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

AGENDA

- The problem
- RDF / NTriples overview
- Lucene
- SIREn
- Searching RDF with SIREn
- Questions

THE PROBLEM

- Build a web application utilizing RDF data that can:
  - Provide responsive search results and/or "auto-complete" or "type-ahead" functionality
  - Query literals in a case in-sensitive manner
  - Find relationships between entities (and counts)
  - Scale well over millions of triples
  - Provide a responsive user interface

WHAT'S A TRIPLE?

subject

predicate

object

ENTAGEN
ACCELERATING INSIGHT

WHAT'S A TRIPLE?

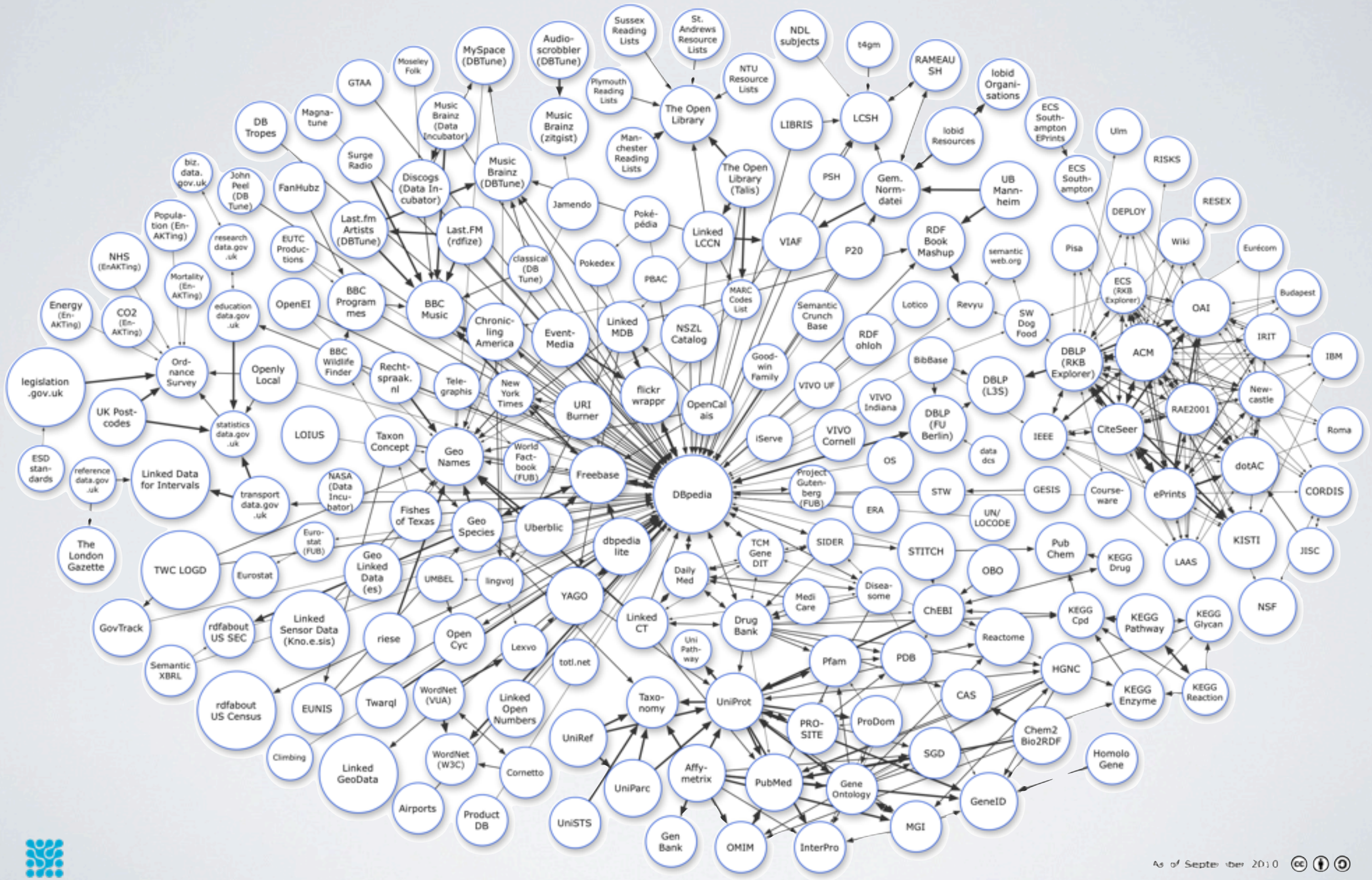
subject "Imatinib"

predicate type

object Pharmaceutical Drug

ENTAGEN
ACCELERATING INSIGHT

LINKING OPEN DATA

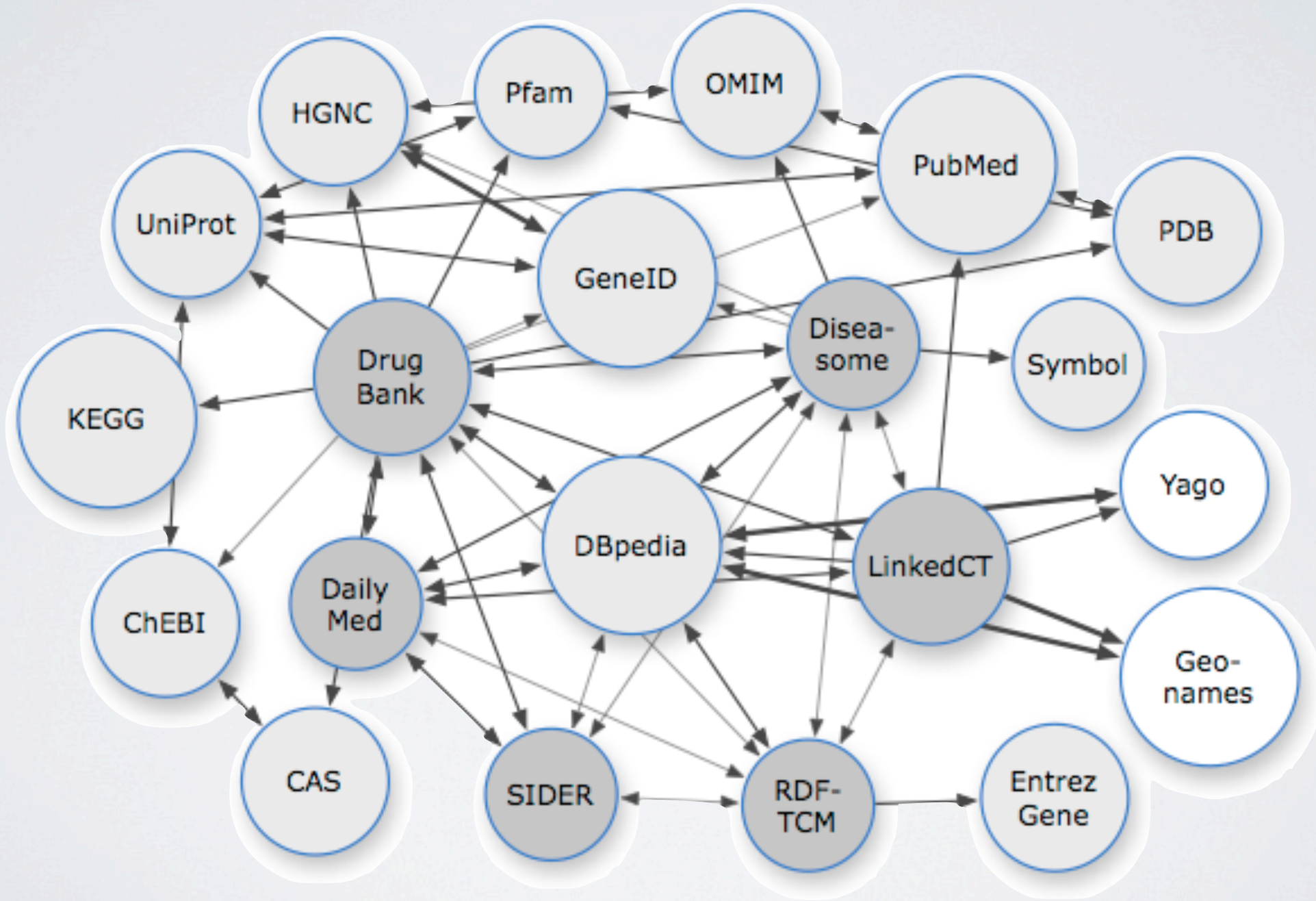


ENTAGEN
ACCELERATING INSIGHT

As of September 2010

LINKING OPEN **DRUG** DATA

http://esw.w3.org/HCLSIG/LODD



ENTAGEN
ACCELERATING INSIGHT

WHAT'S A TRIPLE?

http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs/DB00619

subject	**predicate**	**object**
DB00619	label	"**Imatinib**"
DB00619	type	drugbank:drugs
DB00619	brandName	"Gleevec"
DB00619	drugbank:target	targets:17

ENTAGEN
ACCELERATING INSIGHT

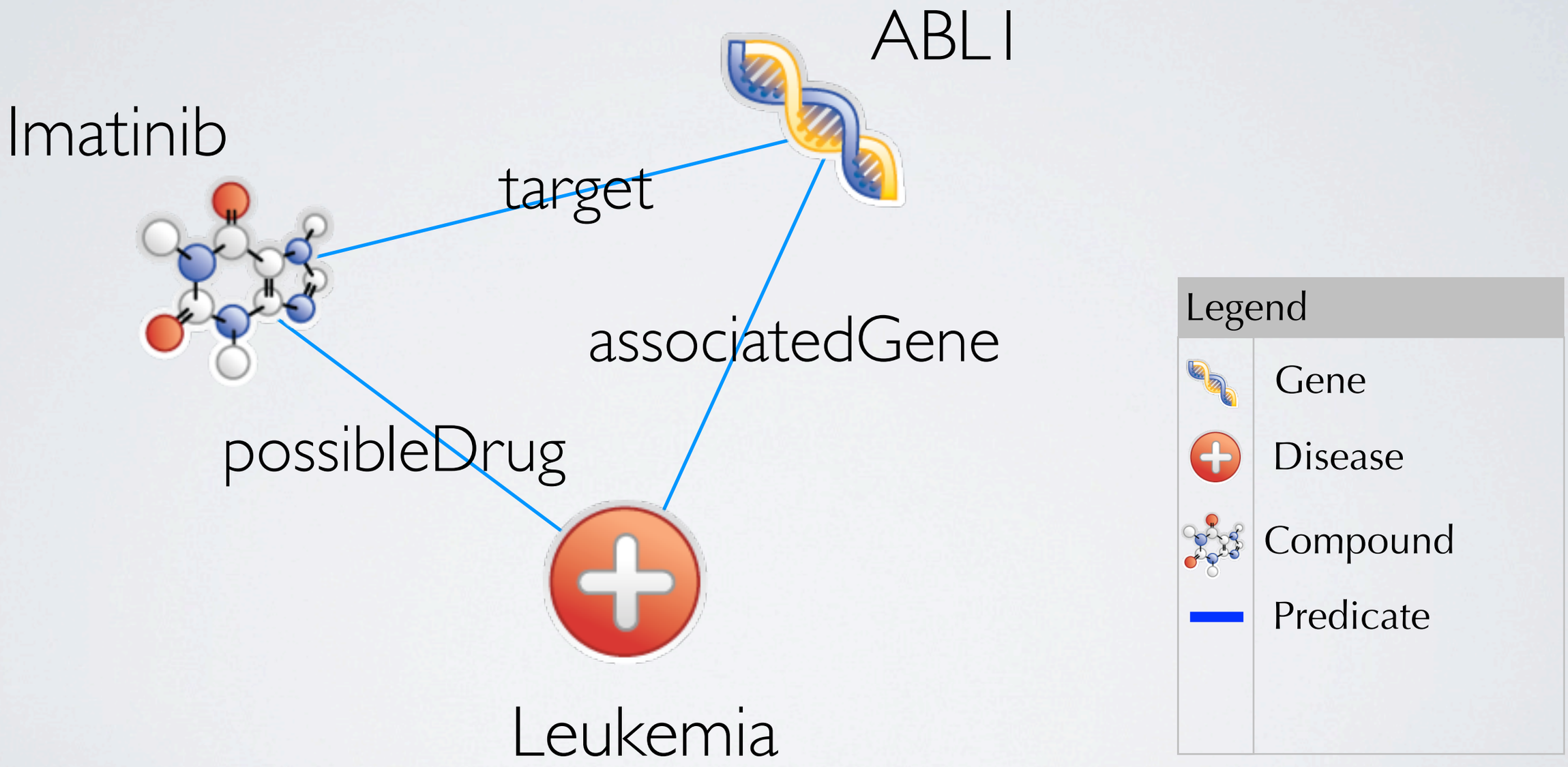
WHAT'S A TRIPLE?

http://www4.wiwiss.fu-berlin.de/drugbank/resource/targets/17

subject	**predicate**	**object**
DB00619	drugbank:target	targets/17
targets/17	type	drugbank:drugs
targets/17	label	"Proto-oncogene tyrosine-protein kinase ABL1"
targets/17	geneName	"ABL1"

ENTAGEN
ACCELERATING INSIGHT

RELATIONSHIPS



ENTAGEN
ACCELERATING INSIGHT

FAST SEARCHING

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

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

```
SELECT ?uri ?label WHERE {
  ?uri rdfs:label ?label .
  ?uri rdf:type drugbank:targets .
  FILTER regex(?label, '\\\\Q${queryValue}\\\\E', 'i')"
}
```

query as literal value

case insensitive

ENTAGEN
ACCELERATING INSIGHT

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

```
SELECT ?uri ?label WHERE {
  ?uri rdfs:label ?label .
  ?uri rdf:type drugbank:targets .
  FILTER regex(?label, '\\\\Q${queryValue}\\\\E', 'i')"
}
```

multiple datasets

ENTAGEN
ACCELERATING INSIGHT

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

```
SELECT ?uri ?label WHERE {
  ?uri rdfs:label ?label .
  ?uri rdf:type drugbank:targets .
  FILTER regex(?label, '\\\\Q${queryValue}\\\\E', 'i')"
}
```

multiple datasets millions of triples

ENTAGEN
ACCELERATING INSIGHT

DEMO

Baseline SPARQL Query Performance

FASTER!

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

def s = new Date().time
List results = sparqlQueryService.executeQuery(queryString)
def e = new Date().time
```

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

def s = new Date().time
List results = sparqlQueryService.executeQuery(queryString)
def e = new Date().time
```

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

def s = new Date().time
List results = sparqlQueryService.executeQuery(queryString)
def e = new Date().time
```

- Java API for adding search and indexing capability to applications
- Widely used in many applications
- Works well with structured and unstructured data
- Complements existing storage layers (RDBS, Triple Store, etc).

ENTAGEN
ACCELERATING INSIGHT

- Index of Content
  - Document
  - Fields

Sample Document

field	value
name	“mike hugo”
company	“Entagen”
bio	“lorem ipsum...”

- Searching
  - Keyword
    - name = mike
  - Phrases
    - name = “Mike Hugo”
  - Wildcards
    - name = “Mike H*”

LUCENE INDEX

- First pass - a lucene index of searchable labels
- For each RDF entity, create a lucene document with fields:
  - subject uri
  - label
  - rdf:type

DEMO

Lucene Index of Searchable Labels

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

sparqlQueryService.executeForEach(repository, queryLabels) {
    def doc = new Document()
    String uri = it.uri.stringValue()
    String label = it.label.stringValue()

    doc.add(new Field(SUBJECT_URI_FIELD, uri,
            Field.Store.YES, Field.Index.ANALYZED))
    doc.add(new Field(LABEL_FIELD, label,
            Field.Store.NO, Field.Index.ANALYZED))

    writer.addDocument(doc)
}
```

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

sparqlQueryService.executeForEach(repository, queryLabels) {
    def doc = new Document()
    String uri = it.uri.stringValue()
    String label = it.label.stringValue()

    doc.add(new Field(SUBJECT_URI_FIELD, uri,
            Field.Store.YES, Field.Index.ANALYZED))
    doc.add(new Field(LABEL_FIELD, label,
            Field.Store.NO, Field.Index.ANALYZED))

    writer.addDocument(doc)
}
```

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

sparqlQueryService.executeForEach(repository, queryLabels) {
    def doc = new Document()
    String uri = it.uri.stringValue()
    String label = it.label.stringValue()

    doc.add(new Field(SUBJECT_URI_FIELD, uri,
            Field.Store.YES, Field.Index.ANALYZED))
    doc.add(new Field(LABEL_FIELD, label,
            Field.Store.NO, Field.Index.ANALYZED))

    writer.addDocument(doc)
}
```

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

    def s = new Date().time
    List results = executeQuery(query)
    def e = new Date().time

    render(view: 'index', model: [results: results, time: e - s])
}

public List executeQuery(Query query) {
    IndexSearcher searcher = luceneSearcherManager.get()
    ScoreDoc[] scoreDocs = searcher.search(query, 10).scoreDocs
    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 << [uri: uri, type: labelAndType.type, label: labelAndType.label]
    }
    connection.close()
    return results
}
```

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

    def s = new Date().time
    List results = executeQuery(query)
    def e = new Date().time

    render(view: 'index', model: [results: results, time: e - s])
}

public List executeQuery(Query query) {
    IndexSearcher searcher = luceneSearcherManager.get()
    ScoreDoc[] scoreDocs = searcher.search(query, 10).scoreDocs
    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 << [uri: uri, type: labelAndType.type, label: labelAndType.label]
    }
    connection.close()
    return results
}
```

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

    def s = new Date().time
    List results = executeQuery(query)
    def e = new Date().time

    render(view: 'index', model: [results: results, time: e - s])
}

public List executeQuery(Query query) {
    IndexSearcher searcher = luceneSearcherManager.get()
    ScoreDoc[] scoreDocs = searcher.search(query, 10).scoreDocs
    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 << [uri: uri, type: labelAndType.type, label: labelAndType.label]
    }
    connection.close()
    return results
}
```

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

    def s = new Date().time
    List results = executeQuery(query)
    def e = new Date().time

    render(view: 'index', model: [results: results, time: e - s])
}

public List executeQuery(Query query) {
    IndexSearcher searcher = luceneSearcherManager.get()
    ScoreDoc[] scoreDocs = searcher.search(query, 10).scoreDocs
    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 << [uri: uri, type: labelAndType.type, label: labelAndType.label]
    }
    connection.close()
    return results
}
```

WHAT ABOUT ENTITY RELATIONSHIPS?

SIREn

{ Semantic Information Retrieval Engine }

- Java API / Lucene Extension
- Specifically built to handle semi-structured data like RDF
- Provides a special Lucene Analyzer (TupleAnalyzer) to index NTriples formatted RDF data
- Can also handle other types of semi-structured data (not limited to RDF)
- Can also work with Solr (Lucene search server)

ENTAGEN
ACCELERATING INSIGHT

SIREn

{ Semantic Information Retrieval Engine }

Sample (Lucene)Document

field	**value**
uri	http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs/DB00619
ntriples	<DB00619> rdfs:label "Imatinib" . <DB00619> rdf:type <drugbank:drugs> . <DB00619> drugbank:brandName "Gleevac" . <DB00619> drugbank:target <targets/1588> .

SIREn

{ Semantic Information Retrieval Engine }

Querying Triples

subject	**predicate**	**object**
DB00619	rdfs:label	"**Imatinib**"

```
SirenTupleQuery tupleQuery = new SirenTupleQuery()
tupleQuery.add(createCellQuery('rdfs:label',
  SirenTupleConstraintPosition.PREDICATE),
  SirenTupleClause.Occur.MUST)
tupleQuery.add(createCellQuery('imatinib',
  SirenTupleConstraintPosition.OBJECT),
  SirenTupleClause.Occur.MUST)
```

ENTAGEN
ACCELERATING INSIGHT

DEMO

SIREn Index of RDF Entities

```
String subjectUris = """
    SELECT distinct ?uri
    WHERE {
        ?uri ?p ?o .
    }
"""

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.ANALYZED))

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

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

    writer.addDocument(doc)
}
```

```
String subjectUris = """
    SELECT distinct ?uri
    WHERE {
        ?uri ?p ?o .
    }
"""
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.ANALYZED))

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

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

    writer.addDocument(doc)
}
```

```
String subjectUris = """
    SELECT distinct ?uri
    WHERE {
        ?uri ?p ?o .
    }
"""

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.ANALYZED))

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

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

    writer.addDocument(doc)
}
```

```
String subjectUris = """
    SELECT distinct ?uri
    WHERE {
        ?uri ?p ?o .
    }
"""
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.ANALYZED))

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

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

    writer.addDocument(doc)
}
```

```
def query = {
    String userQuery = params.query.toLowerCase()
    SirenCellQuery predicateCellQuery = new SirenCellQuery(
            new SirenTermQuery(new Term(TRIPLES_FIELD,
                    RDFS.LABEL.stringValue())))
    predicateCellQuery.constraint = PREDICATE_CELL

    SirenTermQuery objectQuery =
        new SirenTermQuery(new Term(TRIPLES_FIELD, userQuery))
    SirenCellQuery objectCellQuery = new SirenCellQuery(objectQuery)
    objectCellQuery.constraint = OBJECT_CELL

    Query query = new SirenTupleQuery()
    query.add(predicateCellQuery, SirenTupleClause.Occur.MUST)
    query.add(objectCellQuery, SirenTupleClause.Occur.MUST)

    def s = new Date().time
    List results = executeQuery(query)
    def e = new Date().time

    render(view: 'index', model: [results: results, time: e - s])
}
```

ENTAGEN
ACCELERATING INSIGHT

```
def query = {
    String userQuery = params.query.toLowerCase()
    SirenCellQuery predicateCellQuery = new SirenCellQuery(
            new SirenTermQuery(new Term(TRIPLES_FIELD,
                    RDFS.LABEL.stringValue())))
    predicateCellQuery.constraint = PREDICATE_CELL

    SirenTermQuery objectQuery =
        new SirenTermQuery(new Term(TRIPLES_FIELD, userQuery))
    SirenCellQuery objectCellQuery = new SirenCellQuery(objectQuery)
    objectCellQuery.constraint = OBJECT_CELL

    Query query = new SirenTupleQuery()
    query.add(predicateCellQuery, SirenTupleClause.Occur.MUST)
    query.add(objectCellQuery, SirenTupleClause.Occur.MUST)

    def s = new Date().time
    List results = executeQuery(query)
    def e = new Date().time

    render(view: 'index', model: [results: results, time: e - s])
}
```

ENTAGEN
ACCELERATING INSIGHT

```
def query = {
    String userQuery = params.query.toLowerCase()
    SirenCellQuery predicateCellQuery = new SirenCellQuery(
            new SirenTermQuery(new Term(TRIPLES_FIELD,
                    RDFS.LABEL.stringValue())))
    predicateCellQuery.constraint = PREDICATE_CELL

    SirenTermQuery objectQuery =
        new SirenTermQuery(new Term(TRIPLES_FIELD, userQuery))
    SirenCellQuery objectCellQuery = new SirenCellQuery(objectQuery)
    objectCellQuery.constraint = OBJECT_CELL

    Query query = new SirenTupleQuery()
    query.add(predicateCellQuery, SirenTupleClause.Occur.MUST)
    query.add(objectCellQuery, SirenTupleClause.Occur.MUST)

    def s = new Date().time
    List results = executeQuery(query)
    def e = new Date().time

    render(view: 'index', model: [results: results, time: e - s])
}
```

ENTAGEN
ACCELERATING INSIGHT

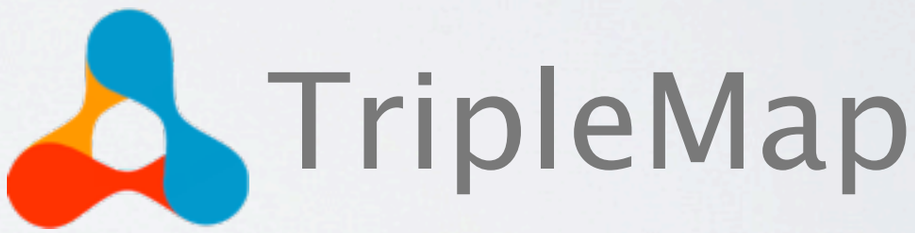
DEMO

SIREn in action on TripleMap.com

QUESTIONS?

mike@entagen.com / twitter: @piragua

http://www.entagen.com



http://www.triplemap.com