I MapReduced a Neo store

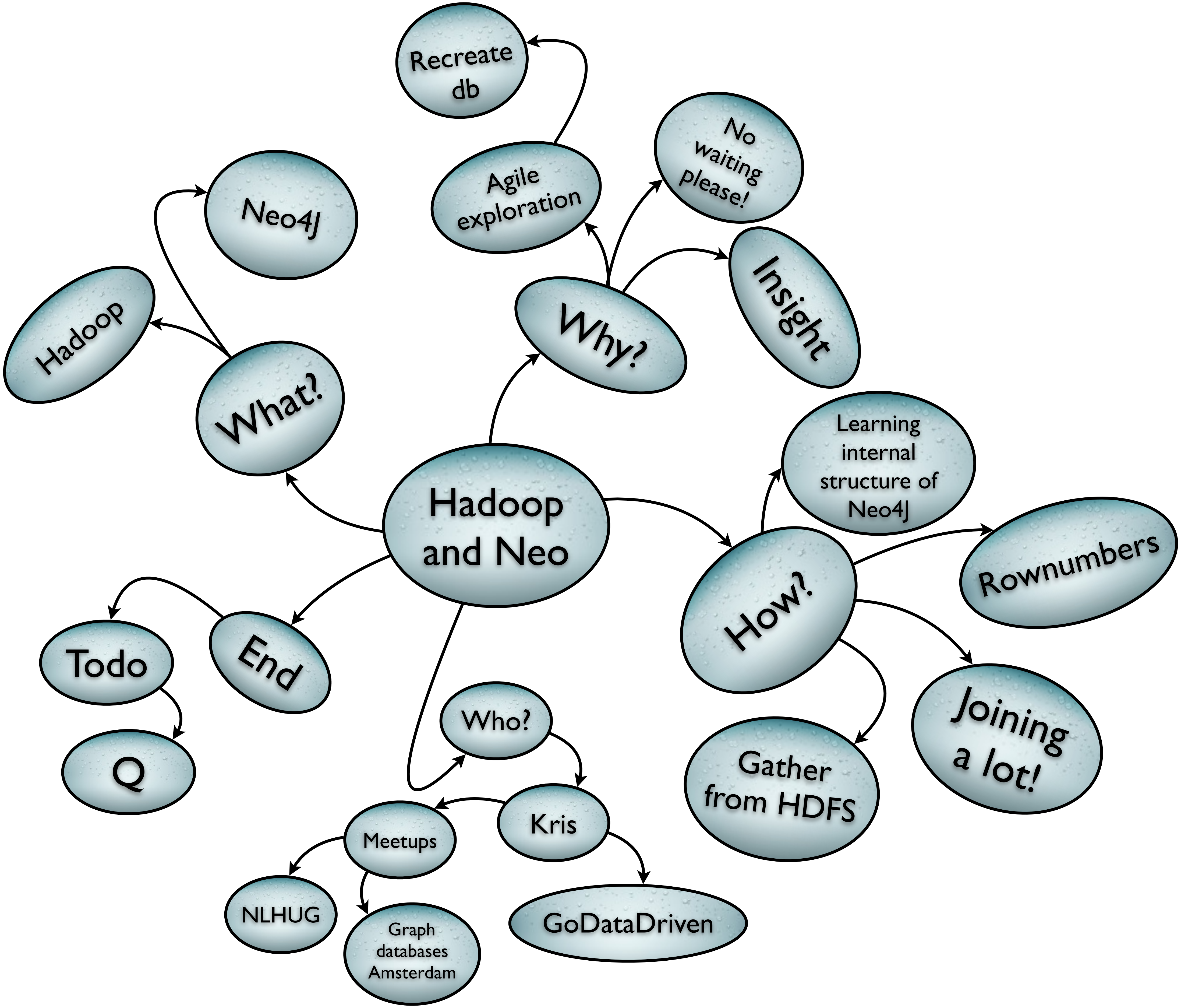
Creating large neo4j databases with hadoop

Kris Geusebroek
Big Data Hacker

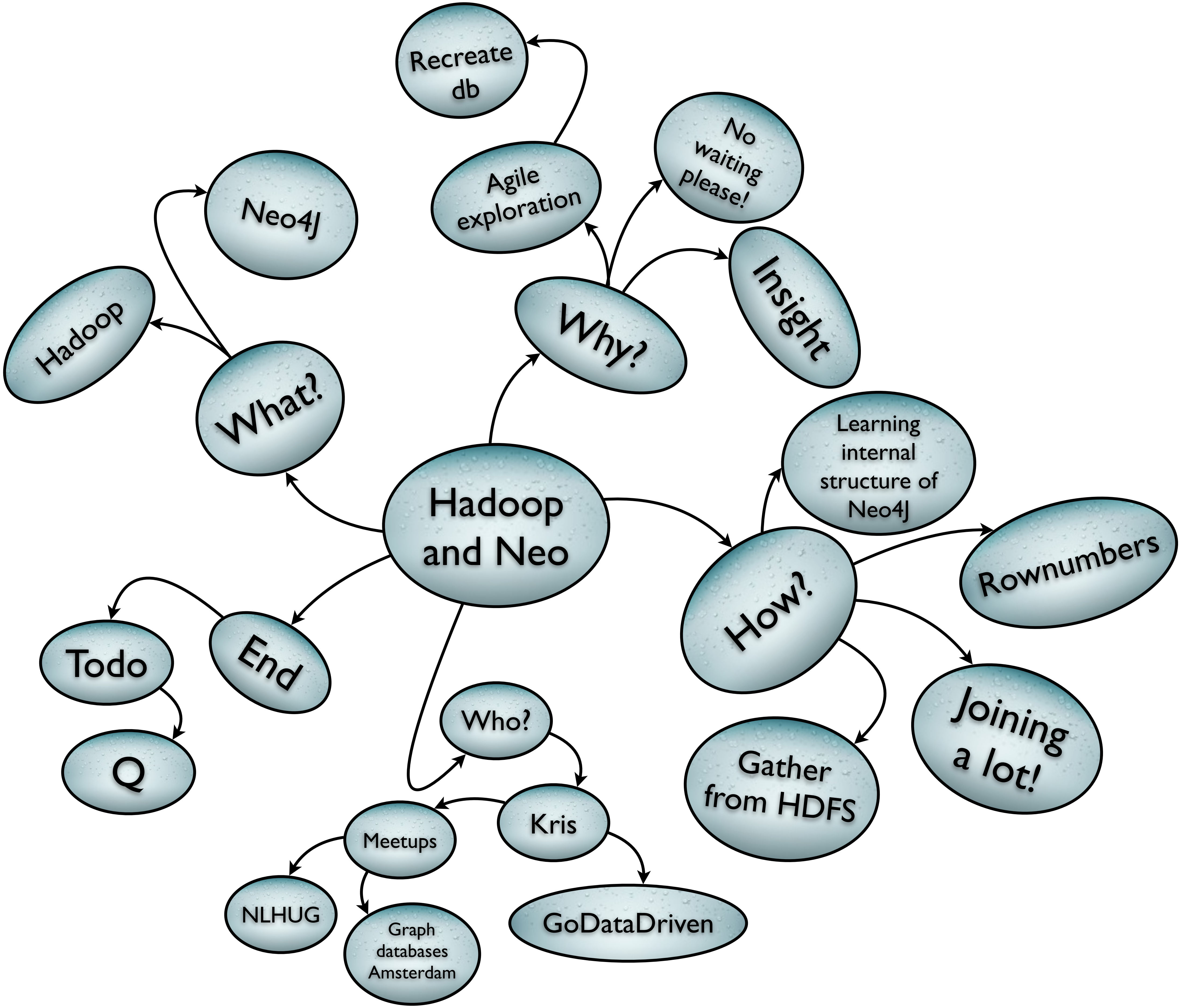
@krisgeus
krisgeusebroek@godatadriven.com

GoDataDriven

PROUDLY PART OF THE XEBIA GROUP



- Hadoop and Neo
  - What?
    - Hadoop
    - Neo4J
  - Why?
    - Agile exploration
      - Recreate db
    - No waiting please!
    - Insight
  - How?
    - Learning internal structure of Neo4J
    - Rownumbers
    - Joining a lot!
    - Gather from HDFS
  - Who?
    - Kris
      - Meetups
        - NLHUG
        - Graph databases Amsterdam
      - GoDataDriven
  - End
    - Todo
      - Q



Hadoop and Neo

What?

Hadoop

Neo4J

Why?

Agile exploration

Recreate db

No waiting please!

Insight

How?

Learning internal structure of Neo4J

Rownumbers

Joining a lot!

Gather from HDFS

Who?

Kris

Meetups

NLHUG

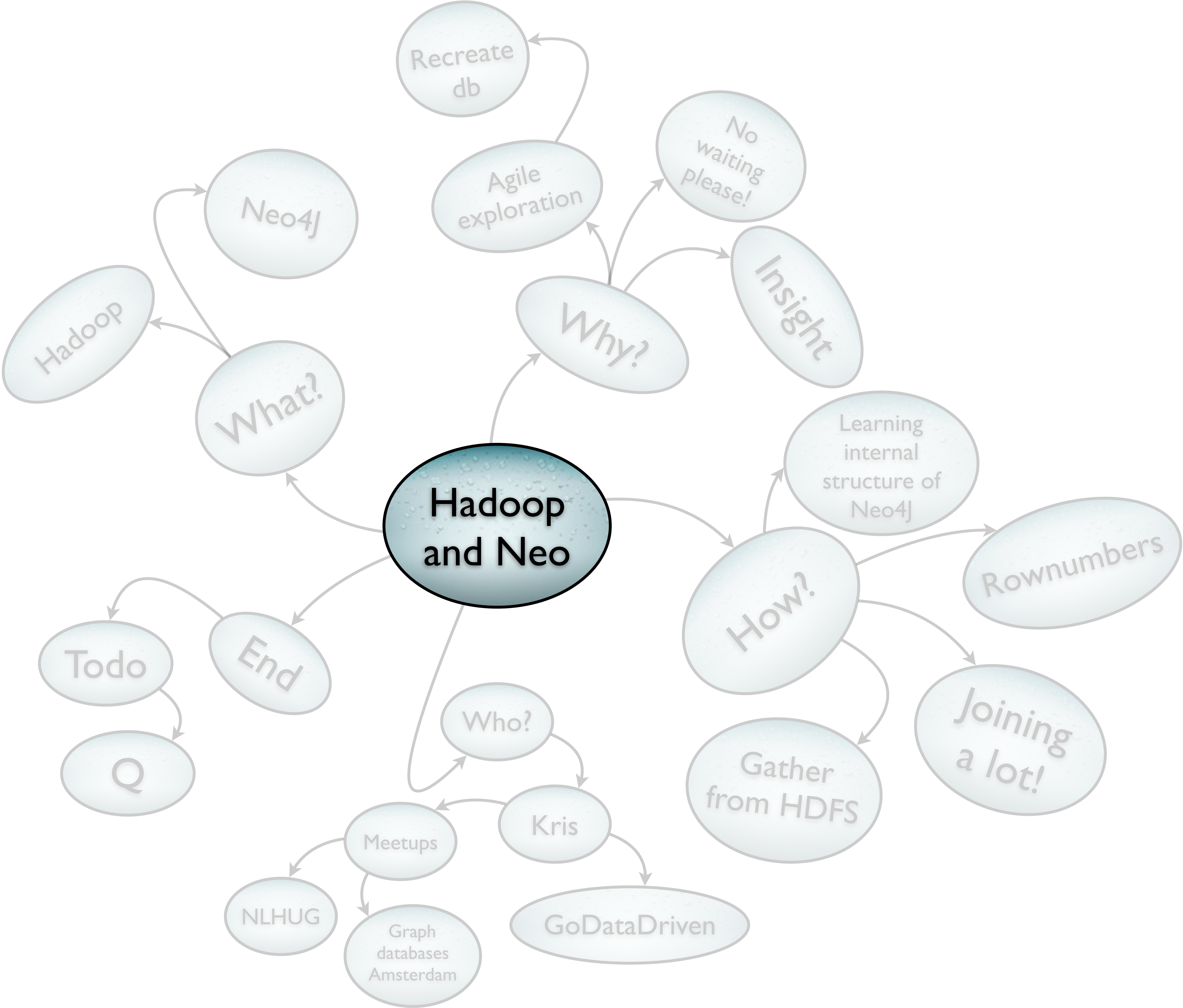
Graph databases Amsterdam

GoDataDriven

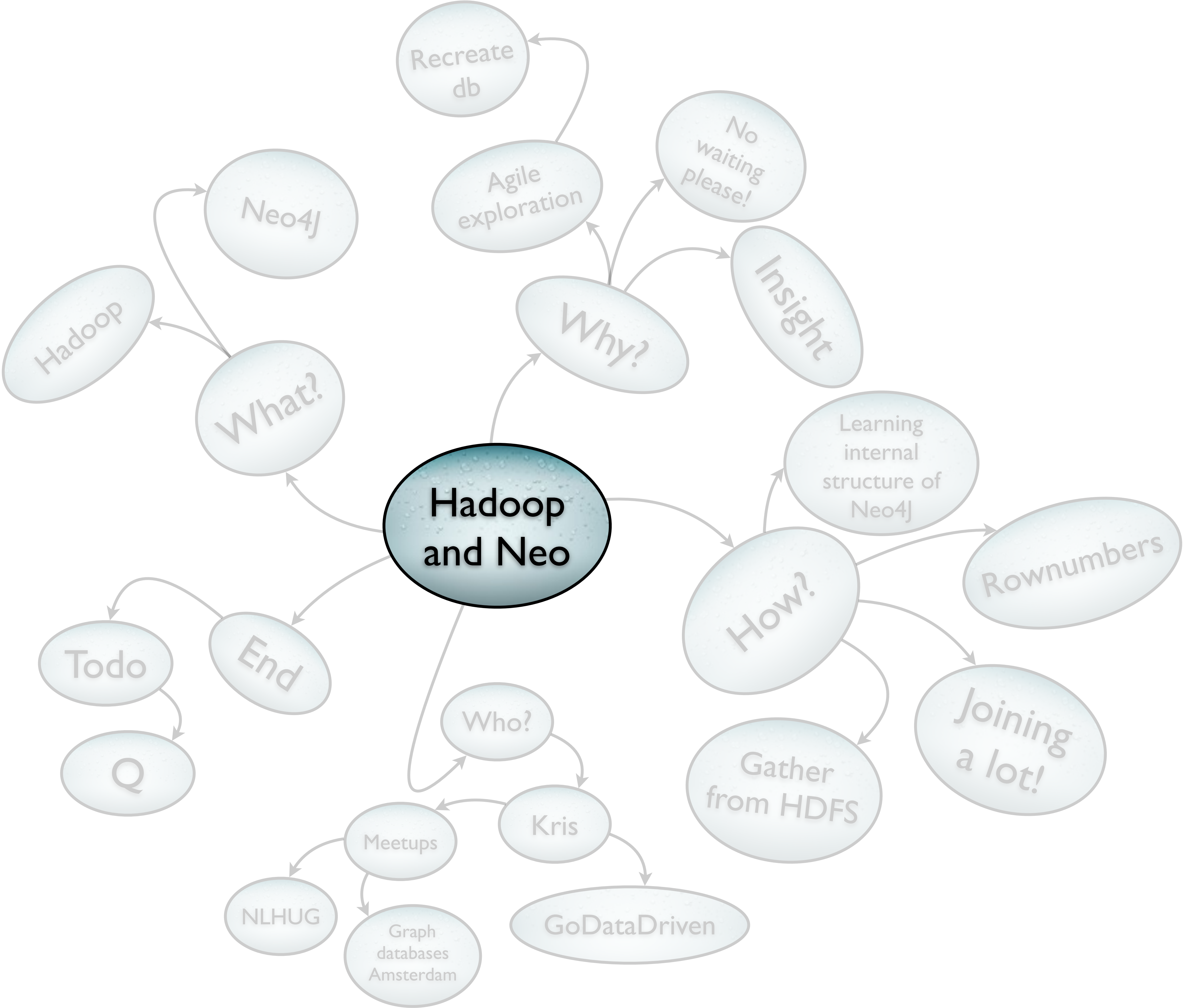
End

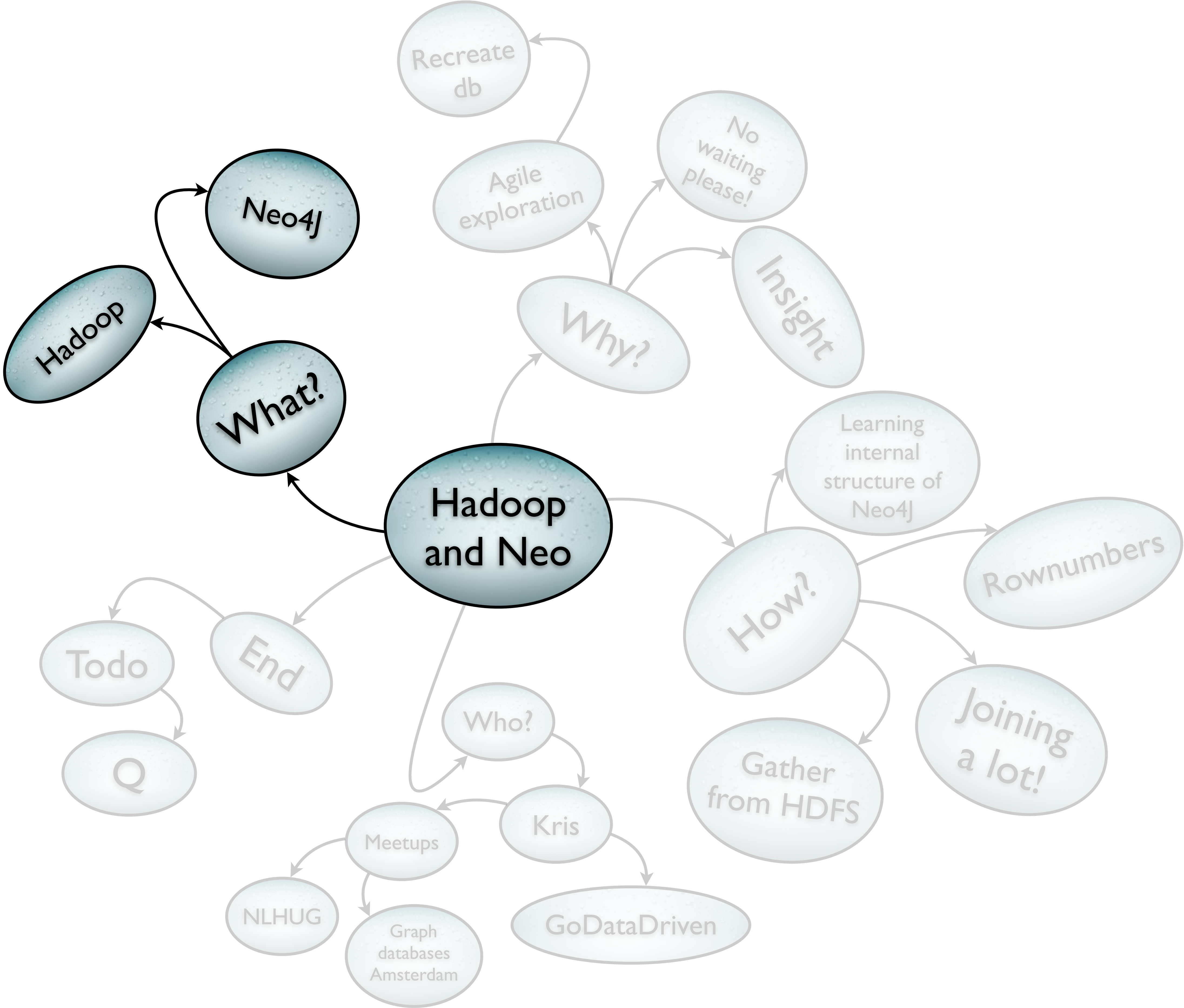
Todo

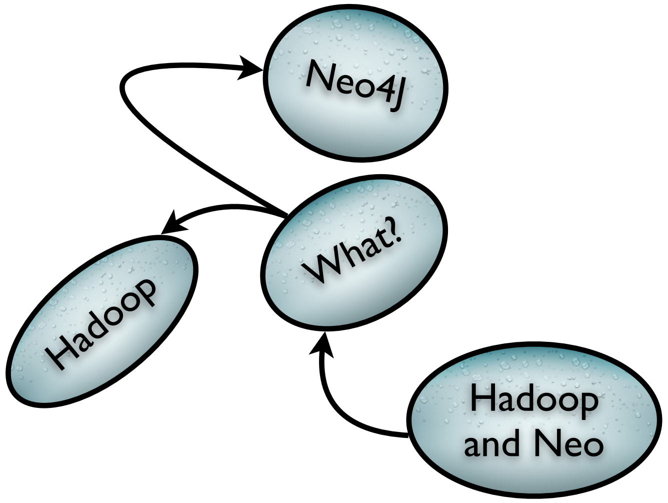
Q



Hadoop and Neo
What?
Hadoop
Neo4J
Why?
Agile exploration
Recreate db
No waiting please!
Insight
How?
Learning internal structure of Neo4J
Rownumbers
Joining a lot!
Gather from HDFS
Who?
Kris
Meetups
NLHUG
Graph databases Amsterdam
GoDataDriven
End
Todo
Q


Hadoop and Neo
What?
Hadoop
Neo4J
Why?
Agile exploration
Recreate db
No waiting please!
Insight
How?
Learning internal structure of Neo4J
Rownumbers
Joining a lot!
Gather from HDFS
Who?
Kris
Meetups
NLHUG
Graph databases Amsterdam
GoDataDriven
End
Todo
Q

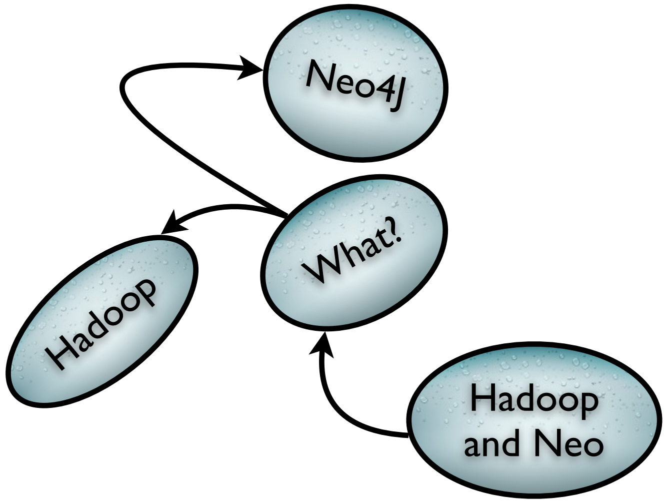

Hadoop and Neo
What?
Hadoop
Neo4J
Why?
Agile exploration
Recreate db
No waiting please!
Insight
How?
Learning internal structure of Neo4J
Rownumbers
Joining a lot!
Gather from HDFS
Who?
Kris
GoDataDriven
Meetups
NLHUG
Graph databases Amsterdam
End
Todo
Q



Hadoop

WIKIPEDIA
The Free Encyclopedia

Apache Hadoop is an open-source framework that supports data-intensive distributed applications



Neo4J

What?

Hadoop

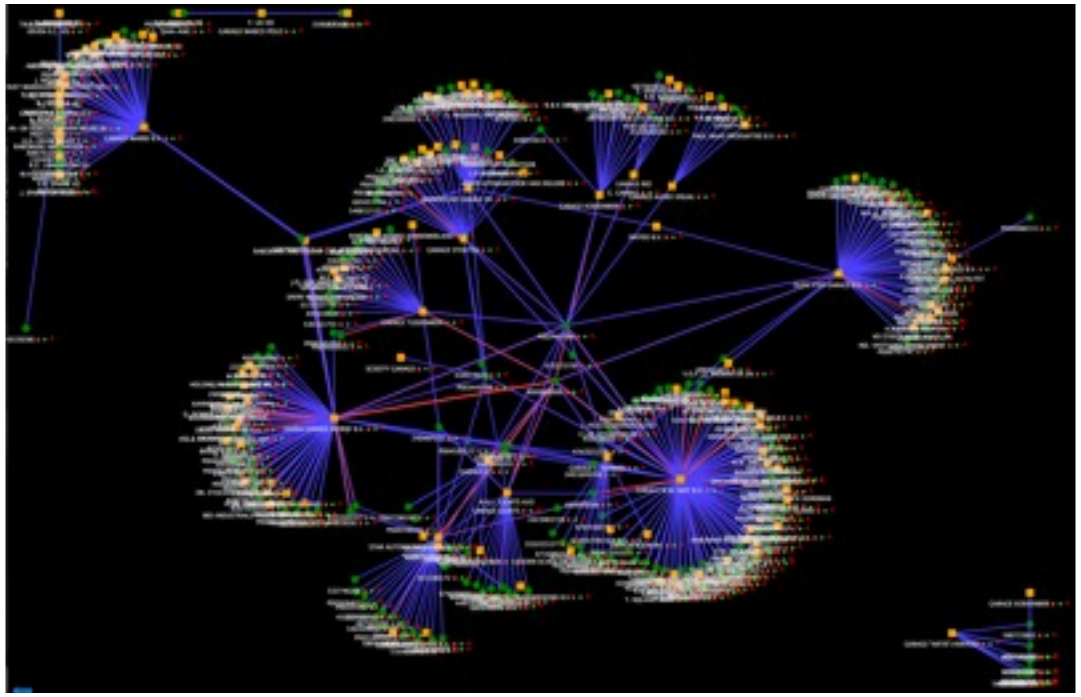
Hadoop and Neo

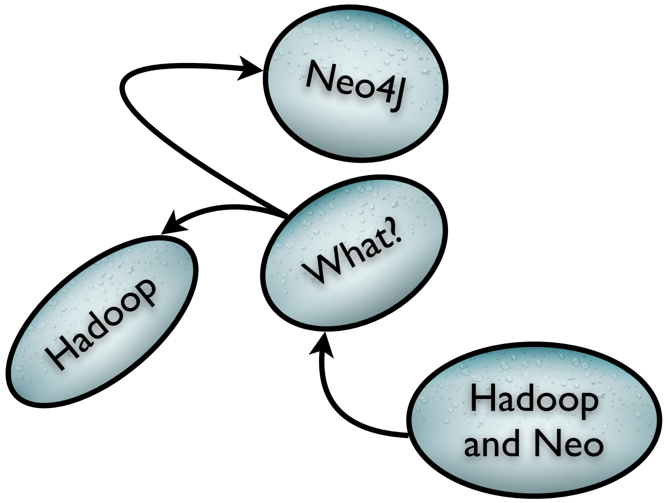
Neo4J

WIKIPEDIA
The Free Encyclopedia

Neo4j
the graph database

Neo4j is an open-source graph database, implemented in Java.The developers describe Neo4j as "embedded, disk-based, fully transactional Java persistence engine that stores data structured in graphs rather than in tables"





Large

hadoop

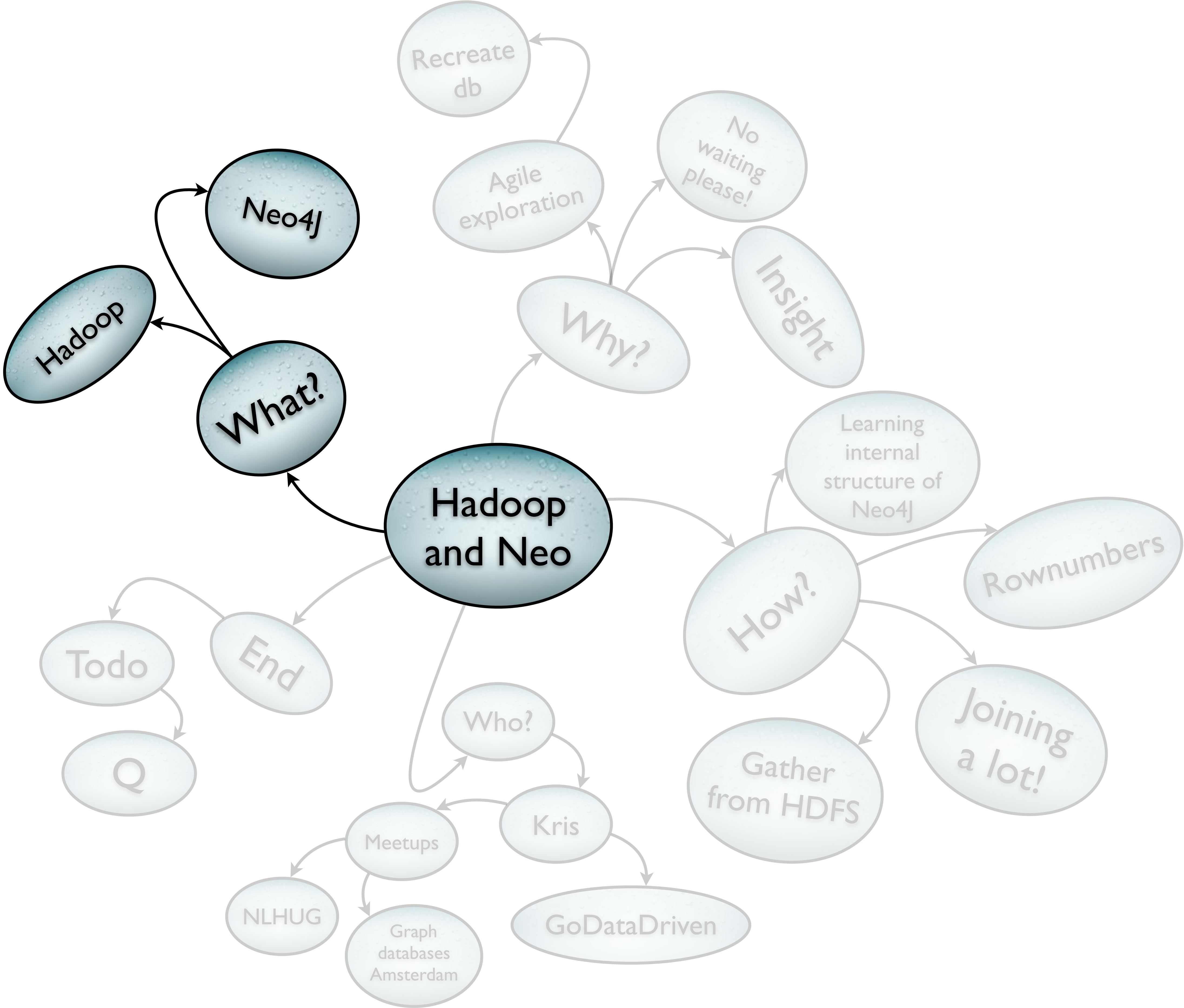
Neo4j
the graph database

Use case: Create large neo4j database

Multiple times!

about 30.000.000 nodes
and 700.000.000 edges
between them.

each node has 9 properties
each edge has 4 properties



Hadoop and Neo

What?

Hadoop

Neo4J

Why?

Agile exploration

Recreate db

No waiting please!

Insight

How?

Learning internal structure of Neo4J

Rownumbers

Joining a lot!

Gather from HDFS

Who?

Kris

Meetups

NLHUG

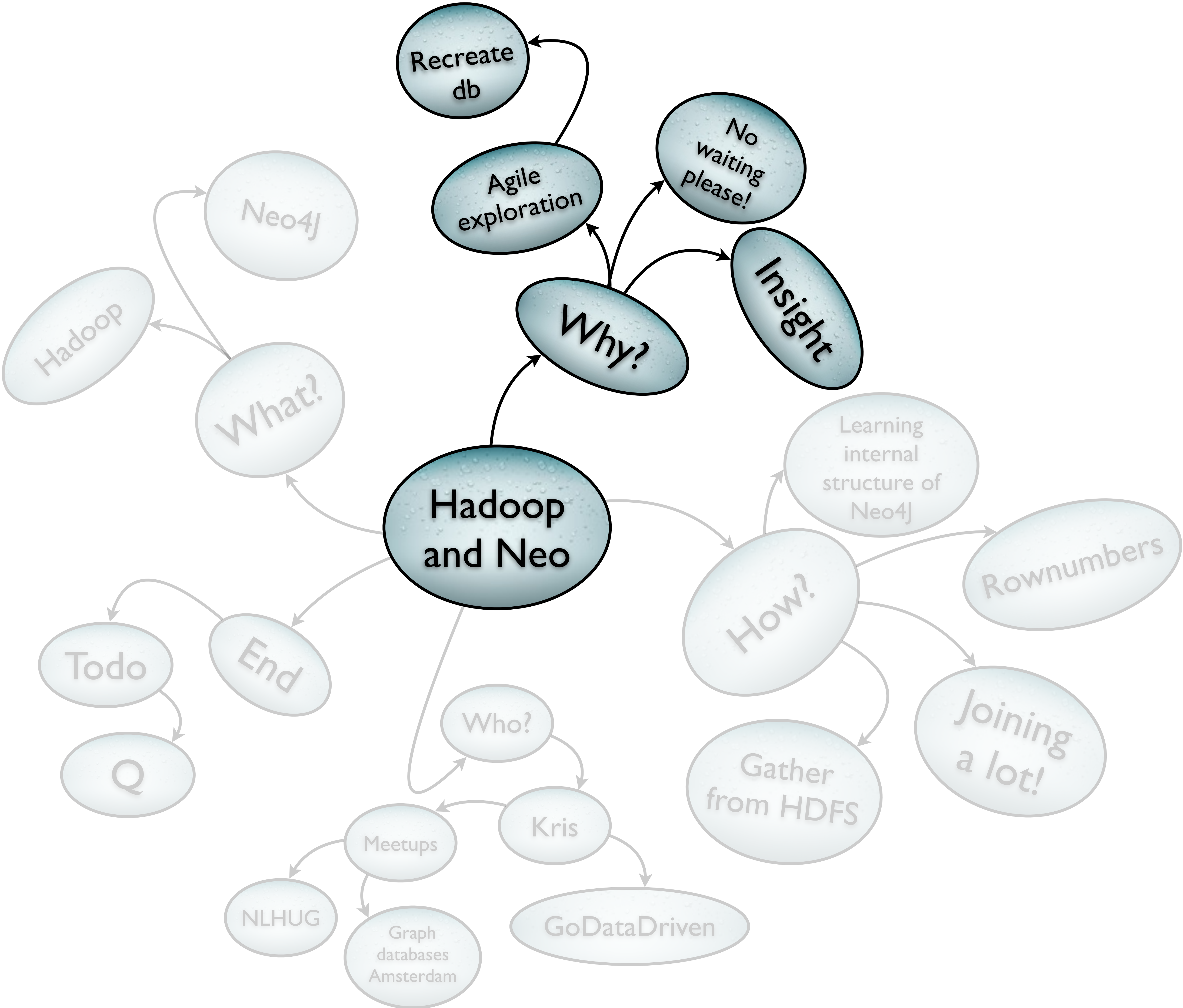
Graph databases Amsterdam

GoDataDriven

End

Todo

Q



Hadoop and Neo

Why?

Agile exploration

Recreate db

No waiting please!

Insight

What?

Hadoop

Neo4J

How?

Learning internal structure of Neo4J

Rownumbers

Joining a lot!

Gather from HDFS

End

Todo

Q

Who?

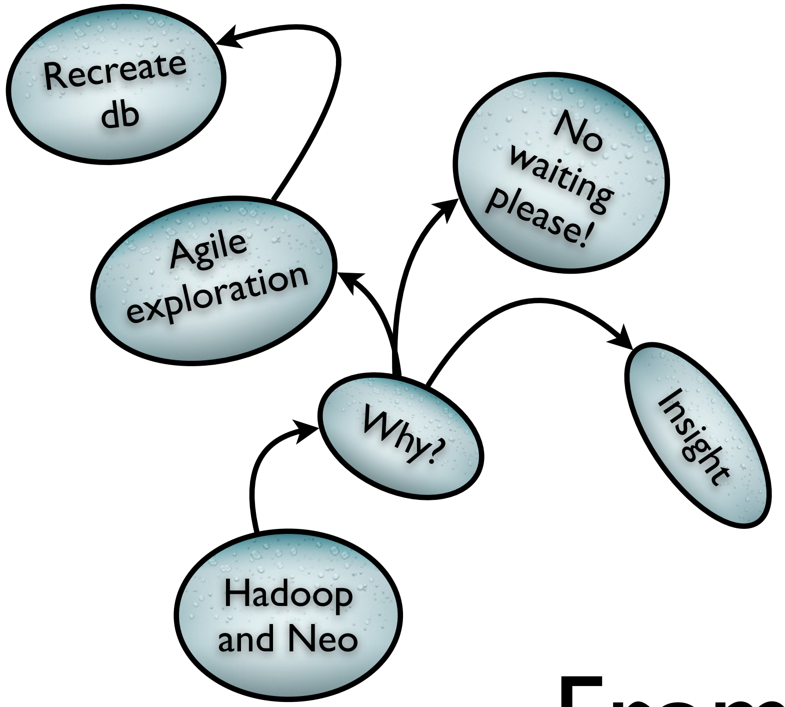
Kris

GoDataDriven

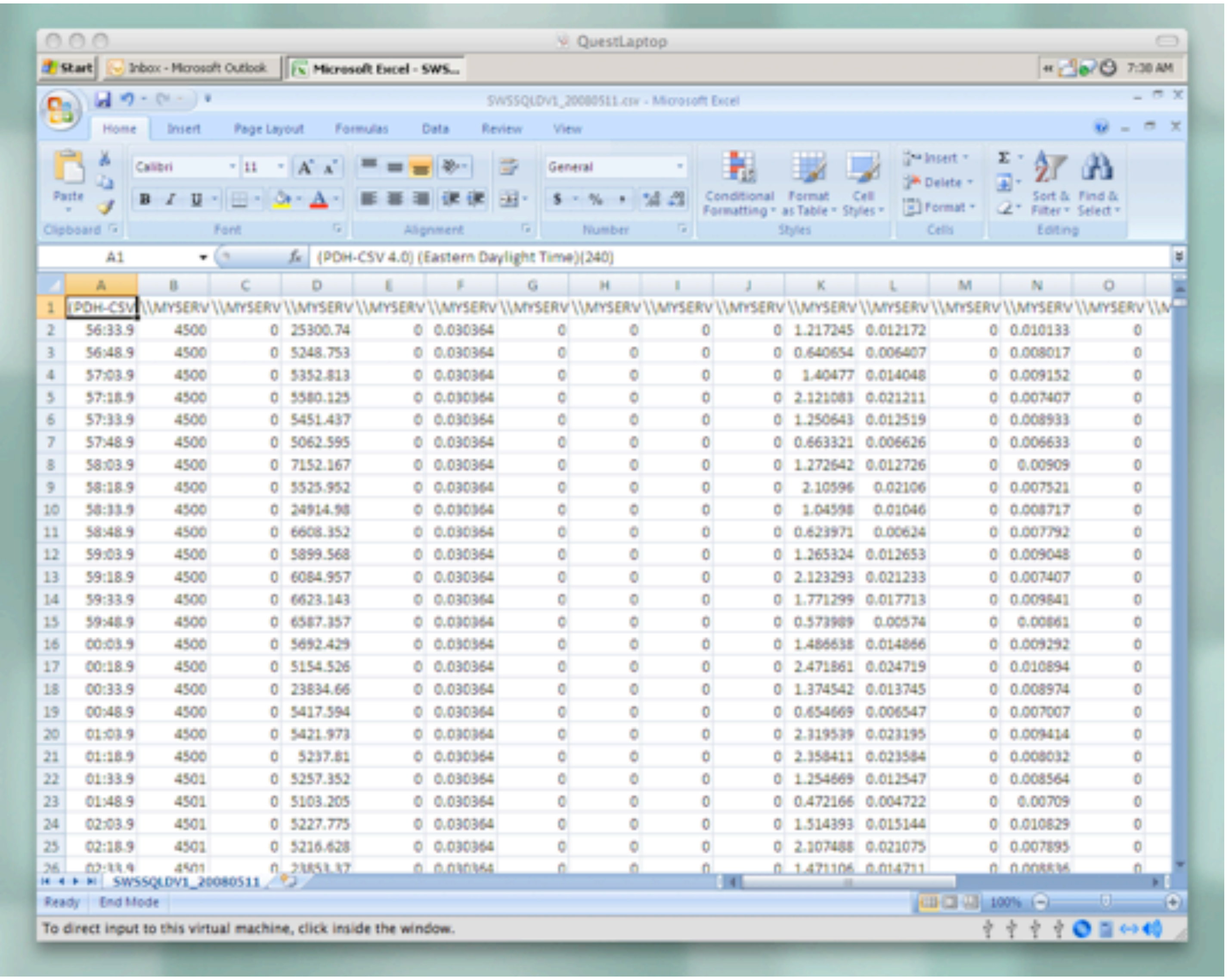
Meetups

NLHUG

Graph databases Amsterdam



From

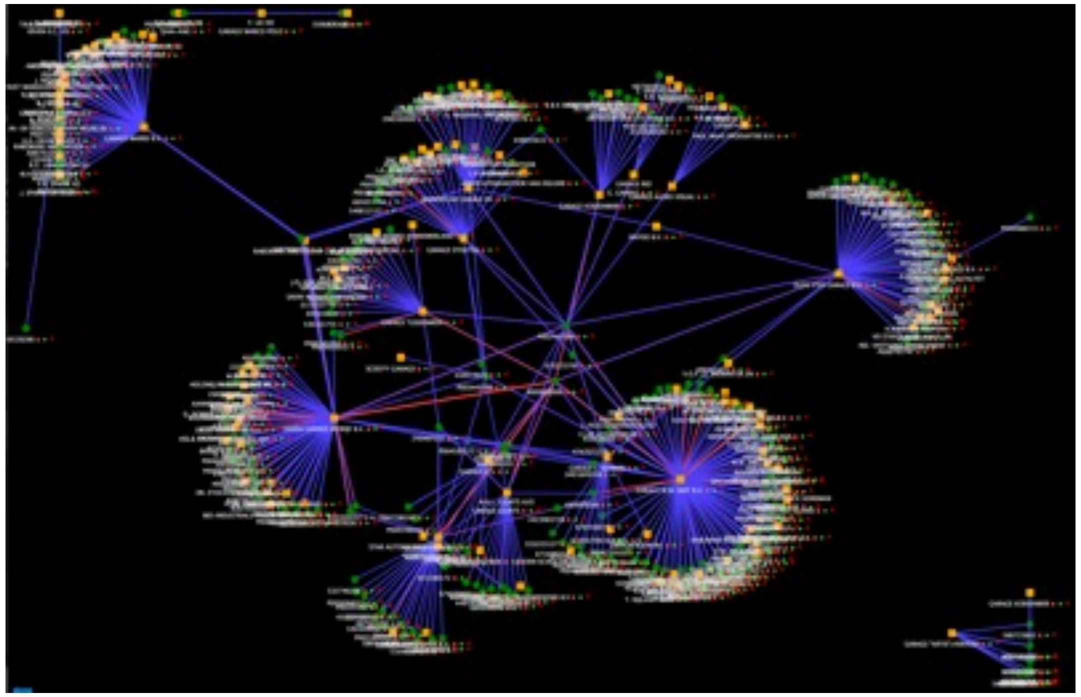


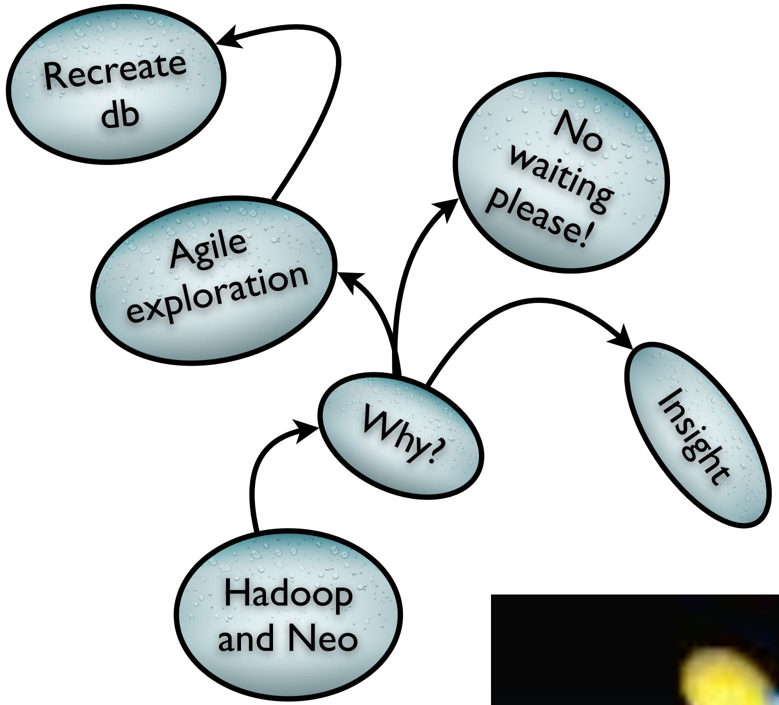
A1 (PDH-CSV 4.0) (Eastern Daylight Time)(240)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	(PDH-CSV	\\MYSERV	\\MYSERV	\\MYSERV	\\MYSERV	\\MYSERV	\\MYSERV	\\MYSERV	\\MYSERV	\\MYSERV	\\MYSERV	\\MYSERV	\\MYSERV	\\MYSERV	\\MYSERV
2	56:33.9	4500	0	25300.74	0	0.030364	0	0	0	0	1.217245	0.012172	0	0.010133	0
3	56:48.9	4500	0	5248.753	0	0.030364	0	0	0	0	0.640654	0.006407	0	0.008017	0
4	57:03.9	4500	0	5352.813	0	0.030364	0	0	0	0	1.40477	0.014048	0	0.009152	0
5	57:18.9	4500	0	5580.125	0	0.030364	0	0	0	0	2.121083	0.021211	0	0.007407	0
6	57:33.9	4500	0	5451.437	0	0.030364	0	0	0	0	1.250643	0.012519	0	0.008933	0
7	57:48.9	4500	0	5062.595	0	0.030364	0	0	0	0	0.663321	0.006626	0	0.006633	0
8	58:03.9	4500	0	7152.167	0	0.030364	0	0	0	0	1.272642	0.012726	0	0.00909	0
9	58:18.9	4500	0	5525.952	0	0.030364	0	0	0	0	2.10596	0.02106	0	0.007521	0
10	58:33.9	4500	0	24914.98	0	0.030364	0	0	0	0	1.04598	0.01046	0	0.008717	0
11	58:48.9	4500	0	6608.352	0	0.030364	0	0	0	0	0.623971	0.00624	0	0.007792	0
12	59:03.9	4500	0	5899.568	0	0.030364	0	0	0	0	1.265324	0.012653	0	0.009048	0
13	59:18.9	4500	0	6084.957	0	0.030364	0	0	0	0	2.123293	0.021233	0	0.007407	0
14	59:33.9	4500	0	6623.143	0	0.030364	0	0	0	0	1.771299	0.017713	0	0.009841	0
15	59:48.9	4500	0	6587.357	0	0.030364	0	0	0	0	0.573989	0.00574	0	0.00861	0
16	00:03.9	4500	0	5692.429	0	0.030364	0	0	0	0	1.486638	0.014866	0	0.009292	0
17	00:18.9	4500	0	5154.526	0	0.030364	0	0	0	0	2.471861	0.024719	0	0.010894	0
18	00:33.9	4500	0	23834.66	0	0.030364	0	0	0	0	1.374542	0.013745	0	0.008974	0
19	00:48.9	4500	0	5417.594	0	0.030364	0	0	0	0	0.654669	0.006547	0	0.007007	0
20	01:03.9	4500	0	5421.973	0	0.030364	0	0	0	0	2.319539	0.023195	0	0.009414	0
21	01:18.9	4500	0	5237.81	0	0.030364	0	0	0	0	2.358411	0.023584	0	0.008032	0
22	01:33.9	4501	0	5257.352	0	0.030364	0	0	0	0	1.254669	0.012547	0	0.008564	0
23	01:48.9	4501	0	5103.205	0	0.030364	0	0	0	0	0.472166	0.004722	0	0.00709	0
24	02:03.9	4501	0	5227.775	0	0.030364	0	0	0	0	1.514393	0.015144	0	0.010829	0
25	02:18.9	4501	0	5216.628	0	0.030364	0	0	0	0	2.107488	0.021075	0	0.007895	0
26	02:33.9	4501	0	23853.37	0	0.030364	0	0	0	0	1.471106	0.014711	0	0.008836	0

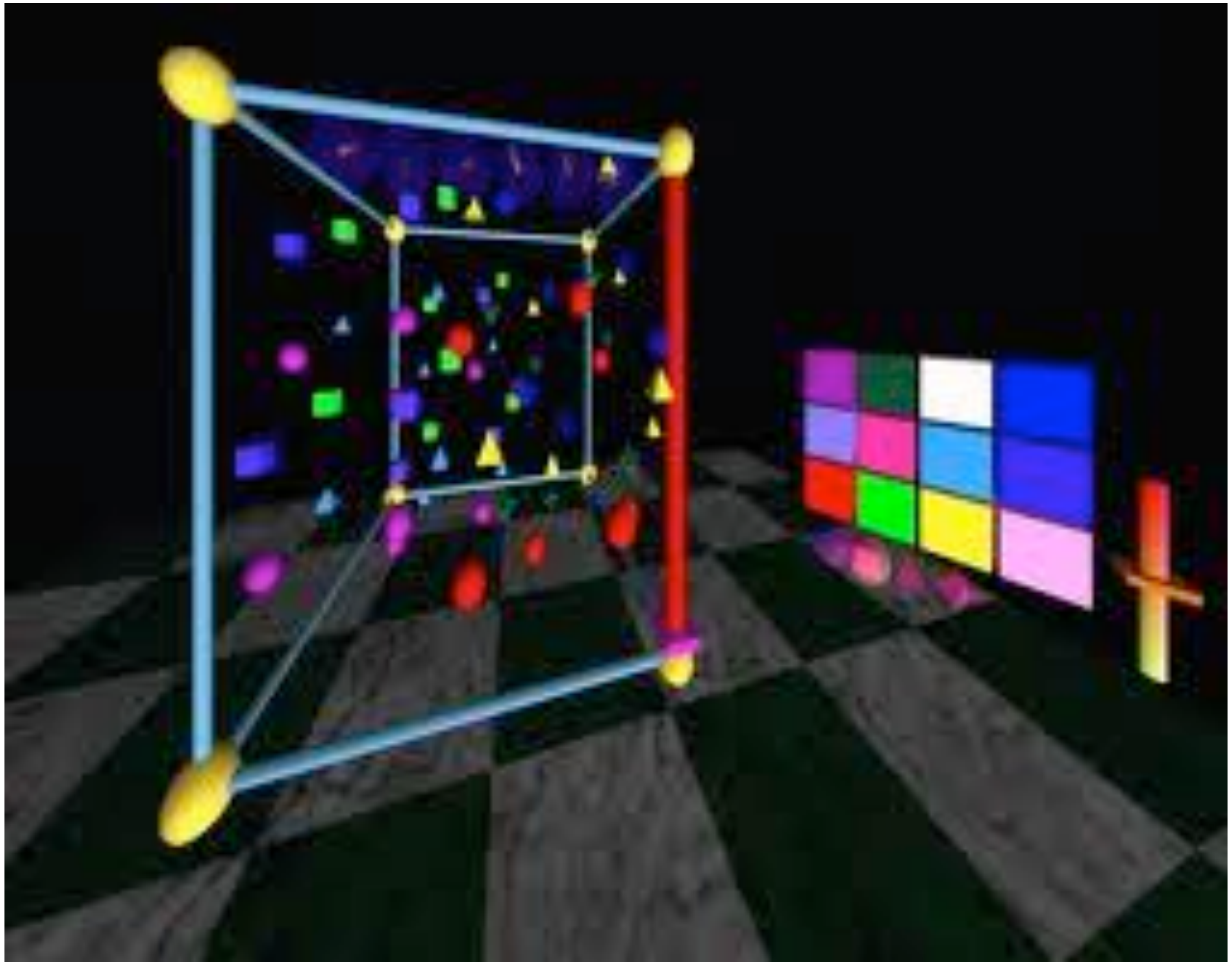
To direct input to this virtual machine, click inside the window.

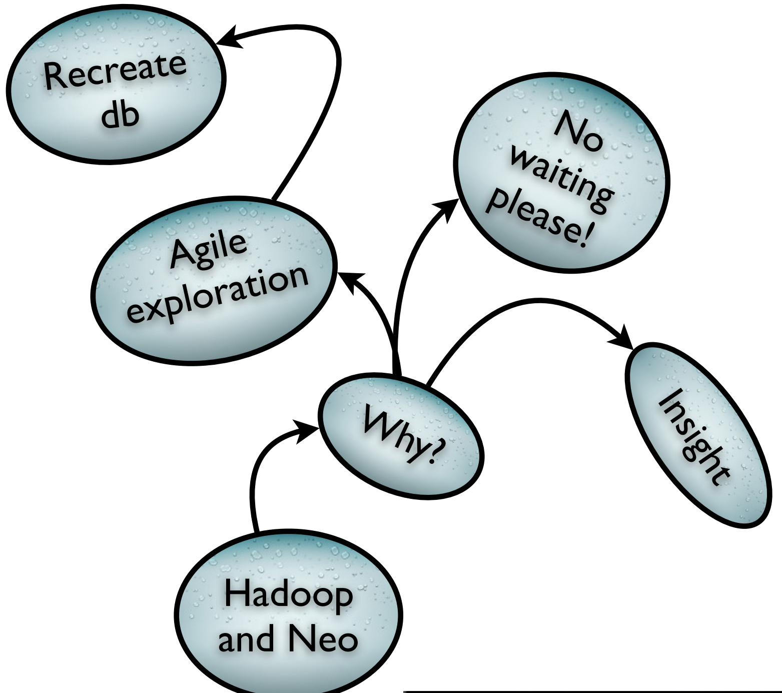
To



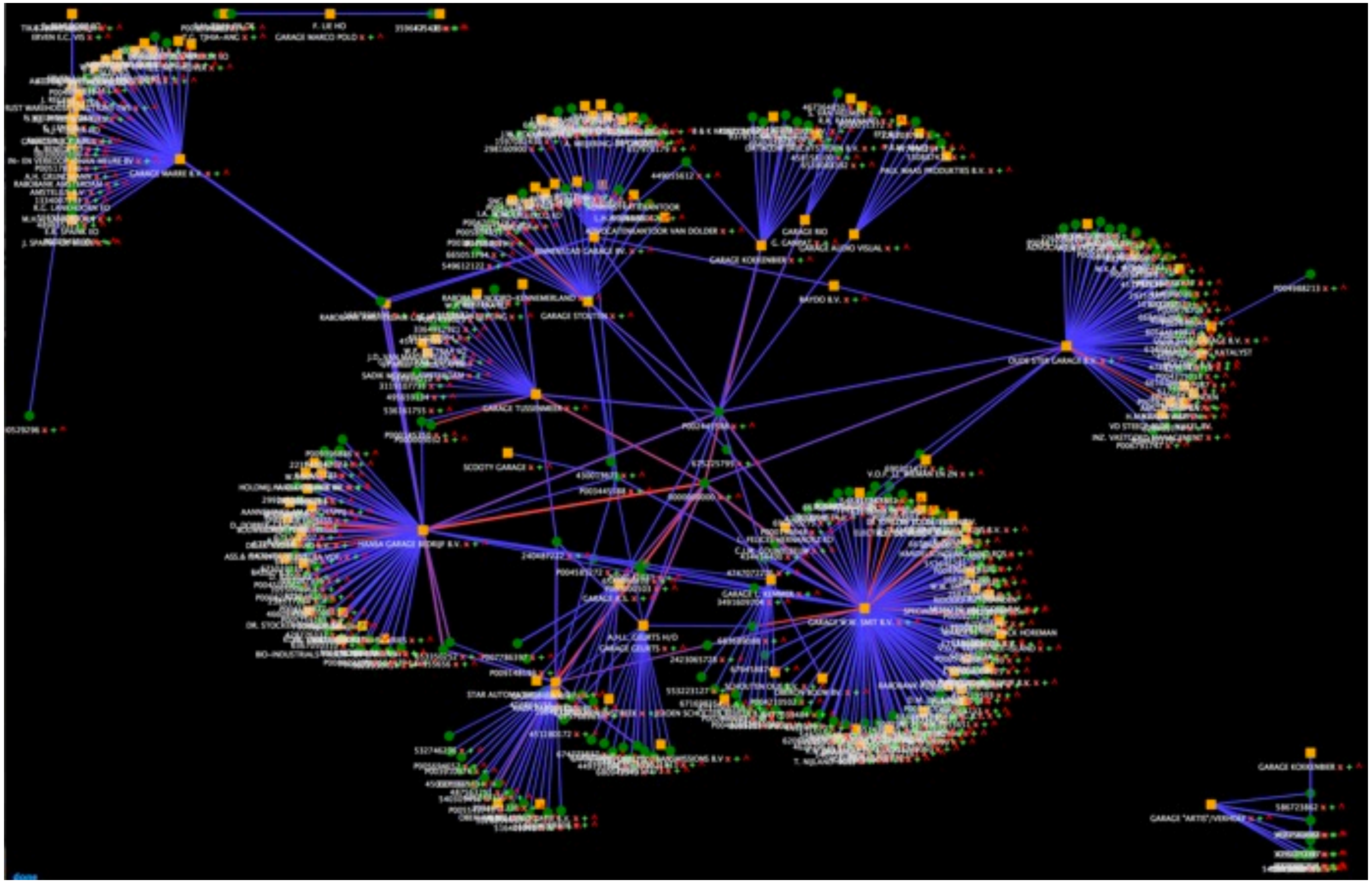


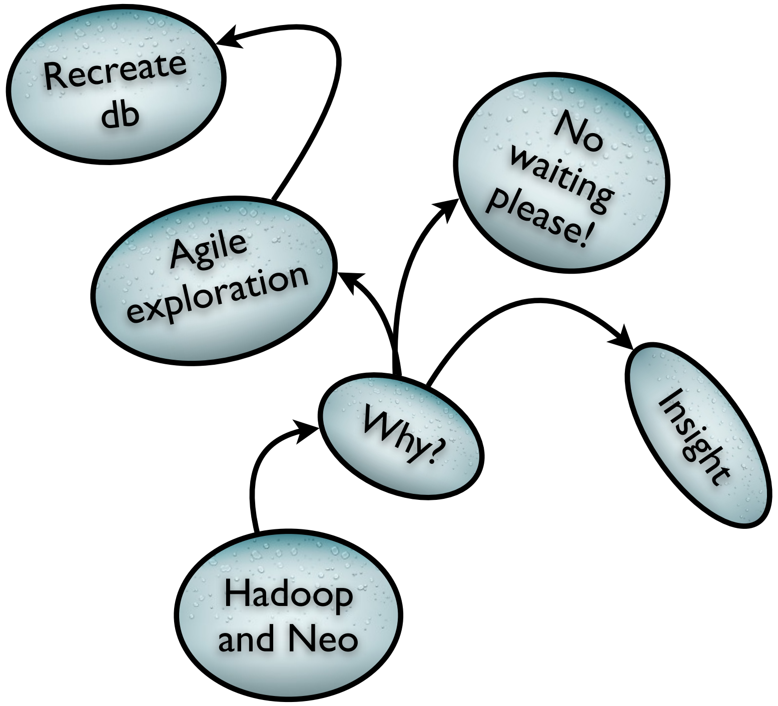
Agile exploration




Recreate db
Agile exploration
No waiting please!
Why?
Insight
Hadoop and Neo

Re-create db





Recreate db

No waiting please!

Agile exploration

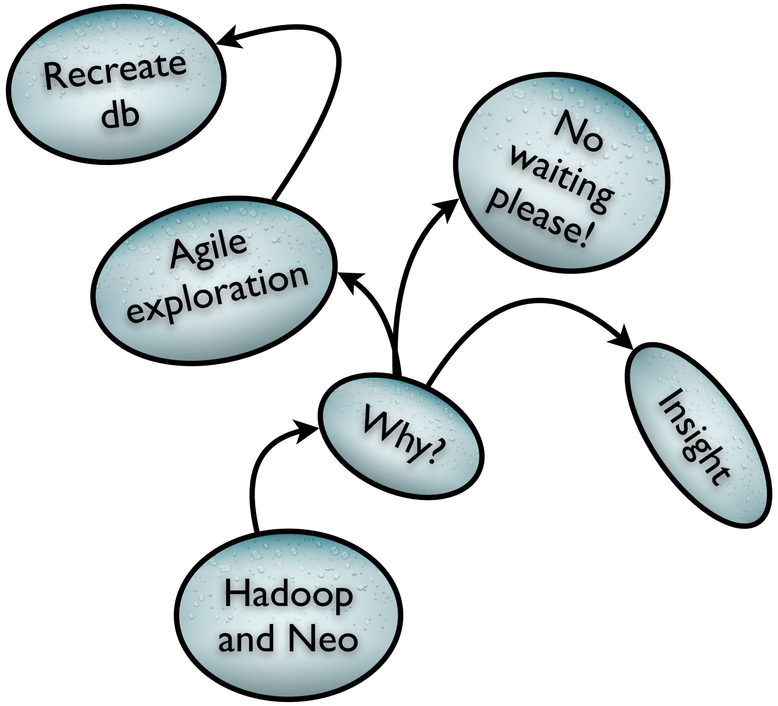
Why?

Insight

Hadoop and Neo

Re-create db

Neo4j
the world's leading graph database



Recreate db

Agile exploration

No waiting please!

Why?

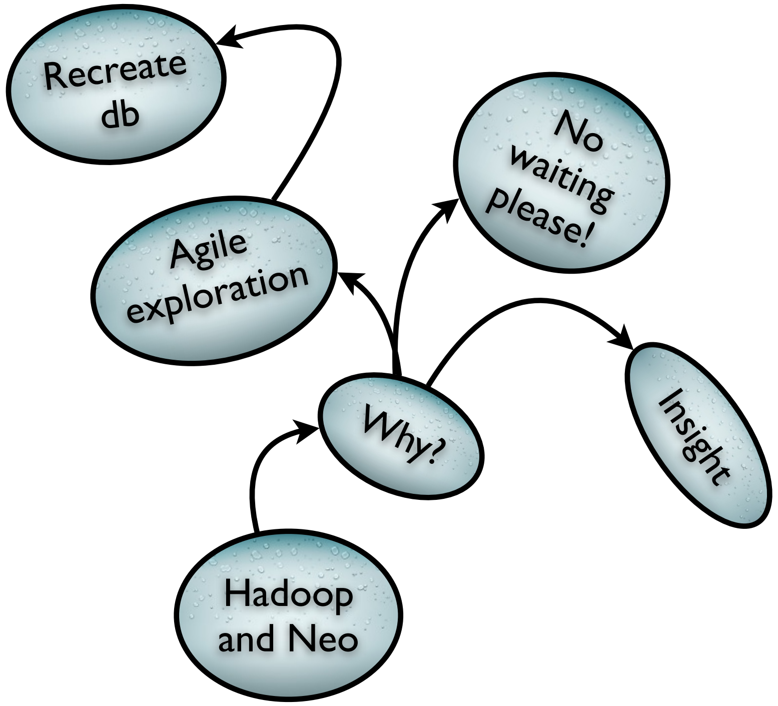
Insight

Hadoop and Neo

Re-create db

Neo4j

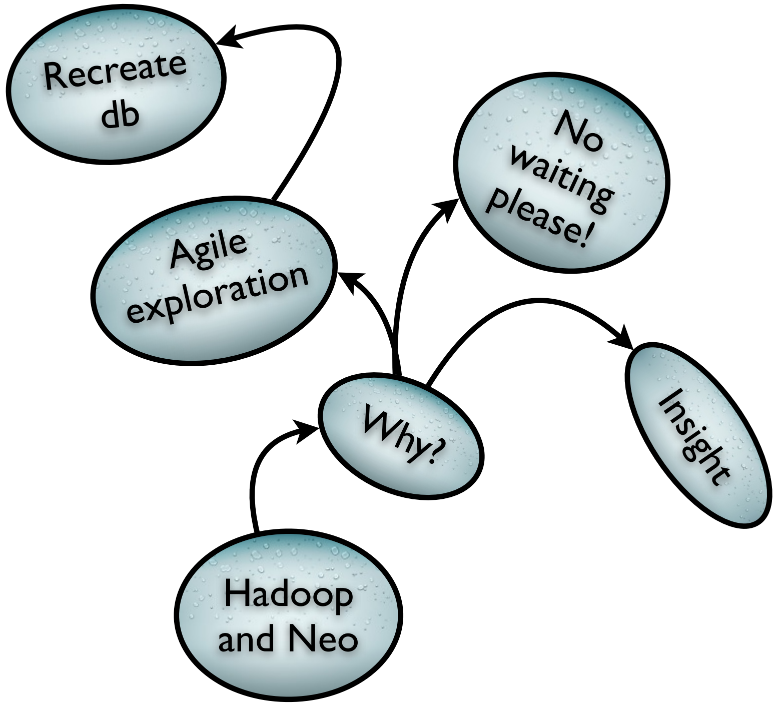
the world's leading graph database



Re-create db

Multiple ways to create a Neo4J database

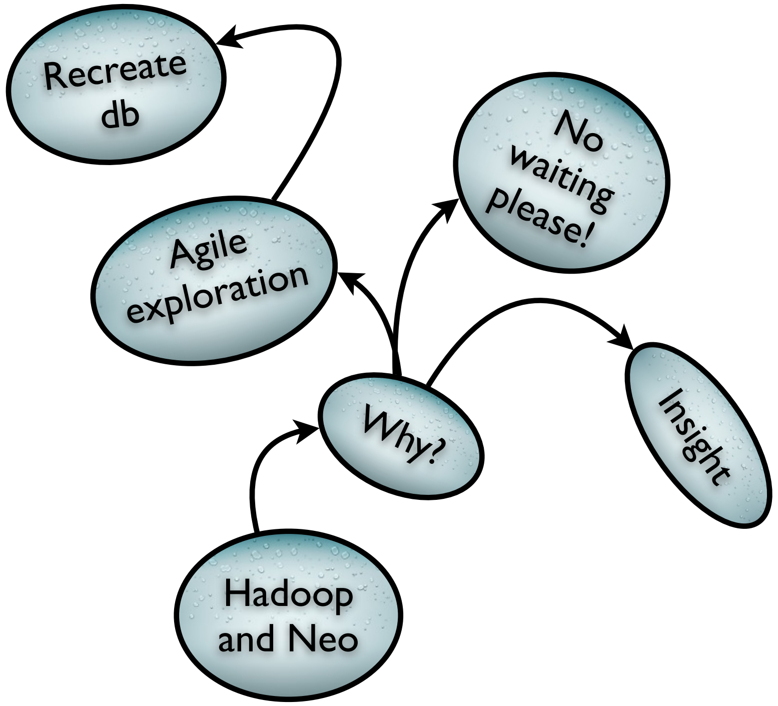
- Just add nodes and edges with:
  - Java api
  - Rest api
  - Cypher via rest
- Batch import functionality
- My way ;-)



Re-create db

Multiple ways to create a Neo4J database

Just adding nodes and edges has some problems.
Neo4J being transactional will give a lot of transaction overhead for large graphs.



Recreate db

Agile exploration

No waiting please!

Why?

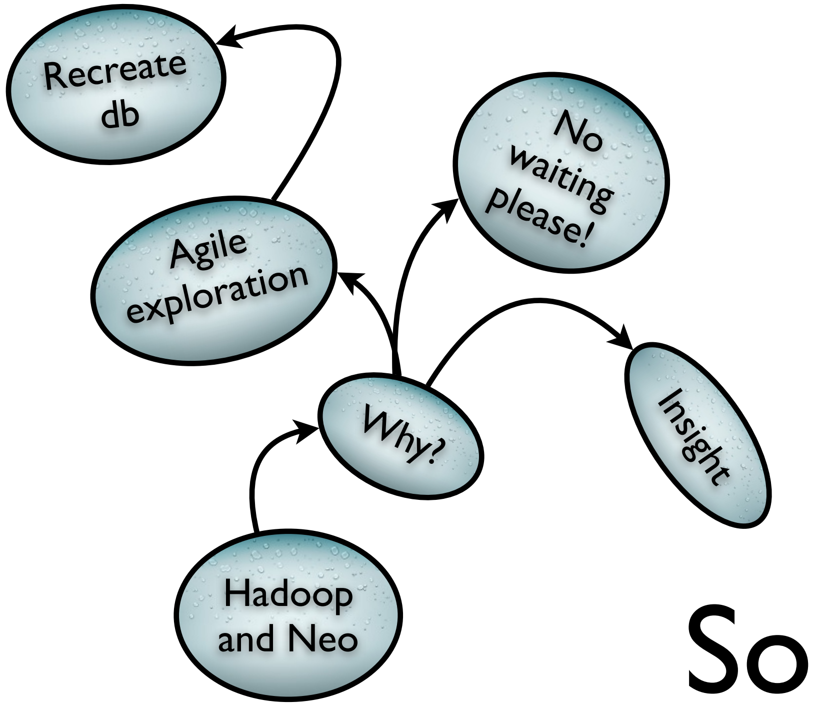
Insight

Hadoop and Neo

Re-create db

Multiple ways to create a Neo4J database

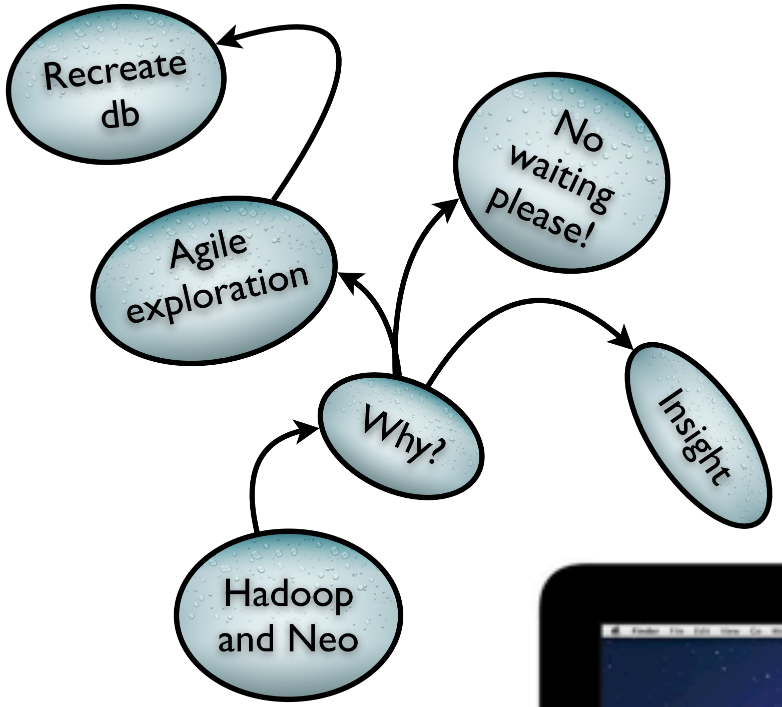
Batch import is non transactional so thats a good way to start



Re-create db

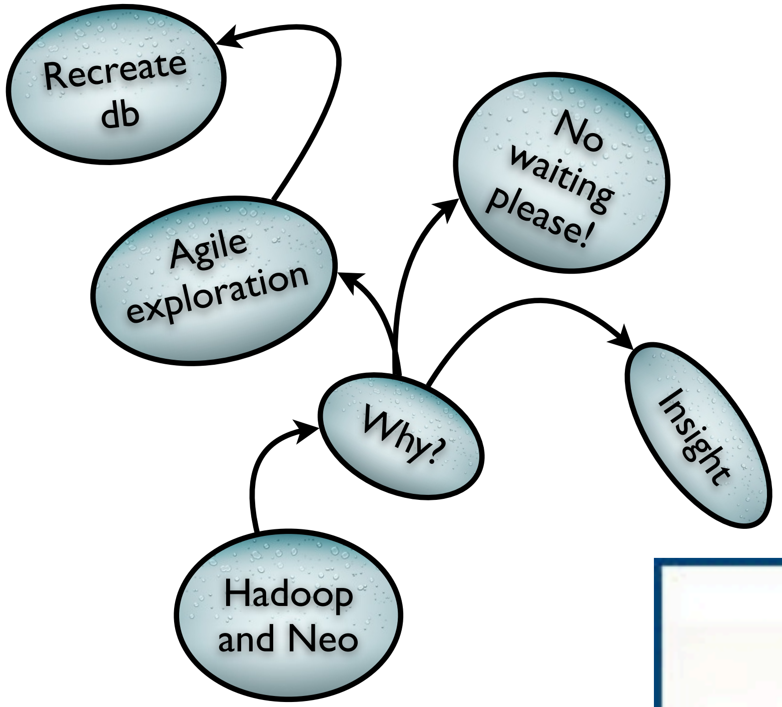
Some batch import code

```
String line;
while ((line = reader.readLine()) != null) {
   String[] parts = line.split(SPLIT_STRING);

   long fileNodeId = Long.parseLong(parts[0]);
   long nodeId;
   for (int c = 0; c < nodeFields.length; c++) {
      properties[1 + c * 2] = objectFromProperty(parts[c + 1].trim());
   }
   nodeId = db.createNode(map(properties));
   index.add(nodeId, map(properties));
}
```



No waiting please!

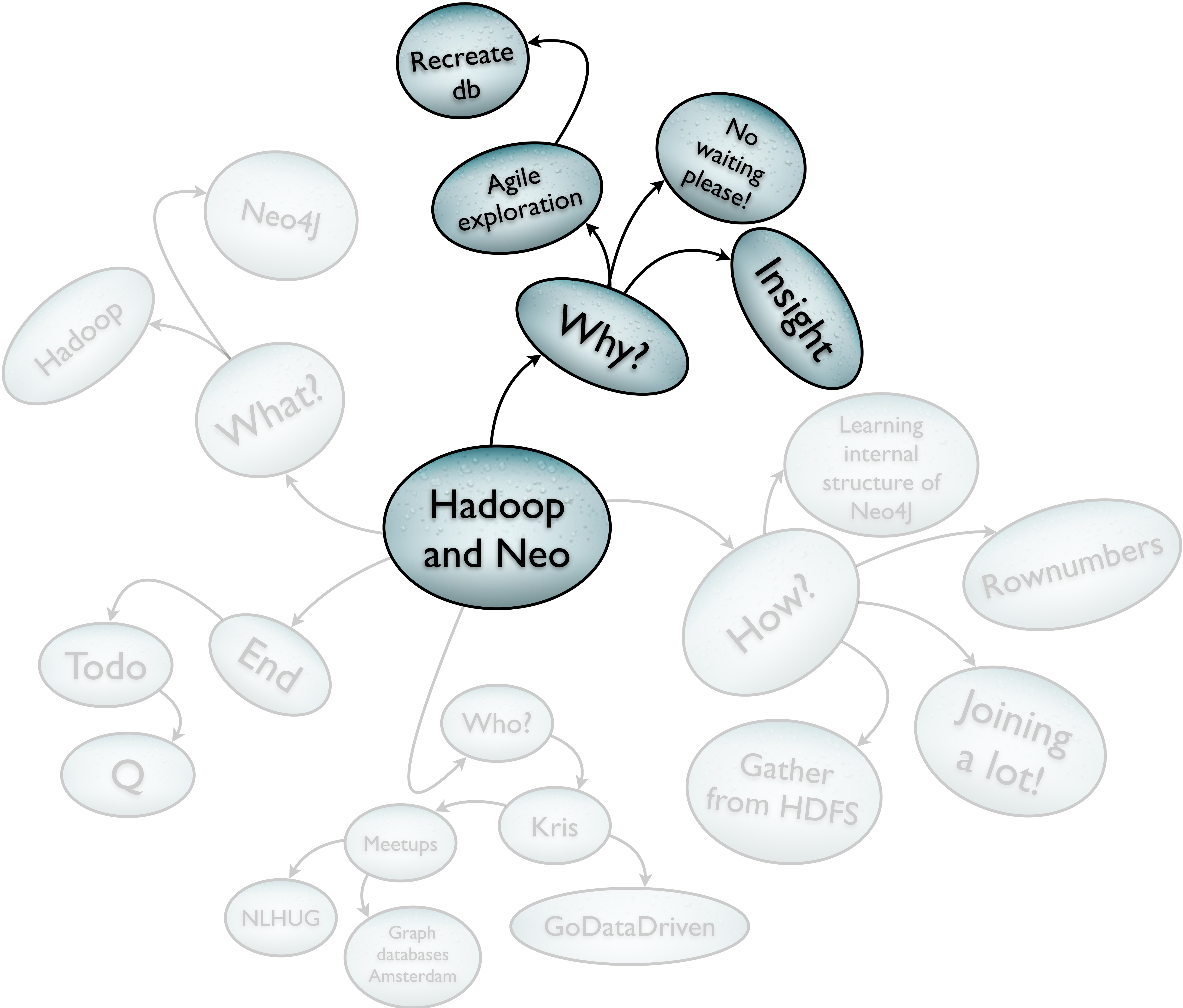
30 hours



No waiting please!

16 (+2) hours

code improved: 3 (+2) hours



Hadoop and Neo

Why?

Agile exploration

Recreate db

No waiting please!

Insight

What?

Hadoop

Neo4J

How?

Learning internal structure of Neo4J

Rownumbers

Joining a lot!

Gather from HDFS

Who?

Kris

GoDataDriven

Meetups

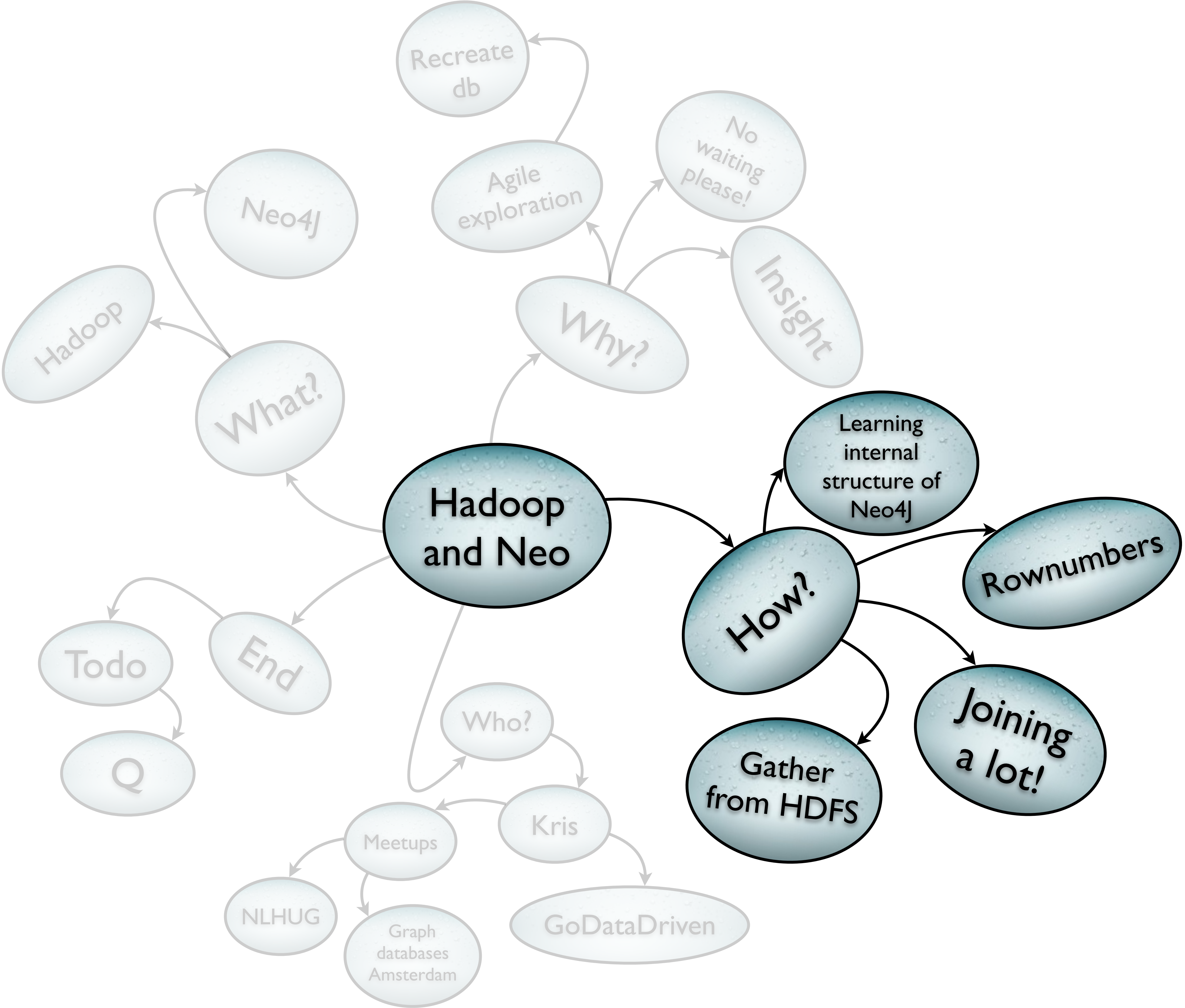
NLHUG

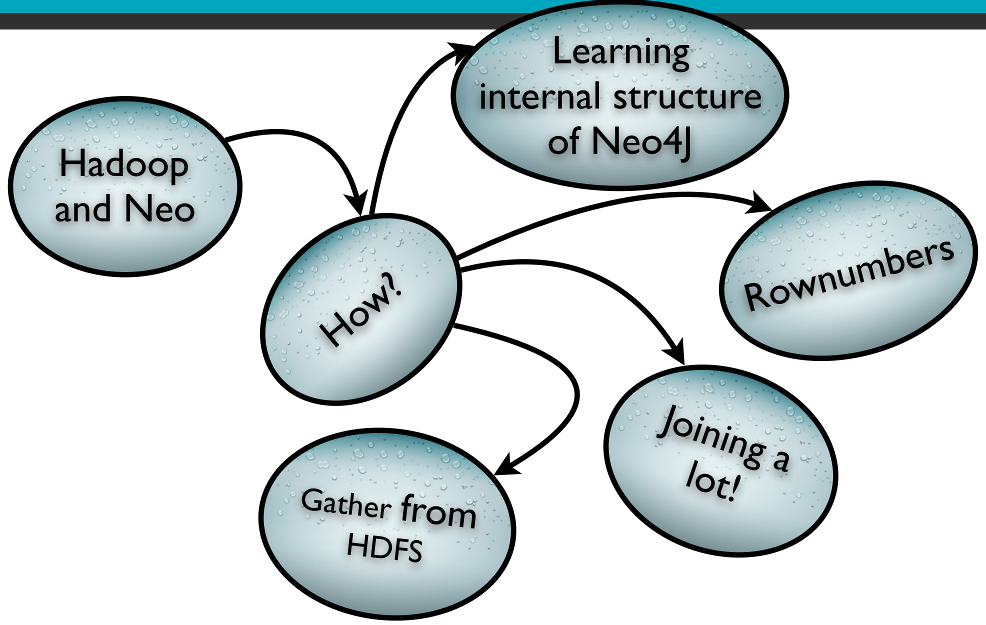
Graph databases Amsterdam

End

Todo

Q


Hadoop and Neo
How?
Learning internal structure of Neo4J
Rownumbers
Joining a lot!
Gather from HDFS
Why?
Agile exploration
Recreate db
No waiting please!
Insight
What?
Hadoop
Neo4J
End
Todo
Q
Who?
Kris
Meetups
NLHUG
Graph databases Amsterdam
GoDataDriven


Hadoop and Neo

How?

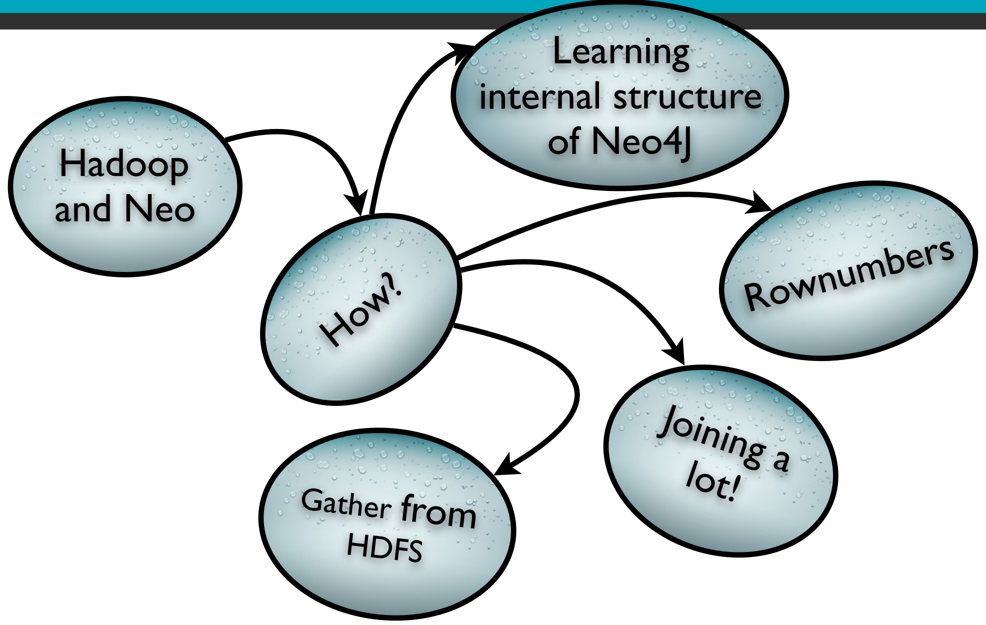
Learning internal structure of Neo4J

Rownumbers

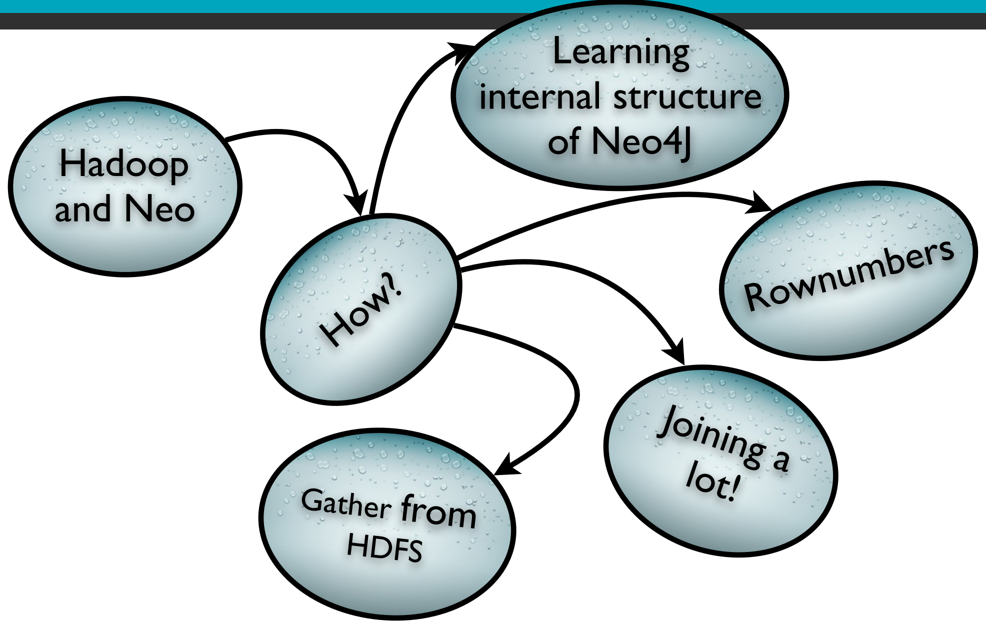
Joining a lot!

Gather from HDFS

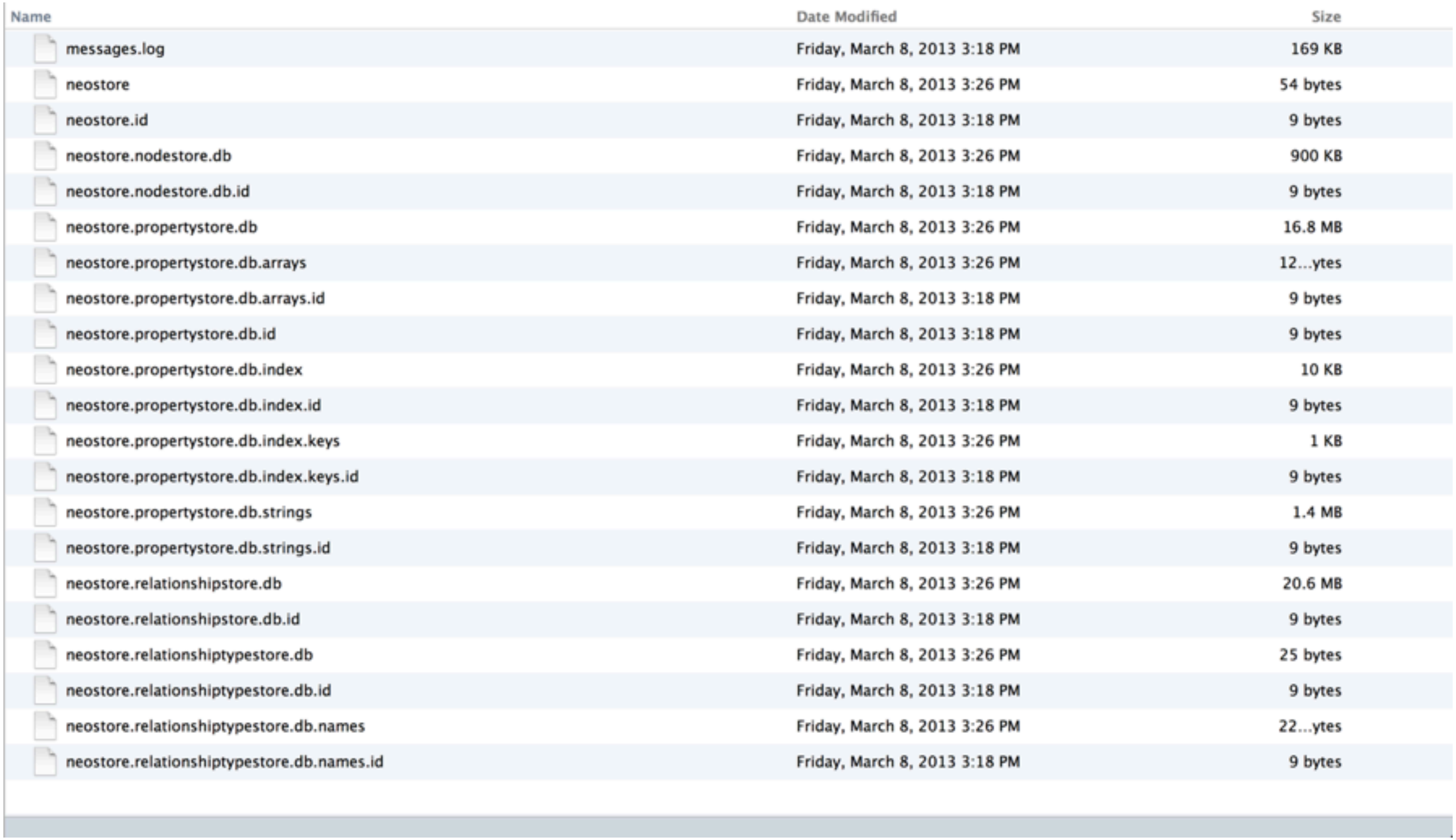
Using Neo4J


Hadoop and Neo
How?
Learning internal structure of Neo4J
Rownumbers
Joining a lot!
Gather from HDFS

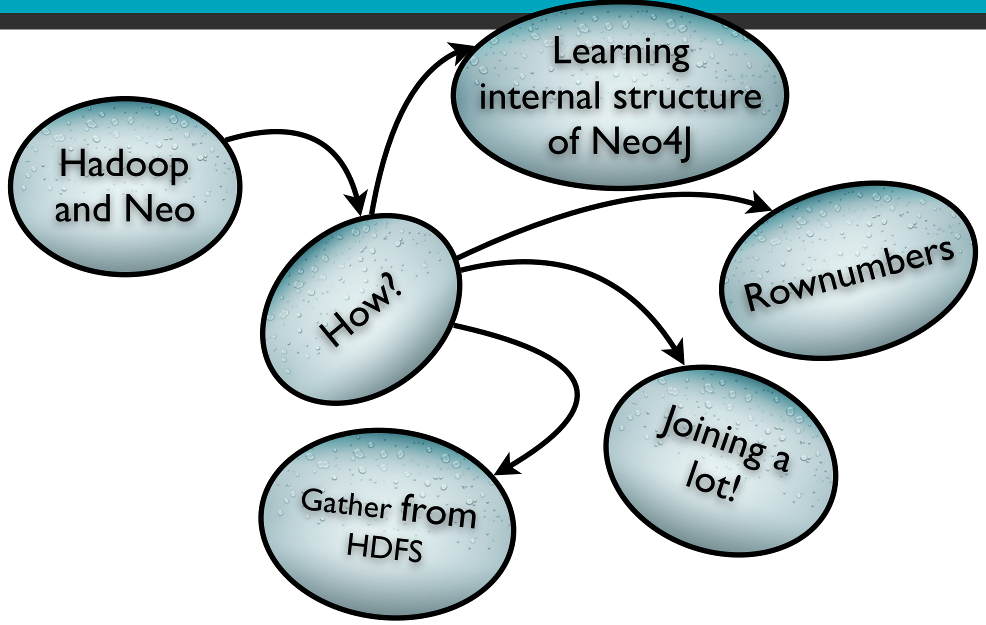
Using Neo4J



Learning internal structure of Neo4J



Name	Date Modified	Size
messages.log	Friday, March 8, 2013 3:18 PM	169 KB
neostore	Friday, March 8, 2013 3:26 PM	54 bytes
neostore.id	Friday, March 8, 2013 3:18 PM	9 bytes
neostore.nodestore.db	Friday, March 8, 2013 3:26 PM	900 KB
neostore.nodestore.db.id	Friday, March 8, 2013 3:18 PM	9 bytes
neostore.propertystore.db	Friday, March 8, 2013 3:26 PM	16.8 MB
neostore.propertystore.db.arrays	Friday, March 8, 2013 3:26 PM	12...ytes
neostore.propertystore.db.arrays.id	Friday, March 8, 2013 3:18 PM	9 bytes
neostore.propertystore.db.id	Friday, March 8, 2013 3:18 PM	9 bytes
neostore.propertystore.db.index	Friday, March 8, 2013 3:26 PM	10 KB
neostore.propertystore.db.index.id	Friday, March 8, 2013 3:18 PM	9 bytes
neostore.propertystore.db.index.keys	Friday, March 8, 2013 3:26 PM	1 KB
neostore.propertystore.db.index.keys.id	Friday, March 8, 2013 3:18 PM	9 bytes
neostore.propertystore.db.strings	Friday, March 8, 2013 3:26 PM	1.4 MB
neostore.propertystore.db.strings.id	Friday, March 8, 2013 3:18 PM	9 bytes
neostore.relationshipstore.db	Friday, March 8, 2013 3:26 PM	20.6 MB
neostore.relationshipstore.db.id	Friday, March 8, 2013 3:18 PM	9 bytes
neostore.relationshiptypestore.db	Friday, March 8, 2013 3:26 PM	25 bytes
neostore.relationshiptypestore.db.id	Friday, March 8, 2013 3:18 PM	9 bytes
neostore.relationshiptypestore.db.names	Friday, March 8, 2013 3:26 PM	22...ytes
neostore.relationshiptypestore.db.names.id	Friday, March 8, 2013 3:18 PM	9 bytes



Hadoop and Neo

How?

Learning internal structure of Neo4J

Rownumbers

Joining a lot!

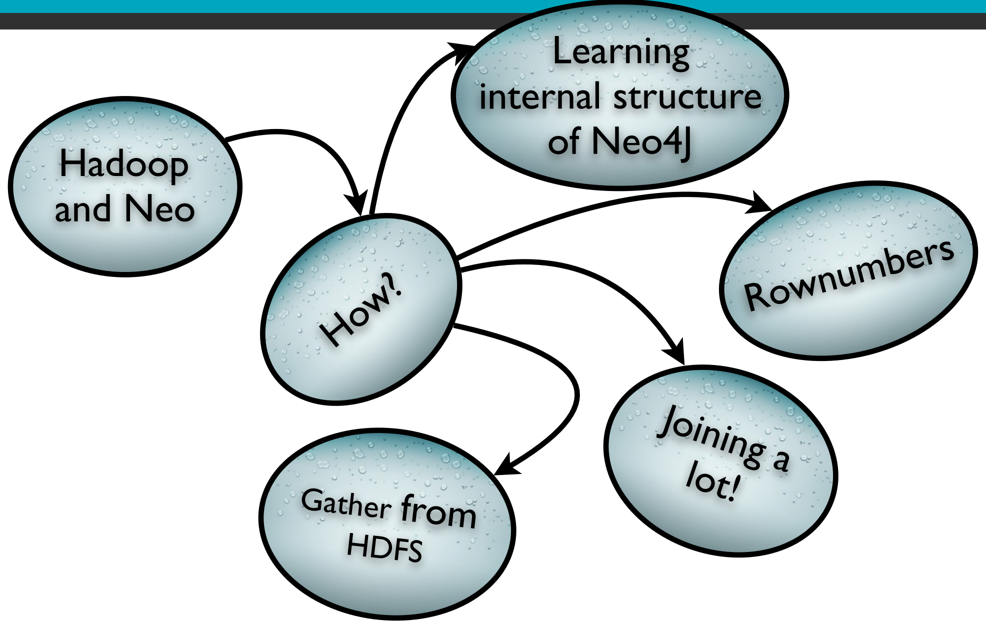
Gather from HDFS

Learning internal structure of Neo4J

Lucky most work already done by

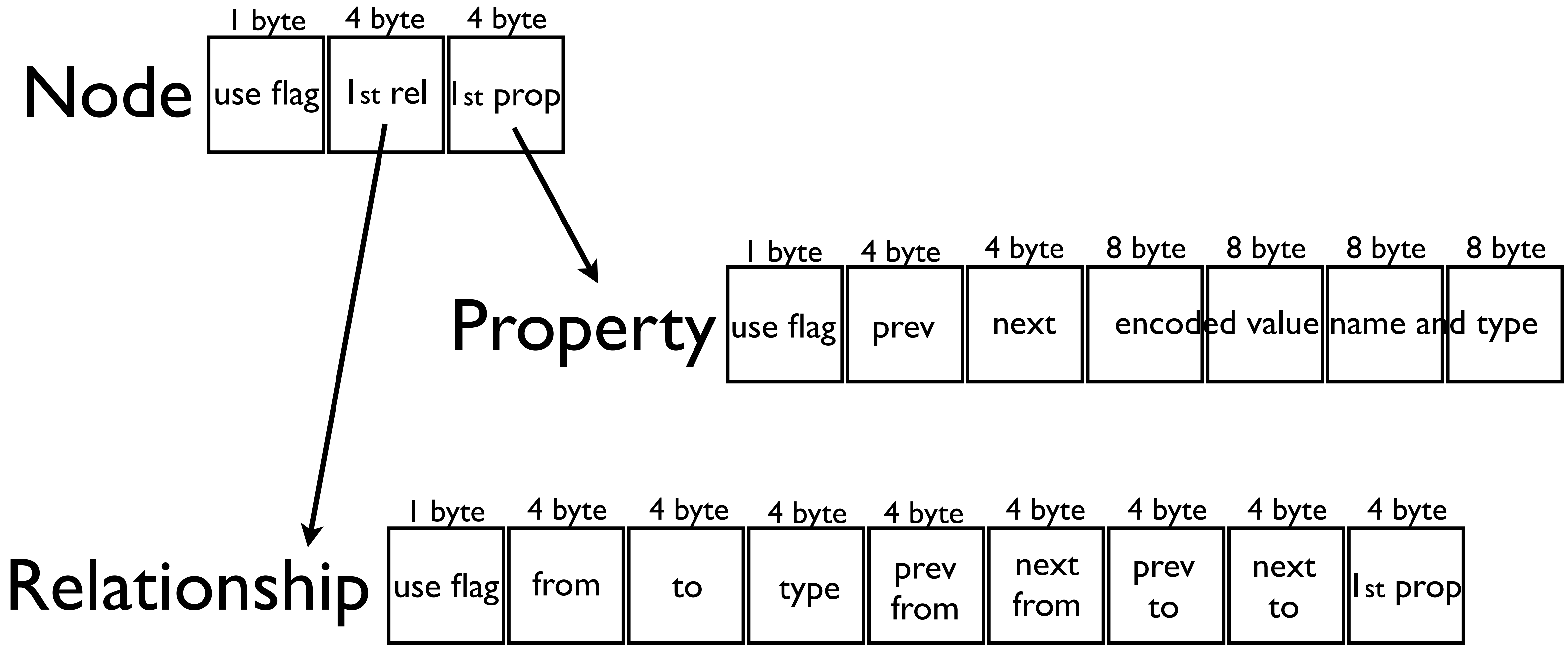
Chris Gioran

In essence it’s all: doubly linked lists sequentially stored on disk



Learning internal structure of Neo4J

In essence it's all: doubly linked lists sequentially stored on disk



Node

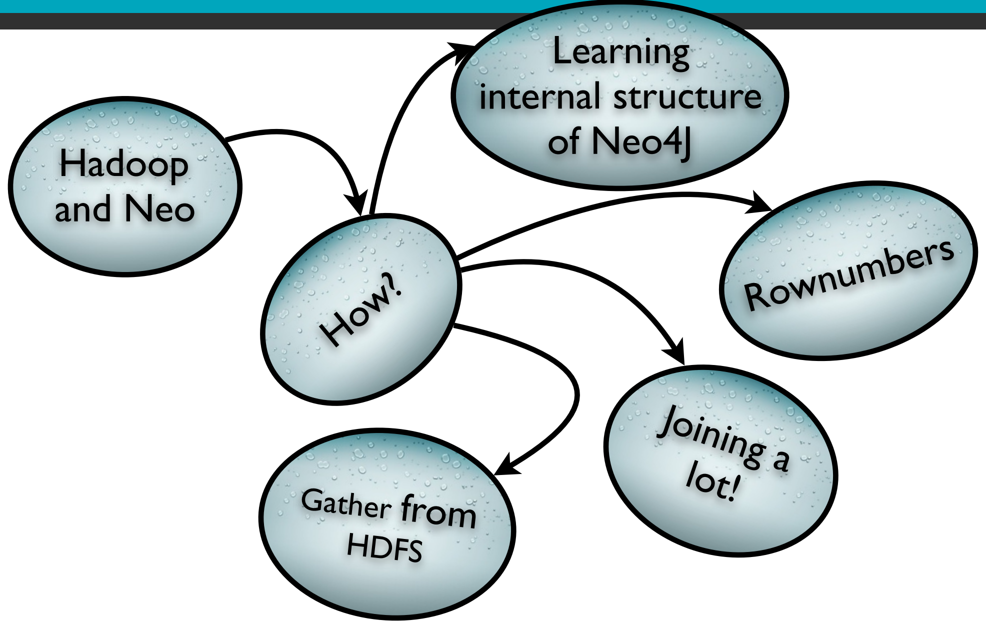
1 byte	4 byte	4 byte
use flag	1st rel	1st prop

Property

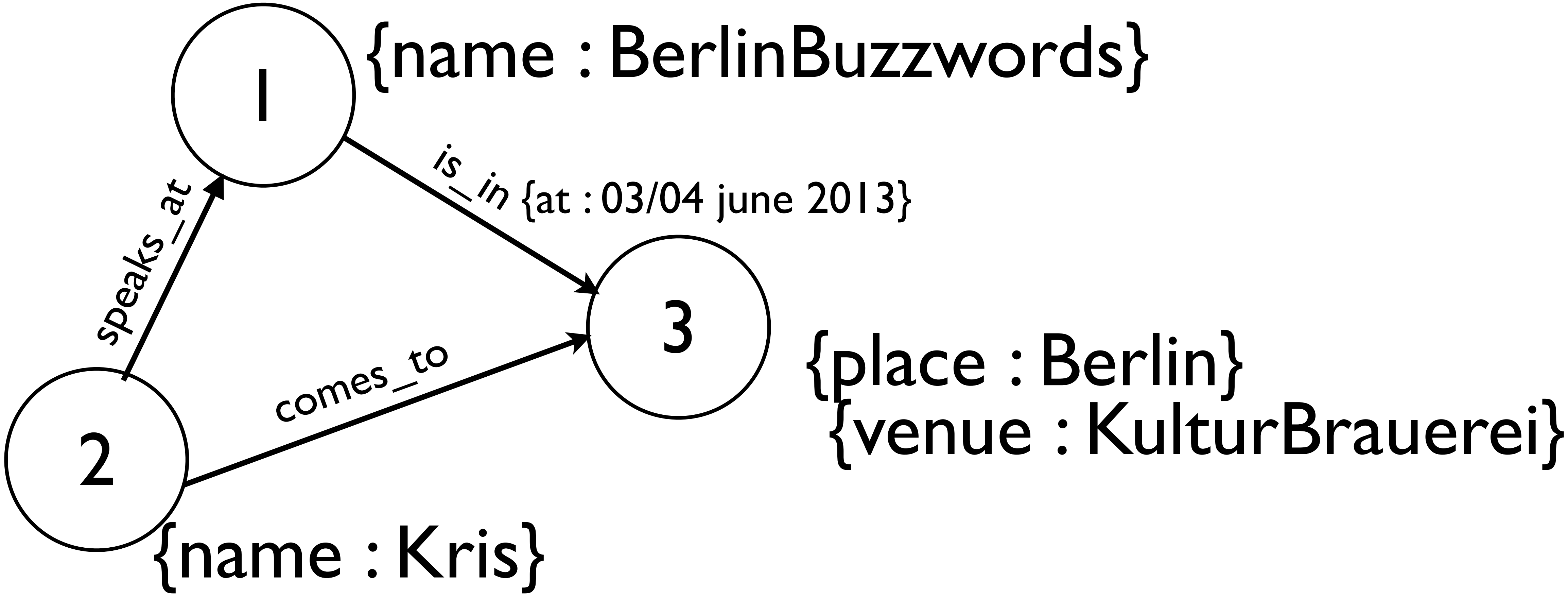
1 byte	4 byte	4 byte	8 byte	8 byte	8 byte	8 byte
use flag	prev	next	encoded value		name and type	

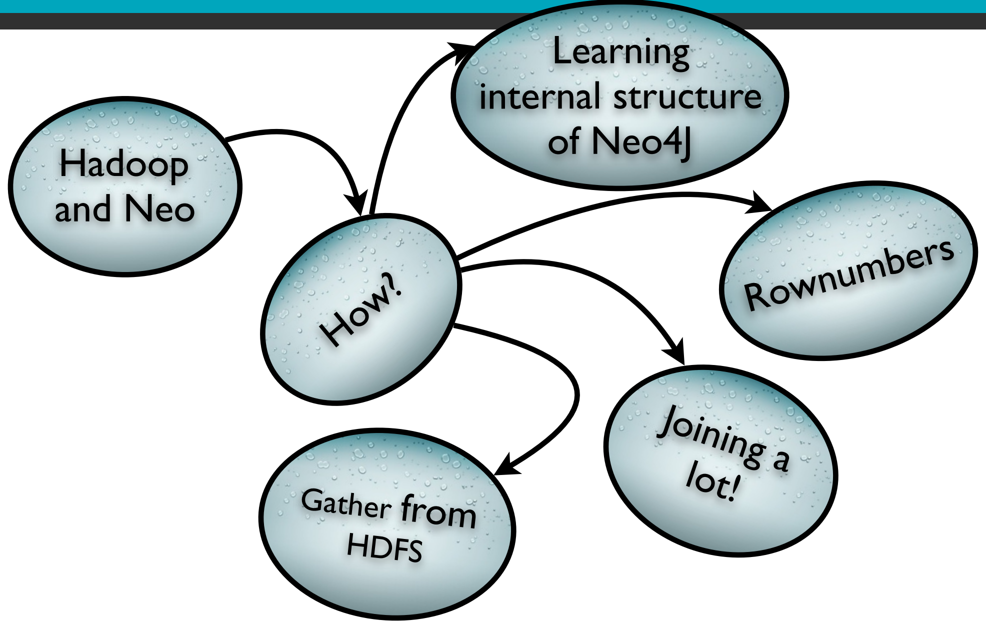
Relationship

1 byte	4 byte	4 byte	4 byte	4 byte	4 byte	4 byte	4 byte	4 byte
use flag	from	to	type	prev from	next from	prev to	next to	1st prop



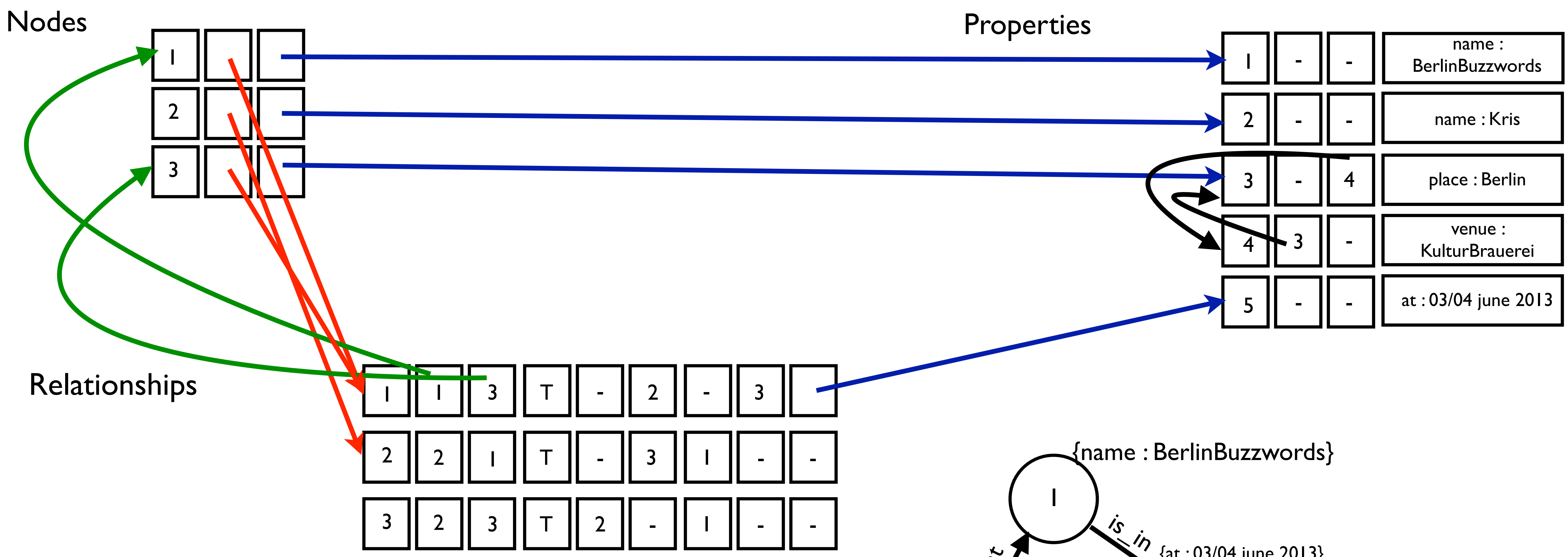
Learning internal structure of Neo4J





Learning internal structure of Neo4J

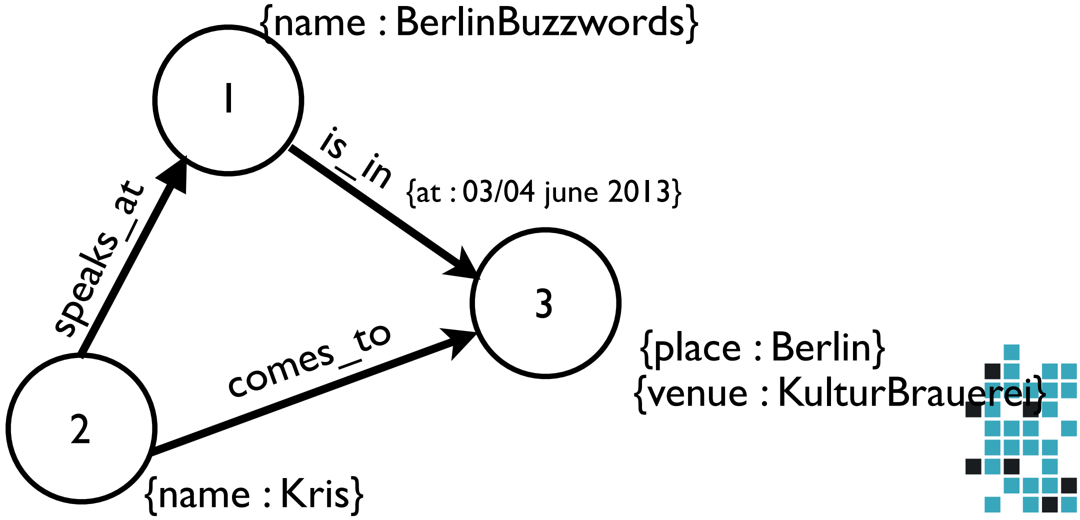
In essence it's all: doubly linked lists sequentially stored on disk

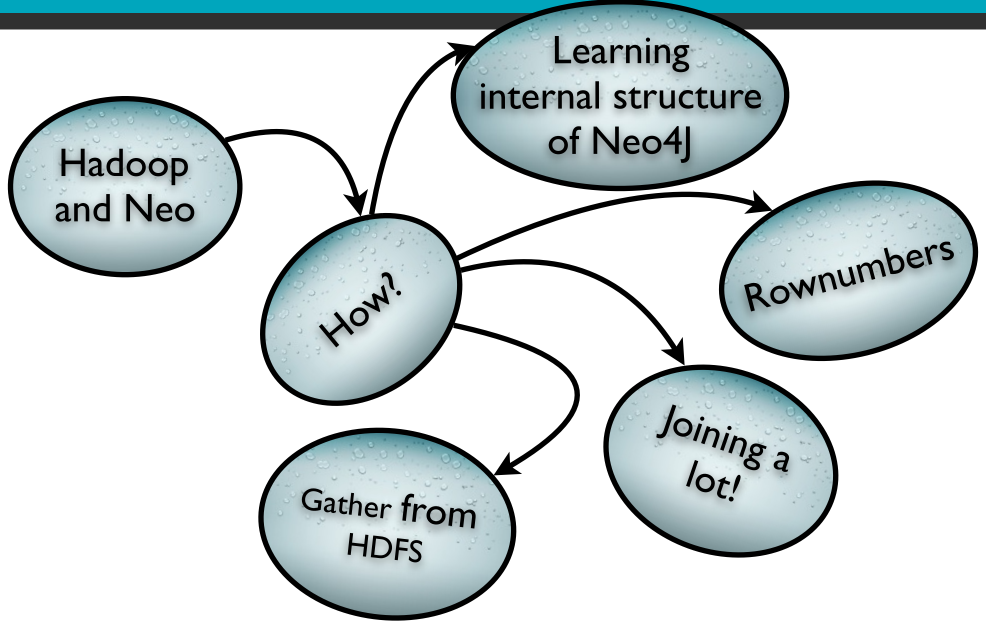


1		
2		
3		

1	-	-	name : BerlinBuzzwords
2	-	-	name : Kris
3	-	4	place : Berlin
4	3	-	venue : KulturBrauerei
5	-	-	at : 03/04 june 2013

1	1	3	T	-	2	-	3	
2	2	1	T	-	3	1	-	-
3	2	3	T	2	-	1	-	-

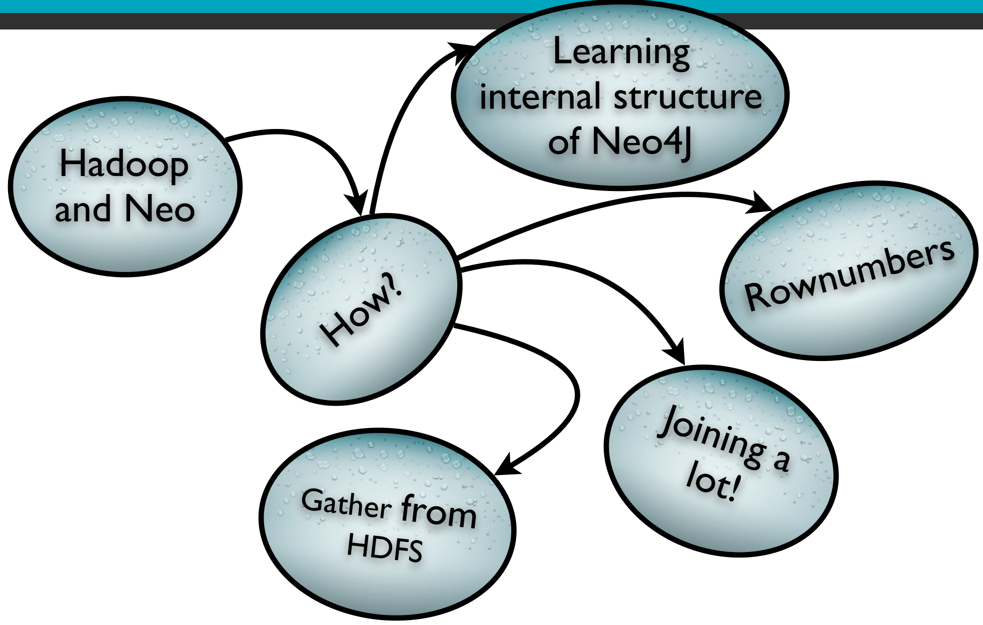




Rownumbers

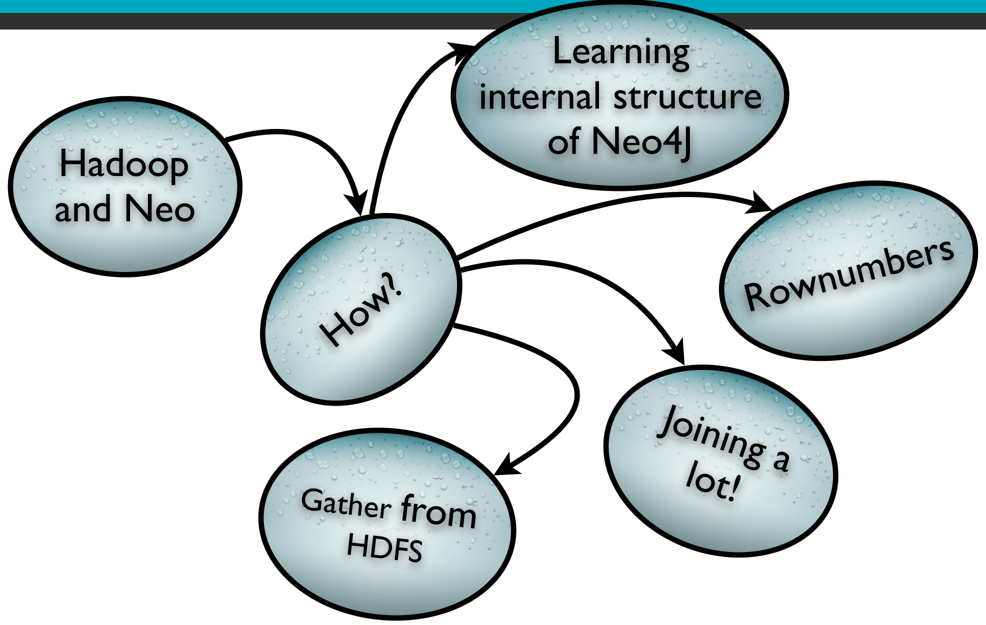
```
01 ff ff ff ff ff ff ff ff 01 00 00 01 ec 00 00
00 00 01 00 00 00 67 00 00 00 01 01 00 00 00 7e
00 00 00 02 01 00 00 00 02 00 00 00 03 01 00 00
00 08 00 00 00 04 01 00 00 00 03 00 00 00 05 01
00 00 00 04 00 00 00 07 01 00 00 02 5c 00 00 00
08 01 00 00 01 f6 00 00 00 09 01 00 00 00 08 00
00 00 0a 01 00 00 00 0c 00 00 00 0b 01 00 00 00
52 00 00 00 0c 01 00 00 00 24 00 00 00 0d 01 00
00 00 0c 00 00 00 0e 01 00 00 02 00 00 00 00 0f
01 00 00 00 0e 00 00 00 10 01 00 00 00 10 00 00
```

Position in the file matters



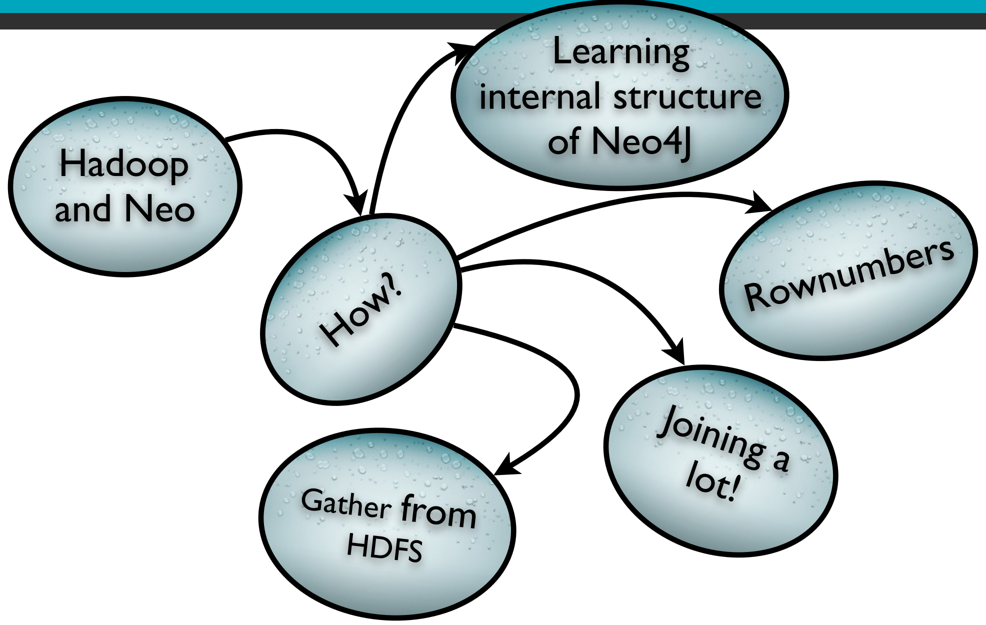
Rownumbers

We need a rownumber generator

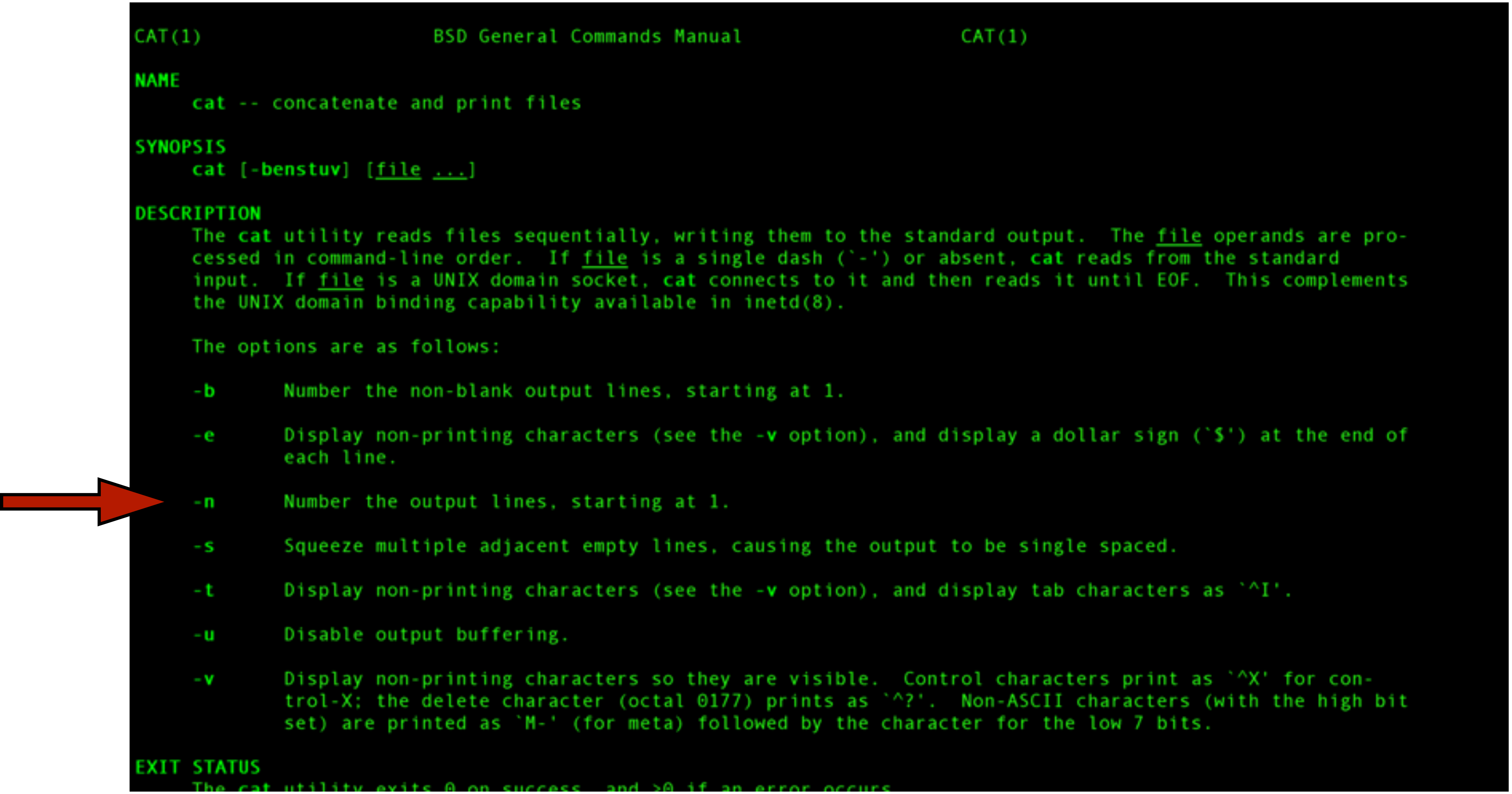


Rownumbers

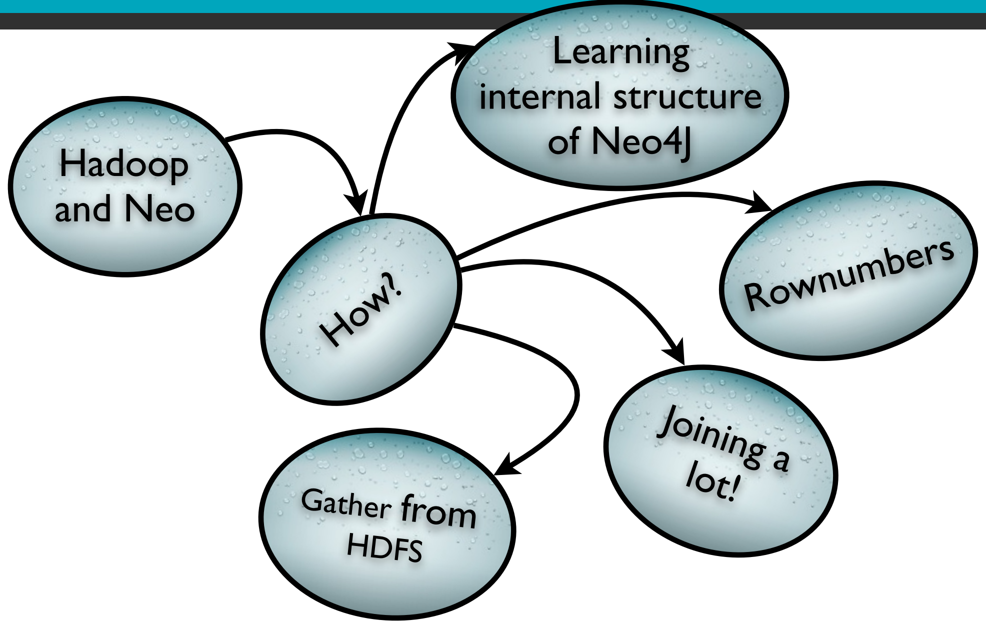
*nix has one build into `cat`



Rownumbers

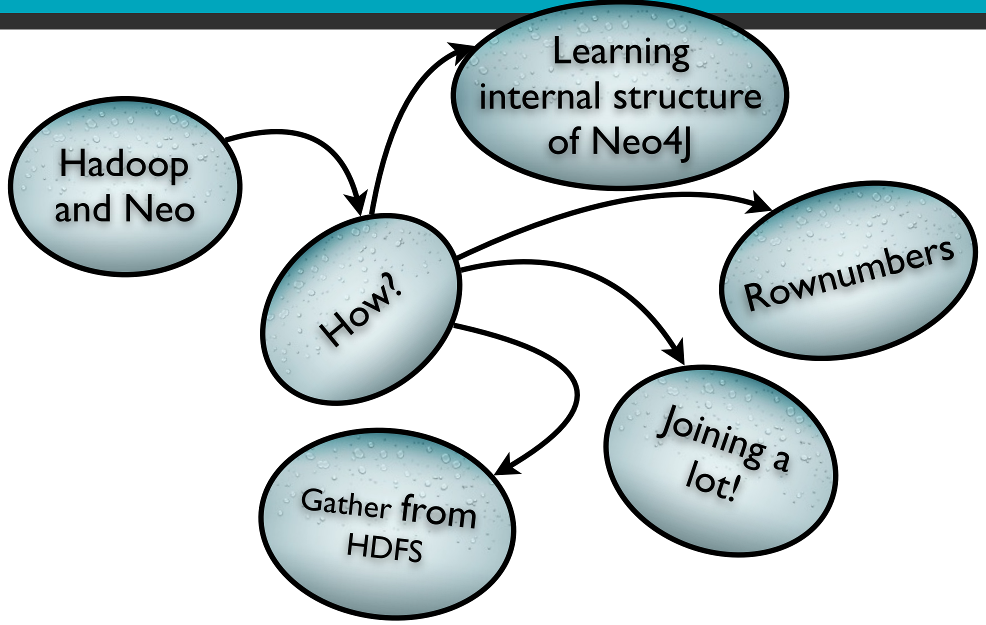


```
CAT(1)                    BSD General Commands Manual                   CAT(1)

NAME
     cat -- concatenate and print files

SYNOPSIS
     cat [-benstuv] [file ...]

DESCRIPTION
     The cat utility reads files sequentially, writing them to the standard output.  The file operands are pro-
     cessed in command-line order.  If file is a single dash (`-') or absent, cat reads from the standard
     input.  If file is a UNIX domain socket, cat connects to it and then reads it until EOF.  This complements
     the UNIX domain binding capability available in inetd(8).

     The options are as follows:

     -b      Number the non-blank output lines, starting at 1.

     -e      Display non-printing characters (see the -v option), and display a dollar sign (`$') at the end of
             each line.

     -n      Number the output lines, starting at 1.

     -s      Squeeze multiple adjacent empty lines, causing the output to be single spaced.

     -t      Display non-printing characters (see the -v option), and display tab characters as `^I'.

     -u      Disable output buffering.

     -v      Display non-printing characters so they are visible.  Control characters print as `^X' for con-
             trol-X; the delete character (octal 0177) prints as `^?'.  Non-ASCII characters (with the high bit
             set) are printed as `M-' (for meta) followed by the character for the low 7 bits.

EXIT STATUS
     The cat utility exits 0 on success, and >0 if an error occurs.
```



Rownumbers

We need a distributed 'cat -n'



Rownumbers

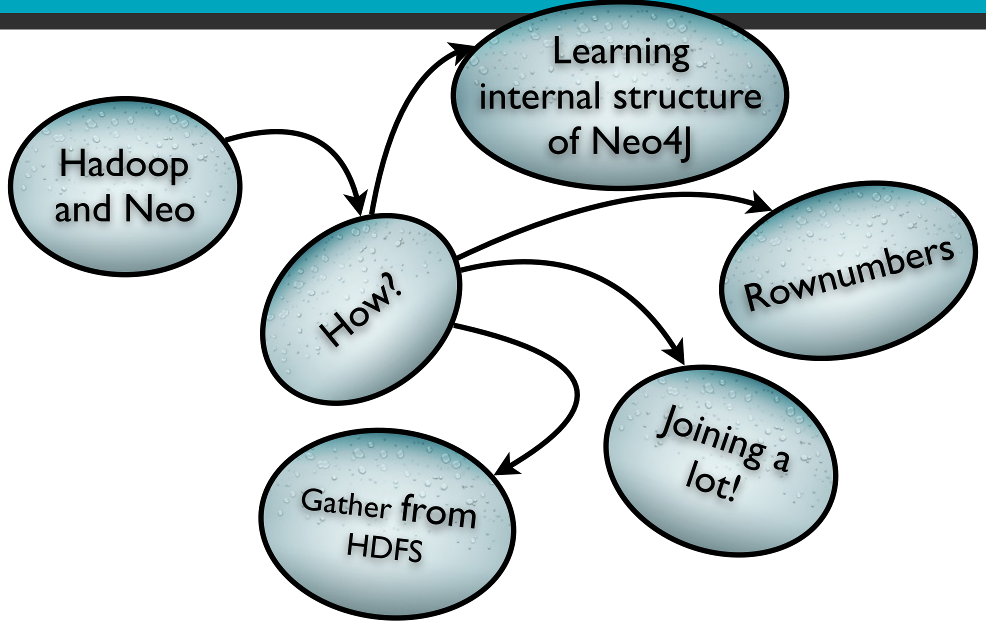
We need a distributed ‘cat -n’

to convert:

ABC
DEF
GHI
JKL
MNO
PQR
STU
VWX
YZ0

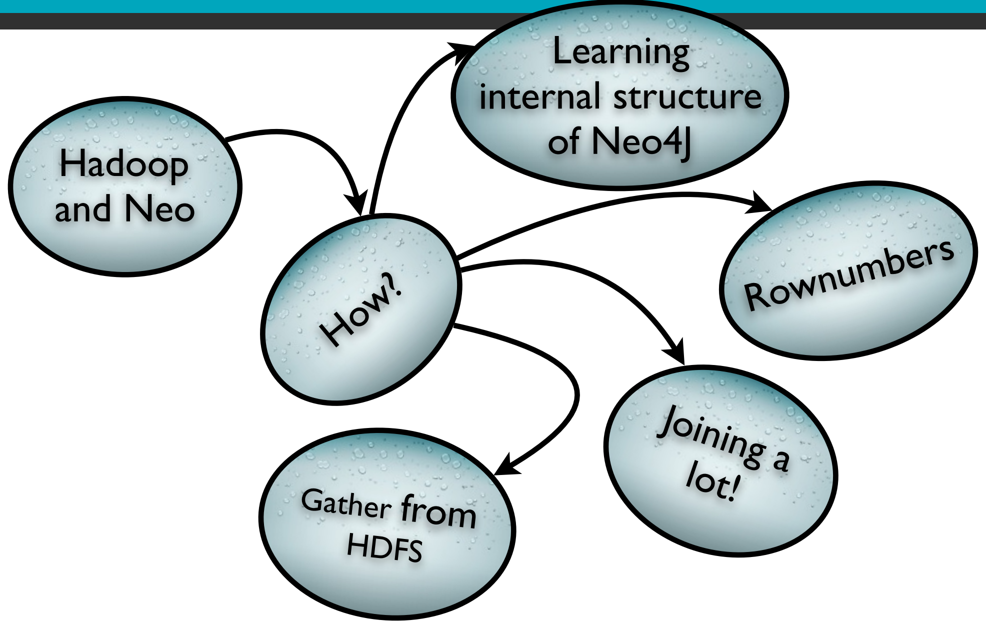
into:

0	ABC
1	DEF
2	GHI
3	JKL
4	MNO
5	PQR
6	STU
7	VWX
8	YZ0



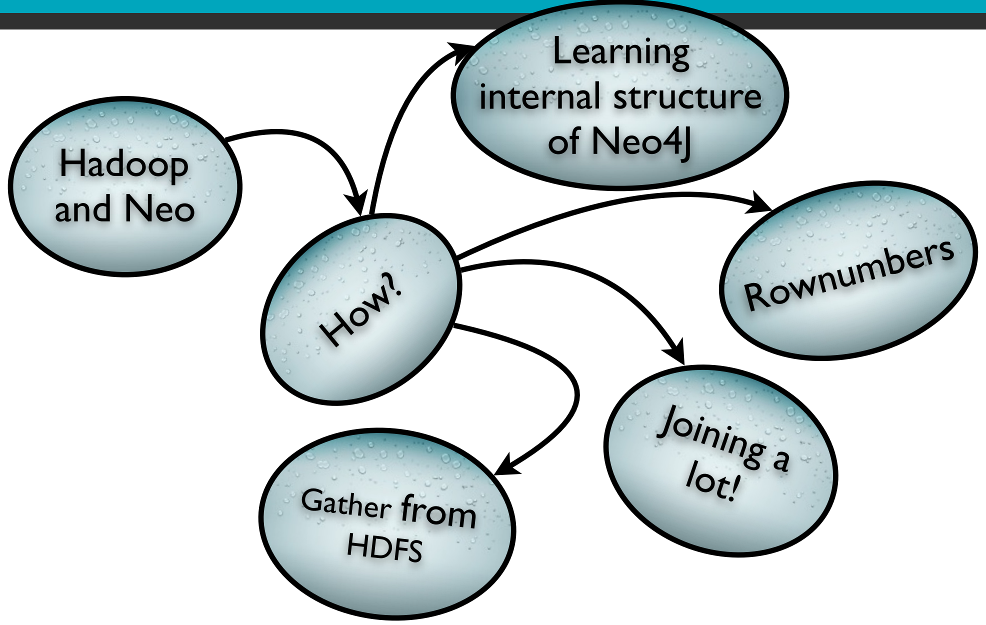
Rownumbers

Easy way out is a 1 (one) reducer job doing the numbering. But that’s no fun right!



Rownumbers

1. Run the data through a mapper and have the mapper emit each record AS IS
2. Have each mapper keep track of how many records it sends to each reducer
3. Have each reducer process the count records and accumulate the counts from each mapper it receives
4. Have each reducer emit each record prepended by a row ID starting the ID sequence at the number calculated

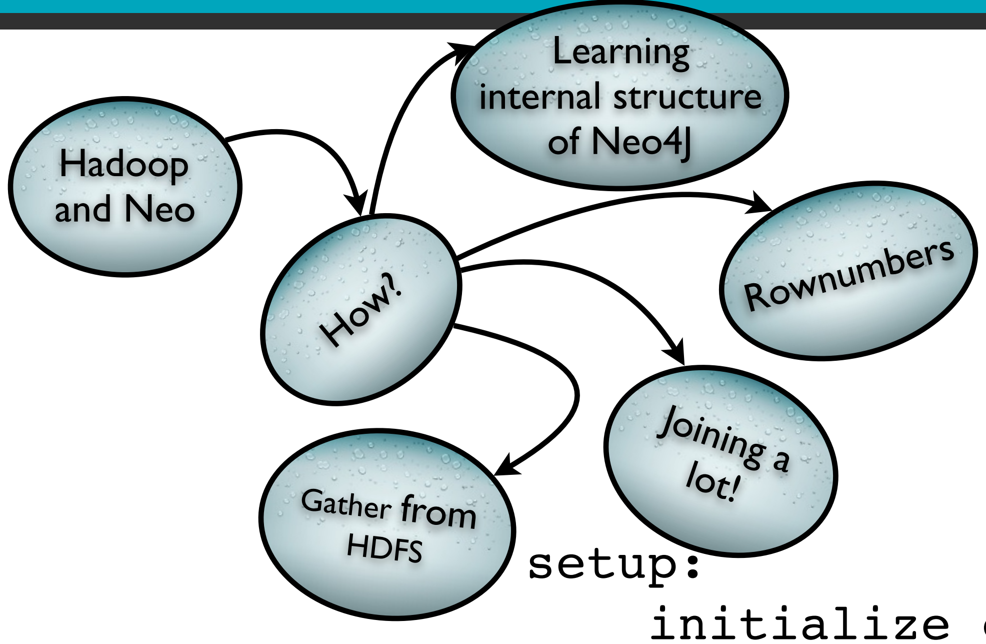


Rownumbers

1. Run the data through a mapper and have the mapper emit each record AS IS
2. Have each mapper keep track of how many records it sends to each reducer

mapper:

```
setup:
    initialize counters

map:
    read input ==> emit as is, increment the correct counter

cleanup:
    emit all counters
```



Rownumbers

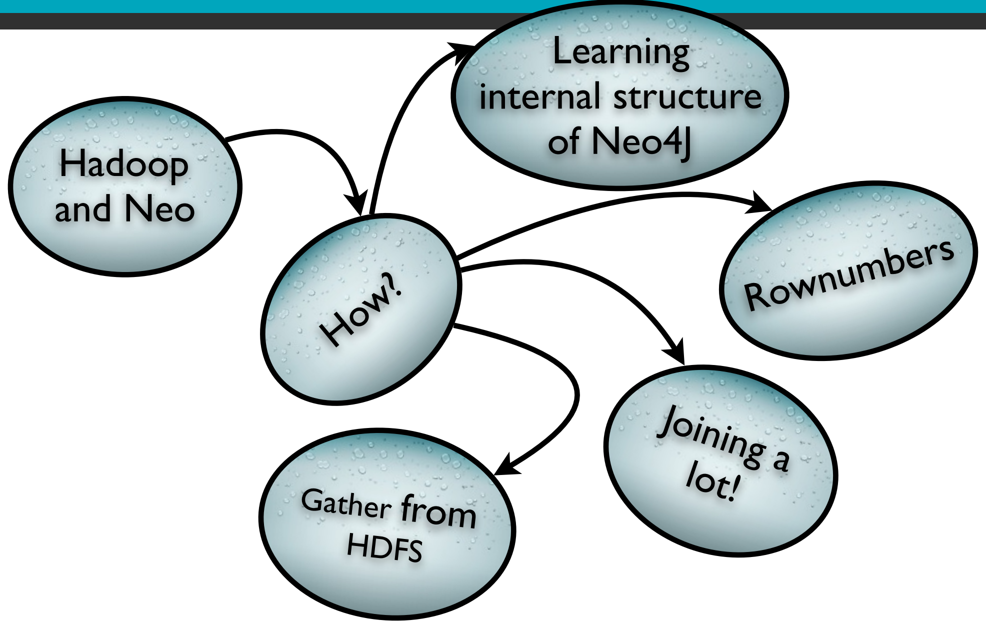
ABC (hash=0)
DEF (hash=1)
GHI (hash=2)
JKL (hash=0)
MNO (hash=1)
------------- split
PQR (hash=2)
STU (hash=0)
VWX (hash=1)
YZ0 (hash=1)

mapper 0:

```
setup:
    initialize counters = [0, 0, 0]
map:
    input "ABC" ==> emit "ABC", increment counters[0]
    input "DEF" ==> emit "DEF", increment counters[1]
    input "GHI" ==> emit "GHI", increment counters[2]
    input "JKL" ==> emit "JKL", increment counters[0]
    input "MNO" ==> emit "MNO", increment counters[1]   //now [2, 2, 1]
cleanup:
    emit counter for partition 0: none
    emit counter for partition 1: counters[0] = 2
    emit counter for partition 2: counters[0] + counters[1] = 4
```

mapper 1:

```
setup:
    initialize counters = [0, 0, 0]
map:
    input "PQR" ==> emit "PQR", increment counters[2]
    input "STU" ==> emit "STU", increment counters[0]
    input "VWX" ==> emit "VWX", increment counters[1]
    input "YZ0" ==> emit "YZ0", increment counters[1]   //now [1, 2, 1]
cleanup:
    emit counter for partition 0: none
    emit counter for partition 1: counters[0] = 1
    emit counter for partition 2: counters[0] + counters[1] = 3
```

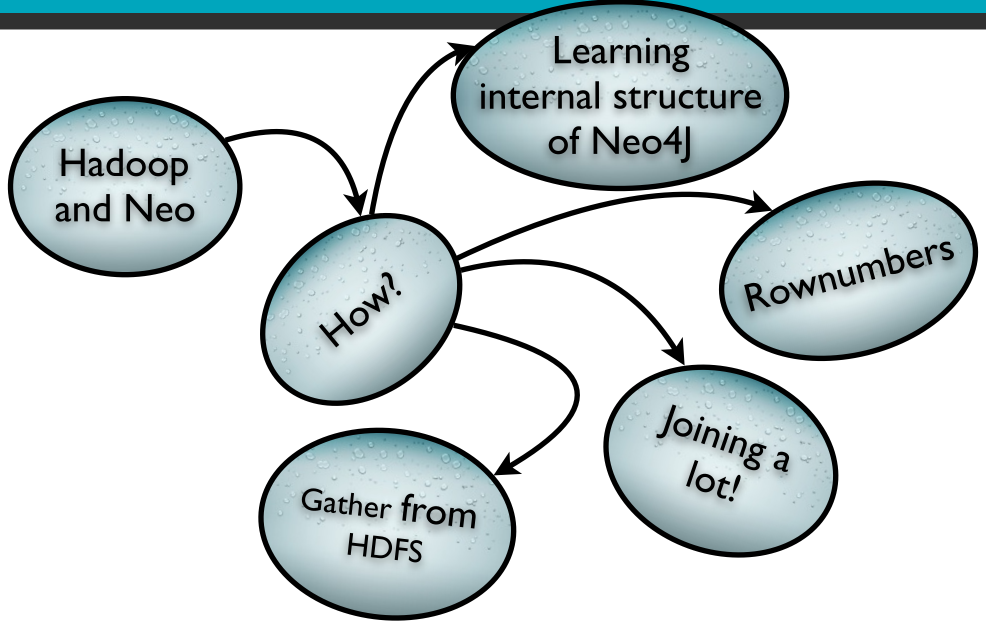


Rownumbers

1. Have each reducer process the count records and accumulate the counts from each mapper it receives
2. Have each reducer emit each record prepended by a row ID starting the ID sequence at the number calculated

reducer:

```
reduce:
    initialize offset = 0
    calculate offset to start numbering based on all counter records from the mappers
    read input ==> emit current offset + input, increment offset
```



Rownumbers

reducer 0:

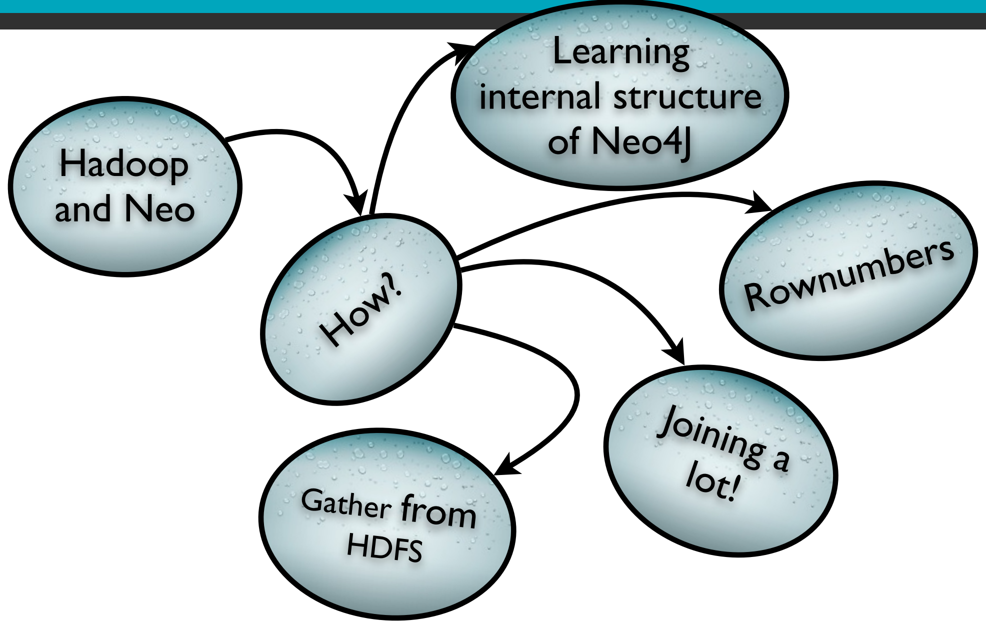
```
reduce:
    initialize offset = 0
    input "ABC" ==> emit "offset <tab> ABC", increment offset   //emits 0<tab>ABC
    input "JKL" ==> emit "offset <tab> JKL", increment offset   //emits 1<tab>ABC, and so on...
    input "STU" ==> emit "offset <tab> STU", increment offset   //last emitted offset is 2
```

reducer 1:

```
reduce:
    initialize offset = 0
    input counter with value 2 ==> offset = offset + 2
    input counter with value 1 ==> offset = offset + 1  //now offset == 3
    input "DEF" ==> emit "offset <tab> DEF", increment offset   //emits 3<tab>DEF
    input "MNO" ==> emit "offset <tab> MNO", increment offset   //emits 4<tab>MNO, and so on...
    input "VWX" ==> emit "offset <tab> VWX", increment offset
    input "XY0" ==> emit "offset <tab> XY0", increment offset   //last emitted offset is 6
```

reducer 2:

```
reduce:
    initialize offset = 0
    input counter with value 4 ==> offset = offset + 4
    input counter with value 3 ==> offset = offset + 3  //now offset == 7
    input "GHI" ==> emit "offset <tab> GHI", increment offset   //emits 7<tab>GHI
    input "PQR" ==> emit "offset <tab> PQR", increment offset   //emits 8<tab>PQR
```



Hadoop and Neo

How?

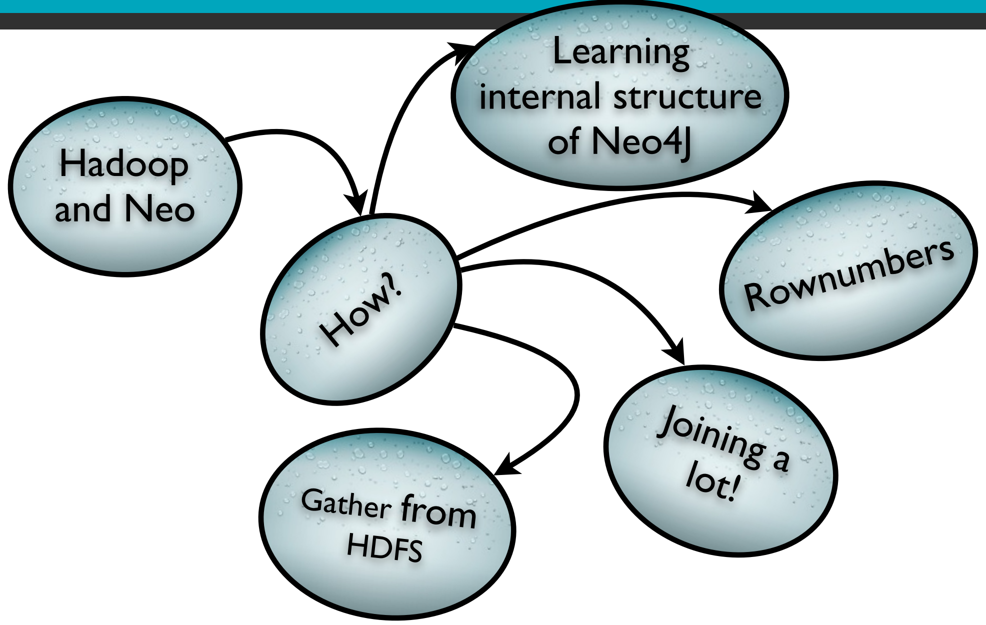
Learning internal structure of Neo4J

Rownumbers

Joining a lot!

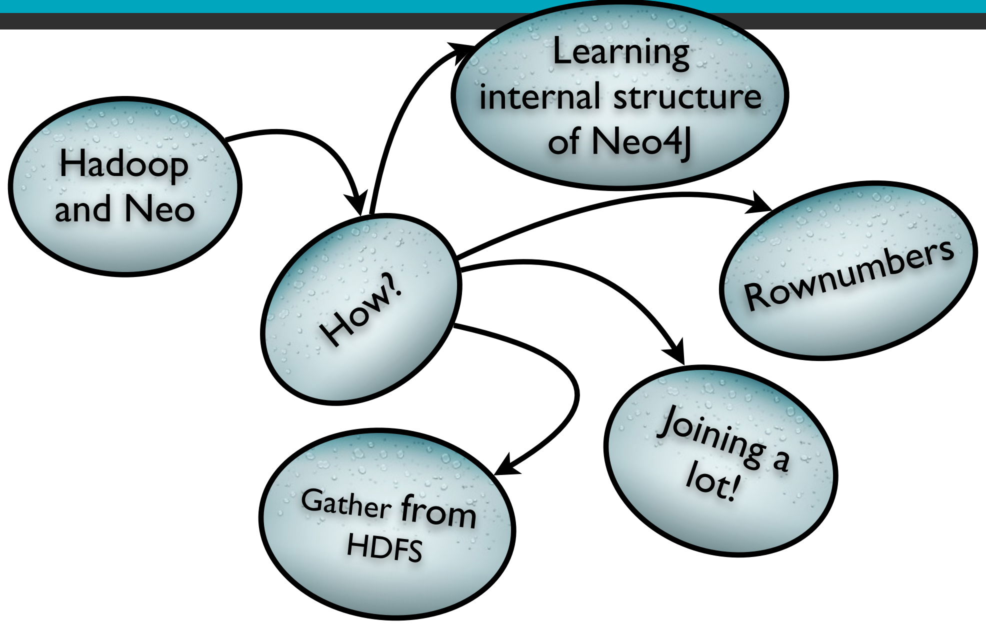
Gather from HDFS

Rownumbers



Rownumbers

Is that all?



Rownumbers

Is that all?

No!

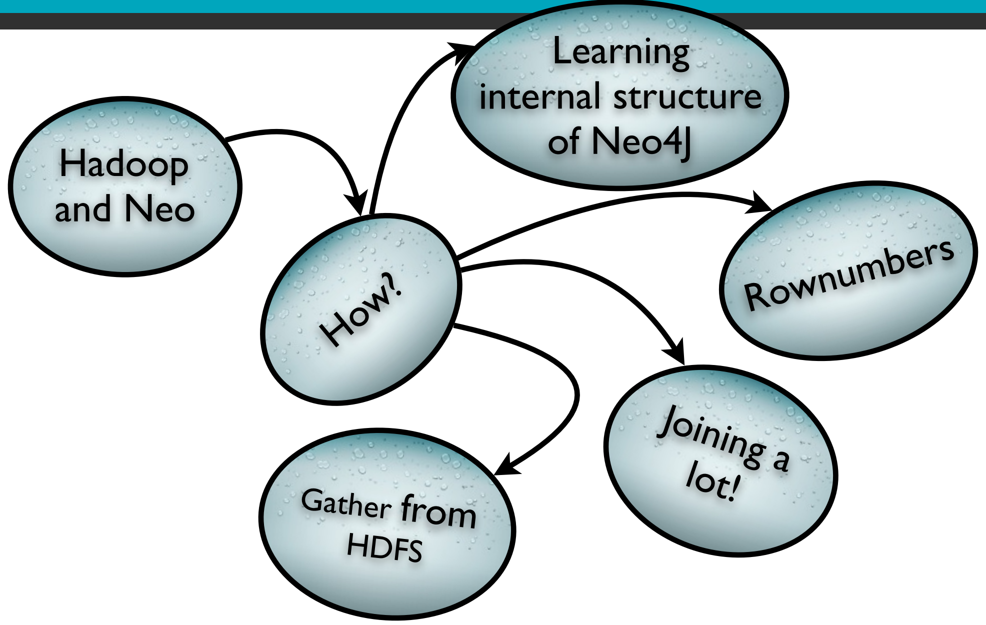
A custom partitioner:

```
public static class Partitioner extends Partitioner<ByteWritable, RowNumberWritable> {
    @Override
    public int getPartition(ByteWritable key, RowNumberWritable value, int numPartitions) {
        if (key.get() == (byte) RowNumberJob.COUNTER_MARKER) {
            return value.getPartition();
        } else {
            return Partitioner.partitionForValue(value, numPartitions);
        }
    }

    public static int partitionForValue(RowNumberWritable value, int numPartitions) {
        return (value.getValue().hashCode() & Integer.MAX_VALUE) % numPartitions;
    }
}
```

A custom grouping comparator:

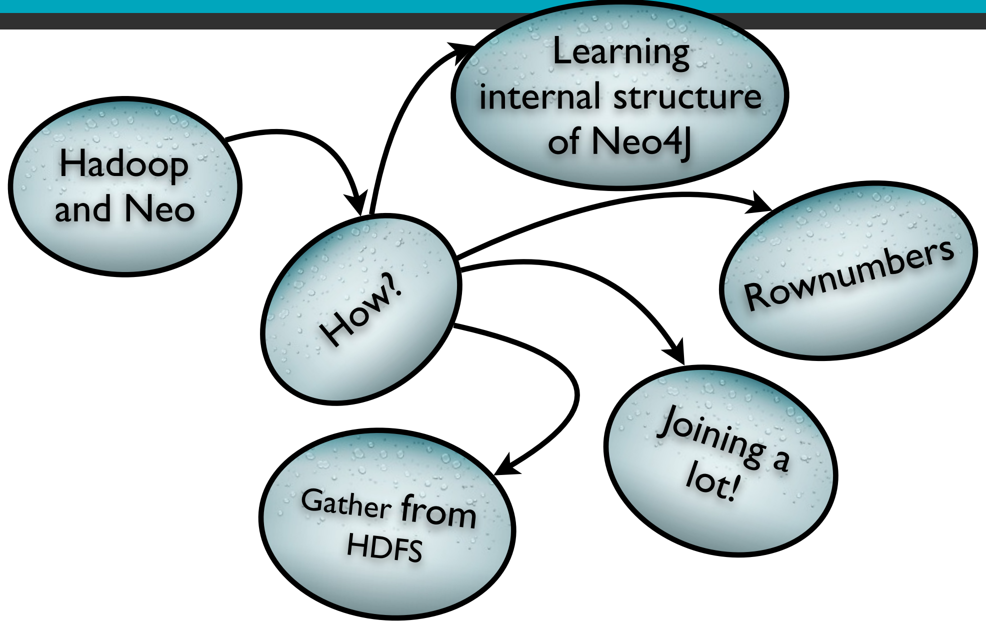
```
public class IndifferentComparator implements RawComparator<ByteWritable> {
    @Override
    public int compare(ByteWritable left, ByteWritable right) {
        return 0;
    }
}
```

A custom writable



Joining a lot!

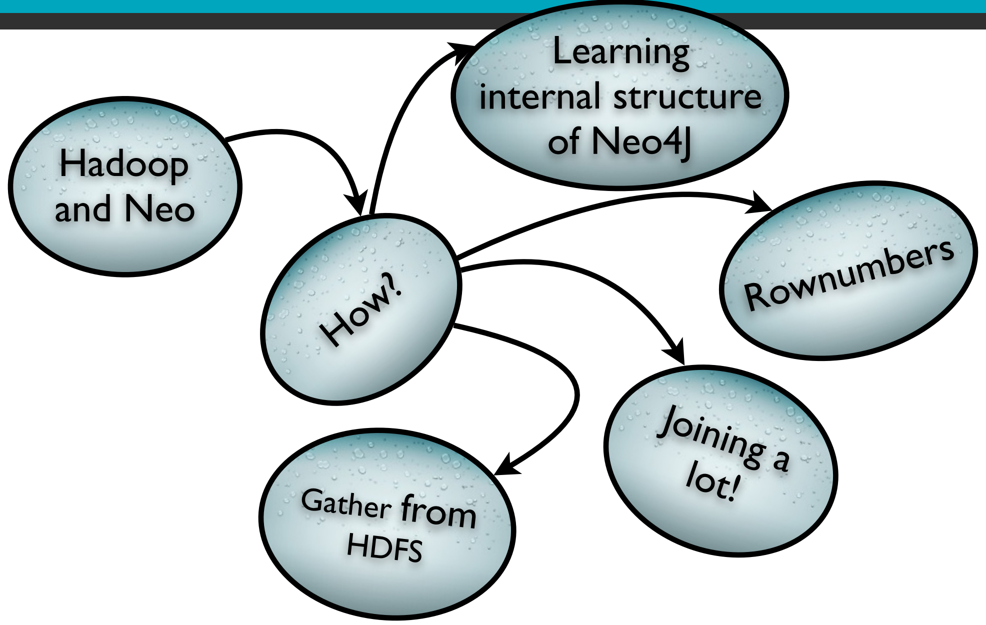
Step 1: Prepare properties
Step 2a: Output properties
Step 2b: Output nodes/edges and first property reference
Step 3: Join edges and nodes to get from-id and to-id
Step 4: Output nodes with first edge and first property reference
Step 5: Output edges



Joining a lot!

Step 1:   Prepare properties

Propertierecords can hold multiple properties
We need to find the first property reference for each node and edge,
so we need to know the total properties structure first



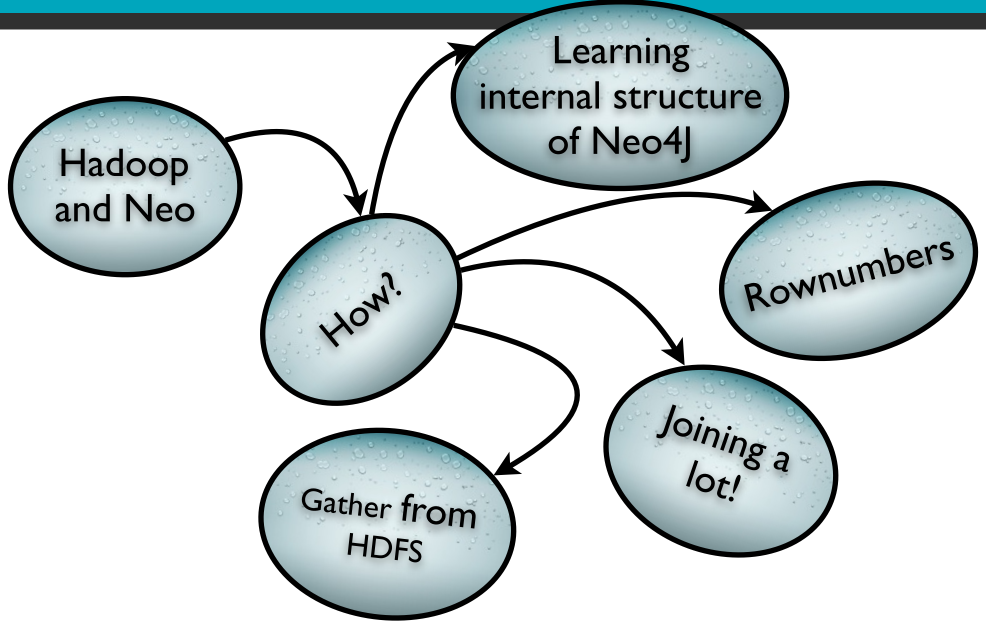
Joining a lot!

Step 2a: Output properties
Step 2b: Output nodes/edges and first property reference

We output the bytearray structure of the neo4j property files

And we can output the node id with functional id and the first property pointer

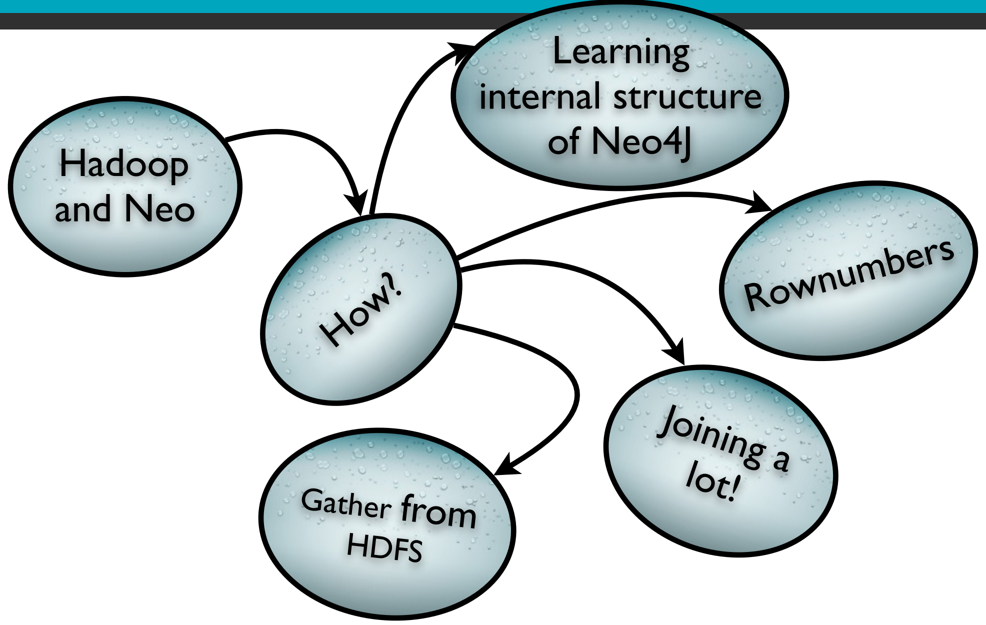
For edges we output the edgeid, fromnode, tonode and first property pointer



Joining a lot!

Step 3: Join edges and nodes to get from-id and to-id

Remember we added rownumbers to the data and we need to use them as our pointers. Not the original functional id.



Hadoop and Neo

How?

Learning internal structure of Neo4J

Rownumbers

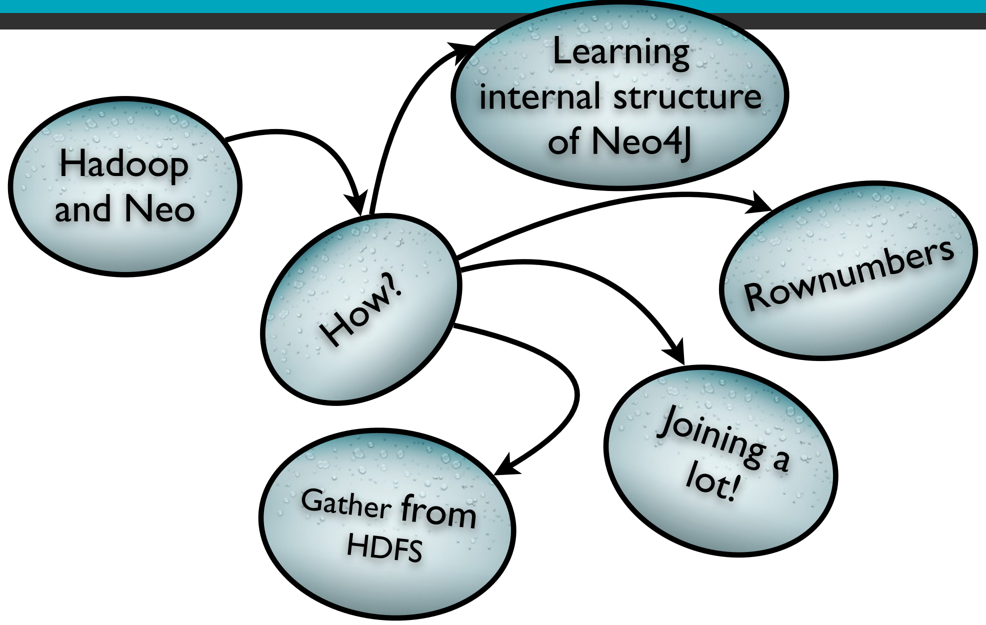
Joining a lot!

Gather from HDFS

Joining a lot!

Step 4: Output nodes with first edge and first property reference

It's actually more complicated. Need to selfjoin all edges of a node to determine the first.



Hadoop and Neo

How?

Learning internal structure of Neo4J

Rownumbers

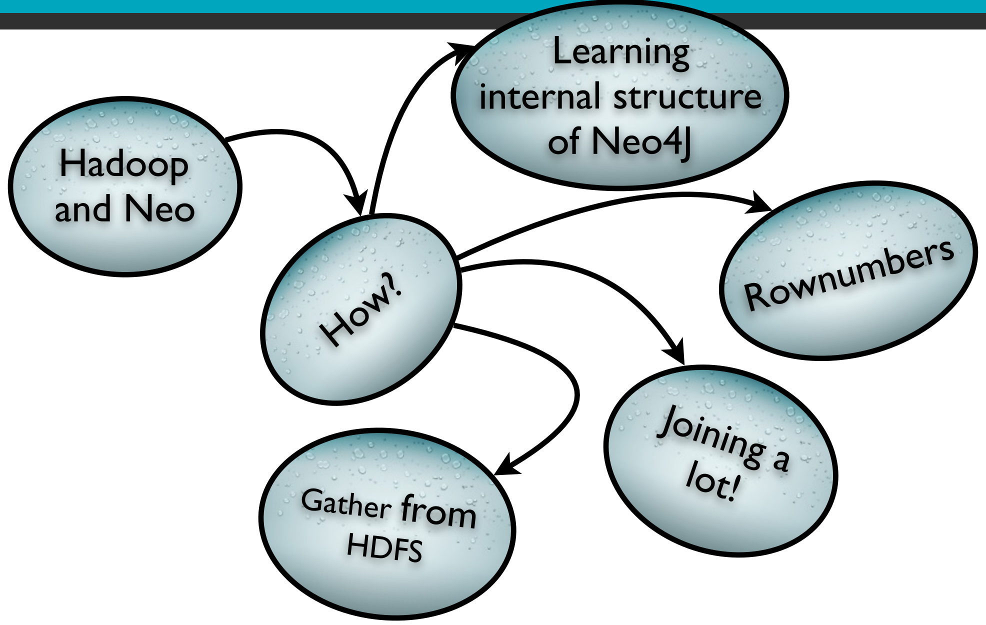
Joining a lot!

Gather from HDFS

Joining a lot!

Step 5: Output edges

Making sure you have them sorted to put in the next and previous edge reference



Hadoop and Neo

How?

Learning internal structure of Neo4J

Rownumbers

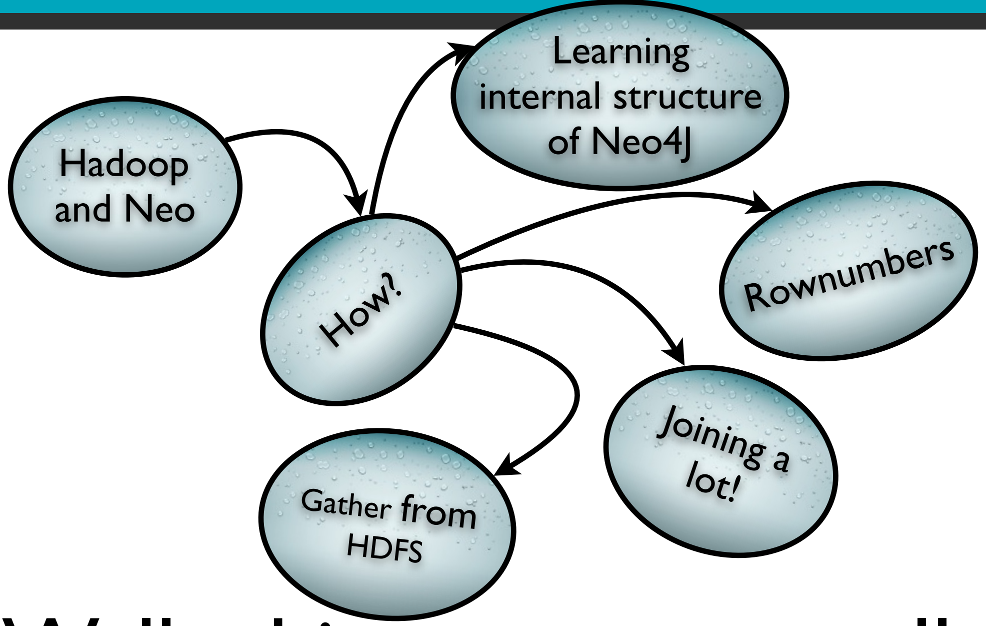
Joining a lot!

Gather from HDFS

Gather from HDFS

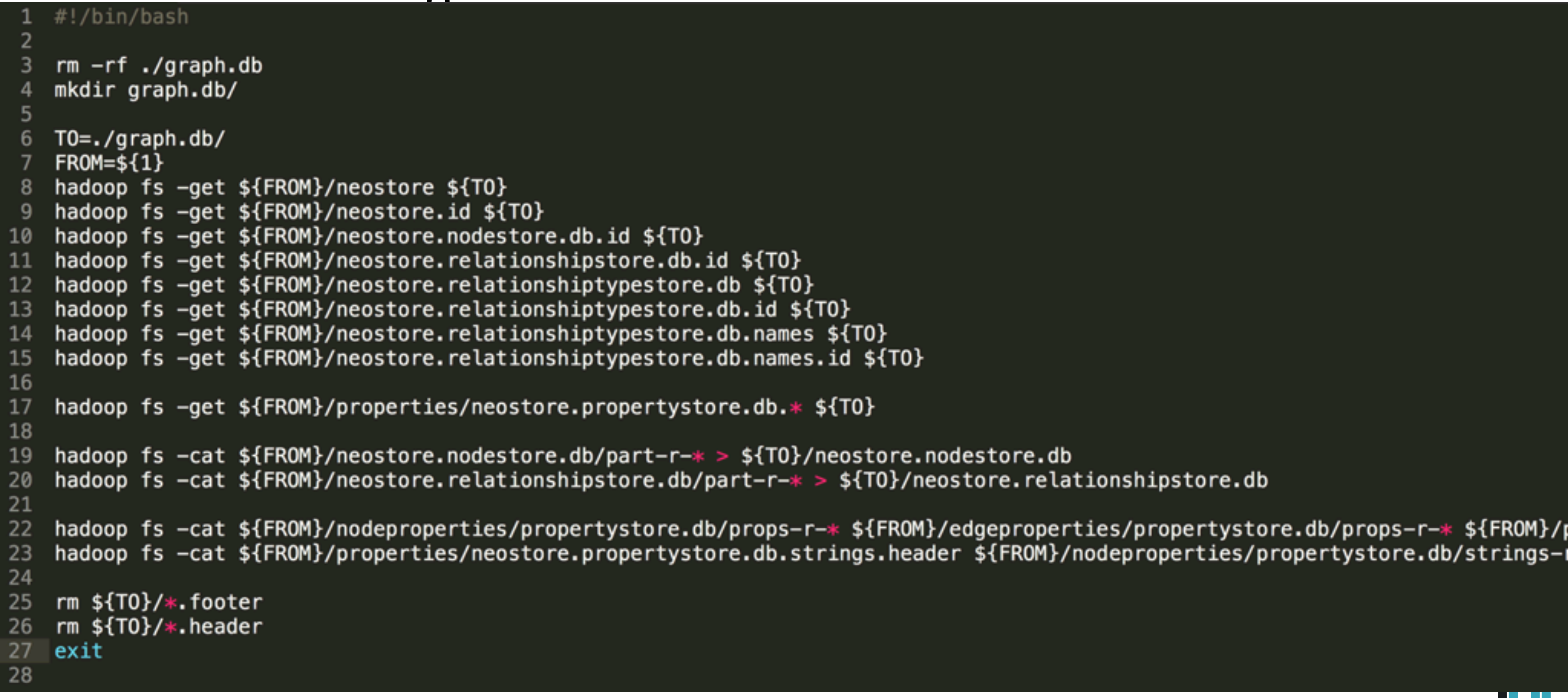
Just a simple

```
hadoop fs -cat <HDFS PATH>/neostore.nodestore.db/part-r-*
```

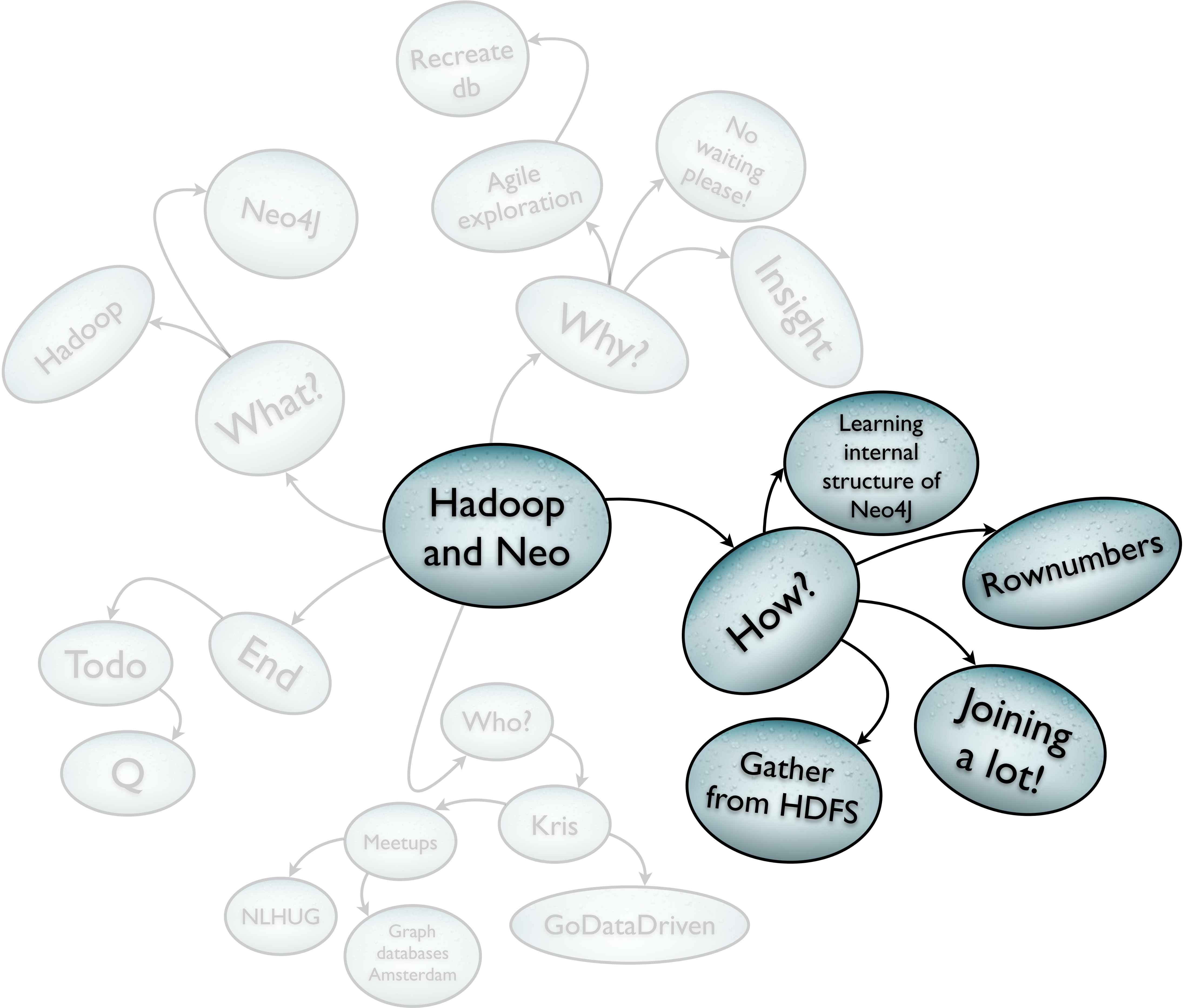


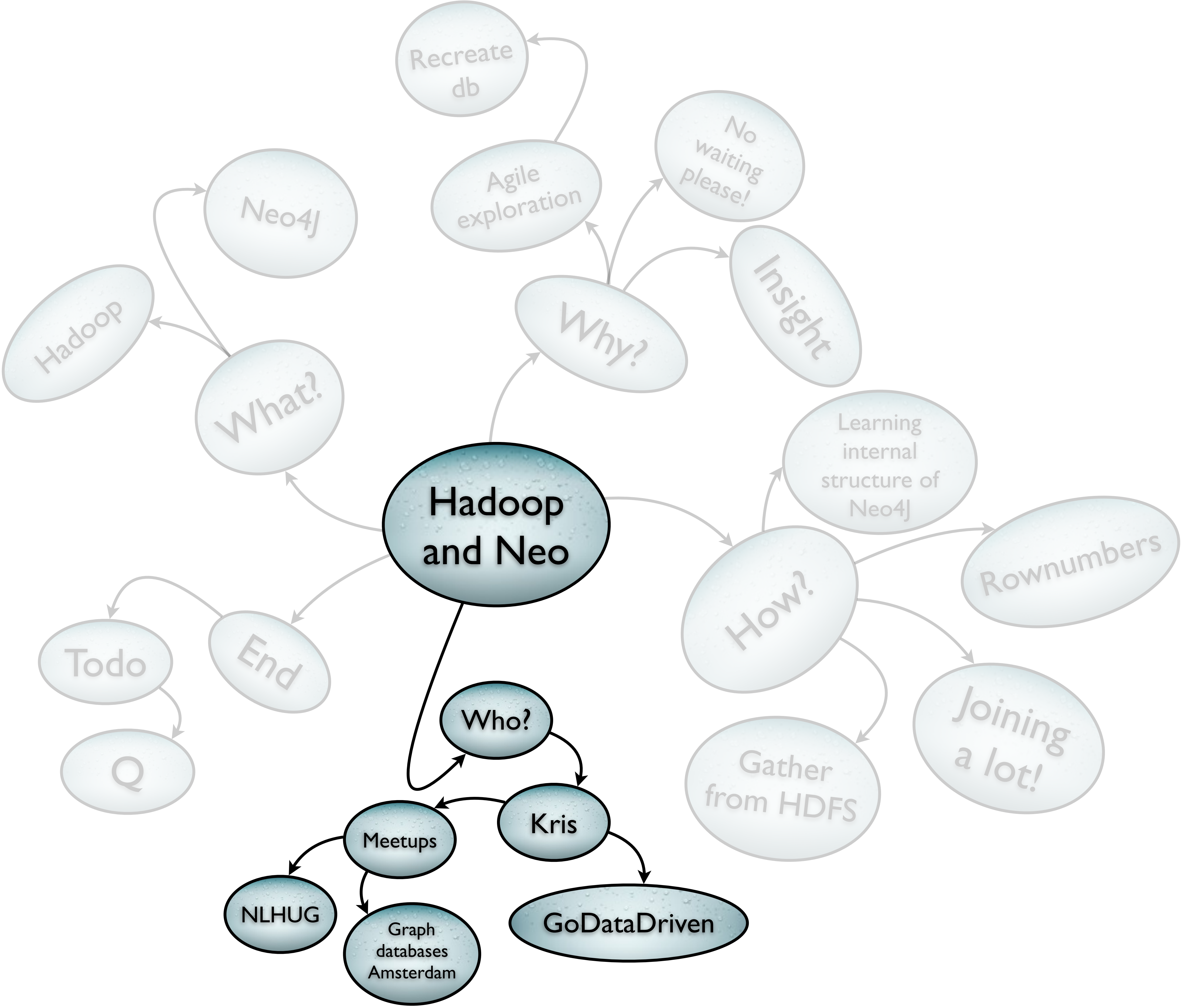
Gather from HDFS

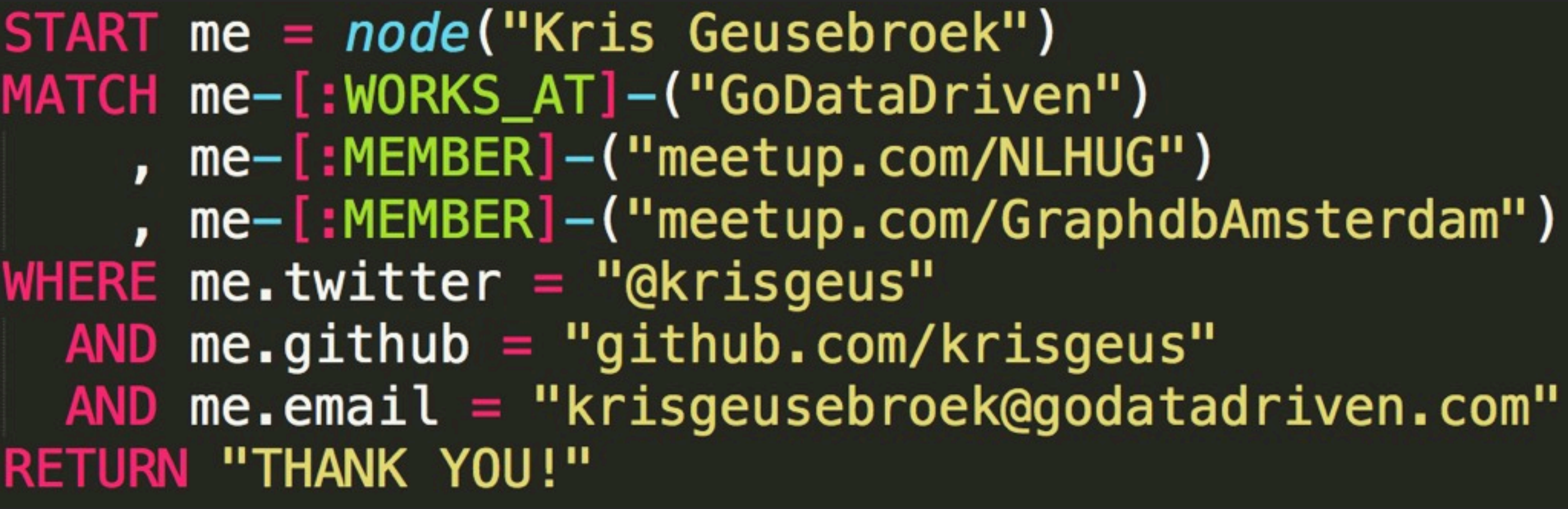
Well a bit more to get all data



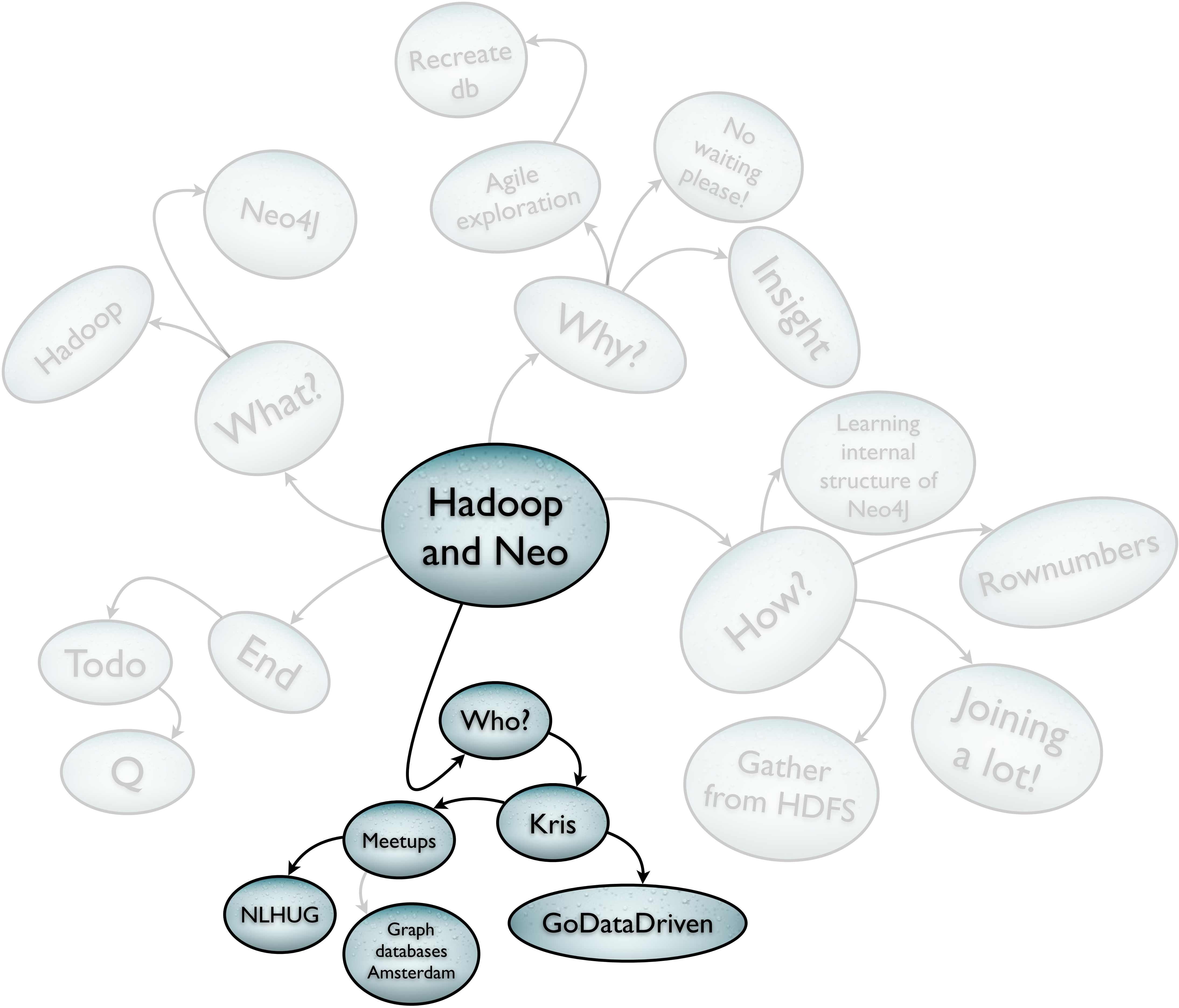
```
#!/bin/bash

rm -rf ./graph.db
mkdir graph.db/

TO=./graph.db/
FROM=${1}
hadoop fs -get ${FROM}/neostore ${TO}
hadoop fs -get ${FROM}/neostore.id ${TO}
hadoop fs -get ${FROM}/neostore.nodestore.db.id ${TO}
hadoop fs -get ${FROM}/neostore.relationshipstore.db.id ${TO}
hadoop fs -get ${FROM}/neostore.relationshiptypestore.db ${TO}
hadoop fs -get ${FROM}/neostore.relationshiptypestore.db.id ${TO}
hadoop fs -get ${FROM}/neostore.relationshiptypestore.db.names ${TO}
hadoop fs -get ${FROM}/neostore.relationshiptypestore.db.names.id ${TO}

hadoop fs -get ${FROM}/properties/neostore.propertystore.db.* ${TO}

hadoop fs -cat ${FROM}/neostore.nodestore.db/part-r-* > ${TO}/neostore.nodestore.db
hadoop fs -cat ${FROM}/neostore.relationshipstore.db/part-r-* > ${TO}/neostore.relationshipstore.db

hadoop fs -cat ${FROM}/nodeproperties/propertystore.db/props-r-* ${FROM}/edgeproperties/propertystore.db/props-r-* ${FROM}/p
hadoop fs -cat ${FROM}/properties/neostore.propertystore.db.strings.header ${FROM}/nodeproperties/propertystore.db/strings-

rm ${TO}/*.footer
rm ${TO}/*.header
exit

```


Recreate db
No waiting please!
Agile exploration
Neo4J
Hadoop
Why?
Insight
What?
Learning internal structure of Neo4J
Hadoop and Neo
How?
Rownumbers
Todo
End
Q
Who?
Joining a lot!
Gather from HDFS
Kris
Meetups
NLHUG
Graph databases Amsterdam
GoDataDriven


Recreate db
Agile exploration
No waiting please!
Neo4J
Why?
Insight
Hadoop
What?
Learning internal structure of Neo4J
Hadoop and Neo
Rownumbers
How?
Todo
End
Q
Who?
Joining a lot!
Gather from HDFS
Kris
Meetups
NLHUG
Graph databases Amsterdam
GoDataDriven



```
START me = node("Kris Geusebroek")
MATCH me-[:WORKS_AT]-("GoDataDriven")
    , me-[:MEMBER]-("meetup.com/NLHUG")
    , me-[:MEMBER]-("meetup.com/GraphdbAmsterdam")
WHERE me.twitter = "@krisgeus"
  AND me.github = "github.com/krisgeus"
  AND me.email = "krisgeusebroek@godatadriven.com"
RETURN "THANK YOU!"
```



Hadoop and Neo

What?

Hadoop

Neo4J

Why?

Agile exploration

Recreate db

No waiting please!

Insight

How?

Learning internal structure of Neo4J

Rownumbers

Joining a lot!

Gather from HDFS

End

Todo

Q

Who?

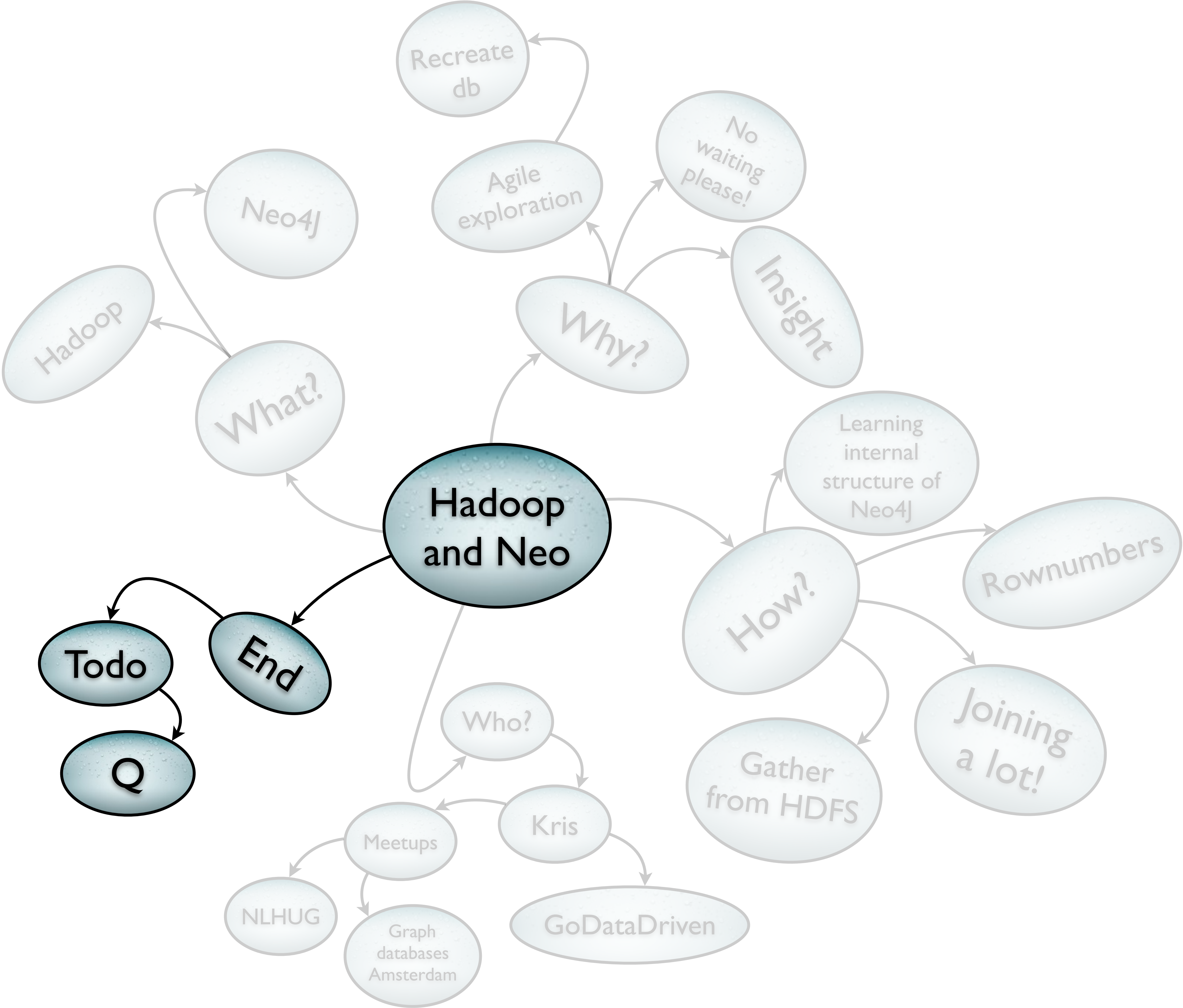
Kris

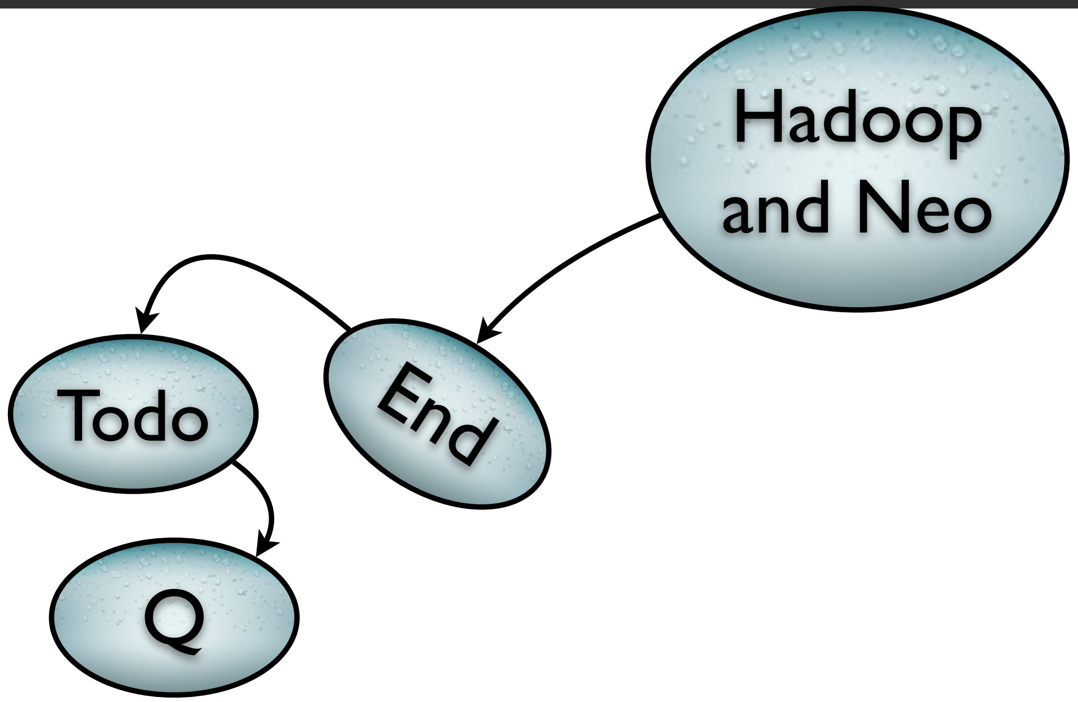
Meetups

NLHUG

Graph databases Amsterdam

GoDataDriven


Hadoop and Neo
What?
Hadoop
Neo4J
Why?
Agile exploration
Recreate db
No waiting please!
Insight
How?
Learning internal structure of Neo4J
Rownumbers
Joining a lot!
Gather from HDFS
Who?
Kris
Meetups
NLHUG
Graph databases Amsterdam
GoDataDriven
End
Todo
Q


Hadoop and Neo

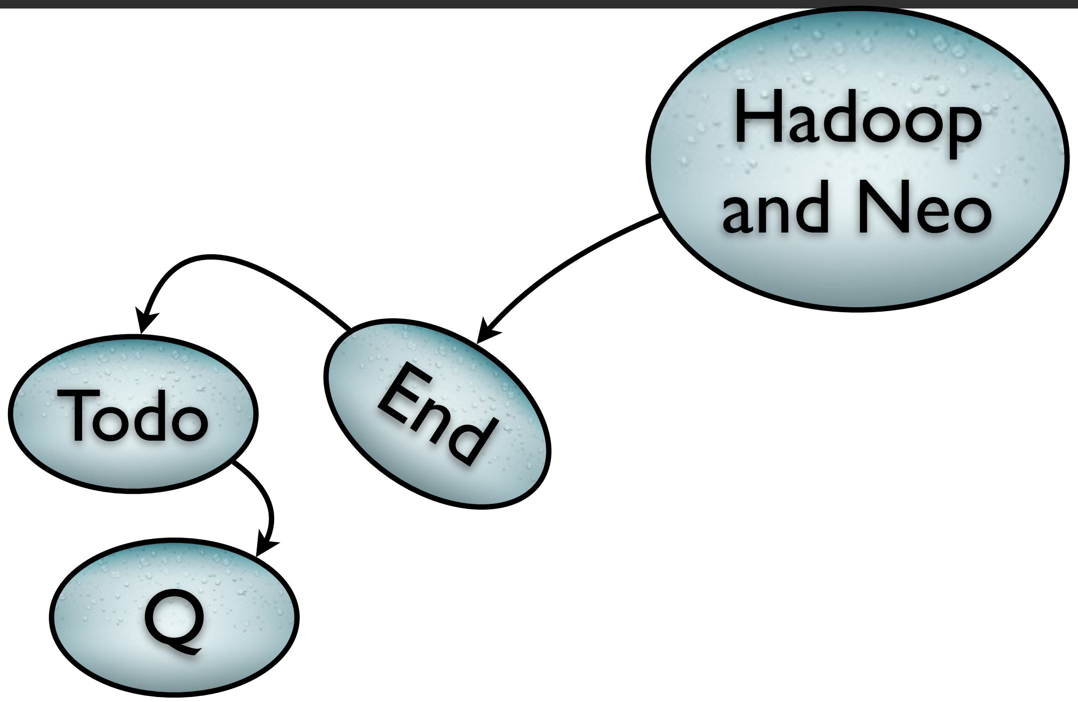
End

Todo

Q

Todo:

- Indexes
- Array properties
- Neo4J 2.0 compatibility
- Inverse direction



Code is on

https://github.com/krisgeus/graphs

GoDataDriven

We're hiring / Questions? / Thank you!

Kris Geusebroek
Big Data Hacker

@krisgeus
krisgeusebroek@godatadriven.com