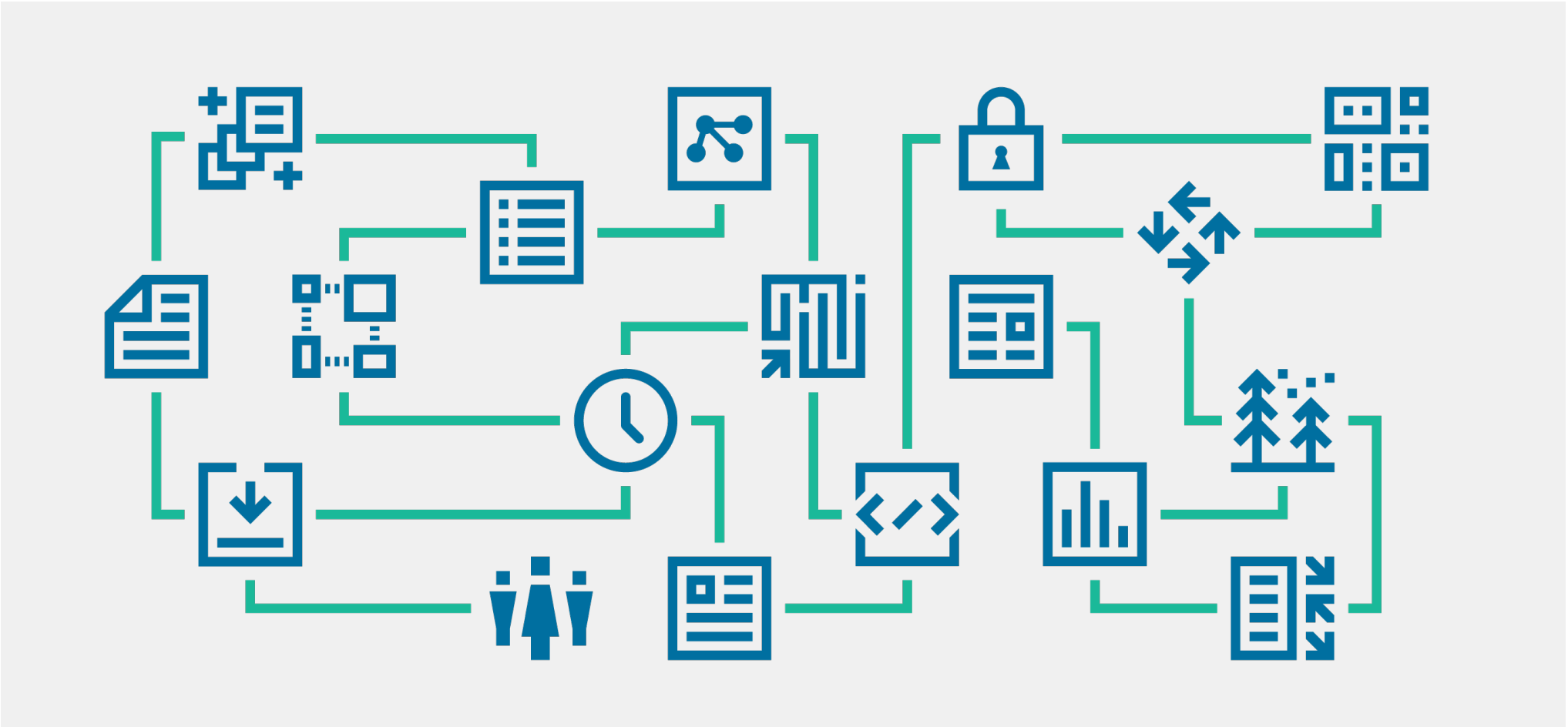
Sparksee Graph Database

Seminaris d'empresa 2017

Arnau Prat-Pérez
Joan Guisado-Gámez

January 2016



Sparksee = Graph Database developed by ***Sparsity**.

Definition

IS a high-performance and out-of-core graph database management system

FOR large scale labeled and attributed multigraphs

BASED ON vertical partitioning and collections of objects identifiers stored as bitmaps.

Let's start!

— All exercises are within a Netbeans project.
Download it from Sparsity Technologies git
https://github.com/SparsityTechnologies/sparksee-handson
— Open the IDE and the project.
— Required data sets are stored into the "data" directory.
— Required libraries are stored into the "libs" directory.
— All exercises have a main method to be executed.
— All exercises are self-explained

Java API

— We are using Sparksee Java for the Training Session

— Java library is a public API

— Private native dynamic library is automatically loaded

— System requirements:

JVM: 5.0 or newer

Operative system: Windows, MacOSX, Linux (32 and 64 bits)

— Javadoc available at the same downloaded package

/libs/sparskeejava-javadoc.jar

Installation

— Open your Netbeans Project.

— Make sure Sparksee library is correctly already linked at

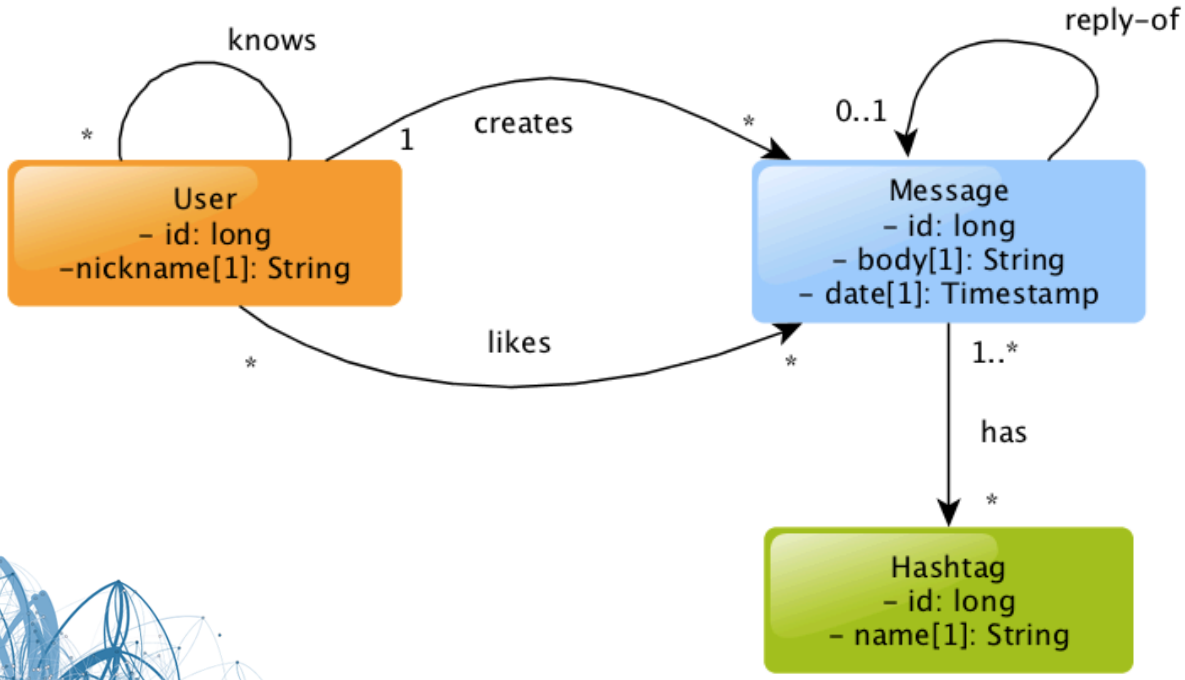
Properties -> Libraries -> Compile

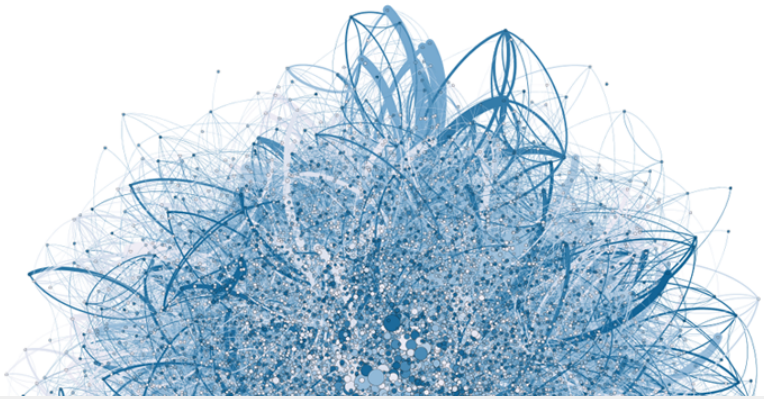
— Make sure Sparksee Javadoc is correctly already linked

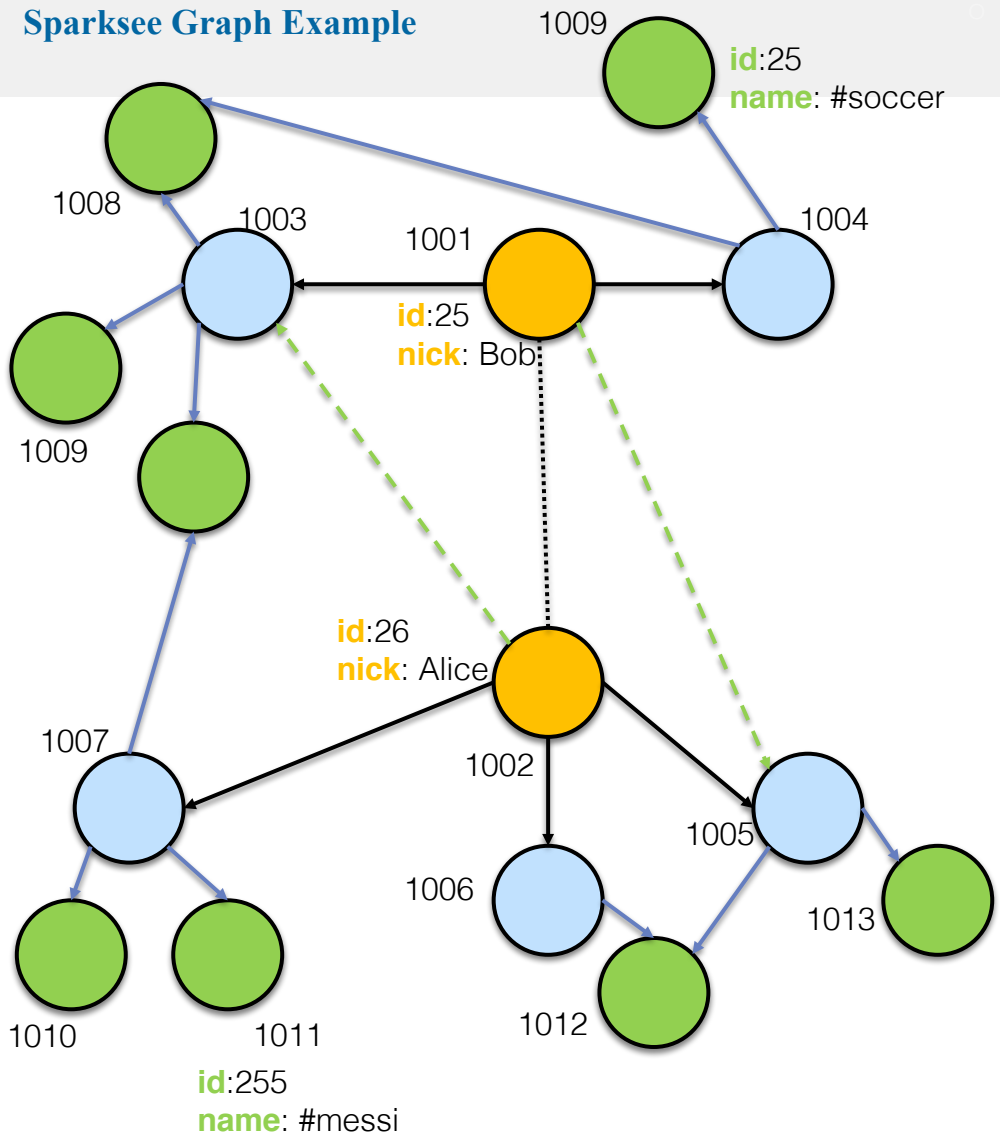
— Otherwise, You should be able to see the sparkseejava.jar, select it and click on the Edit button.

— In the new window where it says Javadoc: write the path to the sparkseejava-javadoc.jar (available at /libs) or click Browse and select it.

Social Network data model







	Type name	Type id (int)
	user	10
	message	11
	hashtag	12
.....	knows	100
→	creates	101
- - →	likes	102
→	has	103
	user.id	201
	user.nick	202
	hashtah.id	203
	hashtag.name	204

Operation	Result
findType("user")	*int*: 10
select(10)	*Objects*: {1001L,1002L}
neighbors(1012L,103,IN)	*Objects*: {1005L,1006L}
getAttribute(1001L,202)	*Value:* v
v.getString()	*String*: "Bob"
getAttribute(1011L,201)	*ERROR*
v.getString()	*String*: "Bob"

WALK-THROUGH

Exercice 1

Calculate Page Rank

$$PR(A) = (1-d) + d\underbrace{\sum_{i=1}^{n} \frac{PR(i)}{C(i)}}$$

For each follower of `A`

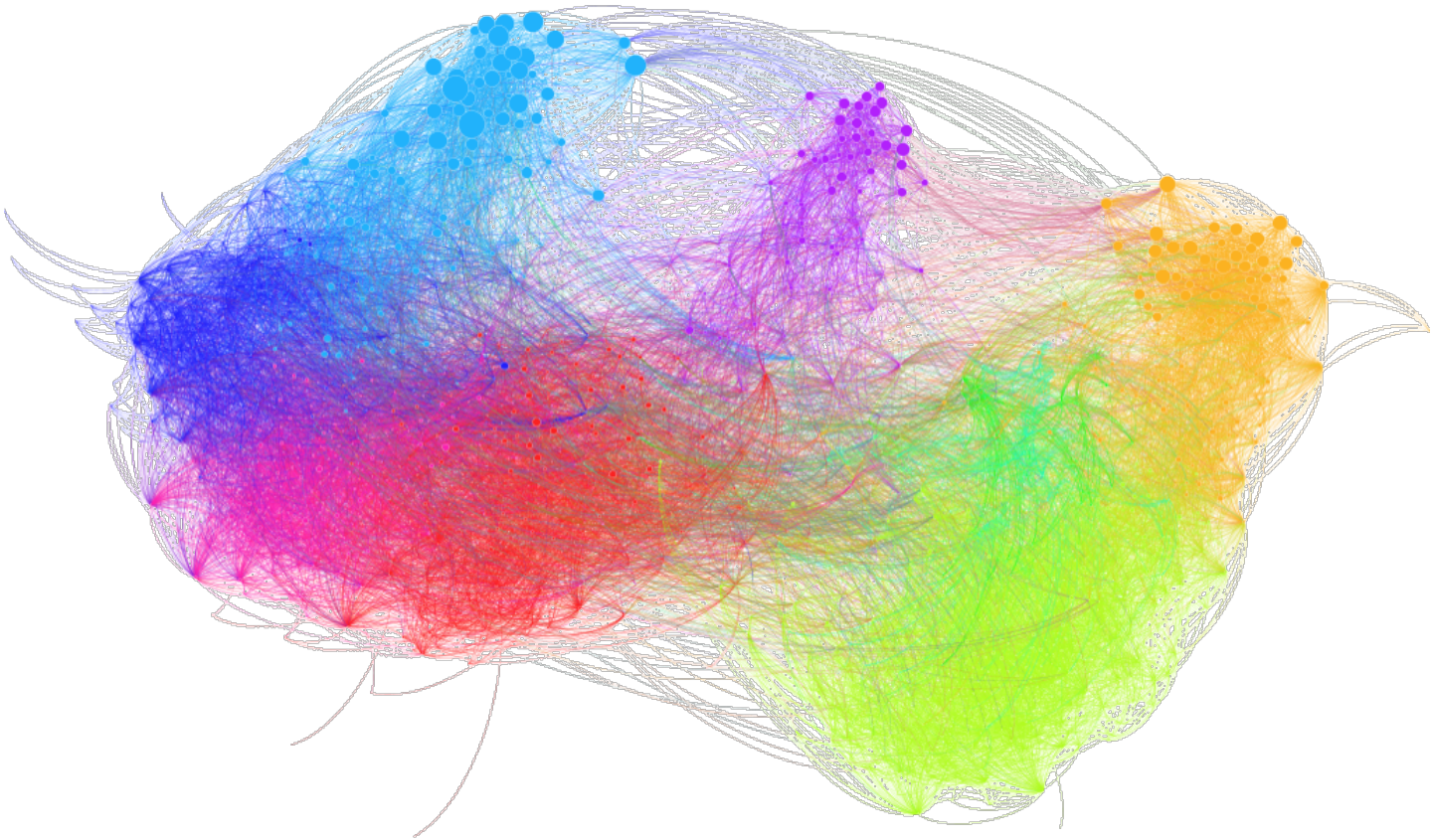
$PR(A)$ **It's the PageRank of User `A`**

d **It's the damping factor**

$C(i)$ **Number of users that `i` follows**

Exercise 2

Finding communities by means of Label Propagation



API Methods Used:

Manage Sparksee

```
create(String filename) → Database
close()
```

Manage Databases

```
newSession() → Session
close()
```

Manage Sessions

```
getGraph() → Graph
close()
```

Manage Graph – Create Schema

Create Types

```
newNodeType(String name) → int
newEdgeType(String name,
            Boolean directed,
            Boolean neighbros) → int
```

Create Attributes

```
newAttribute(int TypeID,
             String name,
             DataType type,
             AttributeKind kind) → int
```

Manage Graph – Populate Database

New Objects

```
newNode(int nodeType) → long
newEdge(int edgeType,
        long nodeId_1,
        long nodeId_2) → long
```

Set attributes

```
setAttribute(long nodeId,
             int attributeType,
             Value v)
```

Manage Value

```
setString(String value)
setBoolean(Boolean value)
setInteger(Integer value)
setDouble(Double value)
setTimestamp(Timestamp value)
setLong(Long value)
```

API Methods Used:

Manage Sparksee

```
open(String filename,
     Boolean readOnly) → Database
close()
```

Manage Graph – Data

```
dumpData(String fileName)
export (String fileName,
        ExportType type,
        ExportManager export)
```

Export Manager

```
prepare(Graph graph)
getGraph(EdgeExport graphExport) → Boolean
getNodeType(int nodeType,
            EdgeExport nodeExport) → Boolean

getEdgeType(int edgeType,
        EdgeExport edgeExport) → Boolean

getNode(long nodeID,
        EdgeExport nodeExport) → Boolean

getEdge(long edgeID,
        EdgeExport edgeExport) → Boolean

enableType(int type) → Boolean
release()
```

API Methods used

Manage Graph – Access Data

```
select(int type) → Objects
select(int attributeType
       Condition condition
       Value value) → Objects
degree(long nodeId,
       int edgeType,
       EdgesDirection direction) → long
getValues(int attributeType) → Values
```

Values

```
Iterator(Order order) → ValuesIterator
close()
```

API Methods used

Manage Graph – Access Data

```
findType(String typeName) → int
findAttribute(int type,
              String name) → int
findObject(int attributeType,
           Value value) → long
neighbors(long nodeId,
          int edgeType,
          EdgesDirection dir) → Objects
getAttribute(long objectId,
             int attributeId,
             Value outValue)
intersection(Objects objectd) → int
```

Manage Objects

```
iterator() → ObjectesIterator
close()
```

Manage ObjectsIterator

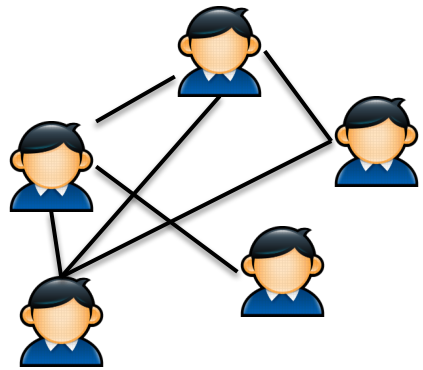
```
hasNext()→Boolean
next() → long
close()
```

Shortest Path

```
SinglePairShortestPathBFS(Session session,
                          long nodeId_1,
                          long nodeId_2)→ShortestPath

SinglePairShortestPathBFS(Session session,
                          long nodeId_1,
                          long nodeId_2)→ShortestPath

addEdgeType(int edgeType,
            EdgesDirection direction)
addNodeType(int nodeType)
run ()
getCost() → double
```



Thanks!

Sparsity-Technologies
@sparsitytech
Sparsity Technologies
Sparsity Technologies