MINES Saint-Étienne

UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Query Language for RDF (SPARQL 1.1)

SPARQL

- The syntax looks similar to SQL
- The features are similar to SQL
- A *family* of standards:
  - **SELECT** queries
  - **Update** (INSERT / DELETE) queries
  - **Protocols**
  - **Reasoning** at query time
- Standards for **managing** RDF data in general
- SQL and SQL DBMS are to the relational data model what SPARQL and its standards are to the RDF data model

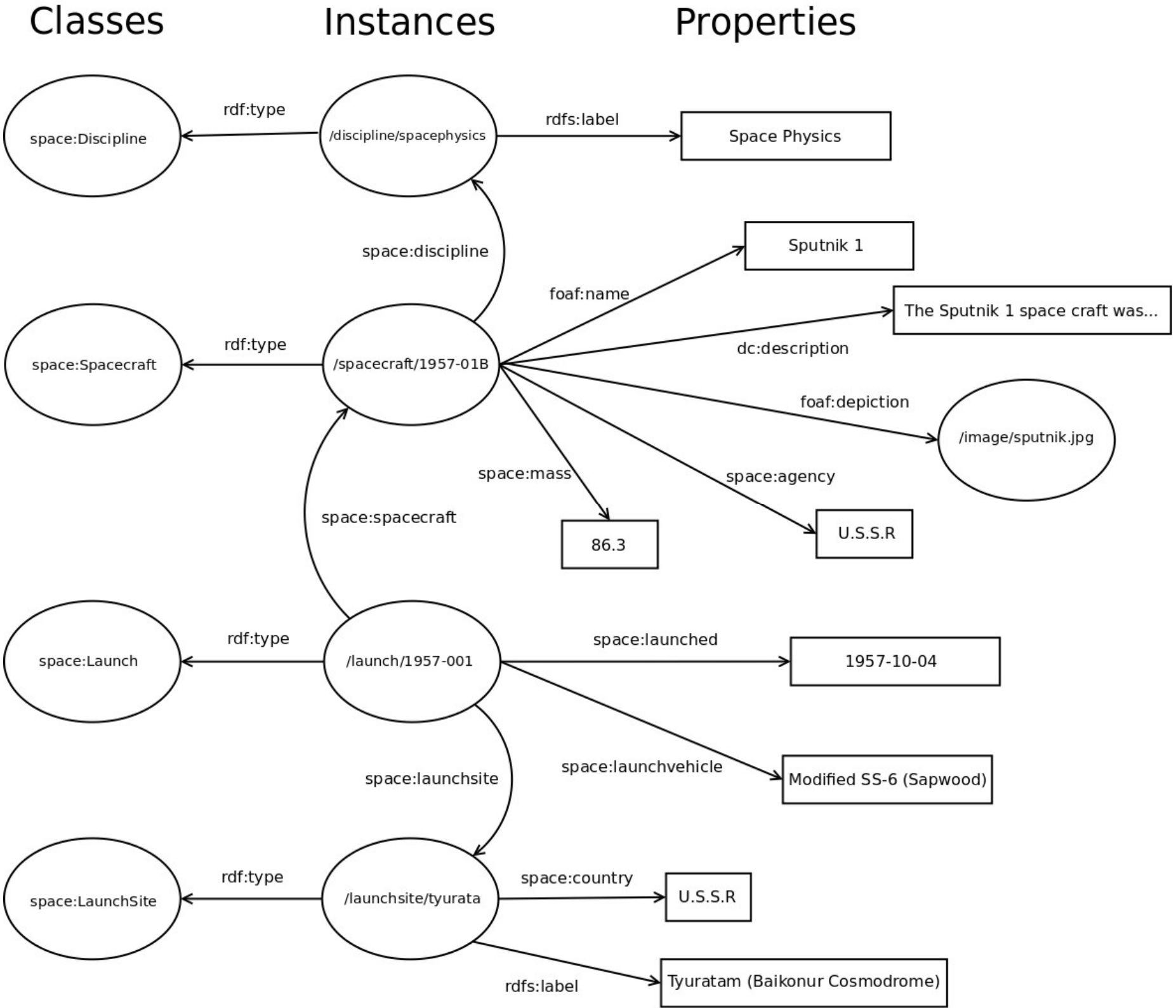
SPARQL SELECT

- **Variable**: an element of a set disjoint from IRIs, literals and blank nodes
- **Basic graph pattern**: an RDF graph where subject, predicate or object can be replaced by a **variable**
- An **answer** to a SELECT query is a mapping from variables in the query to IRIs union literals union blank nodes in the queried graph

Tutorial Schema (SPARQL 1.0)

Based on NASA spaceflight data

(the following slides are adapted from **Leigh Dodds**' tutorial)


Classes
Instances
Properties
space:Discipline
rdf:type
/discipline/spacephysics
rdfs:label
Space Physics
space:discipline
Sputnik 1
foaf:name
The Sputnik 1 space craft was...
dc:description
space:Spacecraft
rdf:type
/spacecraft/1957-01B
foaf:depiction
/image/sputnik.jpg
space:mass
space:agency
space:spacecraft
86.3
U.S.S.R
space:Launch
rdf:type
/launch/1957-001
space:launched
1957-10-04
space:launchvehicle
Modified SS-6 (Sapwood)
space:launchsite
space:LaunchSite
rdf:type
/launchsite/tyurata
space:country
U.S.S.R
rdfs:label
Tyuratam (Baikonur Cosmodrome)

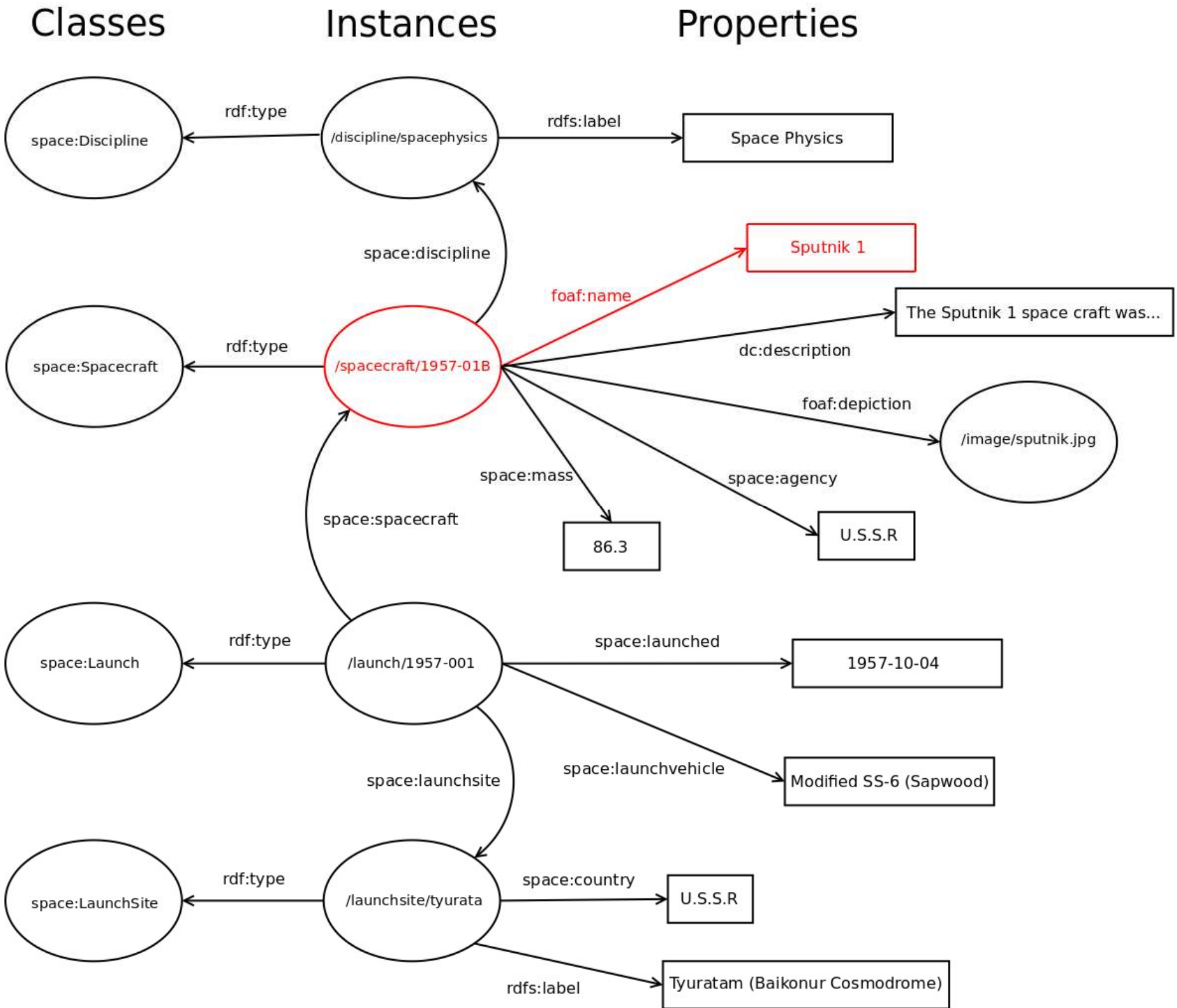
Triple and Graph Patterns

How do we describe the structure of the RDF graph which we are interested in?

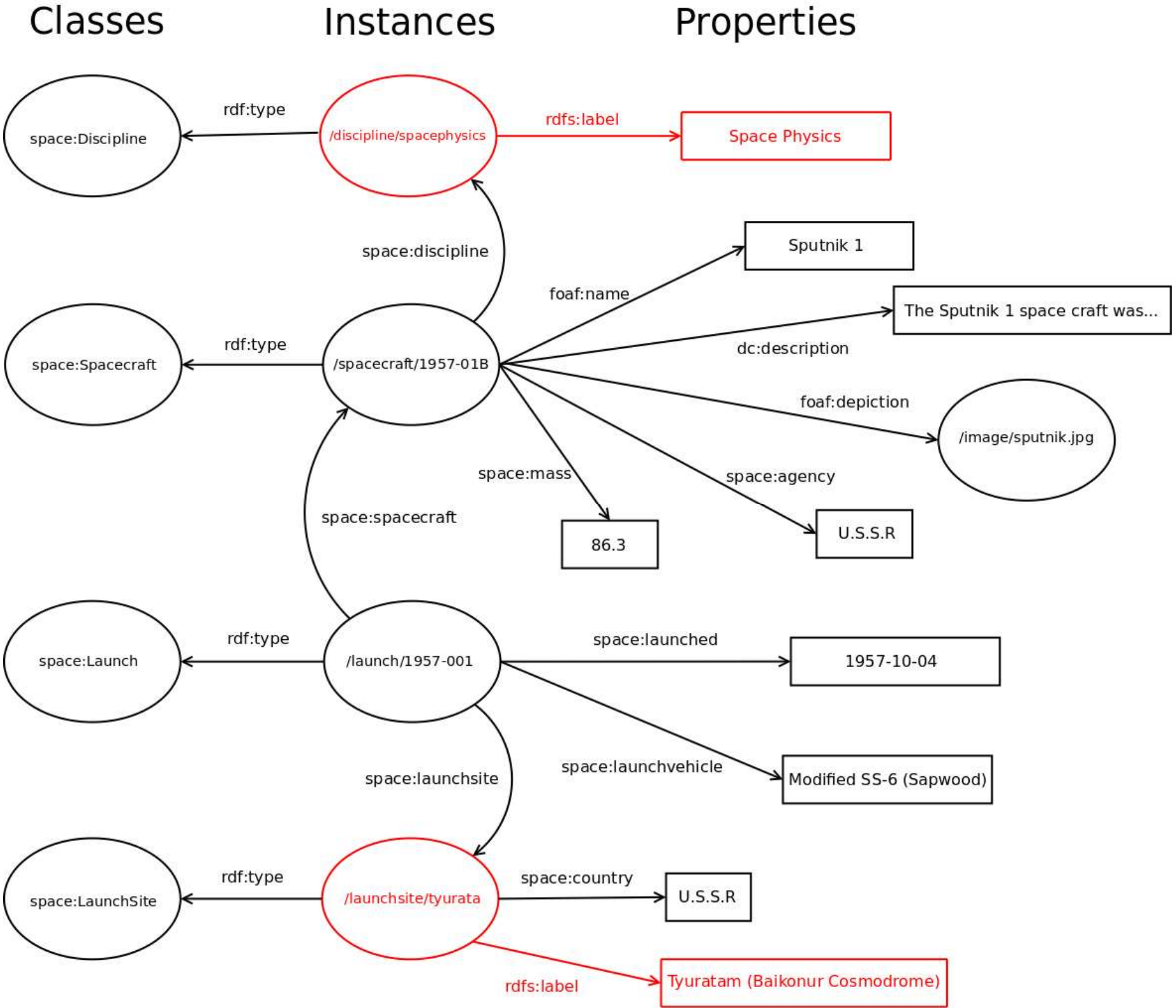
```
#An RDF triple in Turtle syntax

<http://purl.org/net/schemas/space/spacecraft/1957-001B>
foaf:name  "Sputnik 1" .
```

```
#A SPARQL triple pattern, with a single variable

<http://purl.org/net/schemas/space/spacecraft/1957-001B>
foaf:name  ?name .
```

```
#All parts of a triple pattern can be variables

?spacecraft  foaf:name  ?name .
```

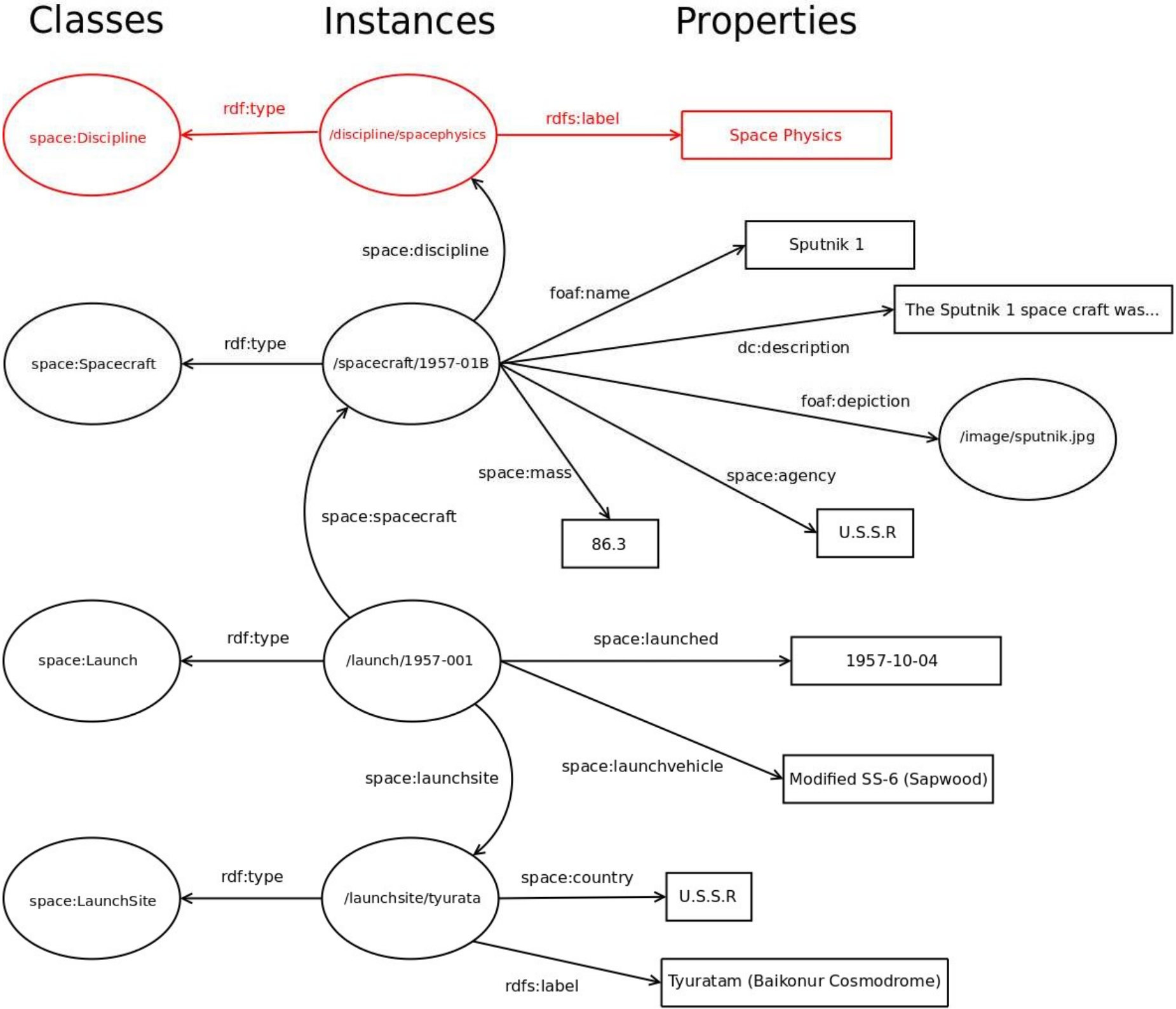

Classes
Instances
Properties
space:Discipline
rdf:type
/discipline/spacephysics
rdfs:label
Space Physics
space:discipline
Sputnik 1
foaf:name
The Sputnik 1 space craft was...
dc:description
space:Spacecraft
rdf:type
/spacecraft/1957-01B
foaf:depiction
/image/sputnik.jpg
space:mass
space:agency
space:spacecraft
86.3
U.S.S.R
space:Launch
rdf:type
/launch/1957-001
space:launched
1957-10-04
space:launchvehicle
Modified SS-6 (Sapwood)
space:launchsite
space:LaunchSite
rdf:type
/launchsite/tyurata
space:country
U.S.S.R
rdfs:label
Tyuratam (Baikonur Cosmodrome)

```
#Matching labels of resources

?subject  rdfs:label  ?label .
```


Classes
Instances
Properties
space:Discipline
rdf:type
/discipline/spacephysics
rdfs:label
Space Physics
space:discipline
Sputnik 1
foaf:name
The Sputnik 1 space craft was...
dc:description
space:Spacecraft
rdf:type
/spacecraft/1957-01B
foaf:depiction
/image/sputnik.jpg
space:mass
space:agency
space:spacecraft
86.3
U.S.S.R
space:Launch
rdf:type
/launch/1957-001
space:launched
1957-10-04
space:launchvehicle
Modified SS-6 (Sapwood)
space:launchsite
space:LaunchSite
rdf:type
/launchsite/tyurata
space:country
U.S.S.R
rdfs:label
Tyuratam (Baikonur Cosmodrome)

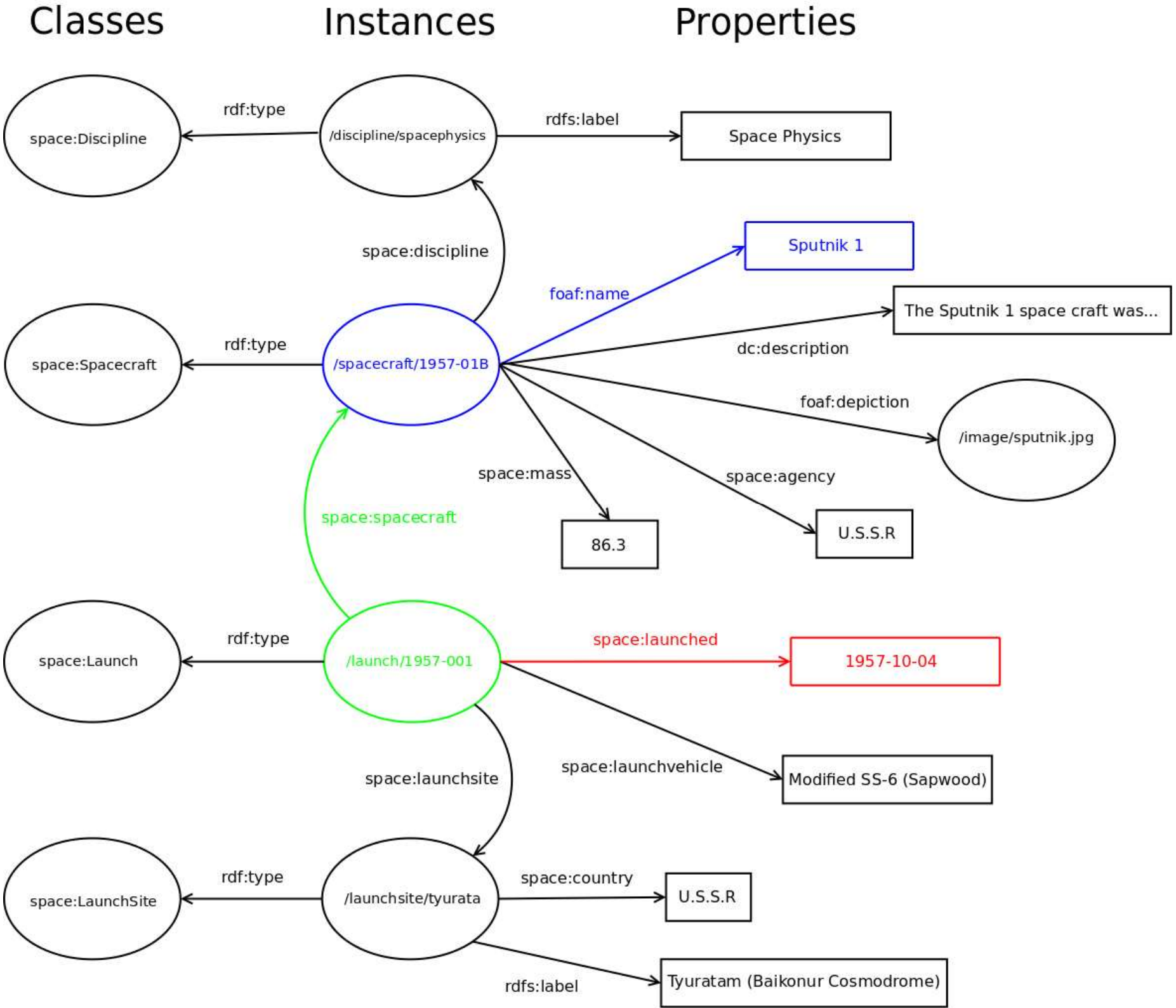
```
#Combine triples patterns to create a graph pattern

?subject  rdfs:label  ?label .
?subject  rdf:type  space:Discipline .
```

```
#The SPARQL syntax is based on Turtle,
#which allows abbreviations
#e.g. predicate-object lists:

?subject  rdfs:label  ?label;
          rdf:type  space:Discipline .
```


Classes
Instances
Properties
space:Discipline
rdf:type
/discipline/spacephysics
rdfs:label
Space Physics
space:discipline
Sputnik 1
foaf:name
The Sputnik 1 space craft was...
dc:description
space:Spacecraft
rdf:type
/spacecraft/1957-01B
foaf:depiction
/image/sputnik.jpg
space:mass
space:agency
space:spacecraft
86.3
U.S.S.R
space:Launch
rdf:type
/launch/1957-001
space:launched
1957-10-04
space:launchvehicle
Modified SS-6 (Sapwood)
space:launchsite
space:LaunchSite
rdf:type
/launchsite/tyurata
space:country
U.S.S.R
rdfs:label
Tyuratam (Baikonur Cosmodrome)

```
#Graph patterns allow us to traverse a graph

?spacecraft  foaf:name  “Sputnik 1” .

?launch  space:spacecraft  ?spacecraft .

?launch  space:launched  ?launchdate .
```

```
#Graph patterns allow us to traverse a graph

?spacecraft  foaf:name  “Sputnik 1”.

?launch  space:spacecraft  ?spacecraft.

?launch  space:launched  ?launchdate.
```


Classes
Instances
Properties
space:Discipline
rdf:type
/discipline/spacephysics
rdfs:label
Space Physics
space:discipline
Sputnik 1
foaf:name
The Sputnik 1 space craft was...
dc:description
space:Spacecraft
rdf:type
/spacecraft/1957-01B
foaf:depiction
/image/sputnik.jpg
space:mass
space:agency
space:spacecraft
86.3
U.S.S.R
space:Launch
rdf:type
/launch/1957-001
space:launched
1957-10-04
space:launchvehicle
Modified SS-6 (Sapwood)
space:launchsite
space:LaunchSite
rdf:type
/launchsite/tyurata
space:country
U.S.S.R
rdfs:label
Tyuratam (Baikonur Cosmodrome)

Structure of a Query

What does a basic SPARQL query look like?

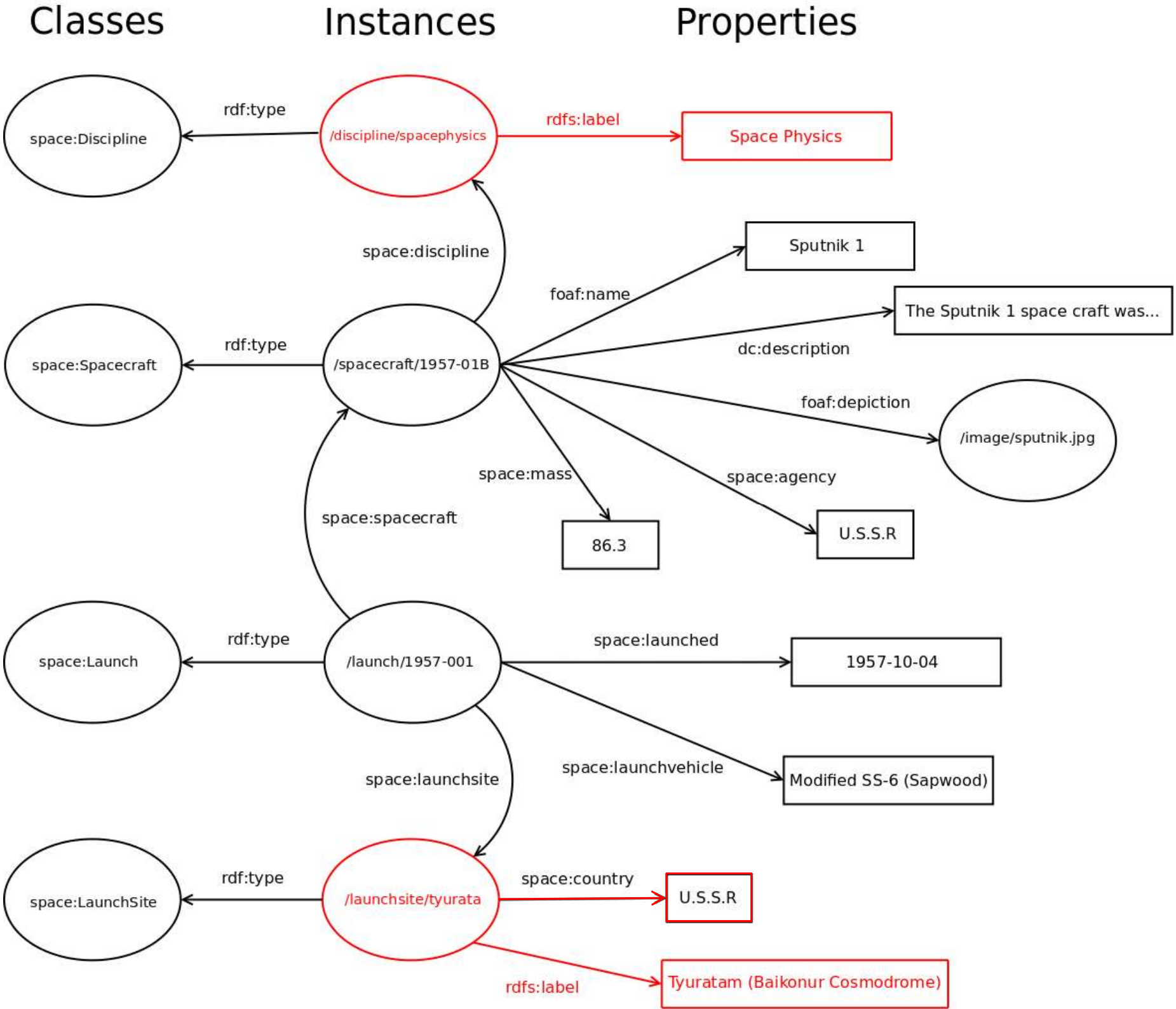
```
#Ex. 1
#Associate URIs with prefixes

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

#Example of a SELECT query, retrieving 2 variables
#Variables selected MUST be bound in graph pattern

SELECT ?subject ?label
WHERE {

  #This is our graph pattern
  ?subject  rdfs:label  ?label;
            rdf:type  space:Discipline .
}
```

```
#Ex. 2

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

#Example of a SELECT query, retrieving all variables

SELECT *
WHERE {

  ?subject  rdfs:label  ?label;
            rdf:type  space:Discipline .
}
```

OPTIONAL bindings

How do we allow for missing or unknown information?

```
#Ex. 3

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name ?country
WHERE {

  #This pattern must be bound
  ?thing  rdfs:label  ?name .

  #Anything in this block doesn't have to be bound
  OPTIONAL {
    ?thing  space:country  ?country .
  }
}
```


Classes
Instances
Properties
space:Discipline
rdf:type
/discipline/spacephysics
rdfs:label
Space Physics
space:discipline
Sputnik 1
foaf:name
The Sputnik 1 space craft was...
dc:description
space:Spacecraft
rdf:type
/spacecraft/1957-01B
foaf:depiction
/image/sputnik.jpg
space:mass
space:agency
space:spacecraft
86.3
U.S.S.R
space:Launch
rdf:type
/launch/1957-001
space:launched
1957-10-04
space:launchvehicle
Modified SS-6 (Sapwood)
space:launchsite
space:LaunchSite
rdf:type
/launchsite/tyurata
space:country
U.S.S.R
rdfs:label
Tyuratam (Baikonur Cosmodrome)

UNION queries

How do we allow for alternatives or variations in the graph?

```
#Ex. 4

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subject ?displayLabel
WHERE {

   {
     ?subject  foaf:name  ?displayLabel .
   }
   UNION
   {
     ?subject  rdfs:label  ?displayLabel .
   }
}
```

Sorting & Restrictions

How do we apply a sort order to the results?

How can we restrict the number of results returned?

```
#Ex.5

#Select the uri and the mass of all the spacecraft

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT  ?spacecraft  ?mass
WHERE {

  ?spacecraft  space:mass  ?mass .

}
```

```
#Ex. 6

#Select the uri and the mass of all the spacecraft
#with highest first

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?spacecraft ?mass
WHERE {

  ?spacecraft  space:mass  ?mass .

}
#Use an ORDER BY clause to apply a sort. Can be ASC or DESC
ORDER BY DESC(?mass)
```

```
#Ex. 7

#Select the uri and the mass of the 10 heaviest spacecraft

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?spacecraft ?mass
WHERE {

  ?spacecraft  space:mass  ?mass .

}
#Order by weight descending
ORDER BY DESC(?mass)
#Limit to first ten results
LIMIT 10
```

```
#Ex. 8

#Select the uri and the mass of the 11-20th most
#heaviest spacecraft

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?spacecraft ?mass
WHERE {

  ?spacecraft  space:mass  ?mass .

}
ORDER BY DESC(?mass)
#Limit to ten results
LIMIT 10
#Apply an offset to get next "page"
OFFSET 10
```

Filtering

How do we restrict results based on aspects of the data rather than the graph, e.g. string matching?

```
#Sample data for Sputnik launch


<http://purl.org/net/schemas/space/launch/1957-001>
   rdf:type  space:Launch;

  #Assign a datatype to the literal,
  #to indicate it is a date
  space:launched  "1957-10-04"^^xsd:date;

  space:spacecraft
<http://purl.org/net/schemas/space/spacecraft/1957-001B> .
```

```
#Ex. 9

#Select name of spacecraft launched between
#1st Jan 1969 and 1st Jan 1970

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?name
WHERE {

  ?launch space:launched ?date;
          space:spacecraft ?spacecraft .
  ?spacecraft foaf:name ?name .

  FILTER (?date > "1969-01-01"^^xsd:date &&
          ?date < "1970-01-01"^^xsd:date)
}
```

```
#Ex. 10

#Select spacecraft with a mass of less than 90kg

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?spacecraft ?name
WHERE {

   ?spacecraft  foaf:name  ?name;
                space:mass  ?mass .

   #Note that we have to cast the data to the right type
   #As it is not declared in the data
   FILTER( xsd:double(?mass) < 90.0 )
}
```

```
#Ex. 11

#Select spacecraft with a name like “ollo”

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?name
WHERE {

   ?spacecraft  foaf:name  ?name .

   FILTER( regex(?name, "ollo", "i" ) )
}
```

Built-In Filters

- Logical: `!`, `&&`, `||`
- Math: `+`, `-`, `*`, `/`
- Comparison: `=`, `!=`, `>`, `<`, ...
- SPARQL tests: `isURI`, `isBlank`, `isLiteral`, `bound`
- SPARQL accessors: `str`, `lang`, `datatype`
- Other: `sameTerm`, `langMatches`, `regex`

DISTINCT

How do we remove duplicate results?

```
#Ex. 12

#Select list of agencies associated with spacecraft

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT ?agency
WHERE {

   ?spacecraft  space:agency  ?agency .

}
```

SPARQL Query Forms

Does SPARQL do more than just SELECT data?

ASK

Test whether the graph contains some data of interest

```
#Ex. 13

#Was there a launch on 16th July 1969?

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>


ASK WHERE {
  ?launch  space:launched  "1969-07-16"^^xsd:date .
}
```

DESCRIBE

Generate an RDF description of a resource(s)

```
#Ex. 14

#Describe launch(es) that occurred on 16th July 1969

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>


DESCRIBE ?launch WHERE {
  ?launch  space:launched  "1969-07-16"^^xsd:date .
}
```

```
#Ex. 15

#Describe spacecraft launched on 16th July 1969

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>


DESCRIBE ?spacecraft WHERE {

  ?launch  space:launched  "1969-07-16"^^xsd:date .

  ?spacecraft  space:launch ?launch .

}
```

CONSTRUCT

Create a custom RDF graph based on query criteria

Can be used to transform RDF data

```
#Ex. 16

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

CONSTRUCT {
  ?spacecraft  rdfs:label  ?name;
               dbpedia:agency  ?agency;
               measure:mass  ?mass .
}
WHERE {
  ?launch  space:launched  "1969-07-16"^^xsd:date .

  ?spacecraft  space:launch  ?launch;
               foaf:name  ?name;
               space:agency  ?agency;
               space:mass  ?mass .
}
```

SELECT

SQL style result set retrieval

```
#Ex. 17

PREFIX space: <http://purl.org/net/schemas/space/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name ?agency ?mass
WHERE {

  ?launch  space:launched  "1969-07-16"^^xsd:date .

  ?spacecraft  space:launch  ?launch;
               foaf:name  ?name;
               space:agency  ?agency;
               space:mass  ?mass .
}
```

Extended Query Language Power (SPARQL 1.1)

- Aggregates
- Sub-queries
- Negation and filtering
- Property paths
- Introducing new variables
- Basic federated query
- Graph Patterns inside FILTERs

Following slides adapted from David Beckett's slides (see ref. at the end)

Aggregates

- AVG(expr)
- COUNT(*) and COUNT(expr)
- GROUP_CONCAT(expr) with optional ;separator = 'string'
- MAX(expr)
- MIN(expr)
- SAMPLE(expr)
- SUM(expr)

Aggregates (cont.)

- All are allowed with and without `DISTINCT` across the arguments.
- Grouping of results is optionally done with `GROUP BY` otherwise the entire result set is 1 group (like SQL). This may bind a variable too.
- `HAVING` executes a filter expression over the results of an aggregation (like SQL)

Sub-queries

- SPARQL 1.1 allows sub-SELECTs

```
#Ex. 18

PREFIX : <http://people.example/>
SELECT ?y ?minName
WHERE {
    :alice :knows ?y .
    {
        SELECT ?y (MIN(?name) AS ?minName)
        WHERE {
            ?y :name ?name .
        }
        GROUP BY ?y
    }
}
```

Negation and Filtering

3 new ways to negate / exclusion:

- `OPTIONAL { graph-pattern }` (1.0)
- `FILTER ... !expr` (1.0)
- `FILTER ... NOT EXISTS { graph-pattern }` (1.1)
- Aggregation using `HAVING` with either of the above (1.1)
- graph-pattern `MINUS` graph-pattern (1.1)
- (Some of these can be done with complex `UNION` and `OPTIONAL` patterns)

Property path

This changes the fundamental SPARQL matching

From:

Triple pattern matches a triple to bind variables.

To:

Triples with property paths regex-like match multiple triples to bind variables.

The essential difference is that *depending on the data*, the query engine could do a simple match or do a **lot** of searching for matches.

There is lots of new syntax to select different properties from a subject node:

```
a/b ^a a|b a* a+ a? a{m,n} a{n} a{m,} a{,n}
```

where a and b are property IRIs.

Basic Federated Queries

A graph pattern that invokes a SPARQL protocol call and remote query returning the usual result formats

Allows querying multiple SPARQL databases in one query

```
#Ex. 19

SELECT variables
WHERE {
      ?person  knows  ?x
      SERVICE <http://social-db.com/sparql/> {
            ?x  foaf:name  ?name;
                ex:birthdate  ?b .
      }
}
```

More

- More functions and operators
- Introducing new variables
- RDF graph database management:
  - INSERT triples / graphs
  - DELETED triples / graphs

Useful Links

- SPARQL FAQ
  - http://www.thefigtrees.net/lee/sw/sparql-faq
- Learn about SPARQL 1.1
  - http://www.dajobe.org/talks/201105-sparql-11/
- SPARQL playground
  - http://sparql-playground.sib.swiss/
- YASGUI: SPARQL query editors
  - http://yasgui.org/