

INDEXING AND SEARCHING RDF DATASETS

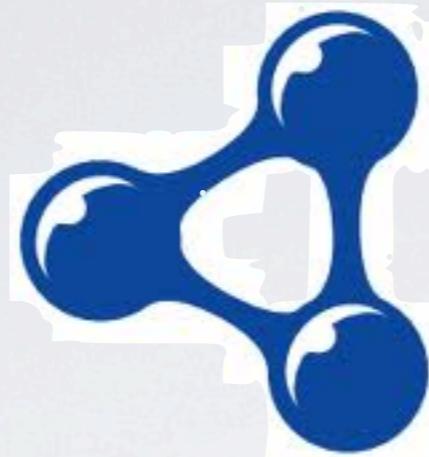
Improving Performance of Semantic Web Applications with
Lucene, SIREn and RDF

Mike Hugo
Entagen, LLC

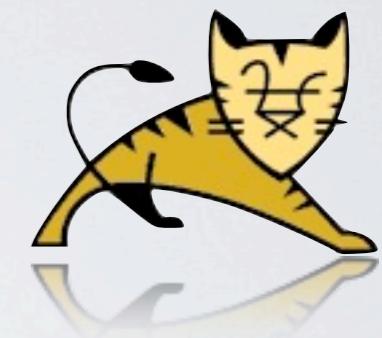
slides and sample code can be found at
<https://github.com/mjhugo/rdf-lucene-siren-presentation>

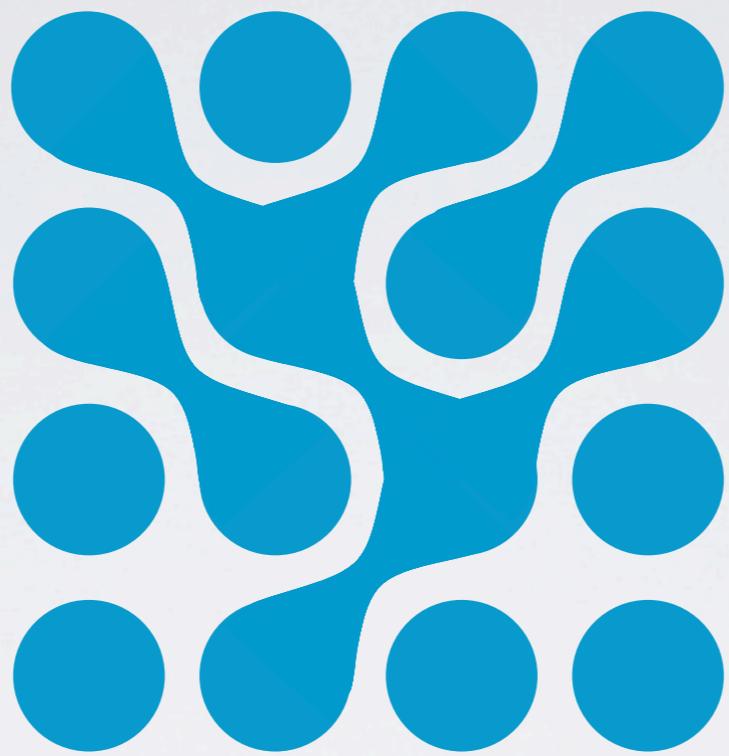


HIBERNATE



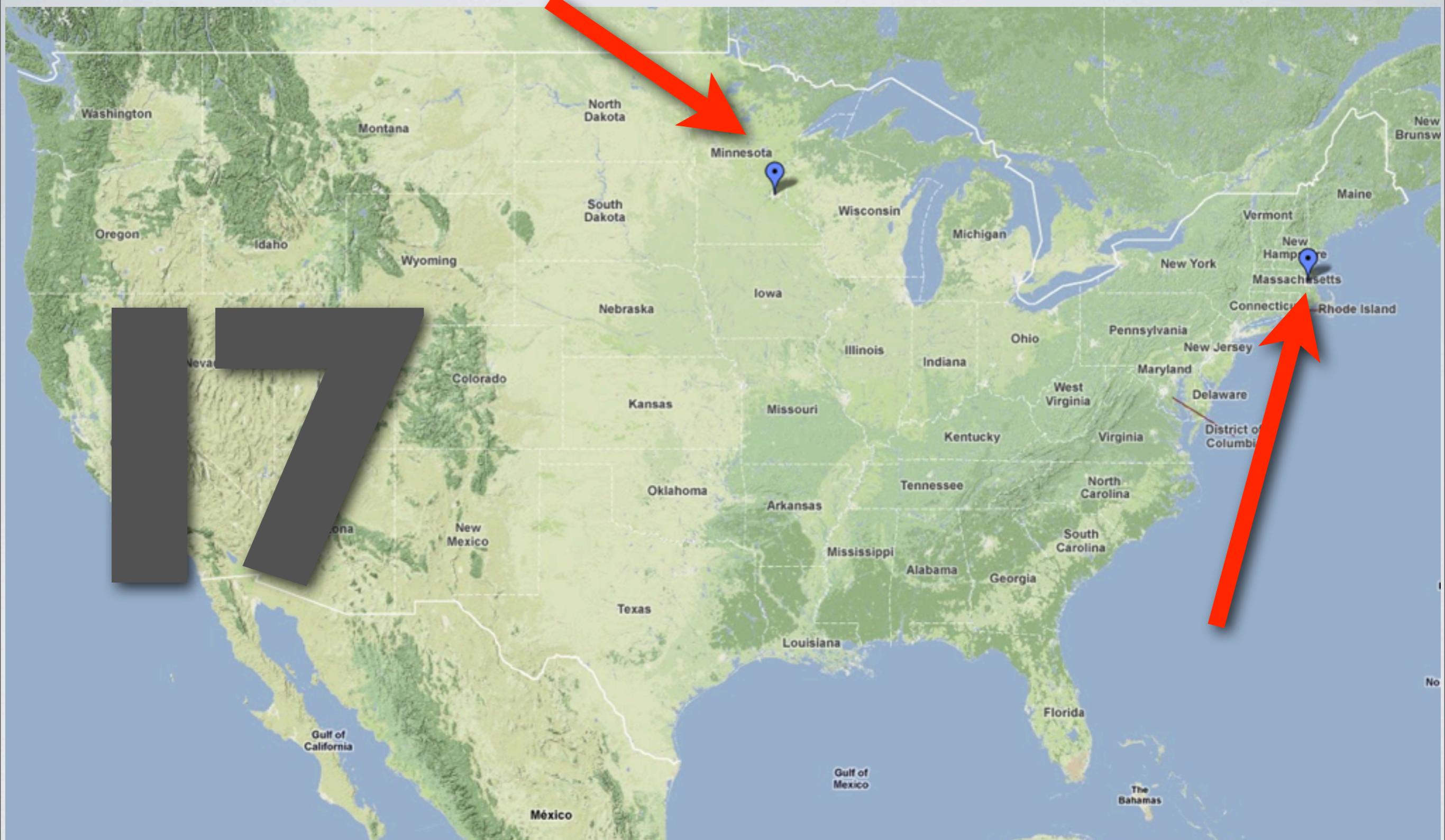
ORACLE®





ENTAGEN

ACCELERATING INSIGHT







TripleMap

Connections Matter



ENTAGEN

ACCELERATING INSIGHT

AGENDA



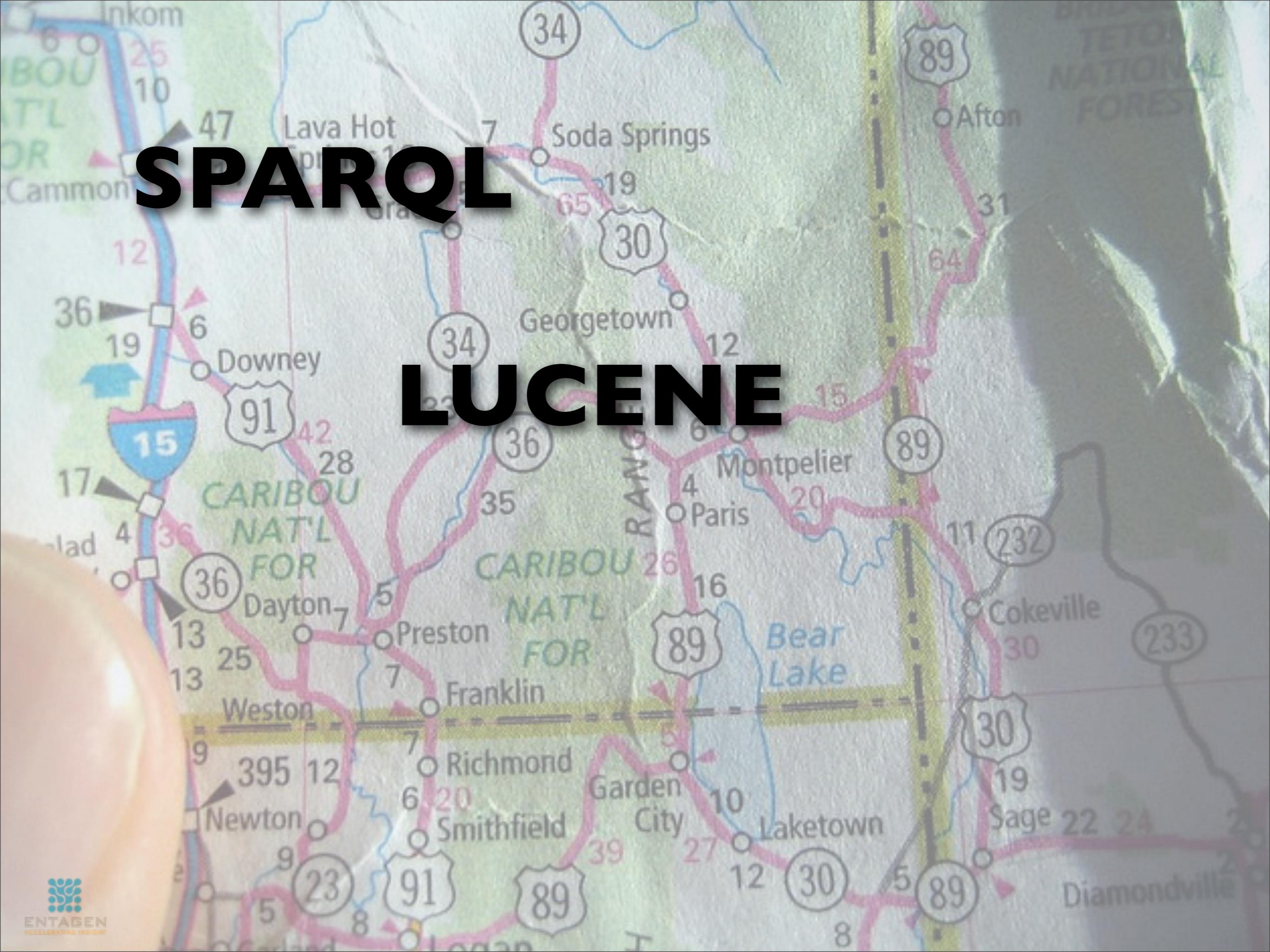


SPARQL



SPARQL

LUCENE

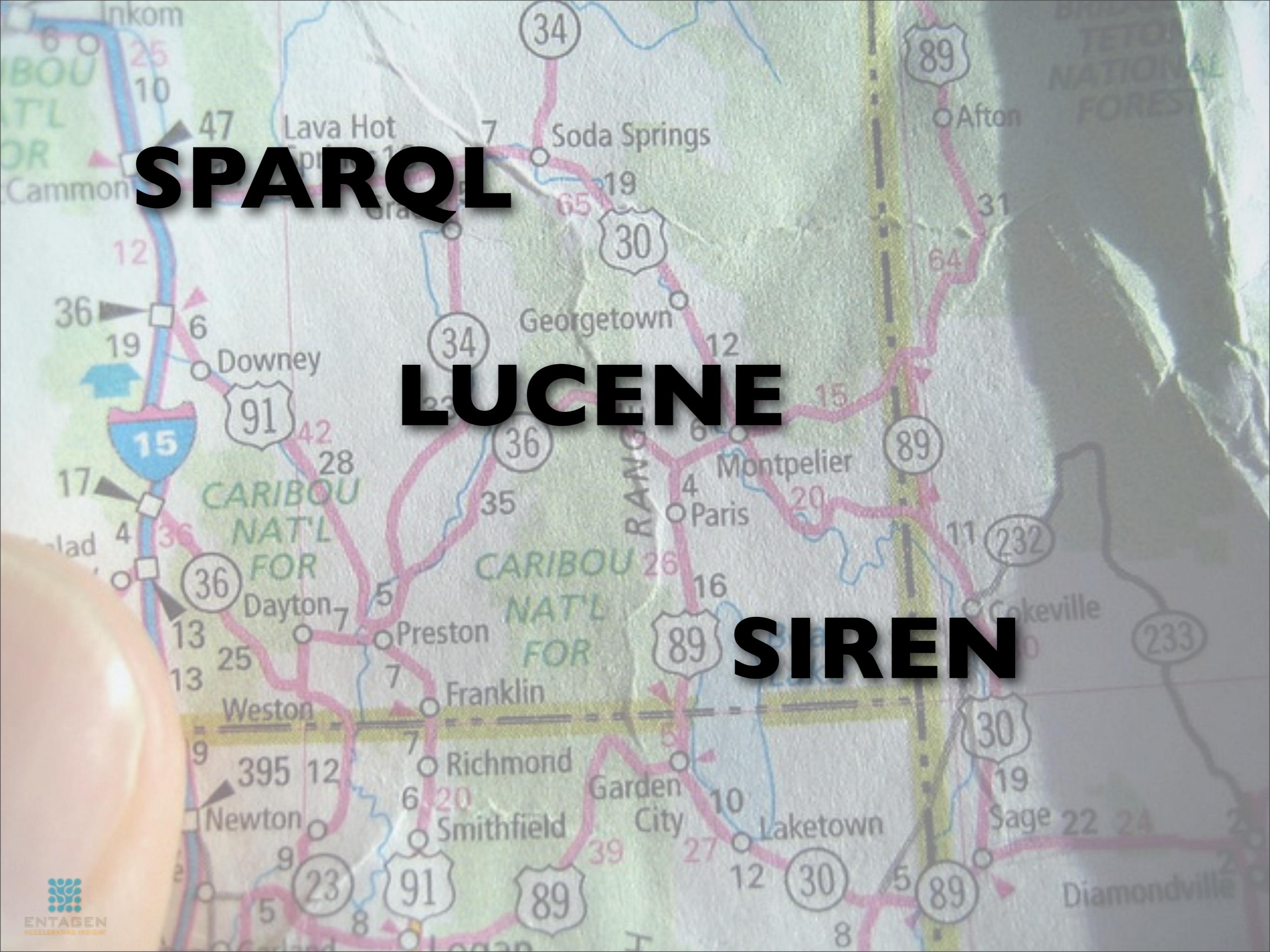


ENTAGEN
ACCELERATING INSIGHT

SPARQL

LUCENE

SIREN



ENTAGEN
ACCELERATING INSIGHT

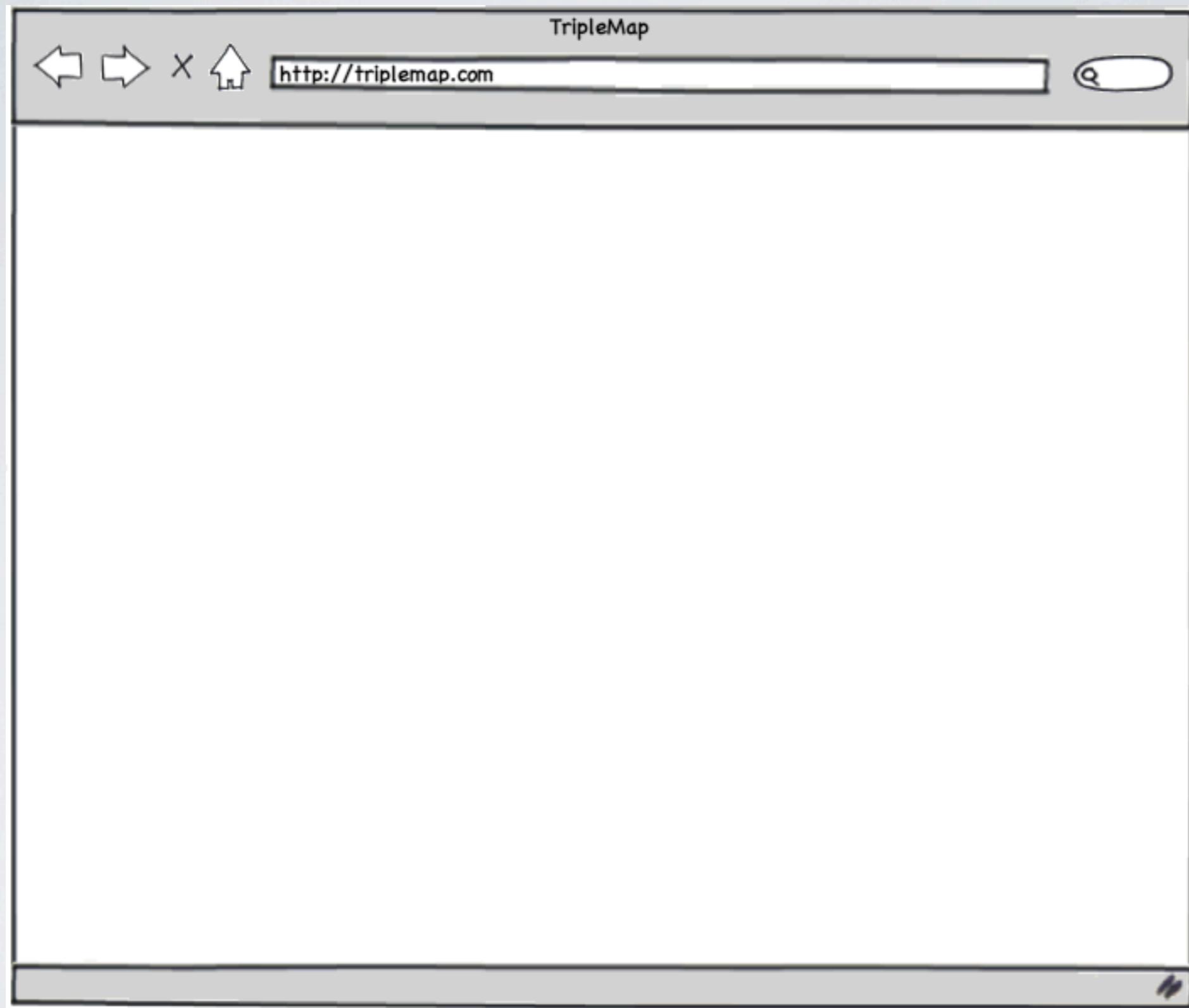
SPARQL

LUCENE

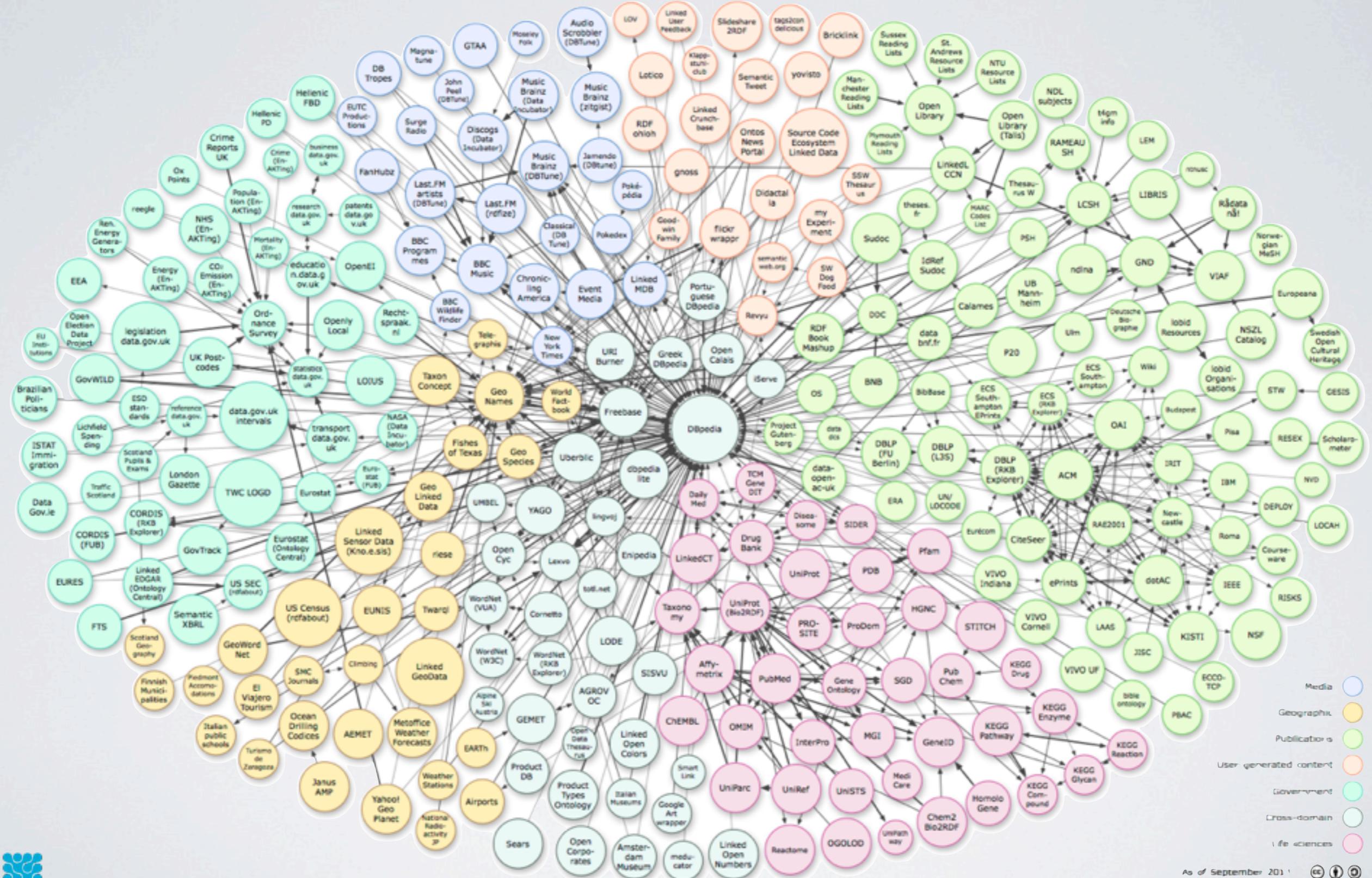
SIREN

TripleMap.com





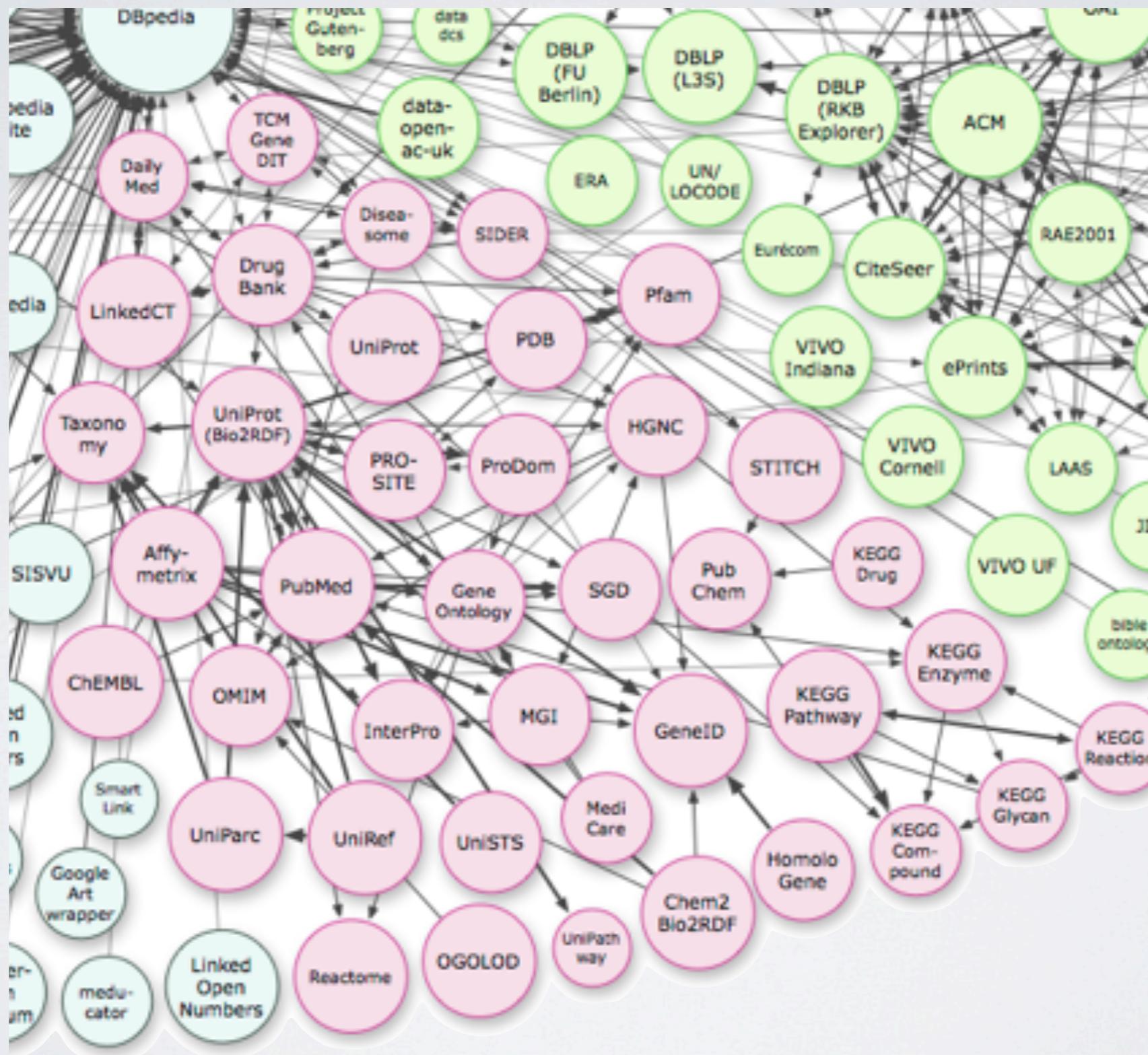
LINKING OPEN DATA



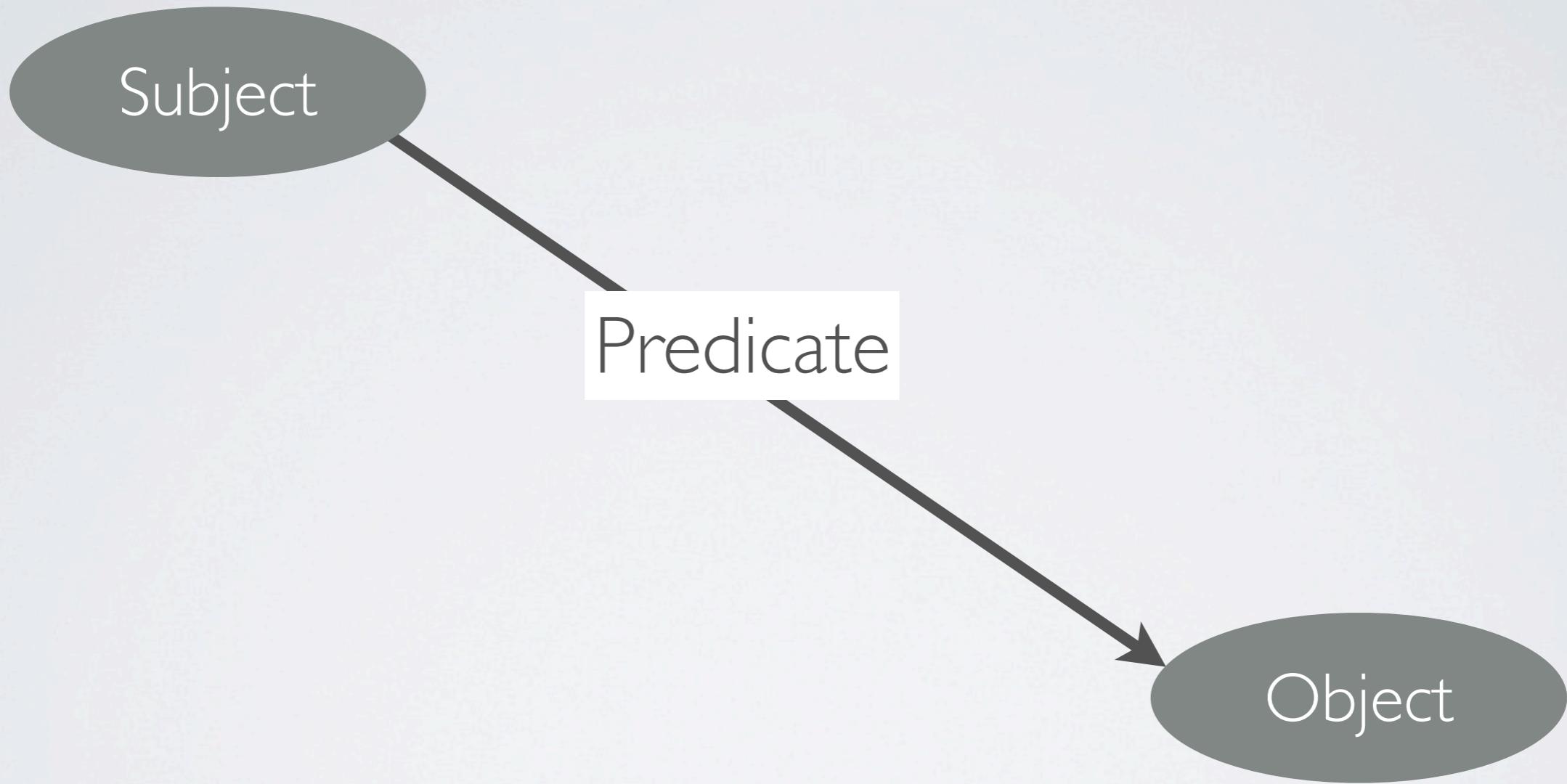
As of September 2011



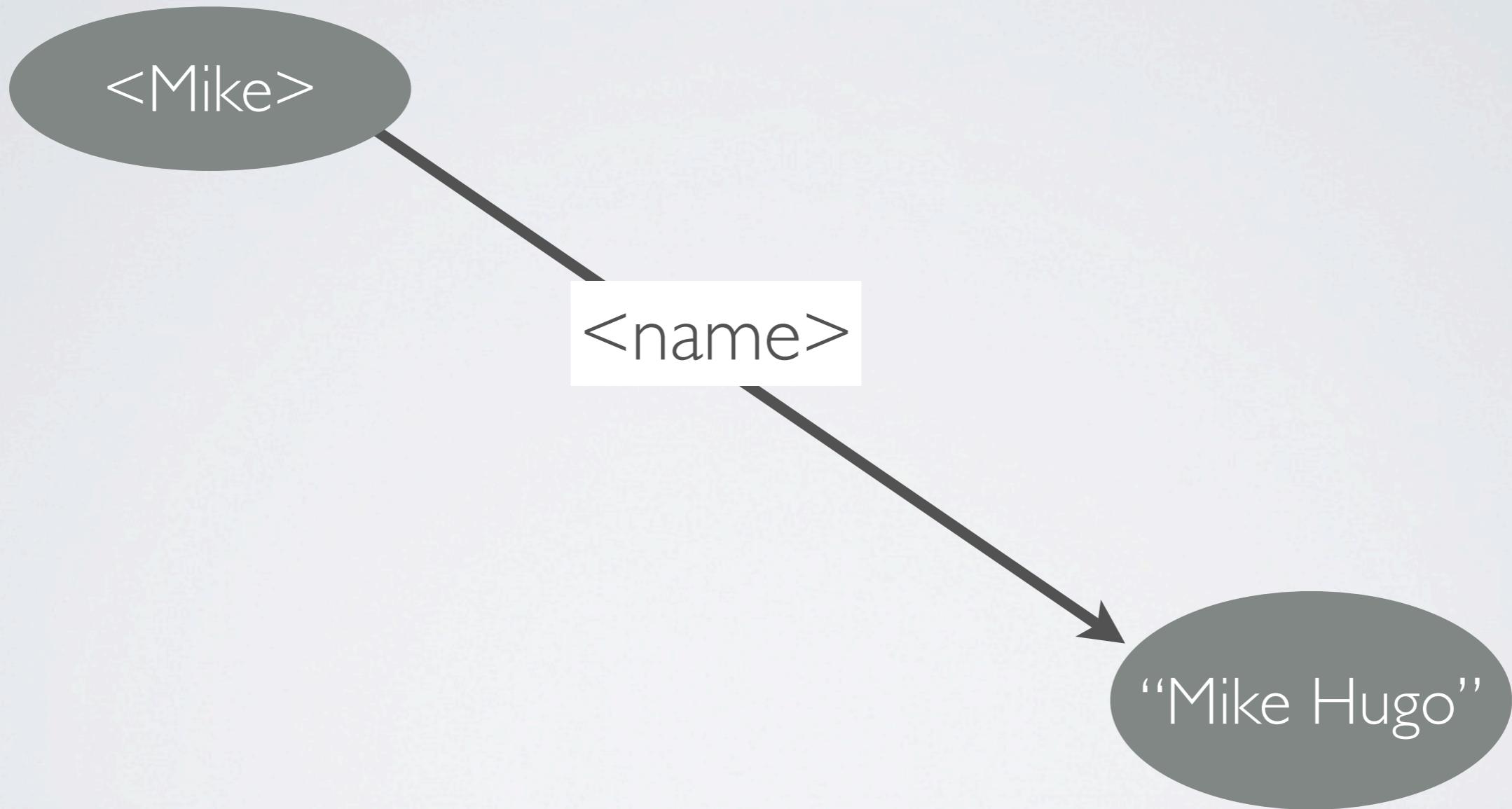
LIFE SCIENCE LINKED DATA



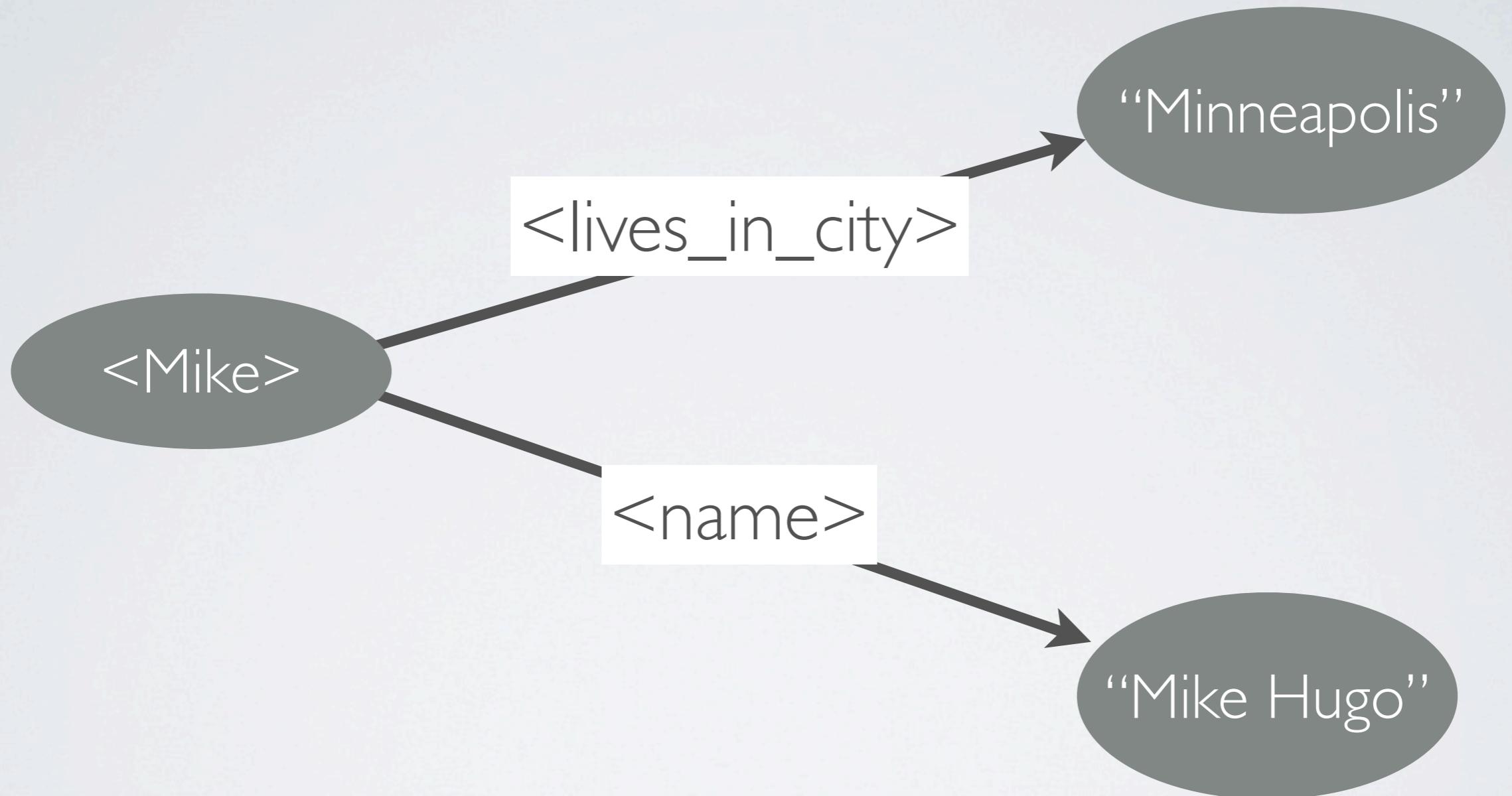
WHAT'S A TRIPLE?



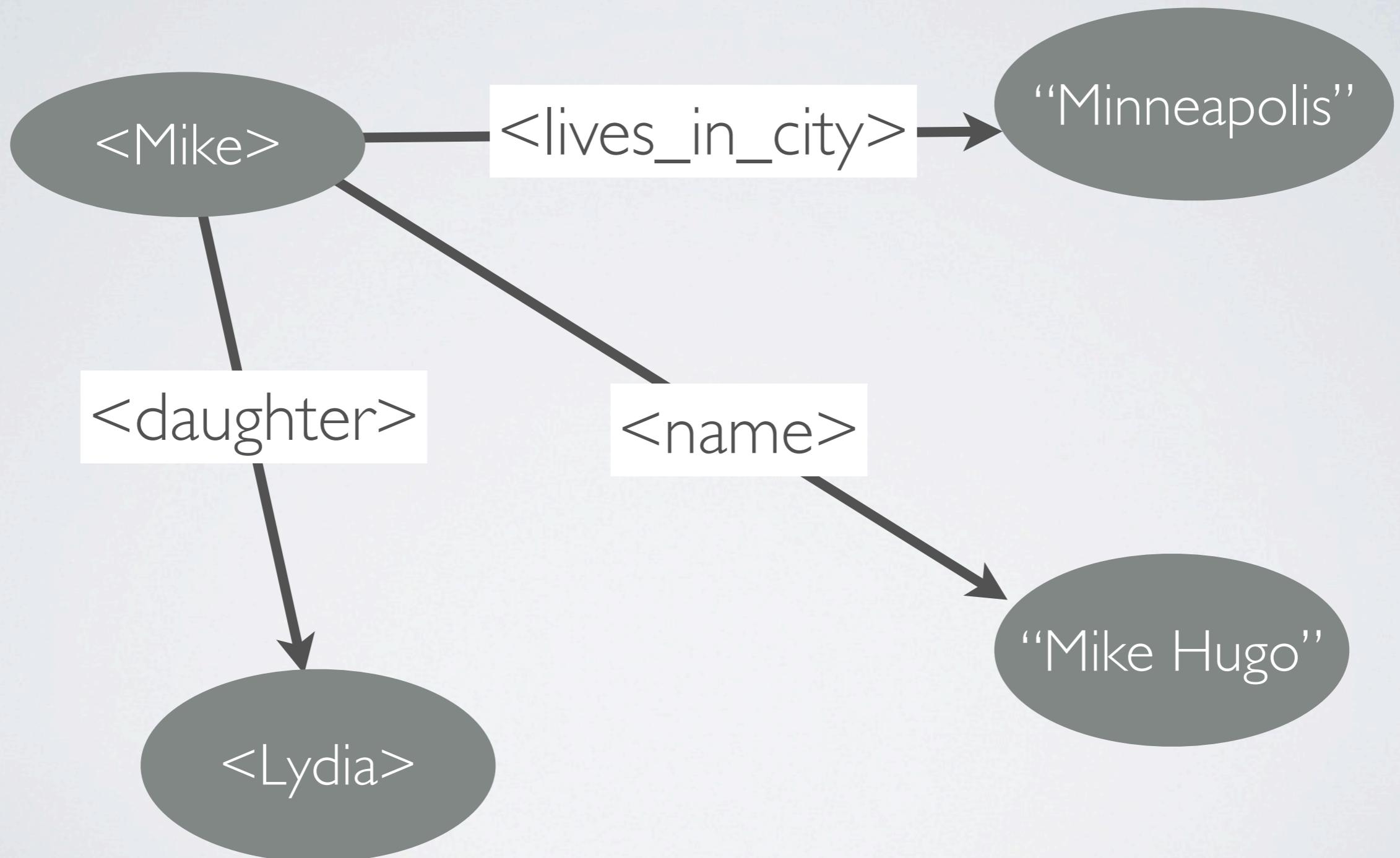
WHAT'S A TRIPLE?



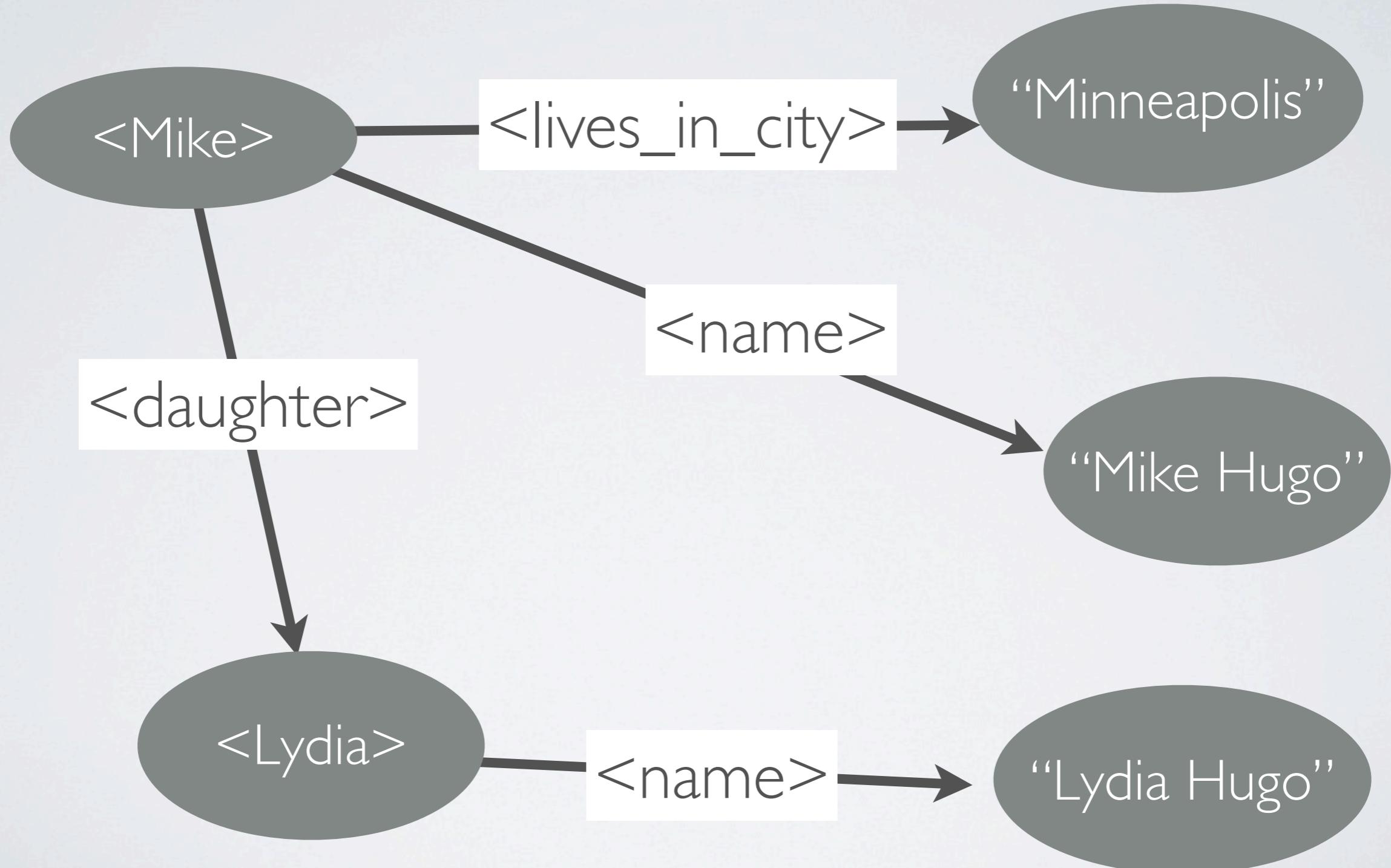
WHAT'S A TRIPLE?

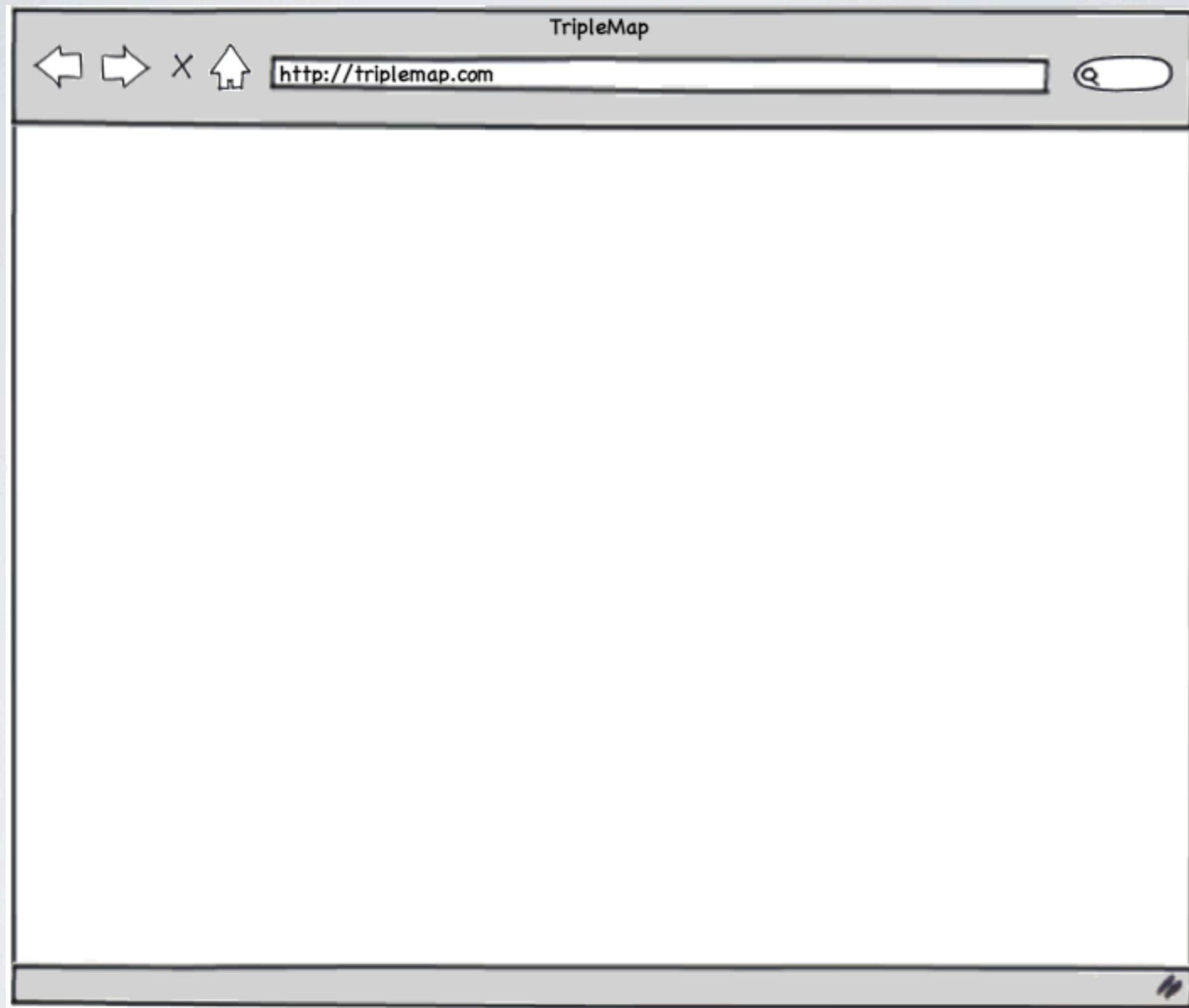


WHAT'S A TRIPLE?



WHAT'S A TRIPLE?





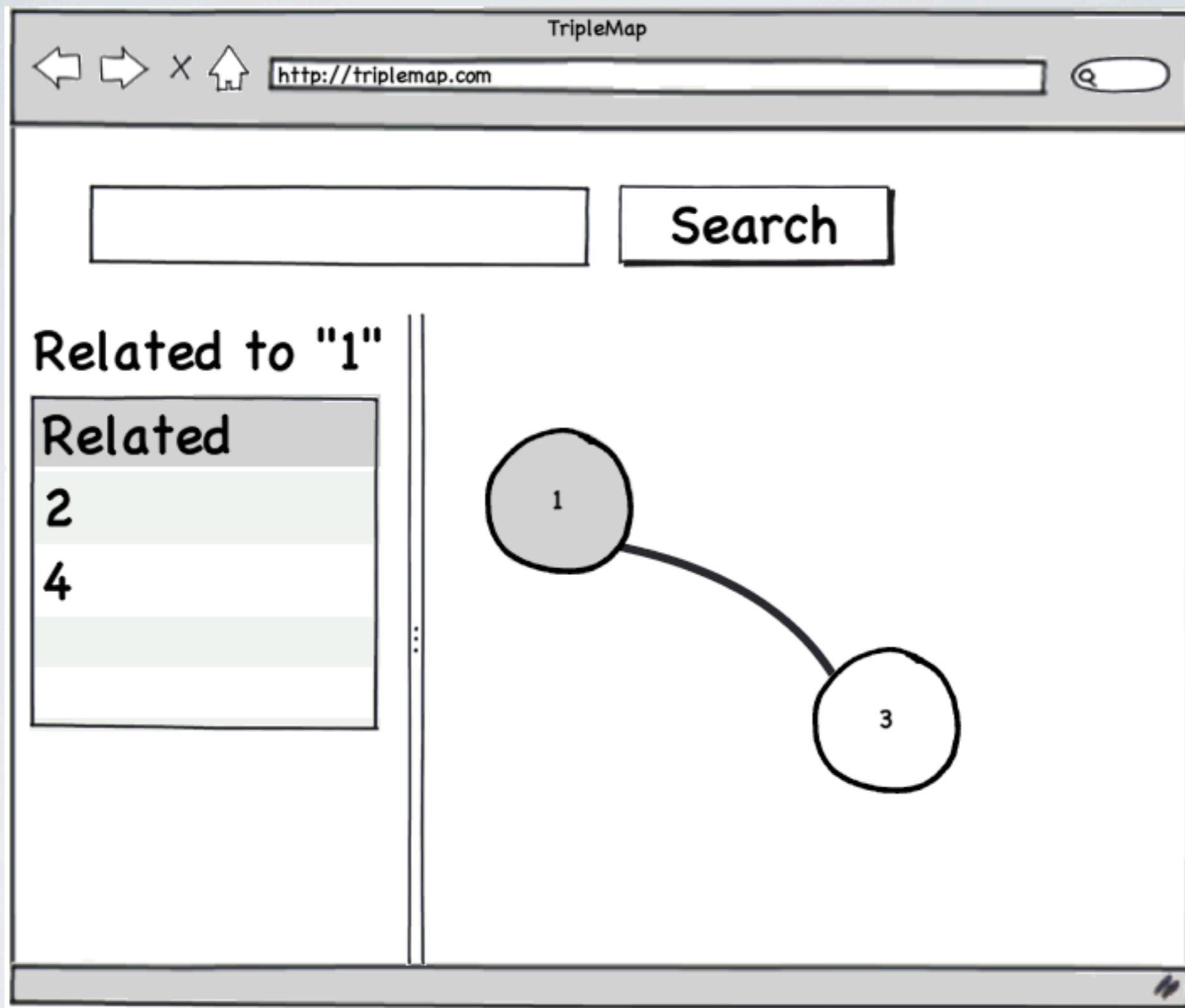
TripleMap

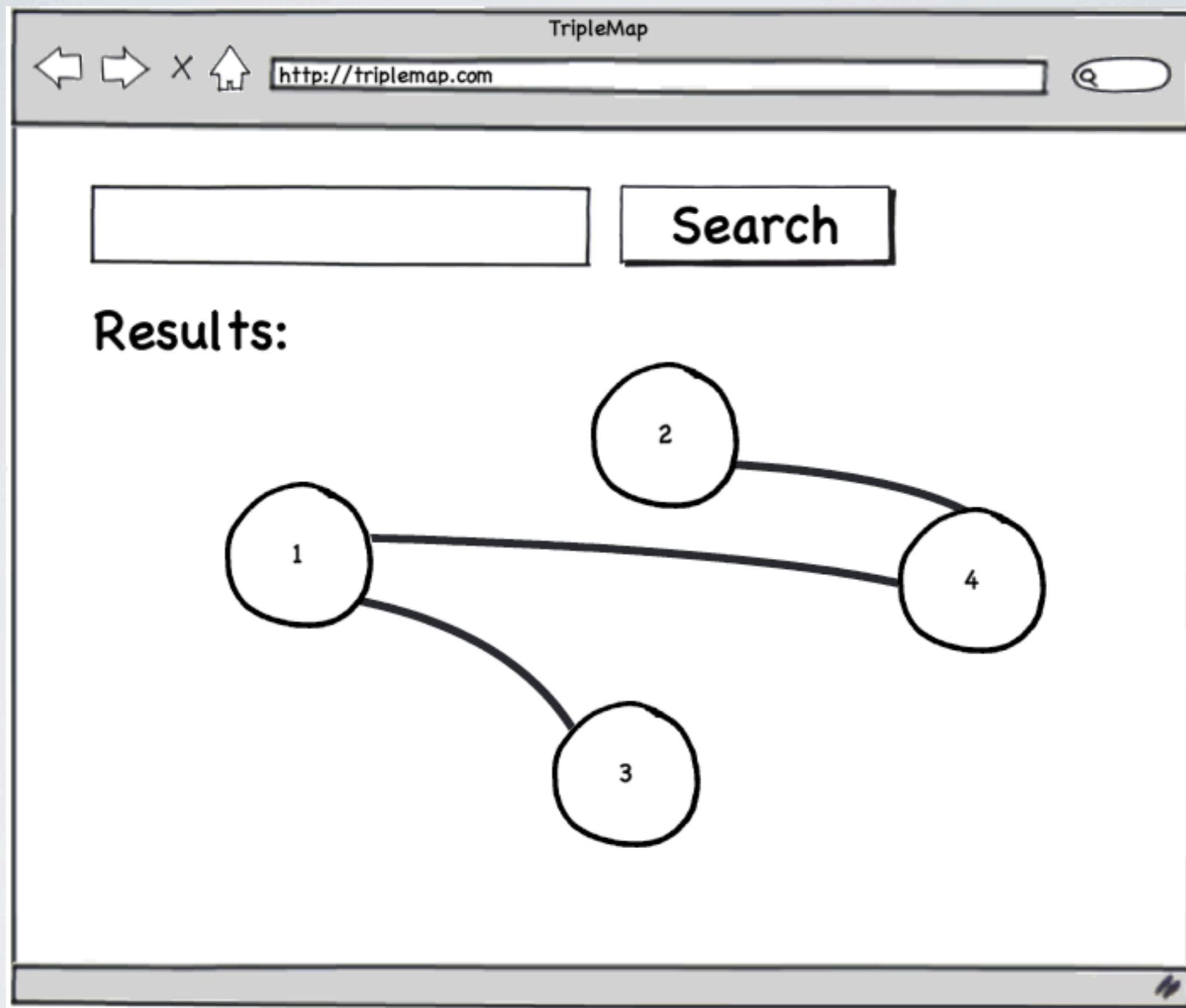
← → X ⌂ http://triplemap.com

Search

Results:

ID	Label
1	Placeholder
2	Placeholder
3	Placeholder
4	Placeholder





Ready, GO!

```
select id, label  
from targets  
where label = '${queryValue}'
```

```
select id, label  
from targets  
where label  
  ilike '%${queryValue}%'
```

```
SELECT ?uri ?type ?label WHERE {  
    ?uri rdfs:label ?label .  
    ?uri rdf:type ?type .  
    FILTER (?label = '${params.query}')  
} LIMIT 10
```

```
SELECT ?uri ?type ?label WHERE {  
  ?uri rdfs:label ?label .  
  ?uri rdf:type ?type .  
  FILTER regex(?label,  
    '\\\\Q${params.query}\\\\E', 'i')  
} LIMIT 10
```

```
SELECT ?uri ?type ?label WHERE {  
    ?uri rdfs:label ?label .  
    ?uri rdf:type ?type .  
    FILTER regex(?label,  
        '\Q${params.query}\E', 'i')  
} LIMIT 10
```

query as literal value

case insensitive

DEMO

Baseline SPARQL Query Performance

FASTER!



ENTAGEN
ACCELERATING INSIGHT



The logo consists of the word "Lucene" in a stylized, italicized font. The letters are a light green color with a dark green outline. The "L" is unique, featuring three curved, upward-pointing strokes above the main body of the letter.



Java API

Indexing and Searching Text

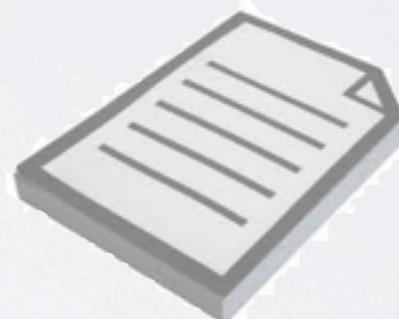


<http://wiki.apache.org/lucene-java/PoweredBy>

indexing

Lucene

storage





Document





Document

field	value
ID	2
name	“Mike Hugo”
company	“Entagen”
bio	“lorem ipsum dolor sum etc...”



Index

field	value
id	2
name	“Mike Hugo”
company	“Entagen”
bio	“lorem ipsum dolor sum etc..”

**Indexed
not
Stored**



Query:

name: mike



Query:

name: mike

Matching
Documents:

field	value
field	value
field	value
field	value
id	2



field	value
id	2





field	value
id	2



Lucene

field	value
id	2



field	value
ID	2
name	“Mike Hugo”
company	“Entagen”
bio	“lorem ipsum dolor sum etc...”

Simplest Solution



ENTAGEN
ACCELERATING INSIGHT

Lucene index of rdfls:label

Build the Index

```
String queryLabels = """
    SELECT ?uri ?label
    WHERE {
        ?uri rdfs:label ?label .
    }
"""
```

**Build a SPARQL
query to find all the
rdfs:label properties**

```
sparqlQueryService.executeForEach(repo)
    def doc = new Document()
    String uri = it.uri.stringValue()
    String label = it.label.stringValue()
```

```
    doc.add(new Field(SUBJECT_URI_FIELD, Field.Store.YES, Field.Indexed))
    doc.add(new Field(LABEL_FIELD, Field.Store.YES, Field.Indexed))
```



```
sparqlQueryService.executeForEach  
(repository, queryLabels) {  
    String uri = it.uri.stringValue()  
    String label = it.label.stringValue()
```

**Execute the
SPARQL query**

```
new Document()  
  
doc.add(new Field(SUBJECT_URI_FIELD,  
                   Field.Store.YES, Field.Indexed)  
        doc.add(new Field(LABEL_FIELD, label,  
                   Field.Store.NO, Field.Indexed)  
  
writer.addDocument(doc)
```

```
String uri = it.uri.stringValue()
String label = it.label.stringValue()
```

```
Document doc = new Document()
doc.add(new Field(SUBJECT_FIELD,
uri,
Field.Store.YES,
Field.Index.ANALYZED),
```

```
doc.add(new Field(LABEL_FIELD,
label,
Field.Store.NO,
Field.Index.ANALYZED))
```

**Instantiate a
new Lucene
Document**

```
Document doc = new Document()  
doc.add(new Field(SUBJECT_URI_FIELD,  
value uri,  
Field.Store.YES,  
Field.Index.ANALYZED))  
doc.add(new Field(LABEL_FIELD,  
label,  
Field.Store.NO  
Field.Index.AN  
writer.addDocument(doc)
```

**Add the Subject
URI to the
Document**

```
Field.Store.YES,  
Field.Index.ANALYZED))  
doc.add(new Field(LABEL_FIELD, key  
value, label,  
Field.Store.NO,  
Field.Index.ANALYZED))
```

```
writer.addDocument
```

**Add the Label field
to the document
(but don't store it)**

```
ly {  
ter.close() // close index
```

```
        Field.Store.NO,  
        Field.Index.ANALYZED) )  
  
writer.addDocument(doc)  
}  
finally {  
    writer.close() //
```

**Add the document
to the Index**

Query the Index

```
query = {  
Query query = new QueryParser(  
    Version.LUCENE_CURRENT,  
    LABEL_FIELD, query this field  
    new StandardAnalyzer())  
    .parse(params.query);
```

for this value

```
def s time  
List time  
def e time  
  
render(view: 'index', model: [results:  
    Create a Lucene  
    Query from user  
    input
```

```
IndexSearcher searcher = luceneSearcher
ScoreDoc[ ] scoreDocs =
    searcher.search(query, 10).scoreDocs
List results = []
def connection = r
scoreDocs.each {
    Document doc =
        String uri = doc[SUBJECT_URI_FIELD]
        Map labelAndType = sparqlQueryService
            results << [uri: uri, type: labelAndType]
}
connection.close()
return results
```

**Search the index
(limit 10) for
matching
documents**

```
list results = []  
def connection = repository.connection  
scoreDocs.each {  
    Document doc = searcher.doc(it.doc)  
    String uri = doc[SUBJECT_URI_FIELD]  
    Map labelAndType =  
        [spans[i].label, spans[i].type]  
    results.add([  
        type: labelAndType.type,  
        label: labelAndType.label])  
}  
connection.close()  
return results
```

For each matching document, get the doc and extract the Subject URI

```
list results = []  
def connection = repository.connection  
scoreDocs.each {  
    Document doc = searcher.doc(it.doc)  
    String uri = doc[SUBJECT_URI_FIELD]  
    Map labelAndType =  
        sparqlQueryService.  
            getLabelAndType(uri, connection)  
    results.add([  
        uri: uri,  
        type: labelAndType[type],  
        label: labelAndType[label]] )  
}  
connection.close()  
return results
```

**Using the Subject
URI, load properties
from the triplestore])**

```
list results = []  
def connection = repository.connection  
scoreDocs.each {  
    Document doc = scoreDocs[i].getDocument()  
    String uri = doc.get("uri")  
    Map labelAndType = doc.get("labelAndType")  
    sparqlQueryService.  
        getLabelAndType(uri, connection)  
    results.add([  
        uri: uri,  
        type: labelAndType.type,  
        label: labelAndType.label])  
}  
connection.close()  
return results
```

**return results
containing Subject
URI, Type, and Label**

DEMO

Lucene Index of Searchable Labels

WHAT ABOUT ENTITY RELATIONSHIPS?

WHAT ABOUT OTHER PROPERTIES?

SIREn

{ *Semantic Information Retrieval Engine* }

Lucene Extension

Indexing and Searching Semi-Structured Data



ENTAGEN
ACCELERATING INSIGHT

SIREn

{ *Semantic Information Retrieval Engine* }

Document

Document

field	value
URI	<DB00619>
triples	<DB00619> rdfs:label "Imatinib" . <DB00619> rdf:type <drugbank:drugs> . <DB00619> drugbank:brandName "Gleevec" . <DB00619> drugbank:target <targets/1588> .

Build the Index

```
Connection connection = repository.getConnection();
String subjectUris = """
    SELECT distinct ?uri
    WHERE {
        ?uri ?p ?o .
    }
"""
sparqlQueryService.executeForEach(repository, new SparqlQueryService.QueryHandler() {
    @Override
    public void handle(SparqlQueryResult result) {
        def doc = new DocumentBuilder().startDocument();
        String subjectUri = result.getResults().get(0).get("uri");
        doc.add(new Field(SUBJECT_URI_FIELD_NAME, subjectUri, Field.Store.YES));
        doc.add(new Field(SUBJECT_URI_FIELD_NAME, subjectUri, Field.Store.NO));
    }
});
```

**Select all Subject
URIs from the
triplestore**

```
"""
sparqlQueryService.executeForEach(
    repository, subjectUris) {
    def doc = new Document()

    String subjectUri = it.uri.stringValue
    doc.add(new Field(SUBJECT_URI_FIELD,
                      subjectUri,
                      Field.Store.YES,
                      Field.Index.NO))

    StringWriter triplesStringWriter =
    NTriplesWriter nTriplesWriter =
        new NTriplesWriter(triplesStringWriter)
    connection.exportStatements(doc, nTriplesWriter)
}
```

**Execute the Sparql Query
For each URI, create a
new Document**

```
def doc = new Document()
```

```
String subjectUri = it.uri.stringValue  
doc.add(new Field(SUBJECT_URI_FIELD,  
subjectUri,  
Field.Store.YES,  
Field.Index.ANALYZED))
```

```
StringWriter triplesStringWriter = new  
NTriplesWriter nT  
new NTriplesW  
connection.exportStatements(  
new URIImpl(subjectUri),  
null, null, false,
```

**Add the Subject URI
to the Document**

```
Field.Index.ANALYZED) )
```

```
StringWriter triplesStringWriter = new  
NTriplesWriter nTriplesWriter =  
new NTriplesWriter(triplesStringWriter)  
connection.exportStatements(  
new URIImpl(subjectUri),  
null, null, false,  
nTriplesWriter)
```

```
doc.add(new Field(T  
triplesStri  
Field.Store  
Field.Index.ANALYZED) )
```

**Get an NTriples
string from the
triplestore**

```
new UriTemplate(subjectURI),  
null, null, false,  
nTriplesWriter)
```

```
doc.add(new Field(TRIPLES_FIELD,  
triplesStringWriter.toString()  
Field.Store.NO,  
Field.Index.ANALYZED))
```

```
writer.addDocument(doc);
```

**Add the NTriples
string to the
document**

```
doc.add(new Field(TRIPLES_FIELD,  
triplesStringWriter.toString(  
Field.Store.NO,  
Field.Index.ANALYZED))
```

```
writer.addDocument(doc)
```

**Add the document
to the index**

Query the Index

```
SirenCellQuery predicate =  
  new SirenCellQuery(  
    new SirenTermQuery(  
      new Term(TRIPLES_FIELD,  
        RDFS.LABEL.stringValue()))));  
predicate.constraint = PREDICATE_CELL
```

```
SirenCellQuery object =  
  new SirenCellQuery(  
    new SirenTermQuery(  
      new Term(TRIPLES_FIELD,  
        params.query.toLowerCase())));  
object.constraint = OBJECT_CELL
```

query the Triples
field

```
SirenCellQuery predicate =  
  new SirenCellQuery(  
    new SirenTermQuery(  
      new Term(TRIPLES_FIELD,  
        RDFS.LABEL.stringValue()))));  
predicate.constraint = PREDICATE_CELL
```

```
SirenCellQuery object =  
  new SirenCellQuery(  
    new SirenTermQuery(  
      new Term(TRIPLES_FIELD,  
        params.query.toLowerCase())));  
object.constraint = OBJECT_CELL
```

```
SirenCellQuery predicate =  
  new SirenCellQuery(  
    new SirenTermQuery(  
      new Term(TRIPLES_FIELD,  
        RDFS.LABEL.stringValue()))));  
predicate.constraint = PREDICATE_CELL
```

SirenCellQuery object

```
  new SirenCellQuery(  
    new SirenTermQuery(  
      new Term(TRIPLES_FIELD,  
        Case()  
        * note: could be any predicate!  
object.constraint = OBJECT_CELL
```

```
SirenCellQuery object =  
  new SirenCellQuery(  
    new SirenTermQuery(  
      new Term(TRIPLES_FIELD,  
        params.query.toLowerCase())))  
object.constraint = OBJECT_CELL
```

```
Query query = new SirenTupleQuery()  
query.add(predicate,  
          SirenTupleClause.  
query.add(object,  
          SirenTupleClause.Occur.MUST)
```

**query the Triples
field**

```
SirenCellQuery object =  
  new SirenCellQuery(  
    new SirenTermQuery(  
      new Term(TRIPLES_FIELD,  
               params.query.toLowerCase())))  
object.constraint = OBJECT_CELL
```

```
Query query = new SirenTupleQuery()  
query.add(predicate,  
          SirenTupleClause.Occur.MUST)  
query.add(object,  
          SirenTupleClause.Occur.MUST)
```

for an object

List results = executeQuery(query)

```
SirenCellQuery object =  
  new SirenCellQuery(  
    new SirenTermQuery(  
      new Term(TRIPLES_FIELD,  
        params.query.toLowerCase())))  
object.constraint = OBJECT_CELL
```

```
Query query = new SirenTupleQuery()  
query.add(predicate,  
          SirenTupleClause.Term.  
query.add(object,  
          SirenTupleClause.Occur.MUST)
```

**matching the
user input**



field	value
URI	<DB00619>
triples	<DB00619> rdfs:label "Imatinib" . <DB00619> rdf:type <drugbank:drugs> . <DB00619> drugbank:brandName "Gleevec" . <DB00619> drugbank:target <targets/1588> .

Query: **“imatinib”**

field	value
URI	<DB00619>
triples	<DB00619> rdfs:label "Imatinib". <DB00619> rdf:type <drugbank:drugs> . <DB00619> drugbank:brandName "Gleevec" . <DB00619> drugbank:target <targets/1588> .

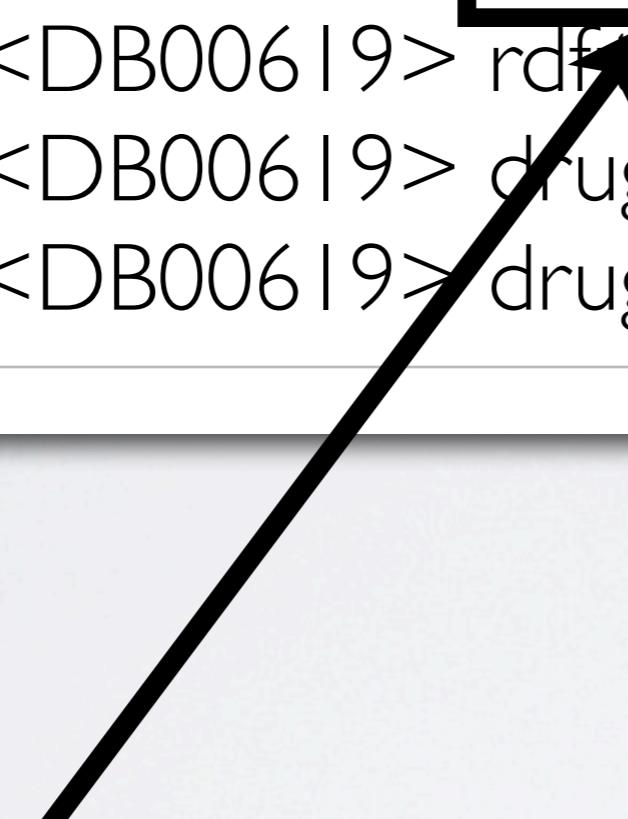
Query:

triples field

field	value
URI	<DB00619>
triples	<DB00619> rdfs:label "Imatinib". <DB00619> rdf:type <drugbank:drugs> . <DB00619> drugbank:brandName "Gleevec" . <DB00619> drugbank:target <targets/1588> .

Query:

predicate = rdfs:label



field	value
URI	<DB00619>
triples	<DB00619> rdfs:label "Imatinib". <DB00619> rdf:type <drugbank:drugs> . <DB00619> drugbank:brandName "Gleevec" . <DB00619> drugbank:target <targets/1588> .

Query:

predicate = rdfs:label

object = “imatinib”

```
List executeQuery(Query query) {  
    IndexSearcher searcher = sirenSearcherM...  
    ScoreDoc[ ] scoreDocs =  
        searcher.search(query, 10).scoreDocs;  
    List results = [ ];  
def connection = reposi...  
    scoreDocs.each {  
        Document doc = sea...  
        String uri = doc[S...  
        Map labelAndType = sparqlQueryServic...  
            getLabelAndType(uri, connection);  
        results.add([  
            uri: uri,  
            type: labelAndType.type,  
            label: labelAndType.label,...  
    }  
}
```

**Search the index
(limit 10) for
matching
documents**

```
list results = []
def connection = repository.connection
scoreDocs.each {
    Document doc = searcher.doc(it.doc)
    String uri = doc[ SUBJECT_URI_FIELD ]
    Map labelAndType = sparqlQueryService
        .getLabelAndType(uri)
    results.add( new Type( type.type,
        label: labelAndType.label ) )
}
connection.close()
return results
```

For each matching document, get the doc and extract the Subject URI

```
connection = repository.connection
reDocs.each {
    Document doc = searcher.doc(it.doc)
    String uri = doc[SUBJECT_URI_FIELD]
    Map labelAndType = sparqlQueryService.
        getLabelAndType(uri, connection)
    results.add([
        uri: uri,
        type: lab
        label: la
    ])
}
connection.close()
return results
```

**Using the Subject
URI, load properties
from the triplestore**

```
String uri = doc[SUBJECT_URI_FIELD]
Map labelAndType = sparqlQueryService.
    getLabelAndType(uri, connection)
results.add([
    uri: uri,
    type: labelAndType.type,
    label: labelAndType.label])
```

```
connection.close()
return results
```

**return results
containing Subject
URI, Type, and Label**

DEMO

SIREn Index of RDF Entities

FLEXIBILITY



field	value
URI	<DB00619>
triples	<DB00619> rdfs:label "Imatinib". <DB00619> rdf:type <drugbank:drugs> . <DB00619> drugbank:brandName "Gleevec" . <DB00619> drugbank:target <targets/1588> .

Query:

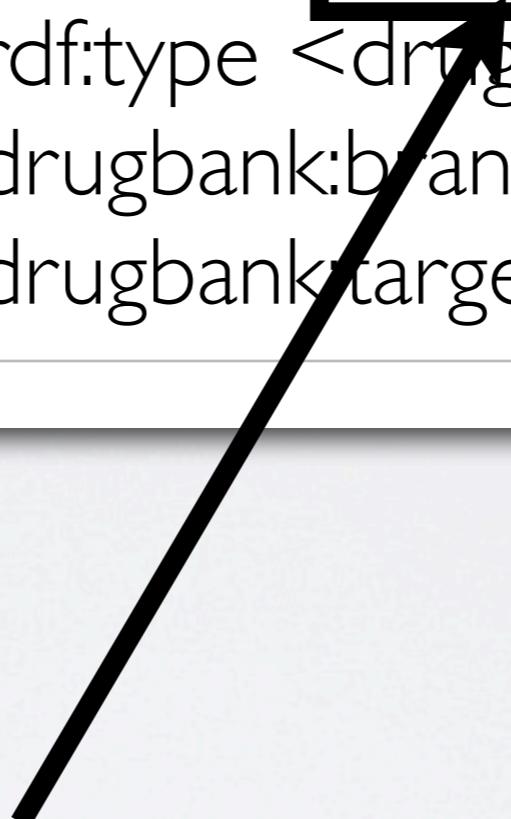
predicate = rdfs:label

object = “imatinib”

field	value
URI	<DB00619>
triples	<DB00619> rdfs:label "Imatinib". <DB00619> rdf:type <drugbank:drugs> . <DB00619> drugbank:brandName "Gleevec" . <DB00619> drugbank:target <targets/1588> .

Query:

object = “imatinib”



field	value
URI	<DB00619>
triples	<p><DB00619> rdfs:label "Imatinib".</p> <p><DB00619> rdf:type <drugbank:drugs> .</p> <p><DB00619> drugbank:brandName "Gleevec"</p> <p><DB00619> drugbank:target <targets/1588> .</p>

Query:

object = “imatinib”

OR

object = “gleevec”

MORE THAN LITERALS



ENTAGEN
ACCELERATING INSIGHT

field	value
URI	<DB00619>
triples	<DB00619> rdfs:label "Imatinib". <DB00619> rdf:type <drugbank:drugs> . <DB00619> drugbank:brandName "Gleevec" . <DB00619> drugbank:target <targets/1588> .

Query:

predicate = brandName



field	value
URI	<DB00619>
triples	<DB00619> rdfs:label "Imatinib". <DB00619> rdf:type <drugbank:drugs> . <DB00619> drugbank:brandName "Gleevec" . <DB00619> drugbank:target <targets/1588> .

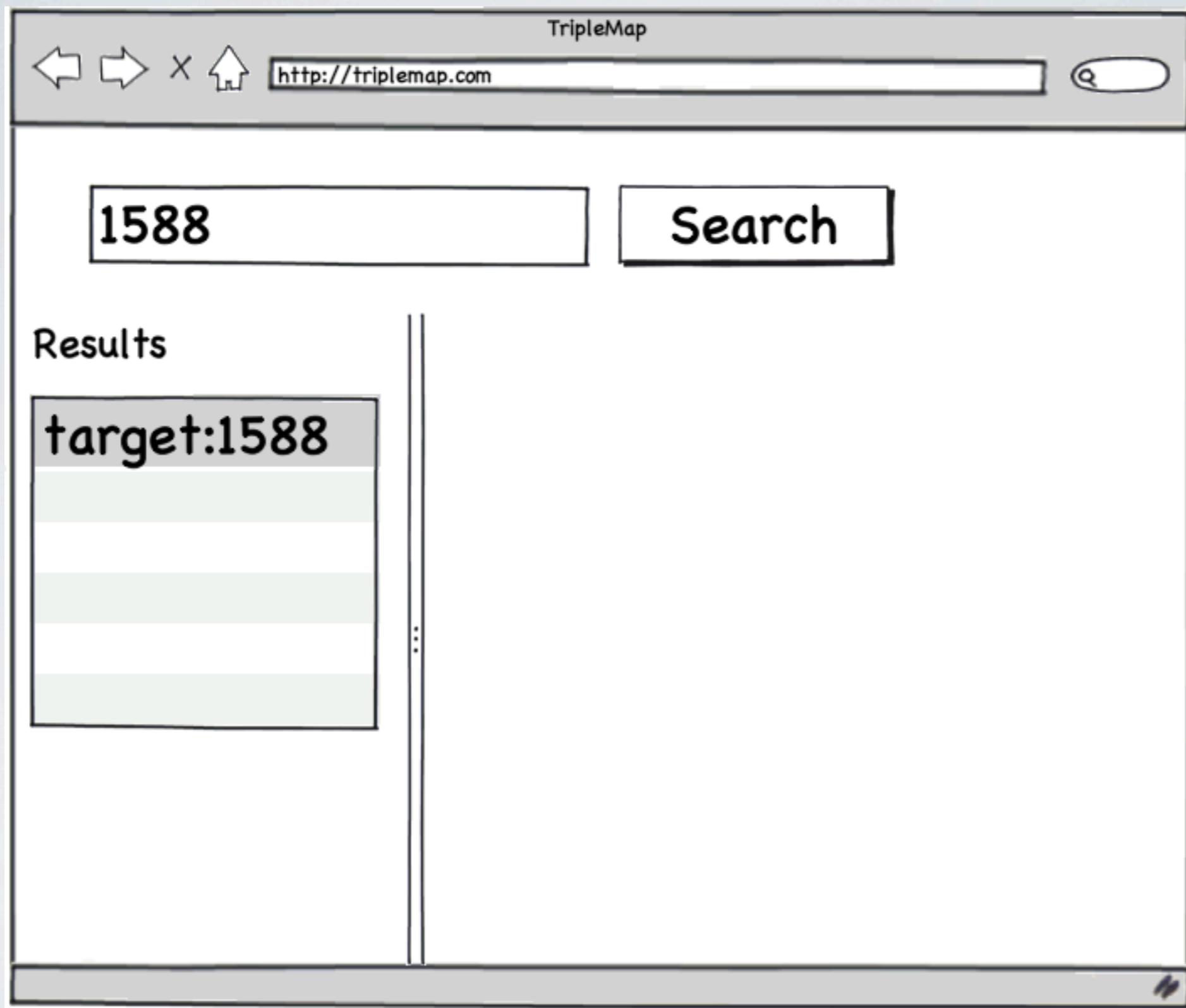
Query:

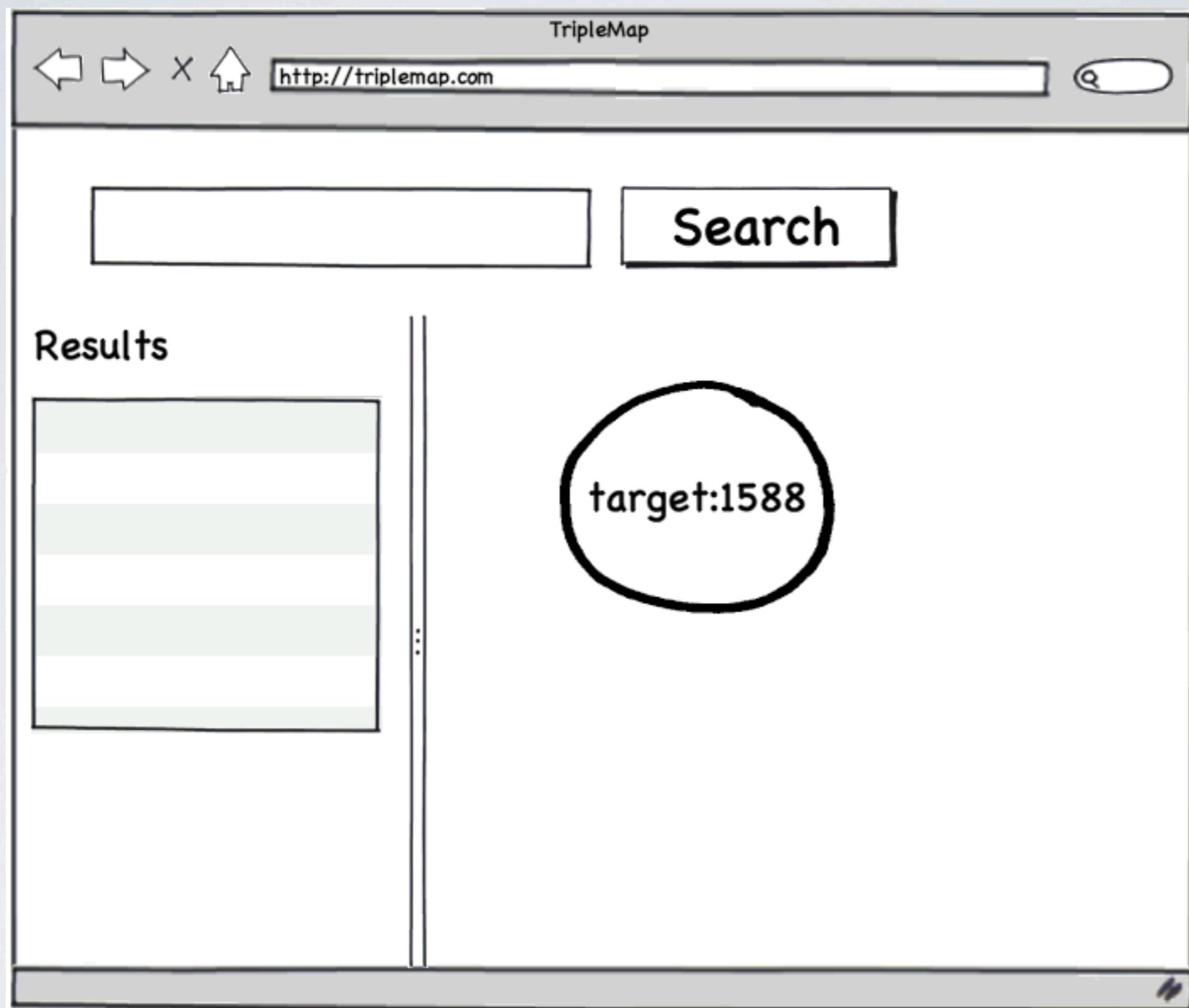
predicate = target

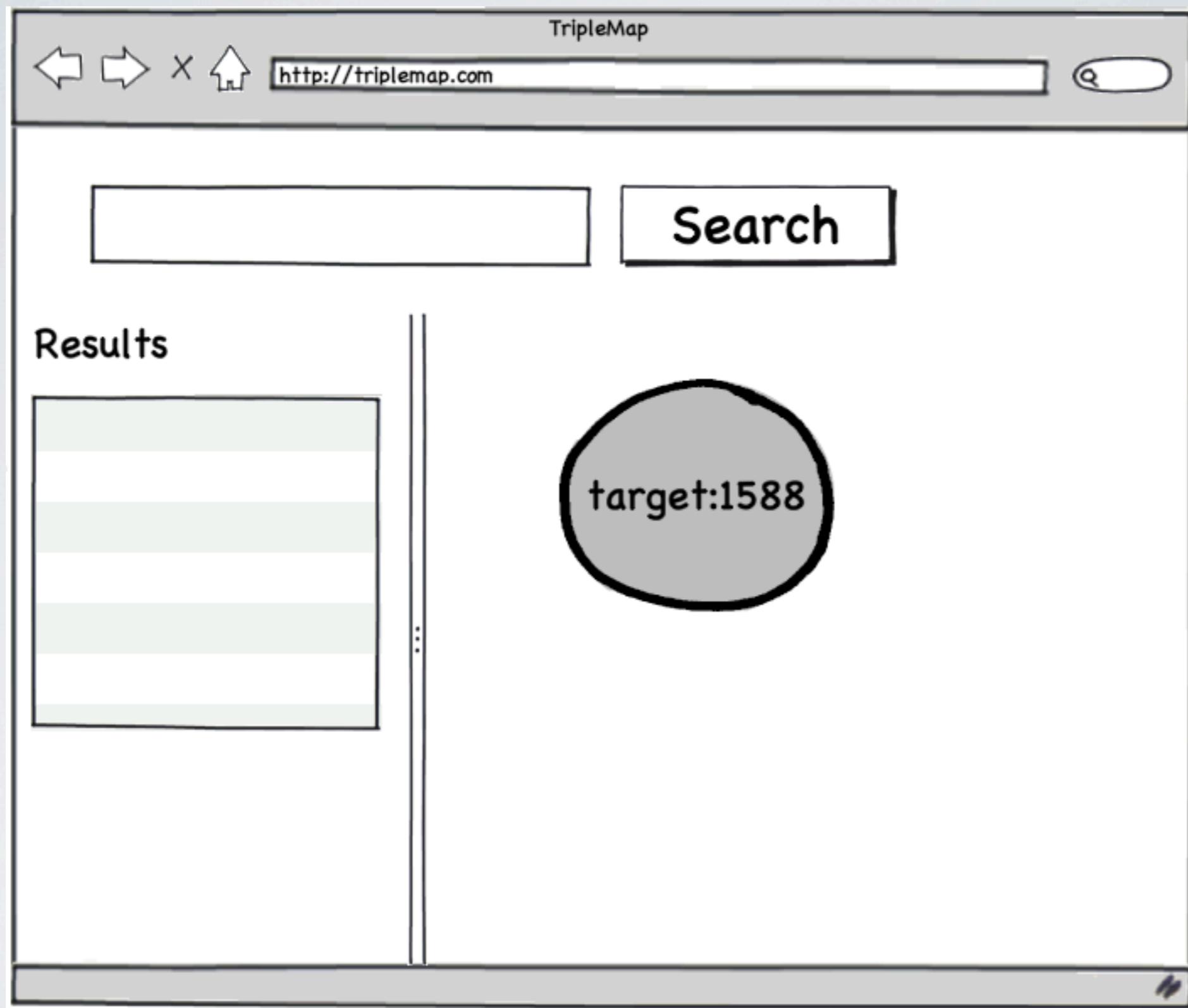


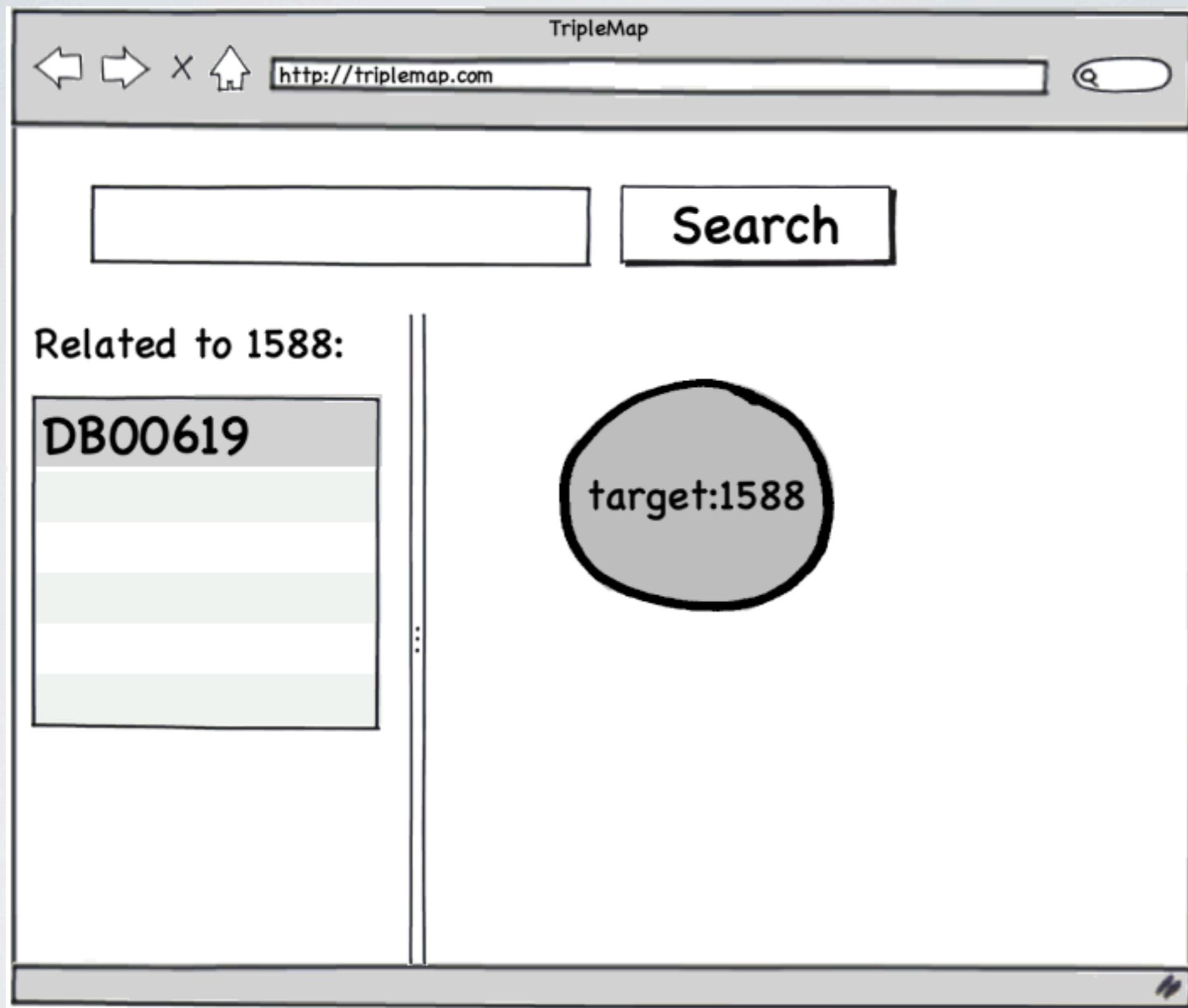
RELATIONSHIPS





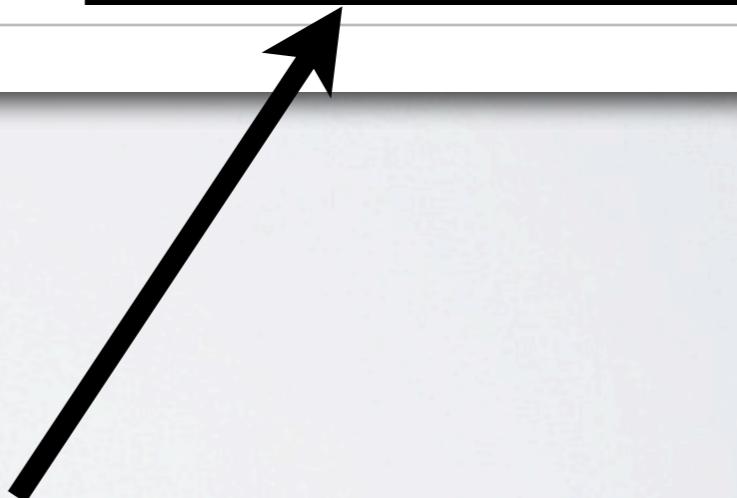






field	value
URI	<DB00619>
triples	<DB00619> rdfs:label "Imatinib". <DB00619> rdf:type <drugbank:drugs> . <DB00619> drugbank:brandName "Gleevec" <DB00619> drugbank:target <targets/1588> .

Query:



object = <targets/1588>

DEMO

Searching SIREn Index for Relationships

SIREn

{ Semantic Information Retrieval Engine }



Distributed Indexing and Searching Semi-Structured Data





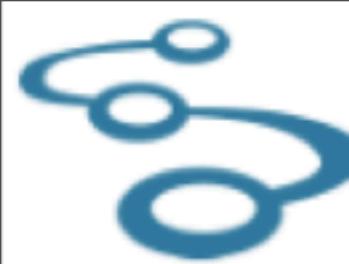
Replication











Search interface type: [Simple](#) [Advanced](#) [Guru](#) [Query Language](#) [Documentation](#)

keyword(s)

ntriple(s)

```
* <name> "Tim Berners Lee"
```

filter(s)

SEARCH

Group By Dataset:

Sorted by:

relevance



Quick filters ([All options](#))

Time range:

[Any date](#)

[Today](#) [Yesterday](#) [Last week](#)

[Last month](#) [Last year](#)

Format:

[Any format](#)

[RDF](#) [RDFA](#) [MICROFORMAT](#) [XFN](#)

[HCARD](#) [HCALENDAR](#) [HLISTING](#)

[HRESUME](#) [LICENSE](#) [GEO](#) [ADR](#)

Sindice search:* <name> "Tim Berners Lee" found 699 documents

<http://dig.csail.mit.edu/2007/01/camp/data> (RDF)

2009-08-12 - 387 triples in 43.2 kB

<http://dig.csail.mit.edu/2007/01/camp/data> ([Search](#)) [Inspect](#): ([Cache](#)) ([Live](#))

http://dbpedia.org/data/Tim_Berners-Lee (RDF)

2011-05-18 - 171 triples in 38.8 kB

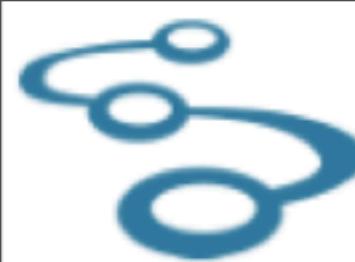
SIGMA

FULL CONTENT

GRAPH

TRIPLES(890)

See all we know about "Tim Berners Lee"



Search interface type: [Simple](#) [Advanced](#) [Guru](#) [Query Language Documentation](#)

keyword(s)

ntriple(s)

* <name> "Tim Berners Lee"

400 Million Documents > 12 Billion Triples

Quick filters (All options)

Sindice search: * <name> "Tim Berners Lee"

Lee" found 699 documents

Time range:

[Any date](#)

[Today](#) [Yesterday](#) [Last week](#)

[Last month](#) [Last year](#)

Format:

[Any format](#)

[RDF](#) [RDFA](#) [MICROFORMAT](#) [XFN](#)

[HCARD](#) [HCALENDAR](#) [HLISTING](#)

[HRESUME](#) [LICENSE](#) [GEO](#) [ADR](#)

<http://dig.csail.mit.edu/2007/01/camp/data> (RDF)

+ 2009-08-12 - 387 triples in 43.2 kB

<http://dig.csail.mit.edu/2007/01/camp/data> ([Search](#)) [Inspect](#): ([Cache](#)) ([Live](#))

http://dbpedia.org/data/Tim_Berners-Lee (RDF)

- 2011-05-18 - 171 triples in 38.8 kB

SIGMA

FULL CONTENT

GRAPH

TRIPLES(890)

See all we know about "Tim Berners Lee"

Query Parser



indice
THE SEMANTIC WEB INDEX

[Home](#)[About](#)[Se](#)

Search interface type: [Simple](#) [Advanced](#) [Guru](#) [Query Language Documentation](#)

keyword(s)

ntriple(s)

```
* <name> "Tim Berners Lee"
```

filter(s)

SEARCH

Group By Dataset:

Sorted by:

relevance



Quick filters ([All options](#))

Time range:

[Any date](#)

[Today](#) [Yesterday](#) [Last week](#)

[Last month](#) [Last year](#)

Format:

Sindice search: * <name> "Tim Berners Lee" found 699 documents

<http://dig.csail.mit.edu/2007/01/camp/data> (RDF)

+ 2009-08-12 - 387 triples in 43.2 kB

<http://dig.csail.mit.edu/2007/01/camp/data> ([Search](#)) Inspect: ([Cache](#)) ([Live](#))

http://dbpedia.org/data/Tim_Berners-Lee (RDF)

- 2011-05-18 - 171 triples in 38.8 kB

Interface type: [Simple](#) [Advanced](#) [Guru](#) [Query Language Documentation](#)

keyword(s)

ntriple(s)

filter(s)

subject

options)

* <name> "Tim Berners Lee"

SEARCH

Group By Dataset.

Sorted by:

relevance

Sindice search:* <name> "Tim Berners Lee" found
predicate **object**

<http://dig.csail.mit.edu/2007/01/camp/data>

 2009-08-12 - 387 triples in 43.2 kB

<http://dig.csail.mit.edu/2007/01/camp/data> ([Search](#)) [Inspect](#)

https://dbpedia.org/data/Tim_Berners_Lee ([View](#))

DEMO

SIREn in action on TripleMap.com

Search Maps Analyze

Search...

Search

Filter

Sort

Select - Total count: 2

< Back Indication



- + Colchicine resistance
- + Colchicine_resistance



18



144



2



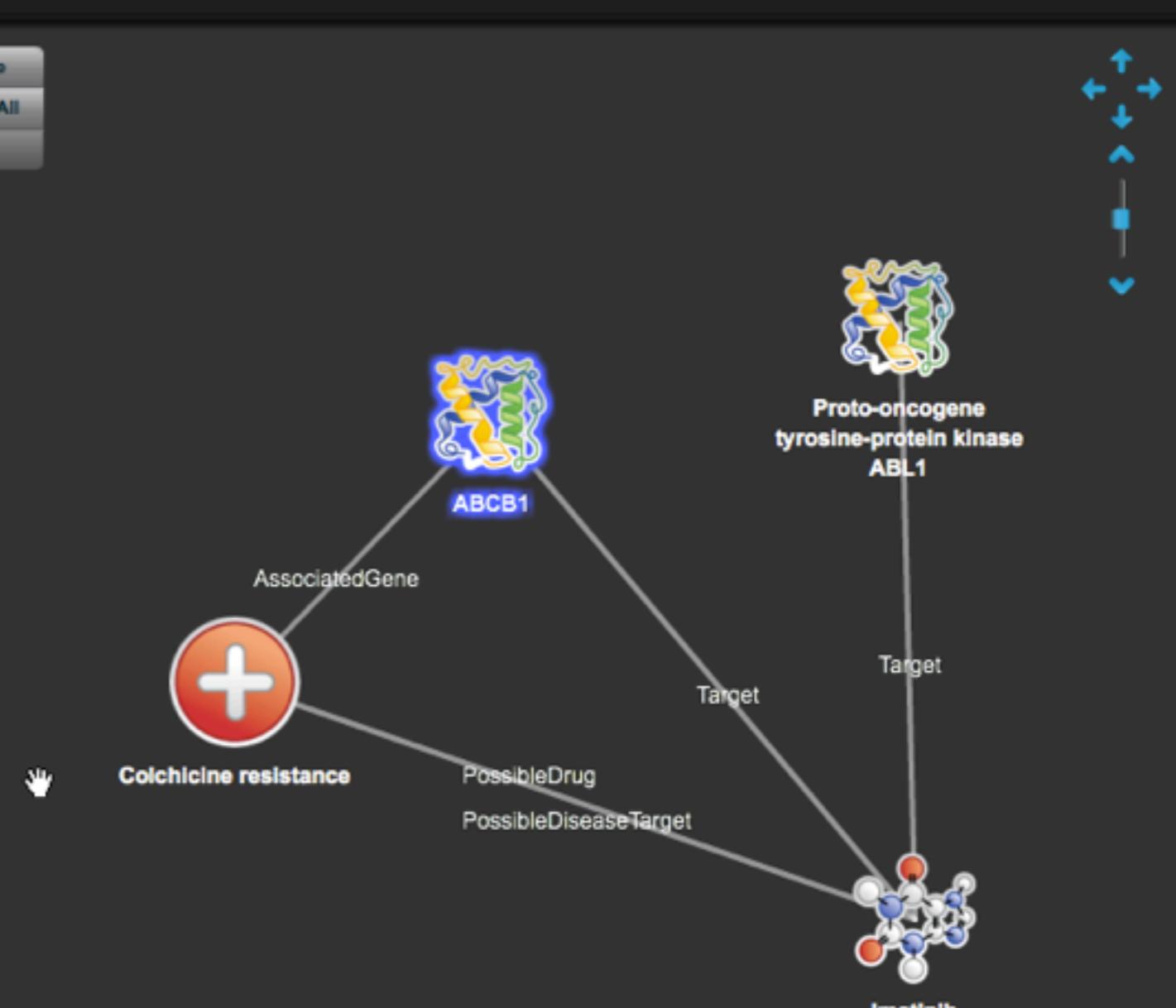
16

Map View

Remove

Remove All

Group



SPARQL

LUCENE

SIREN

TripleMap.com



QUESTIONS?

mike@entagen.com / twitter: @piragua



ENTAGEN
ACCELERATING INSIGHT

<http://www.entagen.com>



TripleMap

<http://www.triplemap.com>



ENTAGEN
ACCELERATING INSIGHT