

WEB INFORMATION SYSTEMS

SPARQL

Stijn Vansummeren

April 23, 2015

WHAT HAVE WE GAINED?

Current – no structure

In modern [molecular biology](#), the **genome** is the entirety of an organism's [hereditary](#) information. It is encoded either in [DNA](#) or, for [many types of virus](#), in [RNA](#).

The genome includes both the [genes](#) and the [non-coding sequences](#) of the DNA.^[1] The term was adapted in 1920 by [Hans Winkler](#), Professor of [Botany](#) at the [University of Hamburg, Germany](#). The Oxford English Dictionary suggests the name to be a [portmanteau](#) of the words **gene** and **chromosome**. A few related *-ome* words already existed, such as [biome](#) and [rhizome](#), forming a vocabulary into which *genome* fits systematically.^[2]

Future – structured by RDF (subject, predicate, object)

b:genome	b:field	b:molecular-bio	.
b:DNA	b:encode	b:genes	.
b:DNA	b:encode	b:non-coding-seq	.
b:genome	b:include	b:non-coding-seq	.
b:genome	b:include	b:gene	.
b:genome	b:related-to	b:rhizome	.

- RDF is meant to assert knowledge ([statements](#)) about [entities](#) ([resources](#))
- By convention is clear what the [subject](#), [predicate](#), and [object](#) are
- With extra knowledge more inferences can be made
(e.g.: John Doe is a person)

WHAT HAVE WE GAINED?

Current – no structure

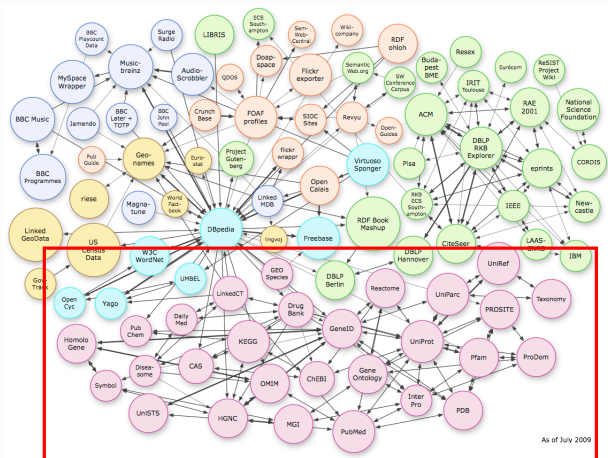
```
<lecturer name="John Doe">
  <teaches>XML Technologies</teaches>
</lecturer>
```

Future – structured by RDF (subject, predicate, object)

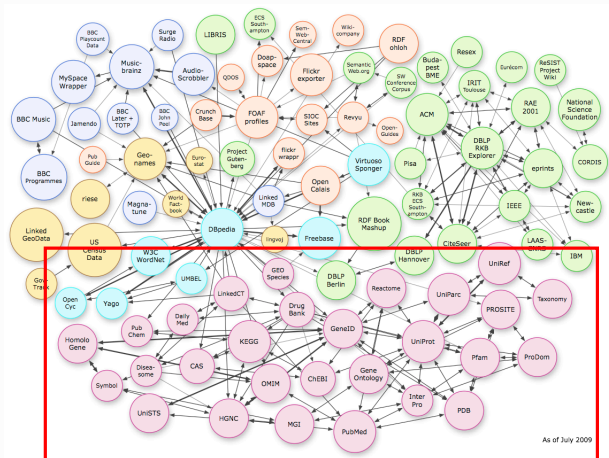
uni:john-doe	a	terms:lecturer .
uni:john-doe	terms:teaches	crs:xml .
crs:xml	a	terms:course .

- RDF is meant to assert knowledge (**statements**) about **entities** (**resources**)
- By convention is clear what the **subject**, **predicate**, and **object** are
- With extra knowledge more inferences can be made
(e.g.: John Doe is a person)

APPLICATION: E-SCIENCE



APPLICATION: E-SCIENCE



- **Query:** What proteins are absent in diabetes patients?
- **Required:** A means to query RDF

SPARQL

- To support **structured queries** like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language
- SPARQL is essentially the SQL for RDF

SPARQL

- To support **structured queries** like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language
- SPARQL is essentially the SQL for RDF

Example

```
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {

}
```

SPARQL

- To support **structured queries** like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language
- SPARQL is essentially the SQL for RDF

Example

```
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
    ?disease      bio:scientific_name      "diabetes mellitus".

}
}
```

SPARQL

- To support **structured queries** like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language
- SPARQL is essentially the SQL for RDF

Example

```
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
    ?disease      bio:scientific_name      "diabetes mellitus".

}
_____
subject
```

SPARQL

- To support **structured queries** like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language
- SPARQL is essentially the SQL for RDF

Example

```
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
    ?disease      bio:scientific_name      "diabetes mellitus".

}
_____
subject          predicate
```

SPARQL

- To support **structured queries** like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language
- SPARQL is essentially the SQL for RDF

Example

```
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
    ?disease      bio:scientific_name      "diabetes mellitus".
}

```

_____ _____ _____
subject predicate object

SPARQL

- To support **structured queries** like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language
- SPARQL is essentially the SQL for RDF

Example

```
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
    ?disease      bio:scientific_name      "diabetes mellitus".
    ?disease      bio:symptom_lack_of      ?protein .
    ?protein      rdf:type                  bio:protein .
}

```

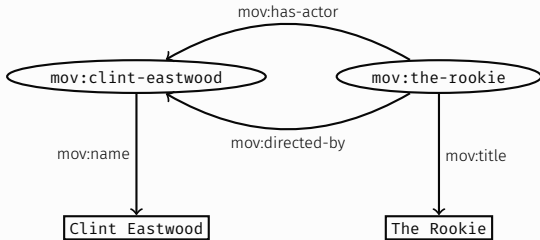
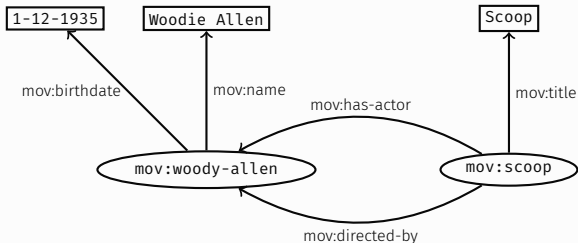
<u>subject</u>	<u>predicate</u>	<u>object</u>
----------------	------------------	---------------

- PREFIX directives can be used to abbreviate URIs
- The basic syntax is a **SELECT** - **WHERE** clause
- A **FROM** clause is optional
- The where clause consists of **graph patterns**: RDF triples with variables
- Variables are denoted by `?var` with `var` a variable name

Example: directors who have acted in their own movies

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie
WHERE {
    ?movie      mov:directed-by    ?director .
    ?movie      mov:has-actor      ?director .
}
```

Example Input:



- PREFIX directives can be used to abbreviate URIs
- The basic syntax is a **SELECT** - **WHERE** clause
- A **FROM** clause is optional
- The where clause consists of **graph patterns**: RDF triples with variables
- Variables are denoted by **?var** with **var** a variable name

Example: directors who have acted in their own movies

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie
WHERE {
    ?movie      mov:directed-by    ?director .
    ?movie      mov:has-actor      ?director .
}
```

- PREFIX directives can be used to abbreviate URIs
- The basic syntax is a **SELECT** - **WHERE** clause
- A **FROM** clause is optional
- The where clause consists of **graph patterns**: RDF triples with variables
- Variables are denoted by **?var** with **var** a variable name

Example: directors who have acted in their own movies

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie
WHERE {
    ?movie      mov:directed-by    ?director .
    ?movie      mov:has-actor      ?director .
}
```

?director	?movie
mov:woody-allen	mov:scoop
mov:clint-eastwood	mov:the-rookie

- PREFIX directives can be used to abbreviate URIs
- The basic syntax is a **SELECT** - **WHERE** clause
- A **FROM** clause is optional
- The where clause consists of **graph patterns**: RDF triples with variables
- Variables are denoted by **?var** with **var** a variable name

Example: directors who have acted in their own movies

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie
FROM <http://example.org/mdb.ttl>
WHERE {
    ?movie      mov:directed-by    ?director .
    ?movie      mov:has-actor      ?director .
}
```

?director	?movie
mov:woody-allen	mov:scoop
mov:clint-eastwood	mov:the-rookie

SPARQL: OPTIONAL

- **OPTIONAL** clauses allow you to use information in the RDF graph if it is present, but does not eliminate solutions if it is missing

Example: director must have a birthdate

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie ?bd
WHERE {
    ?movie      mov:directed-by    ?director .
    ?movie      mov:has-actor      ?director .
    ?director   mov:birthdate      ?bd .
}
```

?director	?movie	?bd
mov:woody-allen	mov:scoop	"1-12-1935"

SPARQL: OPTIONAL

- **OPTIONAL** clauses allow you to use information in the RDF graph if it is present, but does not eliminate solutions if it is missing

Example: return birthdate if available

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie ?bd
WHERE {
    ?movie      mov:directed-by    ?director .
    ?movie      mov:has-actor      ?director .
    OPTIONAL {?director mov:birthdate ?bd . }
}
```

?director	?movie	?bd
mov:woody-allen	mov:scoop	"1-12-1935"
mov:clint-eastwood	mov:the-rookie	

SPARQL: FILTER

- **FILTER** clauses allow you to specify additional constraints on candidate solutions
- These constraints can use a small set of operators from XPath 2.0
- the operator **bound** can be used to test if a variable is bound or not
- the operator **regex** can be used to test if a variable matches a regular expression
- the operators **<**, **>**, **<=**, **>=**, **=** can be used to compare

Example: directors that do not have a birthdate

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director
WHERE {
    ?movie      mov:directed-by    ?director .
    OPTIONAL {?director mov:birthdate ?bd . }
    FILTER (!bound(?bd))
}
```

?director

mov:clint-eastwood

SPARQL: FILTER

- **FILTER** clauses allow you to specify additional constraints on candidate solutions
- These constraints can use a small set of operators from XPath 2.0
- the operator **bound** can be used to test if a variable is bound or not
- the operator **regex** can be used to test if a variable matches a regular expression
- the operators **<**, **>**, **<=**, **>=**, **=**, **!=** can be used to compare

Example: directors who's name contains "allen" as a substring

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director
WHERE {
    ?movie      mov:directed-by    ?director .
    ?director  mov:name           ?name .
    FILTER regex(?name, "allen", "i")
}
```

?director

mov:woody-allen

SPARQL: FILTER

- **FILTER** clauses allow you to specify additional constraints on candidate solutions
- These constraints can use a small set of operators from XPath 2.0
- the operator **bound** can be used to test if a variable is bound or not
- the operator **regex** can be used to test if a variable matches a regular expression
- the operators **<**, **>**, **<=**, **>=**, **=**, **!=** can be used to compare

Example: movies that have rating at least 3

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?movie
WHERE {
    ?movie      mov:rating      ?x .
    FILTER (?x >= 3)
}
```

SPARQL: FILTER NOT EXISTS

- As of SPARQL 1.1 there is an easier way to find data that does not meet certain data using **FILTER NOT EXISTS**.
- **FILTER NOT EXISTS** takes as argument another graph pattern that should not be found in the data.

Example: directors that do not have a birthdate

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director
WHERE {
    ?movie      mov:directed-by    ?director .
    FILTER NOT EXISTS { ?director mov:birthdate ?bd . }
}
```

?director

mov:clint-eastwood

SPARQL: UNION

- Using the **UNION** keyword, we can let patterns be evaluated independently

Example: Directors who's name contains "allen" as a substring or do not have a birthdate

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director
WHERE {
  {
    ?movie      mov:directed-by      ?director .
    ?director   mov:name              ?name .
    FILTER regex(?name, "allen", "i")
  }
  UNION
  {
    ?movie      mov:directed-by      ?director .
    OPTIONAL {?director mov:birthdate ?bd . }
    FILTER (!bound(?bd))
  }
}
```

SPARQL: CONSTRUCT

- SELECT queries returns tables listing variable bindings
- CONSTRUCT queries construct a new RDF graph

Example: assign all courses to John Doe

```
PREFIX terms: <http://ulb.be/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CONSTRUCT
{
  <http://ulb.ac.be/staff/jdoe> terms:teaches ?x
}
WHERE { ?x rdf:type terms:course }
```

- SELECT queries returns tables listing variable bindings
- CONSTRUCT queries construct a new RDF graph

Example: assign all courses to John Doe and give them 5 credits

```
PREFIX terms: <http://ulb.be/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CONSTRUCT
{
  <http://ulb.ac.be/staff/jdoe> terms:teaches ?x .
  ?x terms:credits "5"
}
WHERE { ?x rdf:type terms:course }
```

- SELECT queries returns tables listing variable bindings
- CONSTRUCT queries construct a new RDF graph
- ASK queries check whether the graph pattern occurs in the RDF graph (returns a boolean)

Example: is there some resource with both a birthdate and a name?

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
ASK
{
  ?x mov:name ?n .
  ?x mov:birthdate ?d
}
```

- The **FROM** clause states that the query should be answered on a particular dataset (argument to the FROM clause)
- The dataset is downloaded (if necessary) and then the query answer is computed.

Example: query Tim-Berners-Lee's FOAF file?

```
PREFIX dc: <http://purl.org/dc/elements/1.1>
SELECT ?title
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
    ?s dc:title ?title .
}
```


SPARQL: QUERYING REMOTE SPARQL SERVICES

- The **SERVICE** keyword ships (a part of) a query to a remote SPARQL endpoint, and retrieves the results.
- (New in SPARQL 1.1.)

Example: Ship a subquery to DBpedia?

```
SELECT ?p ?o
WHERE {
  SERVICE <http://dbpedia.org/sparql>
  { SELECT ?p ?o
    WHERE { <http://dbpedia.org/resource/Brussels> ?p ?o. }
  }
}
```

- The **ORDER BY** modifier allows to sort the results (by default ascendingly, but one can specify descending also).
- The **LIMIT** *N* modifies retrieves only the first *N* results.

Example: sort results by price (ascending) and, for items with the same price, by name (descending)

```
PREFIX terms: <http://ulb.be/terms>
SELECT ?name ?price
WHERE {
    ?prod rdf:type terms:product .
    ?prod terms:name ?name .
    ?prod terms:price ?price .
}
ORDER BY ?price DESC(?name)
```

- The **ORDER BY** modifier allows to sort the results (by default ascendingly, but one can specify descending also).
- The **LIMIT** *N* modifies retrieves only the first *N* results.

Example: retrieve the product with the highest price

```
PREFIX terms: <http://ulb.be/terms>
SELECT ?name ?price
WHERE {
    ?prod rdf:type terms:product .
    ?prod terms:name ?name .
    ?prod terms:price ?price .
}
ORDER BY ?price
LIMIT 1
```

- SPARQL 1.1. allows aggregate operators (**MAX**, **MIN**, **AVG**, ...) in the **SELECT** clause to compute aggregate values.

Example: compute the average price

```
PREFIX terms: <http://ulb.be/terms>
SELECT AVG(?price) as ?avgPrice
WHERE {
    ?prod rdf:type terms:product .
    ?prod terms:name ?name .
    ?prod terms:price ?price .
}
```

REFERENCES

- P. Hitzler, M. Krötzsch, S. Rudolph. *Foundations of Semantic Web technologies*. Chapter 7 (7.1.1 – 7.1.8).
- B. DuCharme. *Learning SPARQL*.