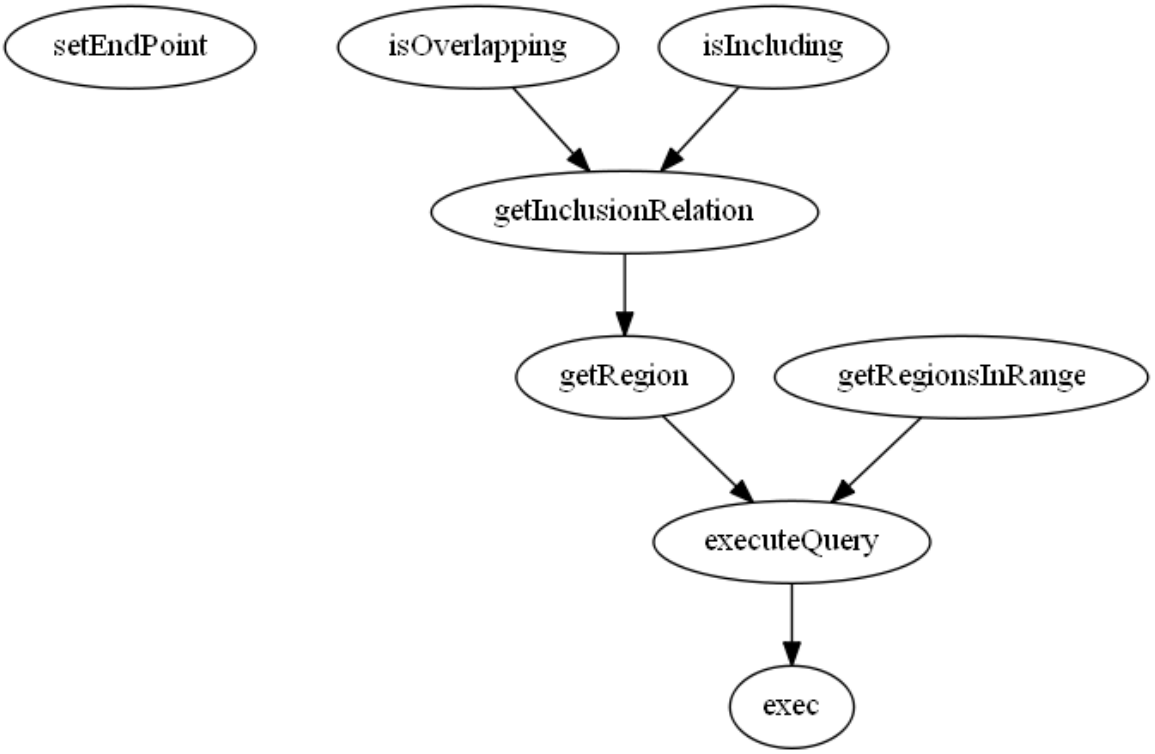

図 4 区間計算関数、呼び出し関係

また、これらの関数は SPARQL 全般を扱った sparql.js と 区間計算に特化した faldo.js に分けて、sparql 機能は区間計算以外でも使いやすくする。

sparql.js

- setEndpoint
- exec
- executeQuery

faldo.js

- getRegion
- getInclusionRelation
- isOverlapping
- isIncluding
- getRegionsInRange

3.4. SPARQL 検索結果を表示する為の Stanza の開発

今回 DBCLS（ライフサイエンス統合データベースセンター）が開発している TogoStanza

を用いて以下の機能を開発する。

- Allele 表示
- Beacon
- SNV リスト

最初の Allele 表示はユーザーが指定した位置の Allele を表示する。

入力

sample: サンプル情報（例 LC2/ad）
chromosome: 検索対象、染色体（例 1, 2, X, Y）
position: 位置（例 10234）

出力

Reference Allele（例 A, T, G, C）
Alternative Allele（例 A, T, G, C）

次の Beacon はユーザーが指定した位置に指定したアレルが存在するか否かを判定する。

入力

sample: サンプル情報（例 LC2/ad）
chromosome: 検索対象、染色体（例 1, 2, X, Y）
position: 位置（例 10234）
allele: アレル（例 A, T, G, C）

出力

指定アレルが Alternative Allele として存在していれば “Existing”
そうでなければ “Not Existing”

最後の SNV リストはユーザーが指定した区間に存在する Allele の一覧を表示する。

入力

sample: サンプル情報（例 LC2/ad）
chromosome: 検索対象、染色体（例 1, 2, X, Y）
range_start: 指定区間開始位置（例 10000）
range_end: 指定区間終了位置（例 11000）

出力

区間内に存在するアレル数、および全てのアレルの
位置、Reference Allele, Alternative Allele