

온톨로지 시각화 도구

```
In [1]: from pyvis.network import Network
import networkx as nx
from SPARQLWrapper import SPARQLWrapper, JSON
from rdflib import Graph
from rdflib.plugins.sparql.results.jsonresults import JSONResultSet
import json

from enum import Enum

class ShowLiteral(Enum):
    Yes = 0
    No = 1
    InNode = 2

class Vis(object):
    """
    SPARQL Endpoint 과 연동하여 특정 트리플들을 시각화하는 도구,
    파일을 읽어 파일의 내용을 시각화할 수도 있다.
    """

    def __init__(self, endpoint=None, filepath=None, ignore_property=[]):
        self.__endpoint = endpoint
        self.__filepath = filepath
        self.__ignore_property = ignore_property if ignore_property
        != None else []
        self.__sparql = SPARQLWrapper(endpoint)

    def __getLocalName(self, node):
        if node.find('/') > 0 and node.find('#') > 0:
            return node[node.rindex('/')+1:] if node.rindex('/') >
node.rindex('#') else node[node.rindex('#')+1:]
        elif node.find('/') > 0:
            return node[node.rindex('/')+1:]
        elif node.find('#') > 0:
            return node[node.rindex('#')+1:]
        else:
            return node

    def __make_short_name(self, name):
        if len(name) > 20 :
            name = name[0:20]
        return name

    def setIgnoreProperty(self, ignore_property):
        self.__ignore_property = ignore_property

    def __show_graph(self, results, show_literal):
        nw = Network(height='800px', width='100%', notebook=True)
```

```

node = {}
groups = {}
node_num = 0;
group_num = 0;
edge_list = []

if show_literal:
    groups['literal'] = group_num
    group_num += 1

    for result in results["results"]["bindings"]:
        # print(result["s"]["value"], "\t", result["p"]["value"]
        ], "\t", result["o"]["value"])
        # print(result)
        if not result["p"]["value"] in self.__ignore_property:
            # 존재하지 않는 경우 디폴트로 처리
            if 'slabel' not in result:
                result['slabel'] = {'type': 'literal', 'value':
self.__getLocalName(result['s']['value'])}
            if 'olabel' not in result:
                result['olabel'] = {'type': 'literal', 'value':
self.__getLocalName(result['o']['value'])}
            if 'stype' not in result:
                result['stype'] = {'type': 'literal', 'value':
''}

            if 'otype' not in result:
                result['otype'] = {'type': 'literal', 'value':
''}

            # 노드
            if result['s']['type'] == 'uri' and not result['s']
['value'] in node:
                node[result['s']['value']] = {'id':node_num}
                node_num += 1

            if result['o']['type'] == 'uri' and not result['o']
['value'] in node:
                node[result['o']['value']] = {'id':node_num}
                node_num += 1

            if 'stype' in result and not result['stype']['valu
e'] in groups:
                groups[result['stype']['value']] = group_num
                group_num += 1
            if 'otype' in result and not result['otype']['valu
e'] in groups:
                groups[result['otype']['value']] = group_num
                group_num += 1

            node[result['s']['value']].update({'label':result['
slabel']['value'], 'type':result['stype']['value']})
            if result['o']['type'] == 'uri':
                node[result['o']['value']].update({'label':resu
lt['olabel']['value'], 'type':result['otype']['value']})
            edge_list.append((result['s']['value'], result[
'o']['value'], result['p']['value']))

            # 리터널도 표시하는 경우
            if show_literal == ShowLiteral.Yes:
                if result['o']['type'] == 'literal':
                    node[result['o']['value']+str(node_num)] =
{'id':node_num}

                    node[result['o']['value']+str(node_num)].un

```

```

date({'label':result['o']['value'], 'type':'literal'})
        edge_list.append((result['s']['value'], res
ult['o']['value'], result['p']['value'], result['o']['value']+str(n
ode_num)))

        node_num += 1

        # 노드 안에 넣는 경우
        if show_literal == ShowLiteral.InNode:
            if result['o']['type'] == 'literal':
                if 'datatype' in node[result['s']['value']]
:
                    dt = node[result['s']['value']]['dataty
pe'] + '<br>' + self.__getLocalName(result['p']['value']) + ":" + r
esult['o']['value']
                    node[result['s']['value']].update({'dat
atype': dt})
                else:
                    node[result['s']['value']]['datatype']
= self.__getLocalName(result['p']['value']) + ":" + result['o']['va
lue']
                    node[result['s']['value']].update({'dat
atype': node[result['s']['value']]['datatype']})

            for n in node:
                if node[n]['type'] == 'literal':
                    nw.add_node(n, shape='box',label=self.__make_short_
name(node[n]['label']), title=node[n]['label'], group=groups[node[n
]['type']])
                else:
                    if 'datatype' in node[n]:
                        nw.add_node(n, label=self.__make_short_name(nod
e[n]['label']), title=n+"<br>"+node[n]['datatype'], group=groups[no
de[n]['type']])
                    else:
                        nw.add_node(n, label=self.__make_short_name(nod
e[n]['label']), title=n, group=groups[node[n]['type']])
                for e in edge_list:
                    if len(e)==4:
                        nw.add_edge(e[0], e[3], arrows='to', label=self.__g
etLocalName(e[2]), title=e[2])
                    else:
                        if e[2]=='http://www.w3.org/1999/02/22-rdf-syntax-n
s#type':
                            nw.add_edge(e[0], e[1], width=4, color='black',
arrows='to', label=self.__getLocalName(e[2]), title=e[2])
                        else:
                            nw.add_edge(e[0], e[1], arrows='to', label=self
.__getLocalName(e[2]), title=e[2])

            nw.set_edge_smooth('dynamic')
            nw.show_buttons(filter_=['physics'])

        return nw

def __run_query(self, query_string, show_literal):
    self.__sparql.setQuery(query_string)
    self.__sparql.setReturnFormat(JSON)
    results = self.__sparql.query().convert()
    nw = self.__show_graph(results, show_literal)
    return nw

```

```

def vis(self, search_keyword : str='', uri: str='', limit: int=
500, show_literal: ShowLiteral=ShowLiteral.No):
    if limit > 1000:
        limit = 1000

    if uri != '':
        query_string = """
            prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#
>
            SELECT ?s ?p ?o ?slabel ?olabel ?stype ?totype
            WHERE {
                {
                    filter(?s=<%s>)
                    ?s ?p ?o.
                    optional{?s rdfs:label ?slabel .}
                    optional{?s a ?stype}
                    optional{?o rdfs:label ?olabel .}
                    optional{?o a ?otype}
                } UNION {
                    filter(?o=<%s>)
                    ?s ?p ?o.
                    optional{?s rdfs:label ?slabel .}
                    optional{?s a ?stype}
                    optional{?o rdfs:label ?olabel .}
                    optional{?o a ?otype}
                }
            } LIMIT %i
            """ % (uri, uri, limit)
    elif search_keyword != '':
        query_string = """
            prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#
>
            SELECT ?s ?p ?o ?slabel ?olabel ?stype ?totype
            WHERE {
                ?s ?pp ?oo.
                filter(regex(?oo, '%s'))
                ?s ?p ?o.
                optional{?s rdfs:label ?slabel .}
                optional{?s a ?stype}
                optional{?o rdfs:label ?olabel .}
                optional{?o a ?otype}
            } LIMIT %i
            """ % (search_keyword, limit)
    else:
        query_string = """
            prefix rdfs: <http://www.w3.org/2000/01/rdf-sch
ema#>
            SELECT *
            WHERE {
                ?s ?p ?o.
                optional{?s rdfs:label ?slabel .}
                optional{?s a ?stype}
                optional{?o rdfs:label ?olabel .}
                optional{?o a ?otype}
            } LIMIT %i
            """ % limit

    return self.__run_query(query_string, show_literal)

def vis_file(self, uri: str='', show_literal: ShowLiteral=ShowL
iteral.Yes):
    g = Graph()
    g.parse(self.__filepath, format='turtle')

```

```

if uri != '':
    gres = g.query("""
        prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#

        SELECT ?s ?p ?o ?slabel ?olabel ?stype ?totype
        WHERE {
            {
                filter(?s=<%s>)
                ?s ?p ?o.
                optional{?s rdfs:label ?slabel .}
                optional{?s a ?stype}
                optional{?o rdfs:label ?olabel .}
                optional{?o a ?otype}
            } UNION {
                filter(?o=<%s>)
                ?s ?p ?o.
                optional{?s rdfs:label ?slabel .}
                optional{?s a ?stype}
                optional{?o rdfs:label ?olabel .}
                optional{?o a ?otype}
            }
        }
        """ % (uri, uri))
else:
    gres = g.query("""
        prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#

        SELECT *
        WHERE {
            ?s ?p ?o.
            optional{?s rdfs:label ?slabel .}

            optional{?s a ?stype}
            optional{?o rdfs:label ?olabel .}
            optional{?o a ?otype}
        }
        """)
    f = open('json_result', 'w')
    JSONResultSerializer(gres).serialize(f)
    with open('json_result') as f:
        read_data = json.load(f)

    nw = self.__show_graph(read_data, show_literal)
    return nw

```

SPARQL Endpoint에 대한 사용법

In [2]: # 객체 생성

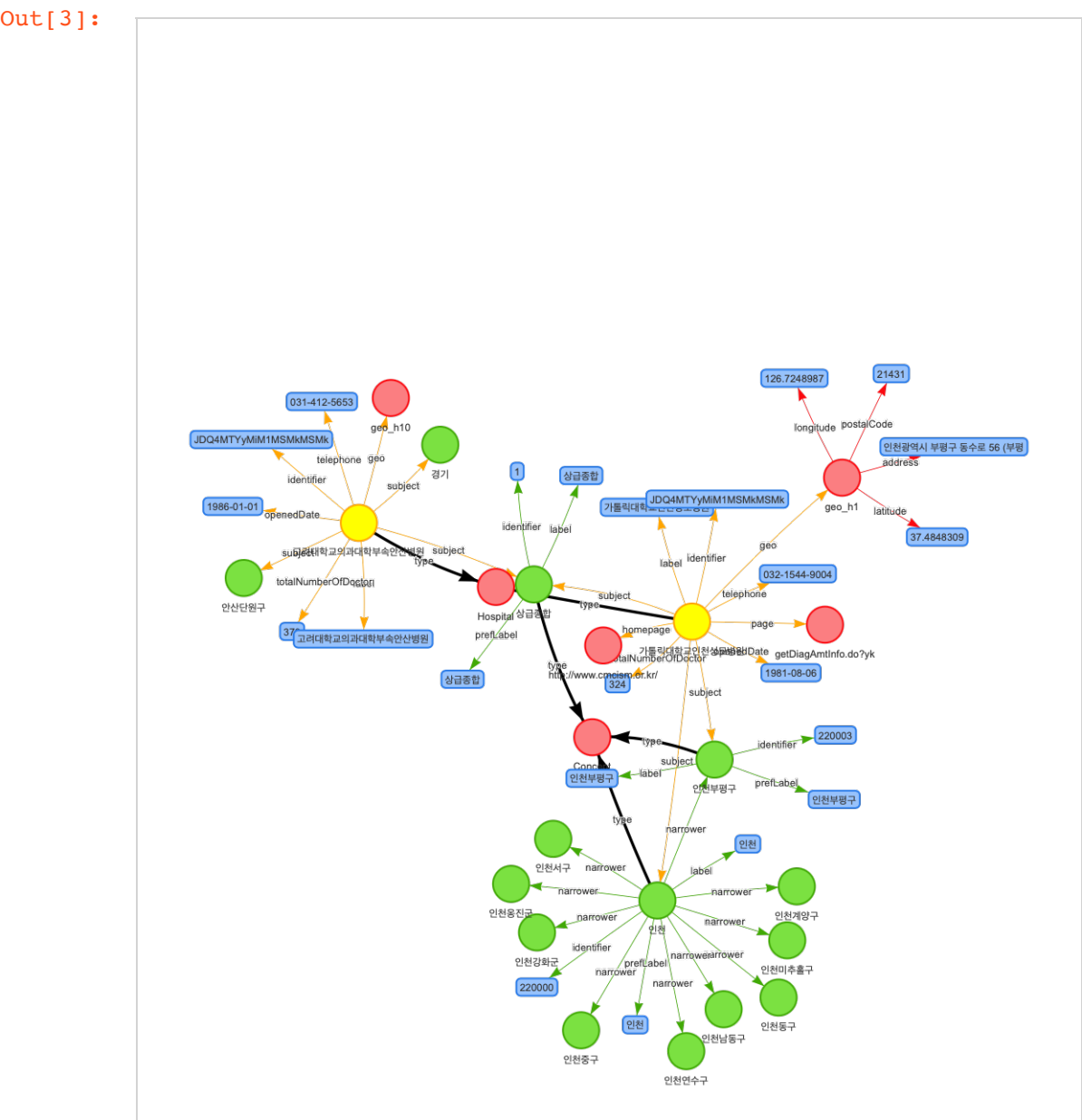
```
v = Vis('http://localhost:3030/publicdata/query')
```

리터널값을 보여주는 방식을 3가지로 정의함

1. ShowLiteral.Yes -> 개별 노드로 구성하여 보여준다.
2. ShowLiteral.No -> 보여주지 않는다.(Resource 인 노드들만 시각화 한다.)(default)
3. ShowLiteral.InNode -> 리터널값을 가지는 주어 노드의 톨팁으로 보여준다.

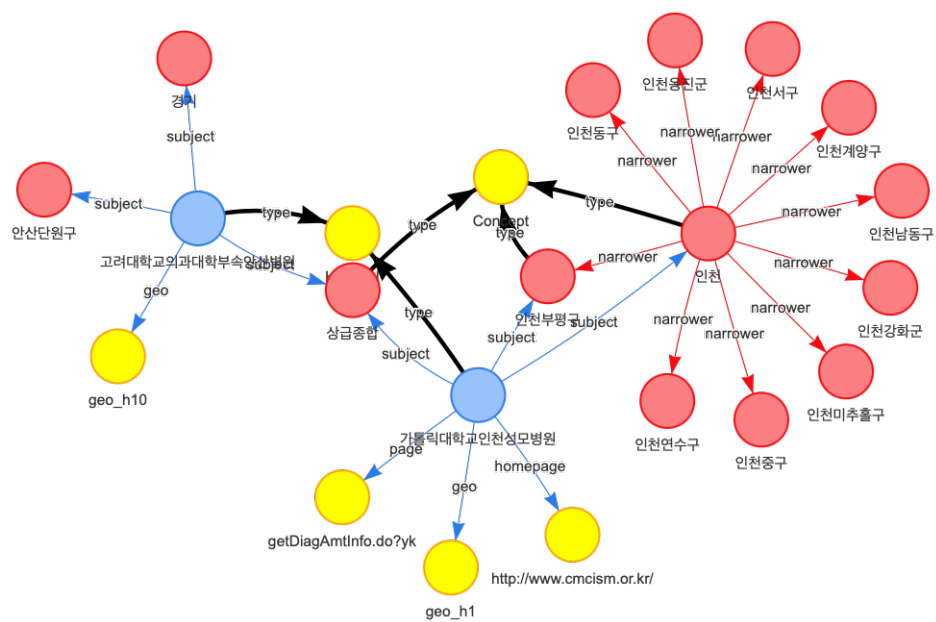
기본 사용

```
In [3]: nw = v.vis(limit=50, show_literal=ShowLiteral.Yes)
nw.show('default1.html')
```



```
In [4]: nw = v.vis(limit=50, show_literal=ShowLiteral.InNode)
nw.show('default2.html')
```

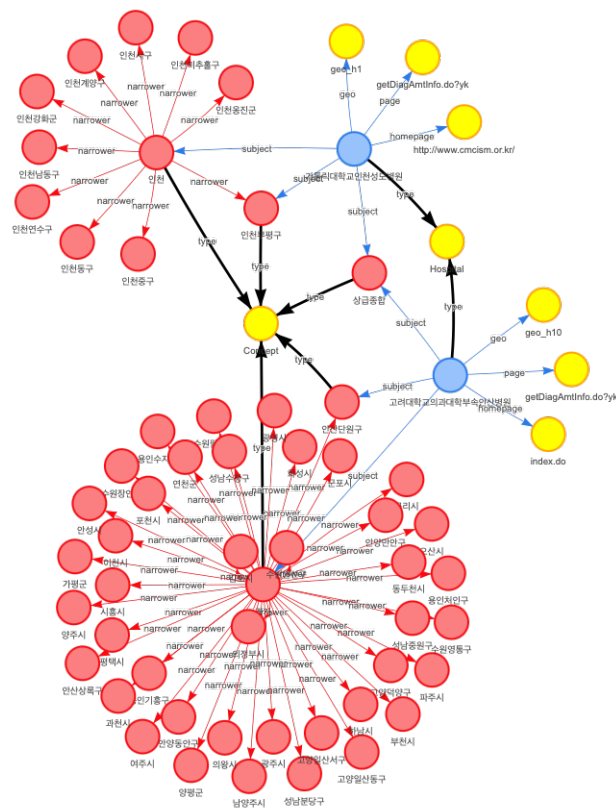
Out[4]:



```
nw = v.vis(limit=100)
nw.show('default3.html')
```

Out[5]:

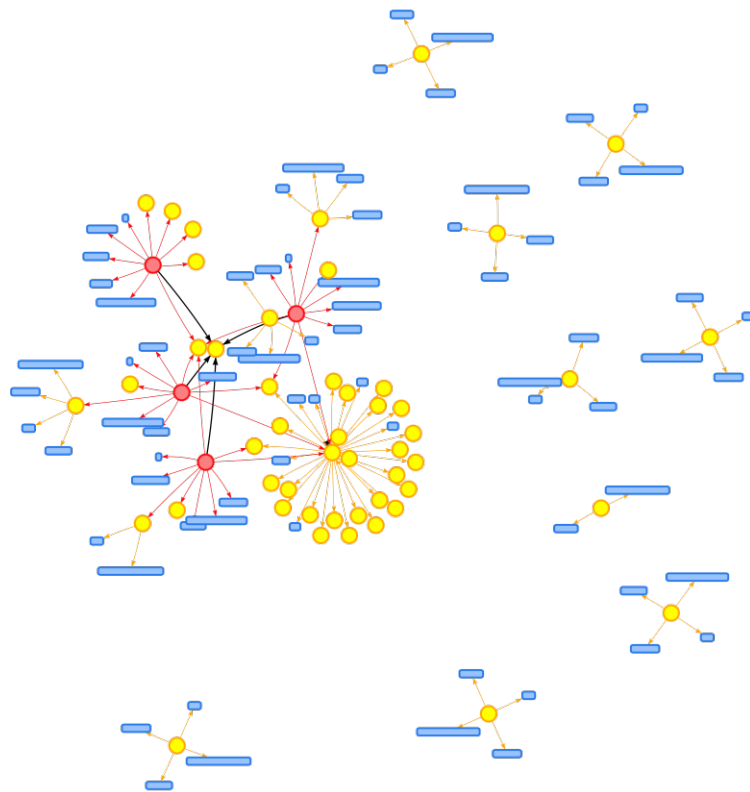
The graph displays a complex network of relationships between various entities. Central nodes (yellow) include '인원' (Personnel), 'Concept', '상급종합' (Superior Comprehensive), 'Hospital', 'geo_h10', 'index.do', and 'getDiagAmtInfo.do?yk'. Peripheral nodes (red) represent specific locations and departments, such as '인천계양구' (Incheon Gyeongsang-gu), '인천남동구' (Incheon Namdong-gu), '인천서구' (Incheon Seogu), '인천중구' (Incheon Junggu), '인천북구' (Incheon Bukgu), '인천광역시' (Incheon Metropolitan City), '인천시' (Incheon City), '인천군' (Incheon-gun), '인천읍면동' (Incheon-eupmyeon-dong), '인천시청' (Incheon City Hall), '인천시의회' (Incheon City Council), '인천시교육청' (Incheon City Education Office), '인천시경찰청' (Incheon City Police Agency), '인천시소방청' (Incheon City Fire Agency), '인천시문화재단' (Incheon City Cultural Foundation), '인천시체육회' (Incheon City Sports Commission), '인천시농업기술센터' (Incheon City Agricultural Technology Center), '인천시수립계획위원회' (Incheon City Urban Planning Committee), '인천시도시관리위원회' (Incheon City Urban Management Committee), '인천시환경보전위원회' (Incheon City Environment and Conservation Committee), '인천시안전대책추진위원회' (Incheon City Safety Policy Implementation Committee), '인천시재난안전대책추진위원회' (Incheon City Disaster Safety Policy Implementation Committee), '인천시교통안전대책추진위원회' (Incheon City Traffic Safety Policy Implementation Committee), '인천시건설안전대책추진위원회' (Incheon City Construction Safety Policy Implementation Committee), '인천시보건안전대책추진위원회' (Incheon City Health Safety Policy Implementation Committee), '인천시식품안전대책추진위원회' (Incheon City Food Safety Policy Implementation Committee), '인천시약물안전대책추진위원회' (Incheon City Drug Safety Policy Implementation Committee), '인천시의료안전대책추진위원회' (Incheon City Medical Safety Policy Implementation Committee), '인천시화재안전대책추진위원회' (Incheon City Fire Safety Policy Implementation Committee), '인천시가스안전대책추진위원회' (Incheon City Gas Safety Policy Implementation Committee), '인천시전기안전대책추진위원회' (Incheon City Electrical Safety Policy Implementation Committee), '인천시건축안전대책추진위원회' (Incheon City Building Safety Policy Implementation Committee), '인천시산업안전대책추진위원회' (Incheon City Industrial Safety Policy Implementation Committee), '인천시노동안전대책추진위원회' (Incheon City Labor Safety Policy Implementation Committee), '인천시관광안전대책추진위원회' (Incheon City Tourism Safety Policy Implementation Committee), '인천시스포츠안전대책추진위원회' (Incheon City Sports Safety Policy Implementation Committee), '인천시문화안전대책추진위원회' (Incheon City Culture Safety Policy Implementation Committee), '인천시복지안전대책추진위원회' (Incheon City Welfare Safety Policy Implementation Committee), '인천시여성안전대책추진위원회' (Incheon City Women's Safety Policy Implementation Committee), '인천시장애인안전대책추진위원회' (Incheon City Disabled Persons' Safety Policy Implementation Committee), '인천시노년층안전대책추진위원회' (Incheon City Elderly Safety Policy Implementation Committee), '인천시청소년안전대책추진위원회' (Incheon City Youth Safety Policy Implementation Committee), '인천시어린이안전대책추진위원회' (Incheon City Children's Safety Policy Implementation Committee), '인천시유아안전대책추진위원회' (Incheon City Toddler Safety Policy Implementation Committee), '인천시가족안전대책추진위원회' (Incheon City Family Safety Policy Implementation Committee), '인천시공동주택안전대책추진위원회' (Incheon City Condominium Safety Policy Implementation Committee), '인천시주택안전대책추진위원회' (Incheon City Housing Safety Policy Implementation Committee), '인천시주택임대차안전대책추진위원회' (Incheon City Rental Housing Safety Policy Implementation Committee), '인천시주택금융안전대책추진위원회' (Incheon City Housing Finance Safety Policy Implementation Committee), '인천시주택정책위원회' (Incheon City Housing Policy Committee), '인천시주택시장조사위원회' (Incheon City Housing Market Research Committee), '인천시주택가격안정대책추진위원회' (Incheon City Housing Price Stabilization Policy Implementation Committee), '인천시주택공급확충대책추진위원회' (Incheon City Housing Supply Expansion Policy Implementation Committee), '인천시주택분양가절감대책추진위원회' (Incheon City Housing Sales Price Reduction Policy Implementation Committee), '인천시주택임대료절감대책추진위원회' (Incheon City Housing Rent Reduction Policy Implementation Committee), '인천시주택보증금환급대책추진위원회' (Incheon City Housing Deposit Refund Policy Implementation Committee), '인천시주택연금제도개선대책추진위원회' (Incheon City Housing Pension System Improvement Policy Implementation Committee), '인천시주택연금제도홍보대책추진위원회' (Incheon City Housing Pension System Promotion Policy Implementation Committee), '인천시주택연금제도지원대책추진위원회' (Incheon City Housing Pension System Support Policy Implementation Committee), '인천시주택연금제도평가대책추진위원회' (Incheon City Housing Pension System Evaluation Policy Implementation Committee), '인천시주택연금제도개선대책추진위원회' (Incheon City Housing Pension System Improvement Policy Implementation Committee).



검색어 입력

```
In [6]: nw = v.vis(search_keyword='서울',limit=150, show_literal=ShowLiteral
        .Yes)
        nw.show('default4.html')
```

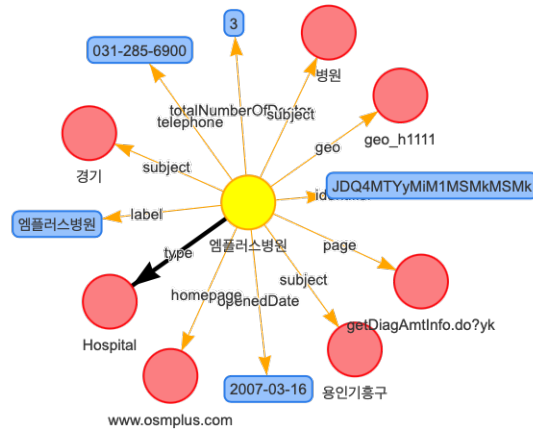
Out[6]:



URI 입력

```
In [7]: nw = v.vis(uri='http://joyhong.tistory.com/resource/h_1111',limit=1
50, show_literal=ShowLiteral.Yes)
nw.show('default5.html')
```

Out[7]:



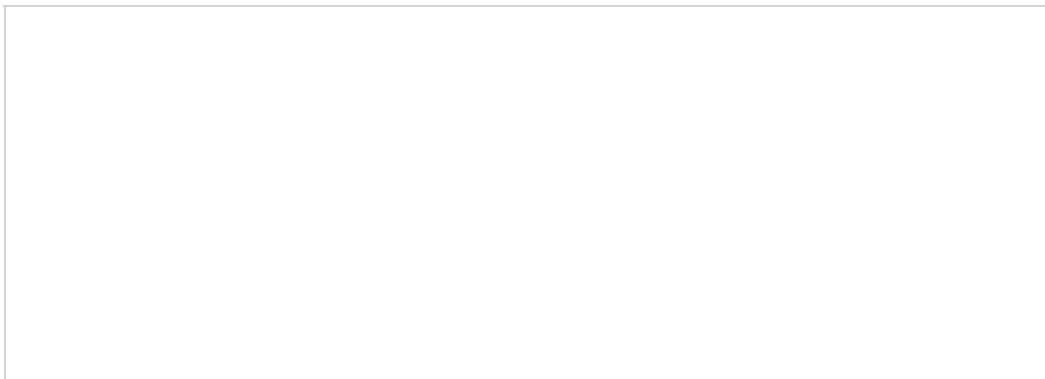
ignore property 설정

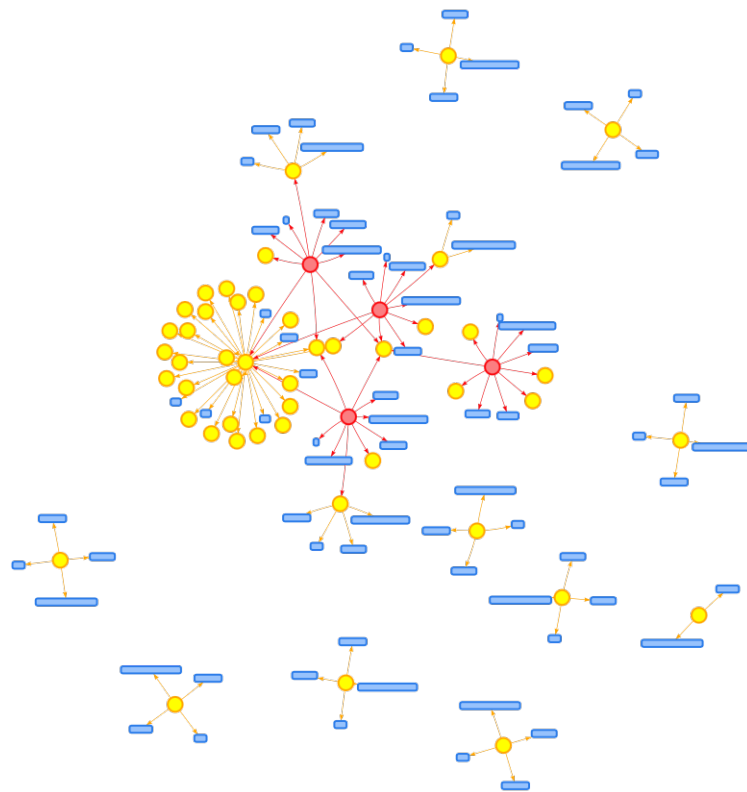
시각화하는 내용 중에 삭제하고자 하는 관계를 ignore property 로 설정하여 사용한다.

```
In [8]: from rdflib.namespace import RDF
```

```
ignore = [str(RDF.type)]  
v.setIgnoreProperty(ignore)  
  
nw = v.vis(search_keyword='서울', limit=150, show_literal=ShowLiteral  
.Yes)  
nw.show('default6.html')
```

Out[8]:





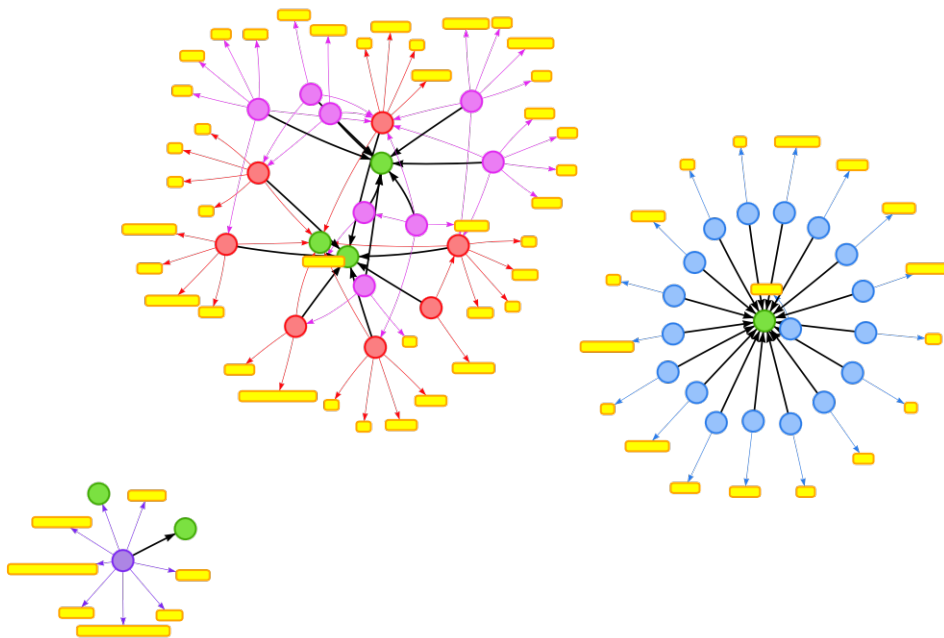
파일에 대한 사용법

기본 사용

```
In [9]: vf = Vis(filepath='sample_model.ttl')
```

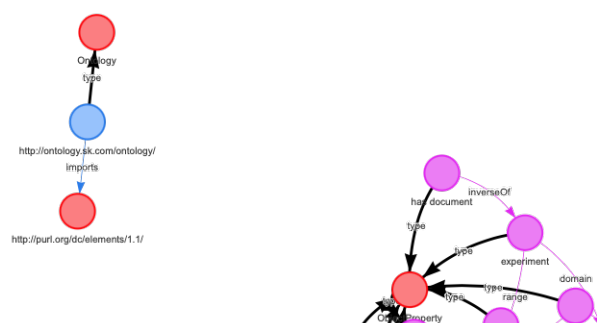
```
In [10]: nw = vf.vis_file()  
nw.show('default7.html')
```

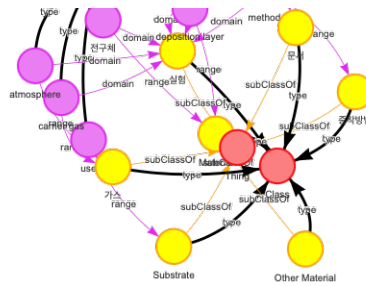
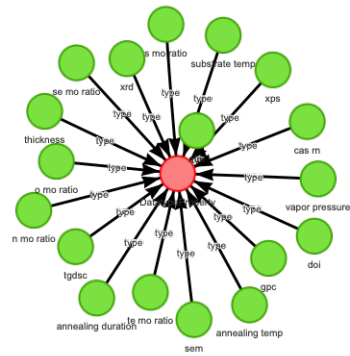
Out[10]:



```
In [11]: nw = vf.vis_file(show_literal=ShowLiteral.InNode)
nw.show('default8.html')
```

Out[11]:

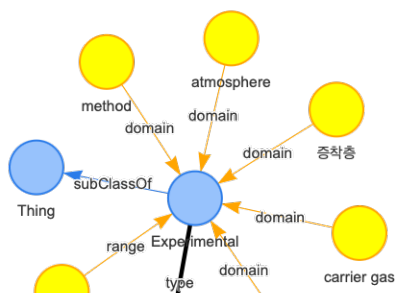




URI 입력

```
In [12]: nw = vf.vis_file(uri='http://test.com/Experimental',show_literal=ShowLiteral.InNode)
nw.show('default9.html')
```

Out[12]:



experiment



precursor

Finish