Week7 & Week8

1. Week7

1.1 Q1

- Who are the creators(including paintings) of Guernica and Sunflowers, respectively

1.1.1 SPARQL语句

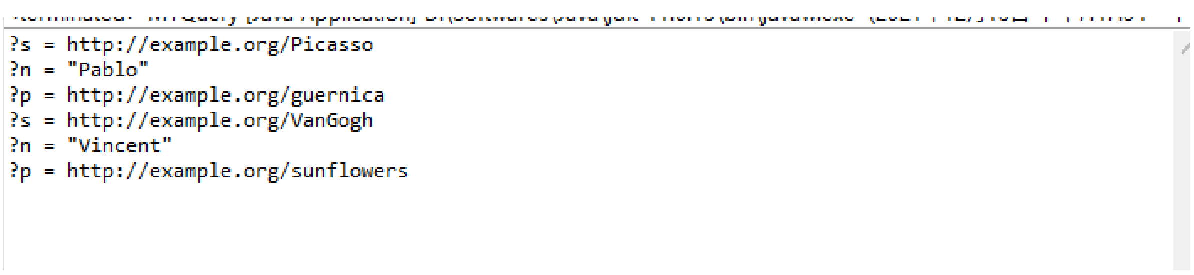
```
PREFIX ex: <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?s ?p ?n
WHERE {
    ?s ex:creatorOf ?p;
       foaf:firstName ?n
    {?p rdfs:label \"Guernica\".}
    UNION{?p rdfs:label \"Sunflowers\".
}
```

1.1.2 code

```
public class MYQuery {

    public static void main(String[] args)
            throws IOException {
        // Create a new Repository.
        Repository db = new SailRepository(new
MemoryStore());

        // Open a connection to the database
```

```java
        try (RepositoryConnection conn = db.getConnection()) {
            String filename = "example-data-artists.ttl";
            try (InputStream input = MYQuery.class.getResourceAsStream("/" + filename)) {
                // add the RDF data from the inputstream directly to our database
                conn.add(input, "", RDFFormat.TURTLE);
            }

            // We do a simple SPARQL SELECT-query that retrieves all resources of type `ex:Artist`,
            // and their first names.
            String queryString = "PREFIX ex: <http://example.org/> \n";
            queryString += "PREFIX foaf: <" + FOAF.NAMESPACE + "> \n";
            queryString += "SELECT ?s ?n ?p \n";
            queryString += "WHERE { \n";
            queryString += "    ?s ex:creatorOf ?p; \n";
            queryString += "       foaf:firstName ?n; \n";
            queryString += "    {?p rdfs:label \"Guernica\".} \n";
            queryString += "    UNION{?p rdfs:label \"Sunflowers\".} \n";
            queryString += "}";

            TupleQuery query = conn.prepareTupleQuery(queryString);

            // A QueryResult is also an AutoCloseable resource, so make sure it gets closed when done.
            try (TupleQueryResult result = query.evaluate()) {
                // we just iterate over all solutions in the result...
                for (BindingSet solution : result) {
                    // ... and print out the value of the variable binding for ?s and ?n
                    System.out.println("?s = " + solution.getValue("s"));
                    System.out.println("?n = " + solution.getValue("n"));
                    System.out.println("?p = " + solution.getValue("p"));
                }
            }
        } finally {
            // Before our program exits, make sure the database is properly shut down.
            db.shutDown();
        }
    }
}

```

1.1.3 Result



```
?s = http://example.org/Picasso
?n = "Pablo"
?p = http://example.org/guernica
?s = http://example.org/VanGogh
?n = "Vincent"
?p = http://example.org/sunflowers
```

1.2 Q2

- List all the artists (including living places) who live in Spain or other places.

1.2.1 SPARQL

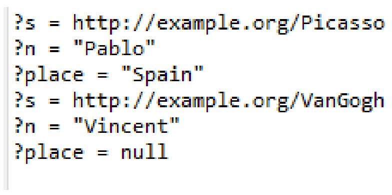
```sparql
PREFIX ex: <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?s ?n ?place
WHERE {
    ?s a ex:Artist;
       foaf:firstName ?n.
    OPTIONAL{?s ex:homeAddress ?p.
            ?p ex:country ?place.}
}
```

1.2.2 code

```java
public class MYQuery2 {

    public static void main(String[] args)
            throws IOException {
        // Create a new Repository.
        Repository db = new SailRepository(new
MemoryStore());

        // Open a connection to the database
        try (RepositoryConnection conn =
db.getConnection()) {
            String filename = "example-data-
artists.ttl";
            try (InputStream input =
MYQuery2.class.getResourceAsStream("/" +
filename)) {
```

```
                // add the RDF data from the inputstream directly to our database
                conn.add(input, "", RDFFormat.TURTLE);
            }

            // We do a simple SPARQL SELECT-query that retrieves all resources of type `ex:Artist`,
            // and their first names.
            String queryString = "PREFIX ex: <http://example.org/> \n";
            queryString += "PREFIX foaf: <" + FOAF.NAMESPACE + "> \n";
            queryString += "SELECT ?s ?n ?place \n";
            queryString += "WHERE { \n";
            queryString += "    ?s a ex:Artist; \n";
            queryString += "       foaf:firstName ?n. \n";
            queryString += "    OPTIONAL{?s ex:homeAddress ?p. \n";
            queryString += "    ?p ex:country ?place.} \n";
            queryString += "}";

            TupleQuery query = conn.prepareTupleQuery(queryString);

            // A QueryResult is also an AutoCloseable resource, so make sure it gets closed when done.
            try (TupleQueryResult result = query.evaluate()) {
                // we just iterate over all solutions in the result...
                for (BindingSet solution : result) {
                    // ... and print out the value of the variable binding for ?s and ?n
                    System.out.println("?s = " + solution.getValue("s"));
                    System.out.println("?n = " + solution.getValue("n"));
                    System.out.println("?place = " + solution.getValue("place"));
                }
            }
        } finally {
            // Before our program exits, make sure the database is properly shut down.
            db.shutDown();
        }
    }
}

```

1.2.3 Result



```
?s = http://example.org/Picasso
?n = "Pablo"
?place = "Spain"
?s = http://example.org/VanGogh
?n = "Vincent"
?place = null
```

1.3 Q3

- List all paintings, their names, and the corresponding techniques.

1.3.1 SPAQL

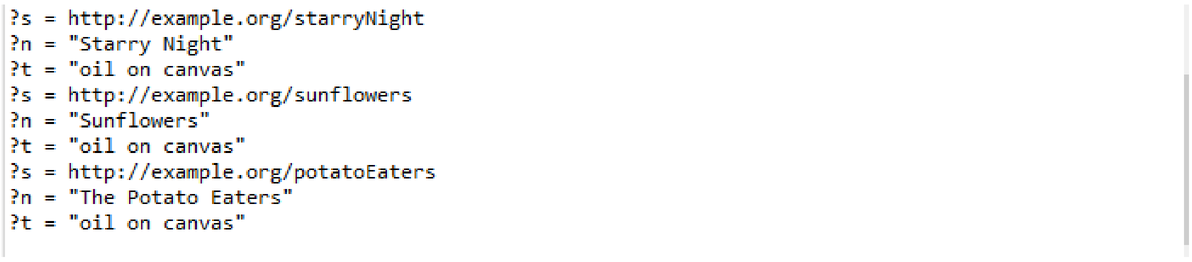
```
PREFIX ex: <http://example.org/>
SELECT ?s ?n ?t
WHERE {
    ?s a ex:Painting;
    rdfs:label ?n;
    ex:technique ?t.
}
```

1.3.2 code

```
public class MYQuery3 {

    public static void main(String[] args)
            throws IOException {
        // Create a new Repository.
        Repository db = new SailRepository(new MemoryStore());

        // Open a connection to the database
        try (RepositoryConnection conn = db.getConnection()) {
            String filename = "example-data-artists.ttl";
            try (InputStream input = MYQuery3.class.getResourceAsStream("/" + filename)) {
                // add the RDF data from the inputstream directly to our database
                conn.add(input, "", RDFFormat.TURTLE);
            }

            // We do a simple SPARQL SELECT-query that retrieves all resources of type `ex:Artist`,
            // and their first names.
```

```java
            String queryString = "PREFIX ex: <http://example.org/> \n";
            queryString += "PREFIX foaf: <" + FOAF.NAMESPACE + "> \n";
            queryString += "SELECT ?s ?n ?t \n";
            queryString += "WHERE { \n";
            queryString += "    ?s a ex:Painting; \n";
            queryString += "       rdfs:label ?n; \n";
            queryString += "       ex:technique ?t. \n";
            queryString += "}";

            TupleQuery query = conn.prepareTupleQuery(queryString);

            // A QueryResult is also an AutoCloseable resource, so make sure it gets closed when done.
            try (TupleQueryResult result = query.evaluate()) {
                // we just iterate over all solutions in the result...
                for (BindingSet solution : result) {
                    // ... and print out the value of the variable binding for ?s and ?n
                    System.out.println("?n = " + solution.getValue("n"));
                    System.out.println("?t = " + solution.getValue("t"));
                }
            }
        } finally {
            // Before our program exits, make sure the database is properly shut down.
            db.shutDown();
        }
    }
}

```

1.3.3 Result

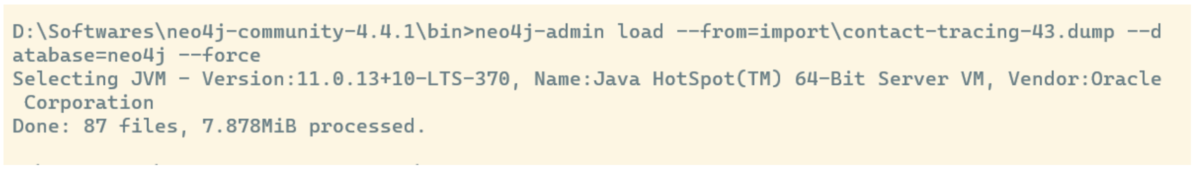


```
?s = http://example.org/starryNight
?n = "Starry Night"
?t = "oil on canvas"
?s = http://example.org/sunflowers
?n = "Sunflowers"
?t = "oil on canvas"
?s = http://example.org/potatoEaters
?n = "The Potato Eaters"
?t = "oil on canvas"
```

2. Week 8

2.1 导入contact-tracing-43.dump文件到数据库neo4j中

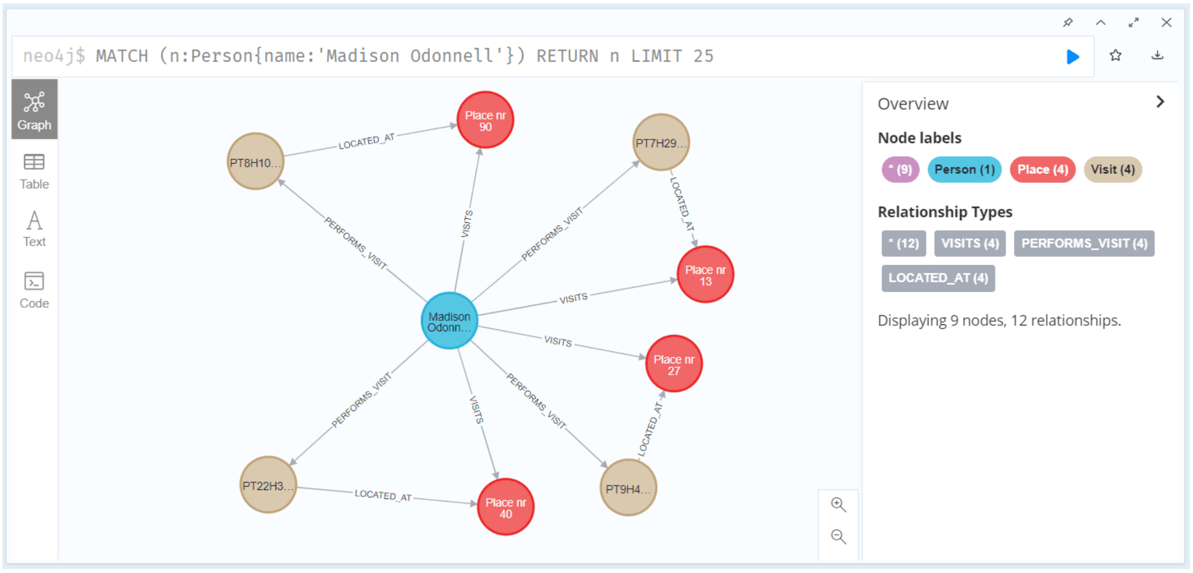
```
neo4j-admin load --from=import\contact-tracing-43.dump --database=neo4j --force
```



```
D:\Softwares\neo4j-community-4.4.1\bin>neo4j-admin load --from=import\contact-tracing-43.dump --d
atabase=neo4j --force
Selecting JVM - Version:11.0.13+10-LTS-370, Name:Java HotSpot(TM) 64-Bit Server VM, Vendor:Oracle
 Corporation
Done: 87 files, 7.878MiB processed.
```

2.2 查询名叫Madison Odonnell的人物节点，并记录下该节点的

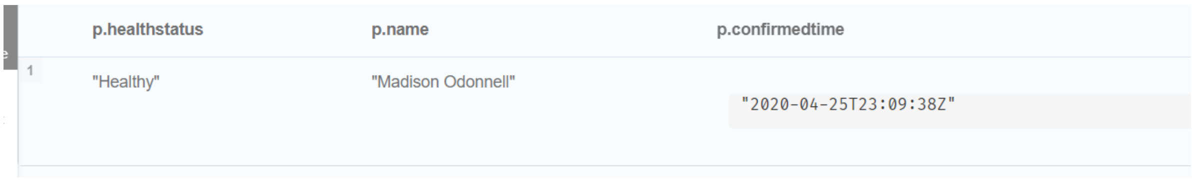
```
MATCH (p:Person{name:'Madison Odonnell'}) Return
LIMIT 25
```



```
neo4j$ MATCH (n:Person{name:'Madison Odonnell'}) RETURN n LIMIT 25
```

Displaying 9 nodes, 12 relationships.

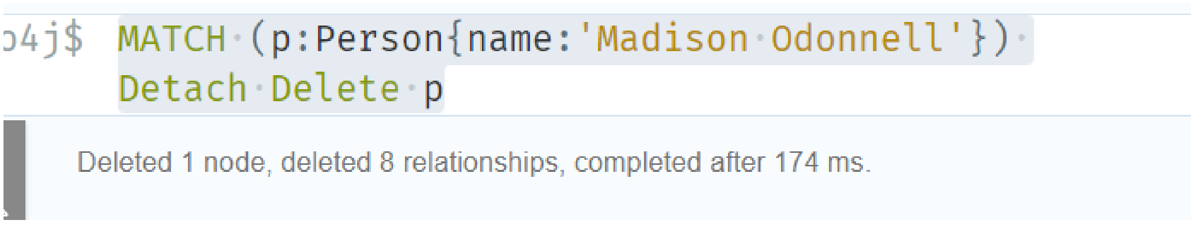
```
MATCH (p:Person{name:'Madison Odonnell'}) return
p.healthstatus,p.name,p.confirmedtime
```



	p.healthstatus	p.name	p.confirmedtime
1	"Healthy"	"Madison Odonnell"	"2020-04-25T23:09:38Z"

2.3 将该人物节点及与其相连的关系删除，并检查是否删除成功

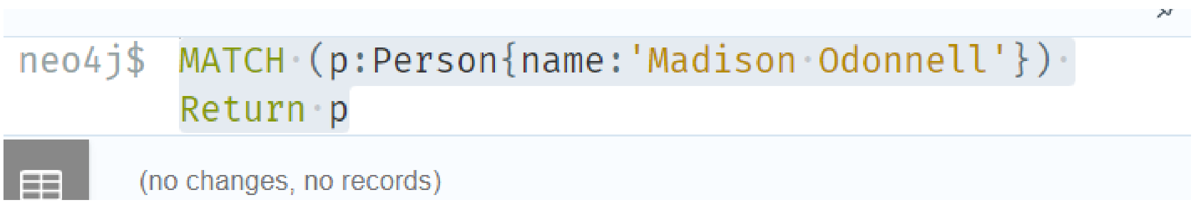
```
MATCH (p:Person{name:'Madison Odonnell'}) Detach
Delete p
```



```
›4j$ MATCH (p:Person{name:'Madison Odonnell'})
     Detach Delete p
```

Deleted 1 node, deleted 8 relationships, completed after 174 ms.

```
MATCH (p:Person{name:'Madison Odonnell'}) Return p
```



```
neo4j$ MATCH (p:Person{name:'Madison Odonnell'})
       Return p
```

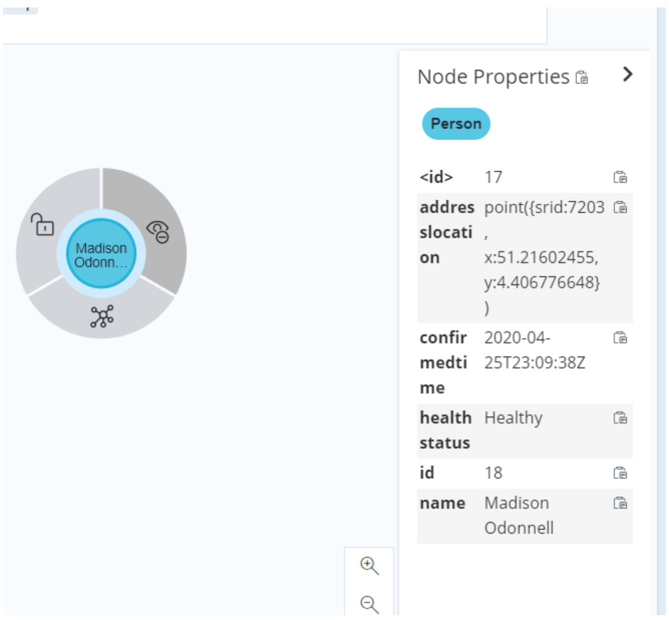
(no changes, no records)

- 删除成功

2.4 重新创建该节点以及第2步记录下来的节点属性;

```
Create
    (p:Person
        {confirmedtime: "2020-04-25T23:09:38Z",
        name: "Madison Odonnell",
        healthstatus: "Healthy",
        id: "18",
        addresslocation: point({srid:7203,
x:51.21602455,             y:4.406776648})})
Return p
```

- 创建成功



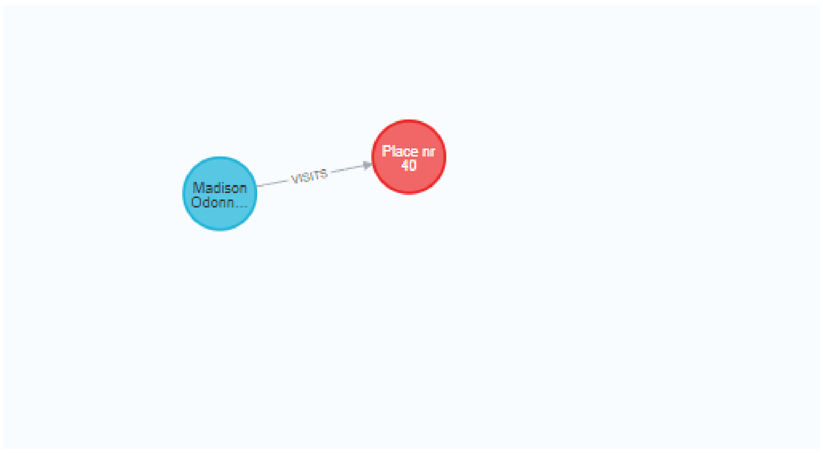
Node Properties

Person

<id>	17
addresslocation	point({srid:7203, x:51.21602455, y:4.406776648})
confirmedtime	2020-04-25T23:09:38Z
healthstatus	Healthy
id	18
name	Madison Odonnell

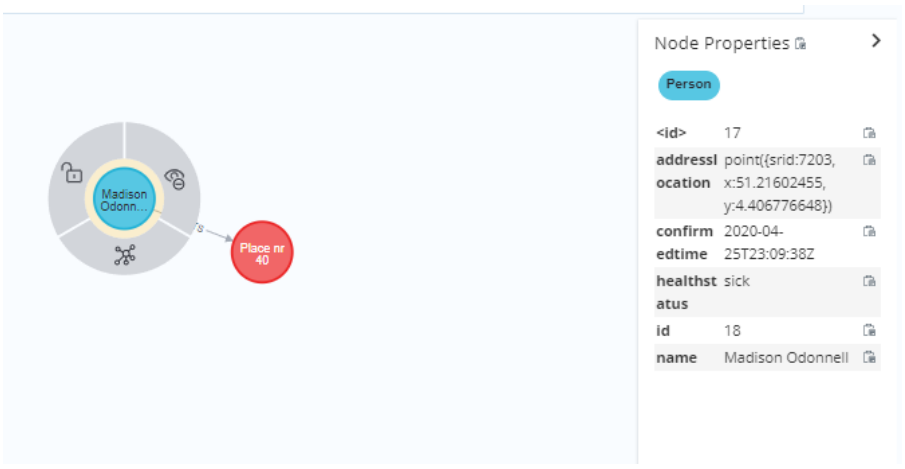
2.5 重新创建关系： Madison Odonnell的人物节点与名为'Place nr 40'的Place节点间的关系，不考虑关系属性;

```
Match
(p:Person{name:'Madison Odonnell'}),
(pl:Place{name:'Place nr 40'})
Create
(p)-[r:VISITS]->(pl)
Return p, r,pl
```



2.6 Madison Odonnell不幸被确诊为新冠（healthstatus='sick'），对图谱进行更新

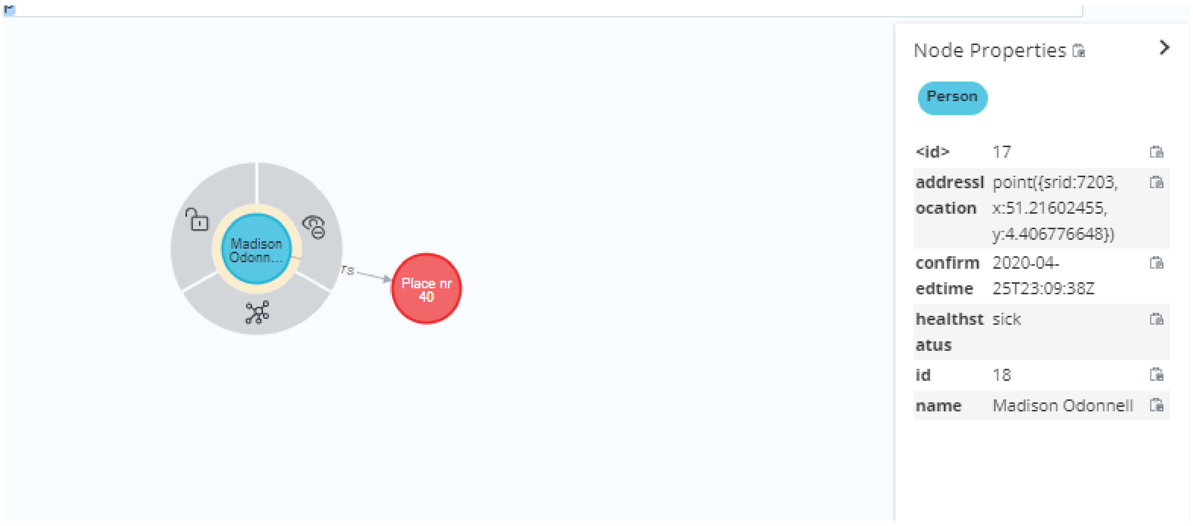
```
Match
(p:Person{name:'Madison Odonnell'})
Set
p.healthstatus='sick'
Return p
```



<id>	17
addresslocation	point({srid:7203, x:51.21602455, y:4.406776648})
confirmedtime	2020-04-25T23:09:38Z
healthstatus	sick
id	18
name	Madison Odonnell

- 检测

```
Match
(p:Person{name:'Madison Odonnell'})
Return p
```



Node Properties

Person

<id>	17
addresslocation	point({srid:7203, x:51.21602455, y:4.406776648})
confirmedtime	2020-04-25T23:09:38Z
healthstatus	sick
id	18
name	Madison Odonnell