Introduction to Data Science

Lecture 10; June 8th, 2015

Ernst Henle
ErnstHe@UW.edu
Skype: ernst.predixion

Agenda

- Social Interactions / Announcements
  - Continue the LinkedIn group for the next 2 quarters
- Hadoop
- Break
- Graph Data (Quiz 10a)
- Break
- SPARQL (Quiz 10b)
- Assignment: Quiz 10c and Quiz 10d

Social Interactions and Announcements

- Continue the LinkedIn group for the next 2 quarters
- Today at 9:10 PM. After-hours with Matt Danielson at Rock Bottom. One Block from Class: 1333 Fifth Ave. Seattle, WA 98101

Python by Matt Danielson

Bio:

Matt Danielson is a Research Software Engineer at Globys, where he transforms data science algorithms into production code. Before that he was a Software Developer at Susquehanna International Group in Philadelphia, developing High Frequency Trading Algorithms for the Futures and Equities Markets. Matt graduated from Western Washington University with a B.S. in Math, and Trinity College in Dublin, Ireland with a M.Sc. in High Performance Computing. When he's not programming in Python, he's likely outside backpacking, cycling, or kayaking!

Hadoop

Sqoop Lab

- **Sq**l + Had**oop** → **Sqoop**
- Use Sqoop to share between a RDBMS and Hadoop

Sqoop Lab (1)

- Use this Sqoop command to familiarize yourself with Sqoop:
  - $ **`sqoop help`**

- Use these sqoop commands to list mysql databases on localhost and then tables in one of those databases:
  - $ **`sqoop list-databases --connect jdbc:mysql://localhost --username training --password training`**
  - $ **`sqoop list-tables --connect jdbc:mysql://localhost/movielens --username training --password training`**

Sqoop Lab (2)

- Use this Sqoop command to import data from a mysql table to an HDFS table. The fields of a table record (row) will be separated by tabs in the corresponding record of the HDFS file:
  - $ **sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movie**

- Note that Sqoop constructs some SQL statements for MySQL and that it converts the SQL statement into a MapReduce job that only has a Map component:
  - **Test SQL: SELECT t.* FROM `movie` AS t LIMIT 1**
  - **SELECT MIN(`id`), MAX(`id`) FROM `movie`**

- Use this HDFS command to list the newly imported file(s) and then view the last part of one of the files:
  - $ **hadoop fs -ls movie**
  - $ **hadoop fs -tail movie/part-m-00000**

To run this command a second time you will need to avoid a naming collision:
sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movie --target-dir /movie2

Sqoop Lab (3)

- Use this Sqoop command to import data from a mysql table to an HDFS table. The fields of a table record (row) will be separated by tabs in the corresponding record of the HDFS file:
  - $ **`sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movierating`**

- Use these HDFS commands to list the newly imported file(s) and then view the last part of one of the files:
  - $ **`hadoop fs -ls movierating`**
  - $ **`hadoop fs -tail movierating/part-m-00000`**

To run this command a second time you will need to avoid a naming collision:
sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movierating --target-dir /movierating2

Hive Lab

"It does look similar—but this one is powered by Hadoop"

Hive Lab (1)

- Check that we have movie and movierating in HDFS
  - $ **`hadoop fs -cat movie/part-m-00000 | head`**
  - $ **`hadoop fs -cat movierating/part-m-00000 | head`**

- Start Hive
  - $ **`hive`**

- Tell Hive that the HDFS files, movie and movierating, are tables:
  - **hive> CREATE EXTERNAL TABLE movie (id INT, name STRING, year INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LOCATION '/user/training/movie';**
  - **hive> CREATE EXTERNAL TABLE movierating (userid INT, movieid INT, rating INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LOCATION '/user/training/movierating';**

To run this command a second time you will need to avoid a naming collision:
hive> DROP TABLE **`movie`**
hive> DROP TABLE **`movie`**rating

Hive Lab (2)

- Ask Hive to show tables and provide metadata on these tables
  - `hive> SHOW TABLES;`
  - `hive> DESCRIBE movie;`
  - `hive> DESCRIBE movierating;`

- Use Hive to find information from your tables
  - **hive> SELECT * FROM movie WHERE year < 1925;**
  - **hive> SELECT * FROM movie WHERE year < 1925 AND year != 0 ORDER BY name;**
  - **hive> SELECT * FROM movierating WHERE userid=149;**

Hive Lab (3)

- hive> CREATE TABLE USERRATING (userid INT, numratings INT, avgrating FLOAT);
- hive> insert overwrite table userrating SELECT userid,COUNT(userid),AVG(rating) FROM movierating GROUP BY userid;

- hive> SELECT AVG(rating) FROM movierating WHERE userid=149;
- hive> SELECT userid, COUNT(userid),AVG(rating) FROM movierating where userid < 10 GROUP BY userid;

Hive Lab (4)

- Query with Join
  - hive> select movieid,rating,name from movierating join movie on movierating.movieid=movie.id where userid=149;
- hive> QUIT;

- Take-home Exercise
  - What is oldest movie in the database that has a top rating?

Hue and Impala

- Use Impala to execute “HiveQL”
- Open Firefox
- Start Hue by clicking on the Hue link in the favorites
- Select Impala and Query Tab
- Enter Query into Query text box for Impala:
  - **select movieid,rating,name from movierating join movie on movierating.movieid=movie.id where userid=149;**

MapReduce (0)

MapReduce (1)

- MapReduce is a pattern (programming model) designed to work in Hadoop
- MapReduce works with HDFS
- The two primary ideas behind MapReduce
  - Send the program to the data as opposed to the data to the program
  - Make use of distributed data where each chunk of data has its own CPUs
- When a MapReduce job is started, then Hadoop sends Map and Reduce jobs to the data nodes. Hadoop manages all the details of data passing among data nodes.
  - The Map portion of the computation occurs on the individual data nodes where the data reside.
  - The reduce portion of the computation occurs on some of the data nodes (not necessarily where the data reside). In some cases part of the reduce operation can occur on the data node where the data reside.

MapReduce (2)

- MapReduce uses three stages:
  - Map
  - Shuffle
  - Reduce
- Then, why don't we call MapReduce: MapShuffleReduce?
  - Because: In most cases, the programmer need only be concerned with Map and Reduce. The Shuffle and Sorting is taken care of by Hadoop.
  - To achieve scalable programs, Hadoop takes care off:
    - Creating tasks on the various data nodes
    - Tracking Tasks
    - Moving data among data nodes
    - File I/O, networking, process synchronization, recovery from failures, re-running jobs, splitting input, Moving Key-Value-Pairs from Map to Reduce
    - Outputs results

MapReduce (3)

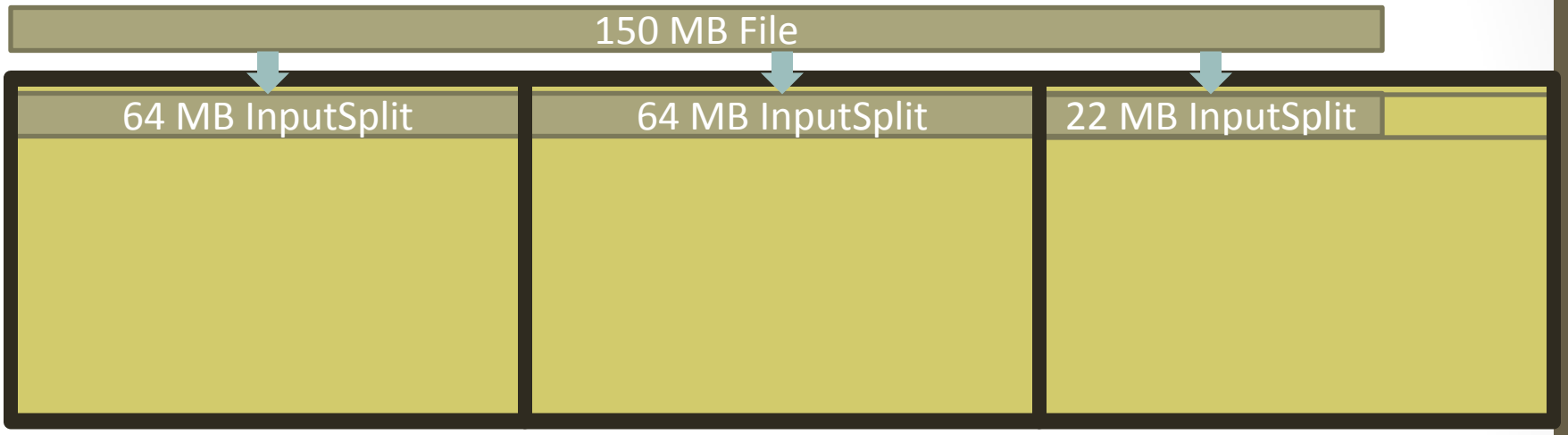
- Parallelization Rules in MapReduce
  - Map tasks occur in parallel independent of each other
  - Reducers can start before Mappers are done. But, reducers cannot complete before Mapping and Shuffle & Sort have completed.
  - Reduce tasks occur in parallel independent of each other

MapReduce (4)

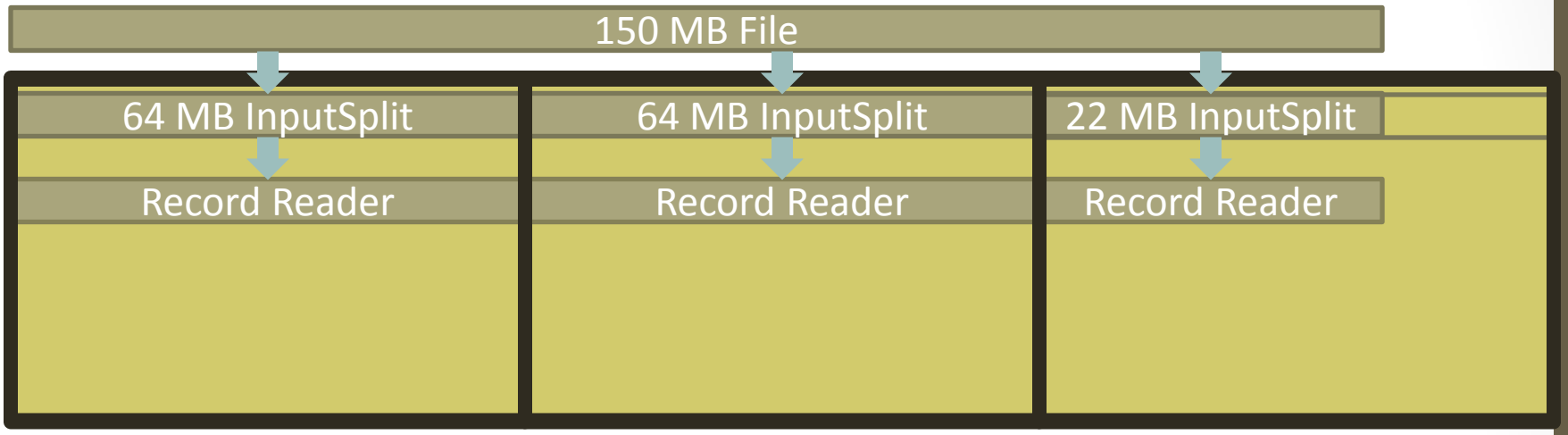
MapReduce (5)


150 MB File
64 MB InputSplit
64 MB InputSplit
22 MB InputSplit

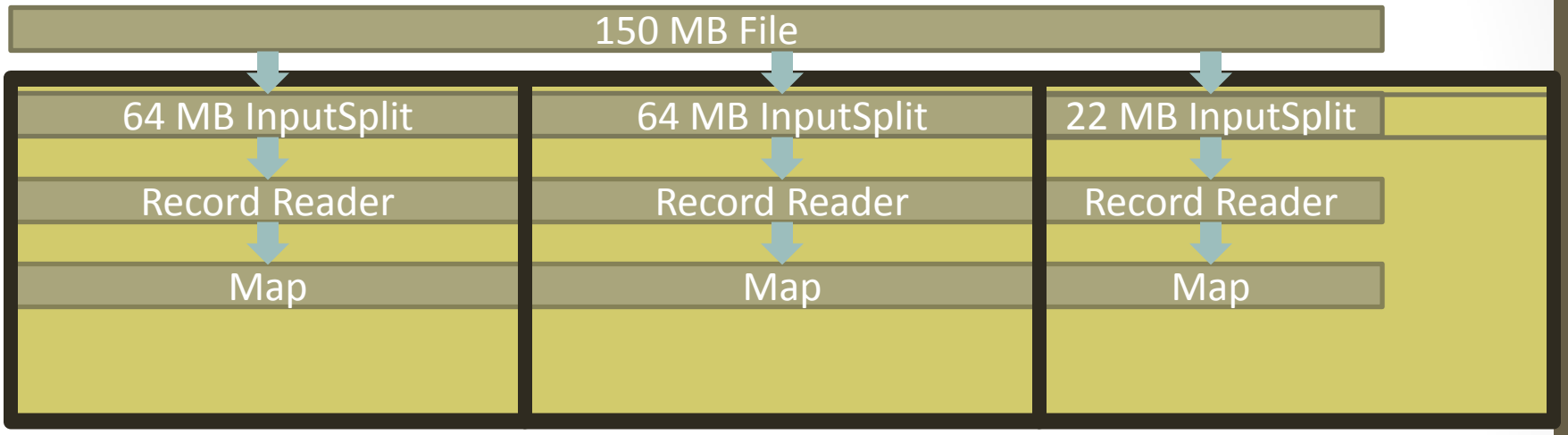
MapReduce (6)


150 MB File
64 MB InputSplit
64 MB InputSplit
22 MB InputSplit

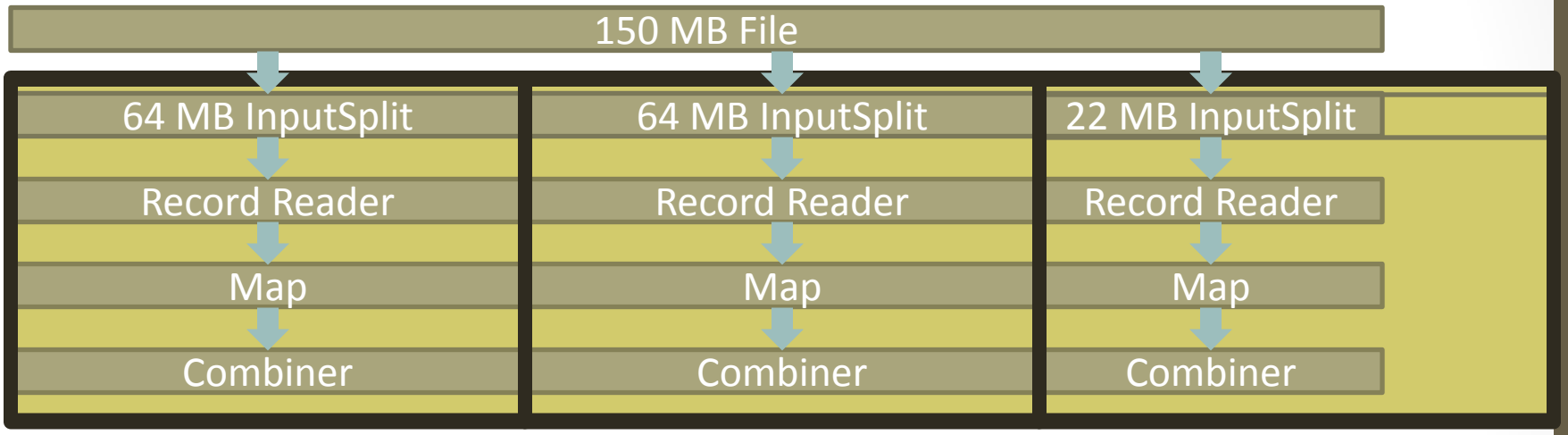
MapReduce (7)


150 MB File
64 MB InputSplit
64 MB InputSplit
22 MB InputSplit
Record Reader
Record Reader
Record Reader

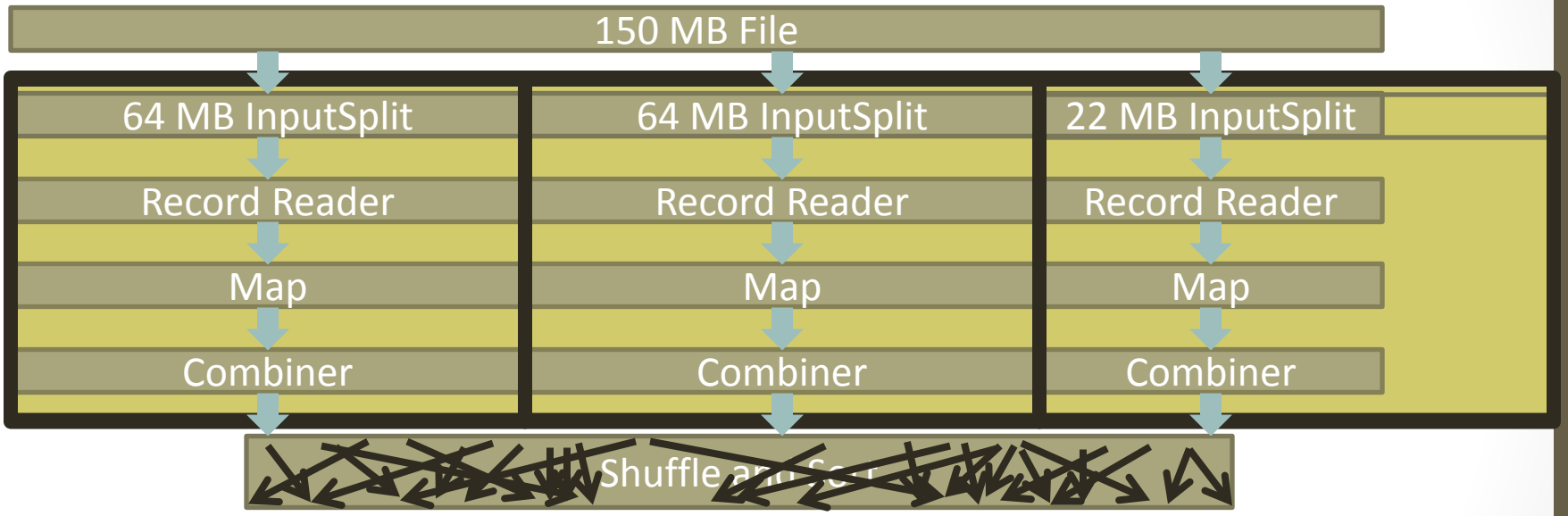
MapReduce (8)


150 MB File
64 MB InputSplit
Record Reader
Map
64 MB InputSplit
Record Reader
Map
22 MB InputSplit
Record Reader
Map

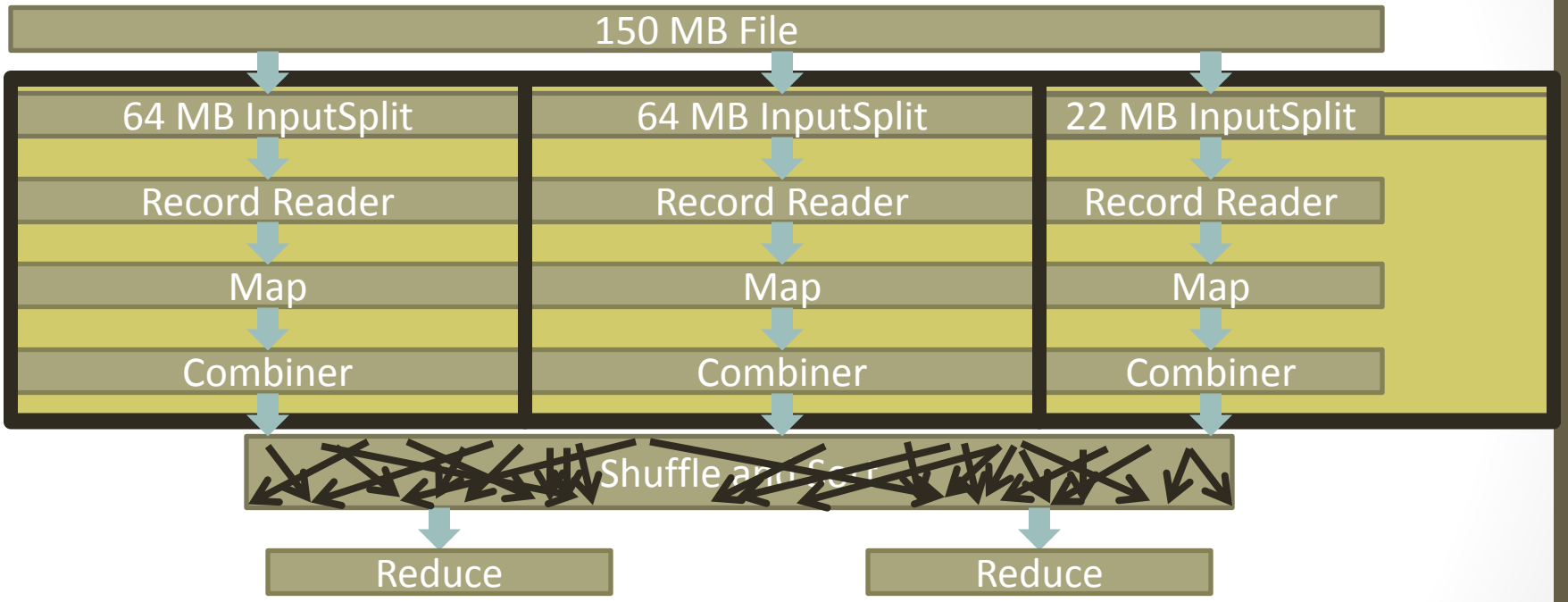
MapReduce (9)


150 MB File
64 MB InputSplit
Record Reader
Map
Combiner
64 MB InputSplit
Record Reader
Map
Combiner
22 MB InputSplit
Record Reader
Map
Combiner

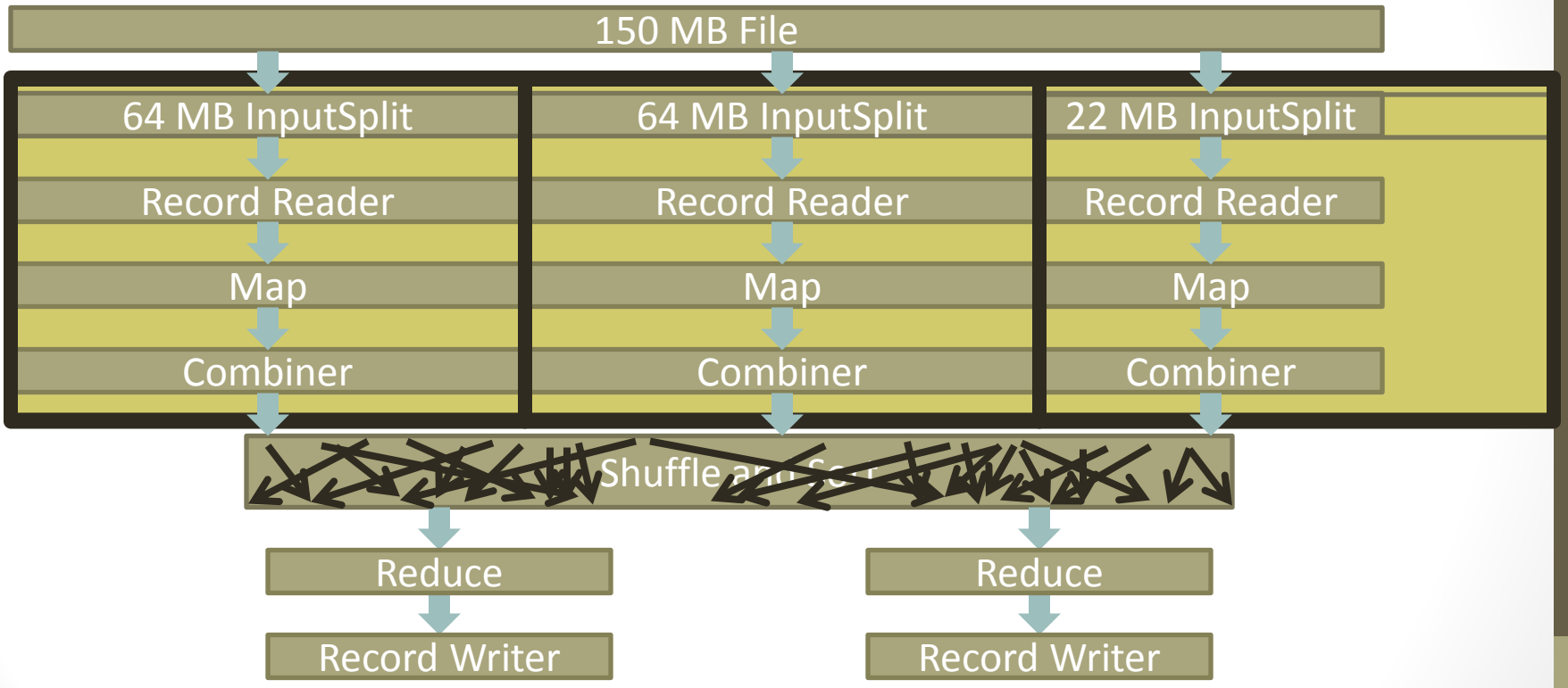
MapReduce (10)



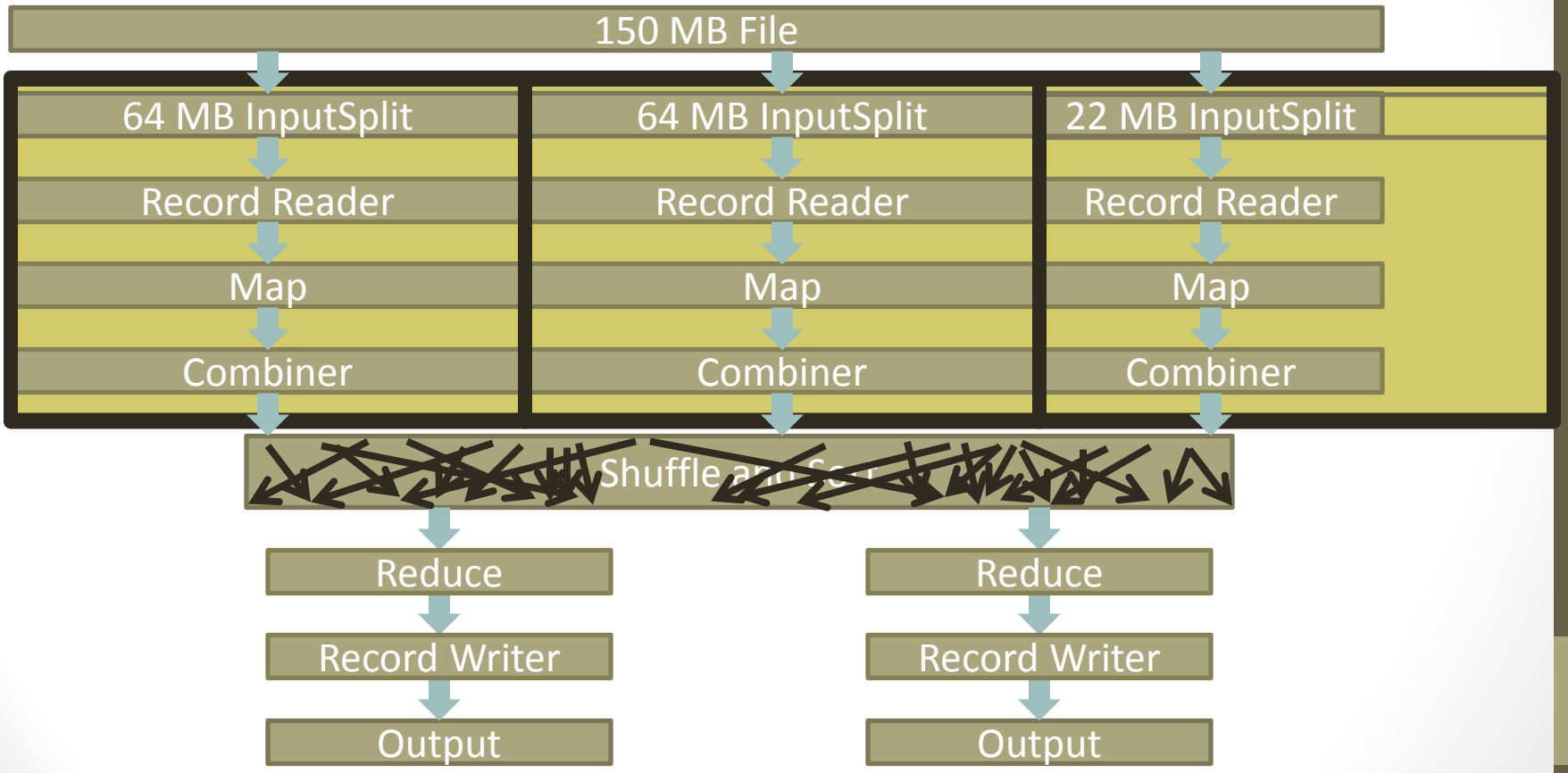
MapReduce (11)



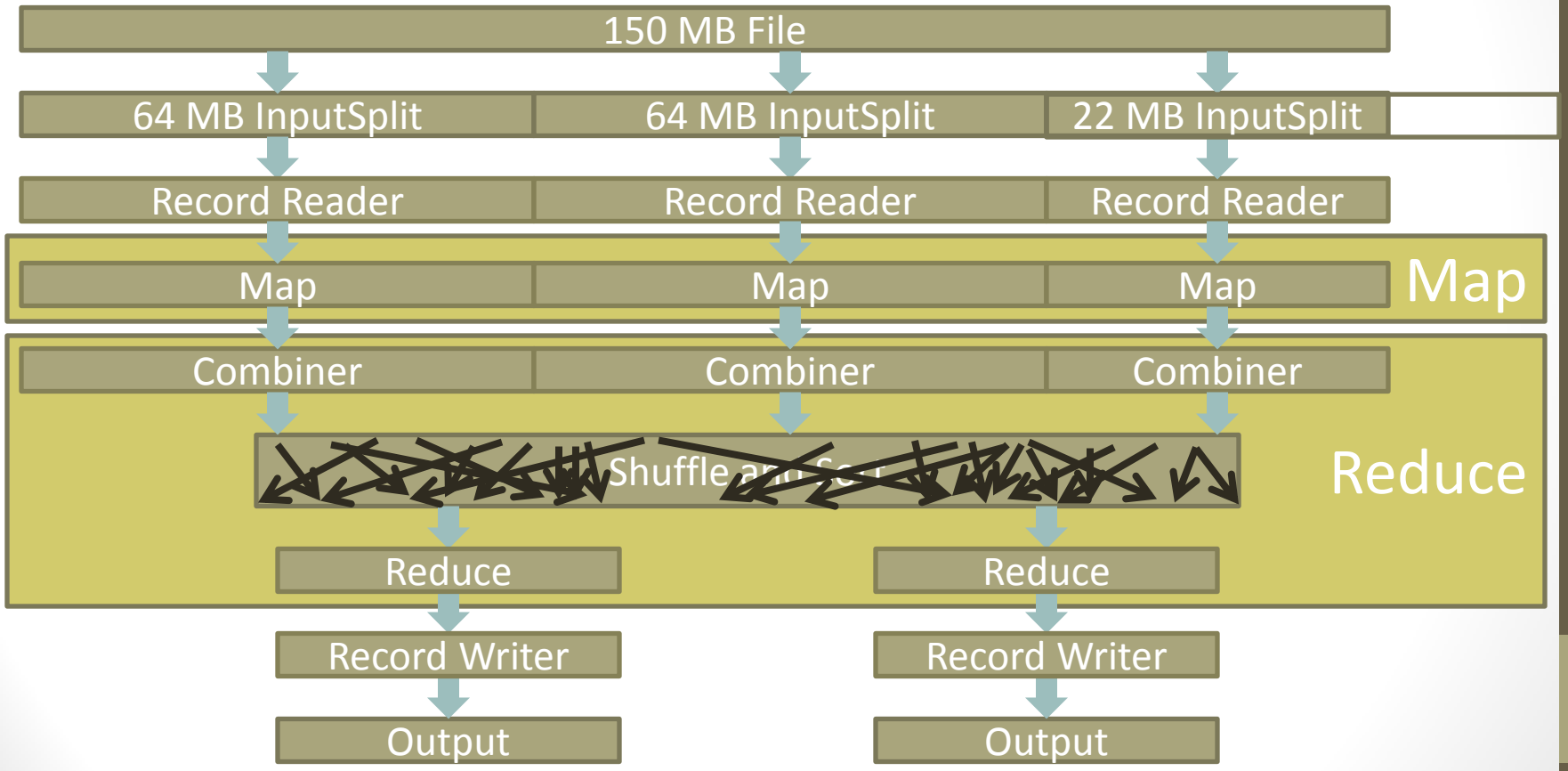
MapReduce (12)



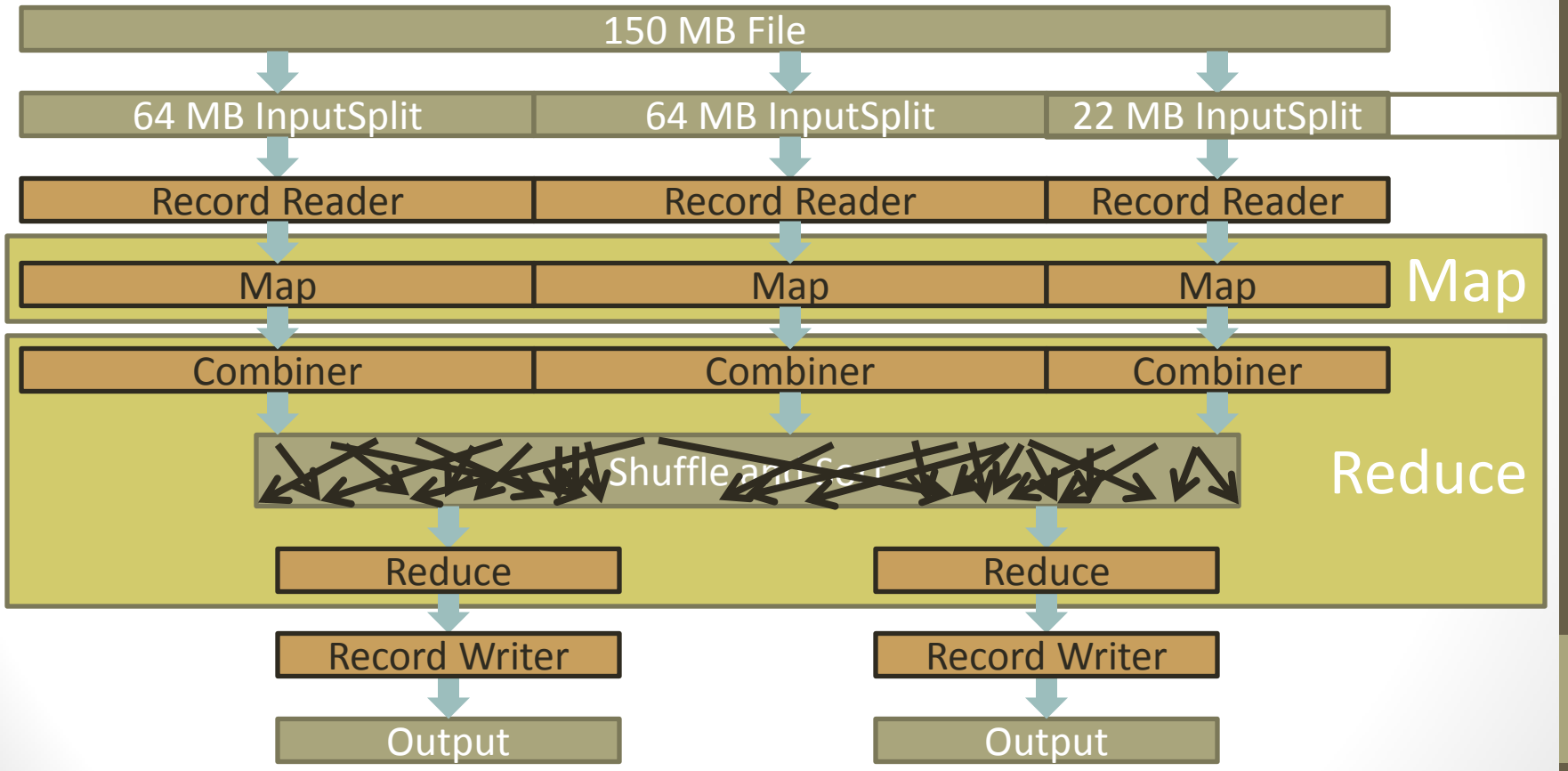
MapReduce (13)



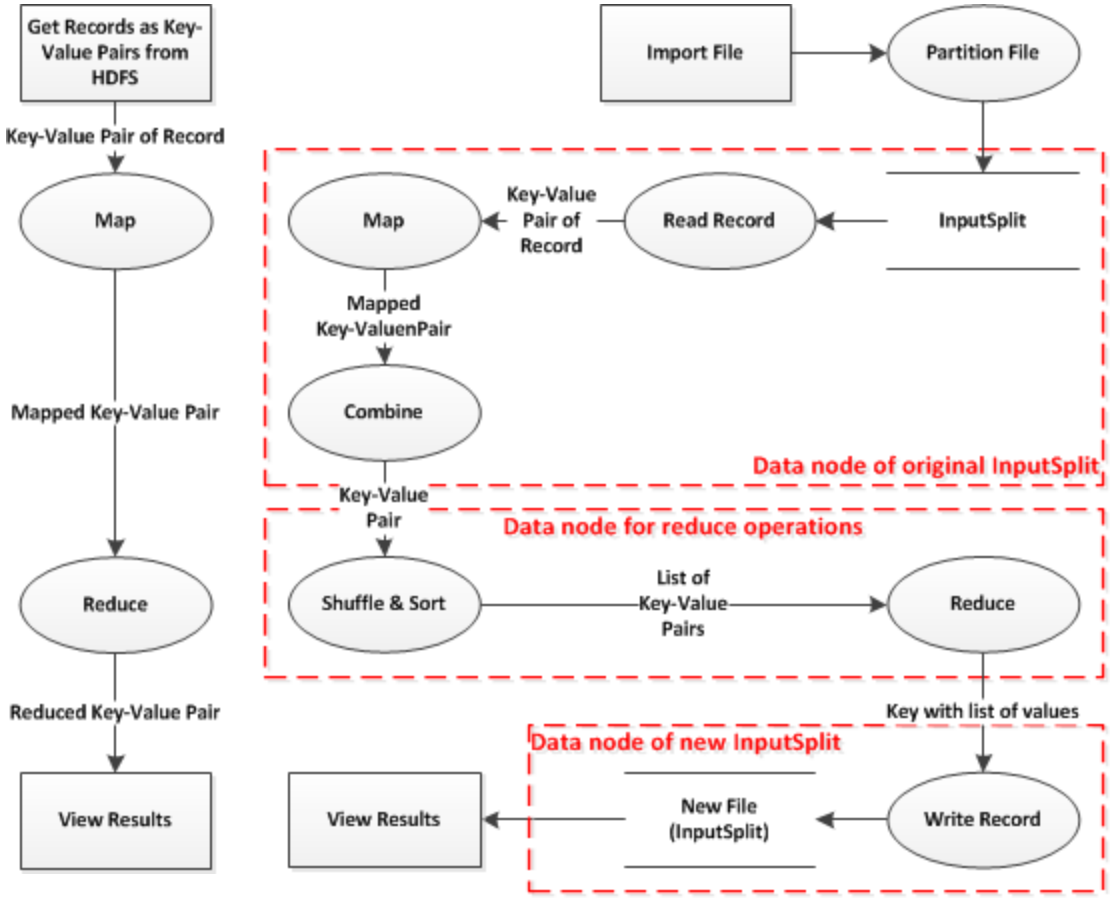
MapReduce(14)



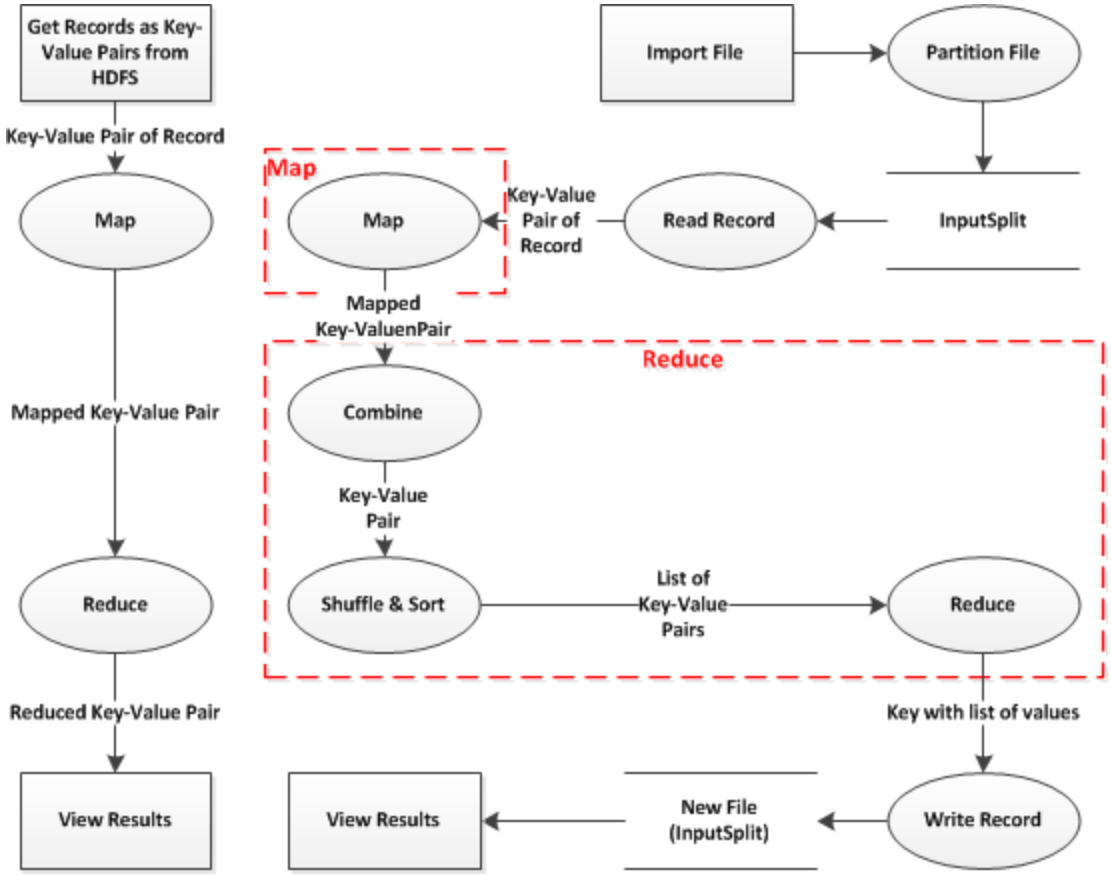
MapReduce (15)



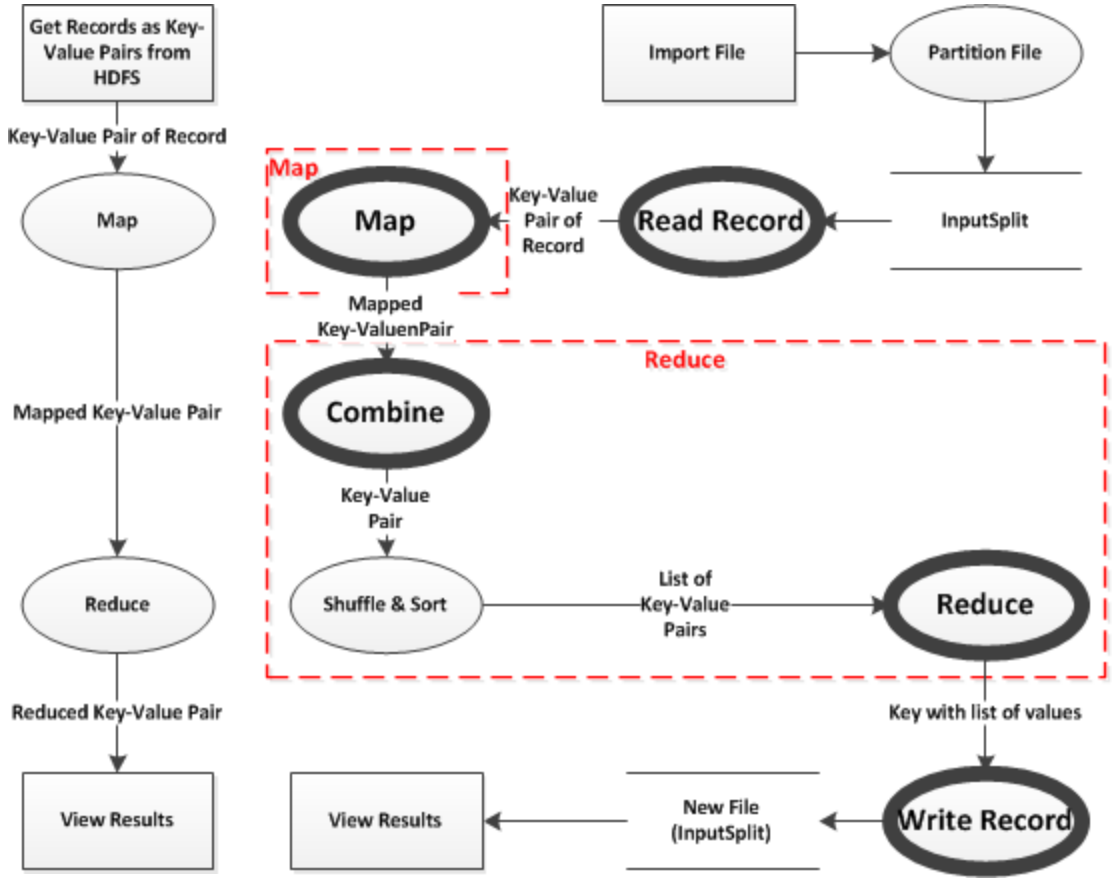
MapReduce (16)



MapReduce (17)



MapReduce (18)



MapReduce (19)

- Record Read
  - Record → (K1,V1)
  - to be or not to be → (0,“to be or not to be”)
- Map
  - (K1,V1) → list(K2,V2)
  - (0,“to be or not to be”) → [(to,1),(be,1),(or,1),(not,1),(to,1),(be,1)]
- Shuffle and Sort
  - list(K2,V2) → (K2,list(V2))
  - [(to,1),(be,1),(or,1),(not,1),(to,1),(be,1)] → (to,[1,1]),(be,[1,1]),(or,[1]),(not,[1])
- Reduce
  - (K2,list(V2)) → list(K3,V3)
  - (to,[1,1]),(be,[1,1]),(or,[1]),(not,[1]) → [(to,2),(be,2),(or,1),(not,1)]
- Record Write
  - (K3,V3) → Records
  - [(to,2),(be,2),(or,1),(not,1)] →

to,2
be,2
or,1
not,1

MapReduce Lab (1)

- Open Eclipse by clicking on the and navigate to WordCount to see the java files that are part of :
  - wordcount\src\solution\
    - WordCount.java
    - WordMapper.java
    - SumReducer.java

- Study these files

- In the terminal:
  - `$ cd ~/workspace/wordcount/src`
  - `$ ls`
  - `$ ls solution`
  - `$ hadoop classpath`
  - `$ javac -classpath `hadoop classpath` solution/*.java`

Note that the apostrophe in this statement is the not the straight or slanted quote (' or ‘) but rather this character (`):  javac -classpath `hadoop classpath` solution/*.java

MapReduce Lab (2)

- In the terminal:
  - `$ jar cvf wc.jar solution/*.class`
  - `$ hadoop jar wc.jar solution.WordCount shakespeare wordcounts`
  - `$ hadoop fs -ls wordcounts`
  - `$ hadoop fs -cat wordcounts/part-r-00000 | less`
  - `$ hadoop jar wc.jar solution.WordCount shakespeare/poems pwords`
  - `$ hadoop fs -rm -r wordcounts pwords`

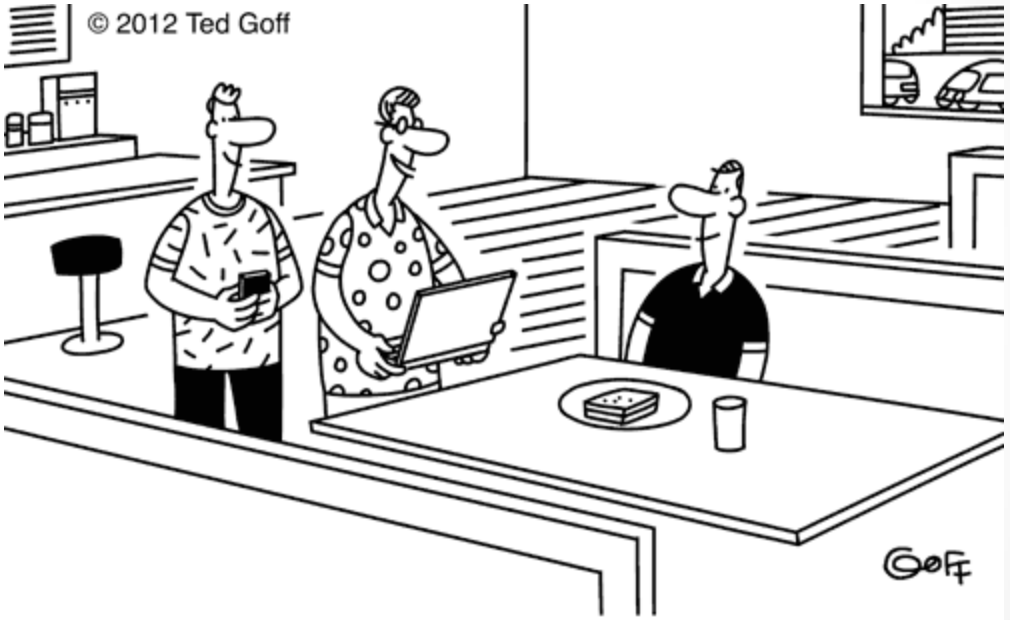
MapReduce Lab (3)

- Open Firefox
- Start Hue
- Start File Browser in Hue
  - Find wordcounts/part-r-00000
  - View Contents of part-r-00000


C'mon, Hadoop — it's bedtime!

Hadoop

Break



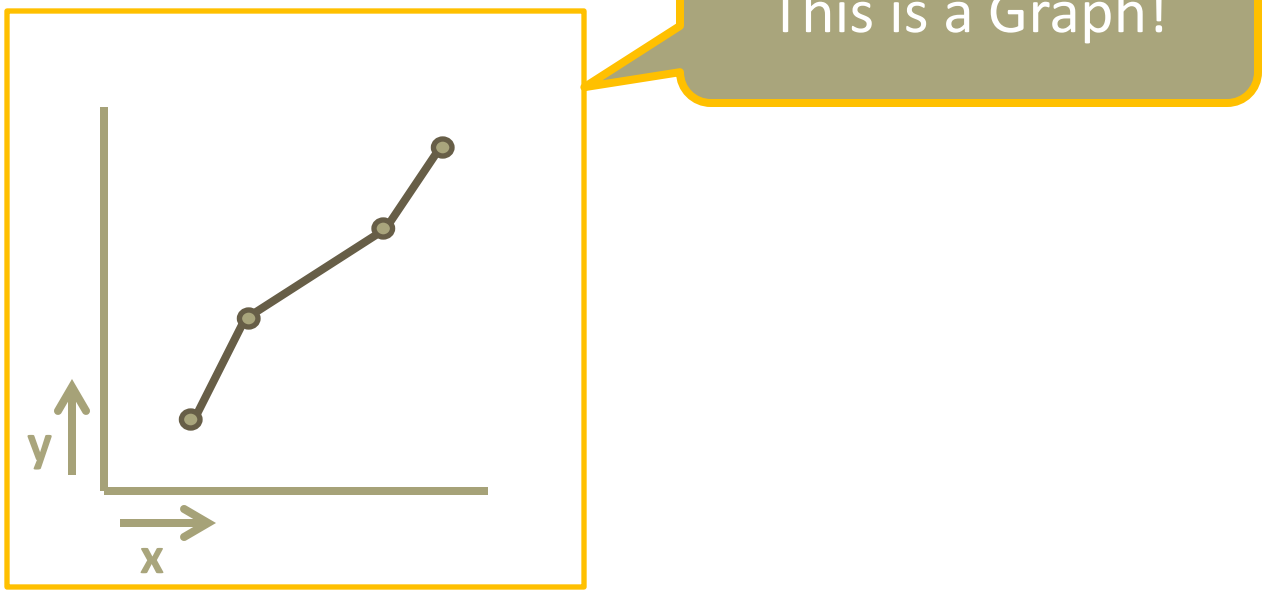
"Twitter and Facebook can't predict the election, but they did predict what you're going to have for lunch: a tuna salad sandwich. You're having the wrong sandwich."

Introduction to Graph Data

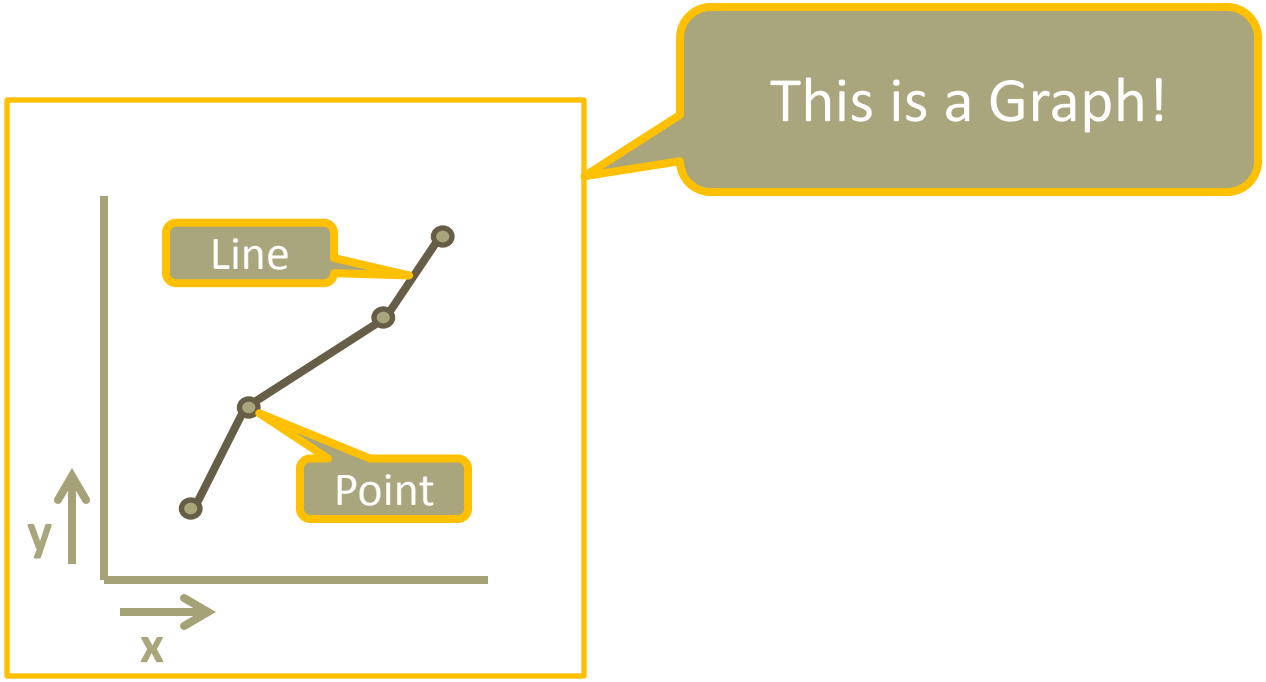
Graph Data: What is it? (0)

What is a Graph?

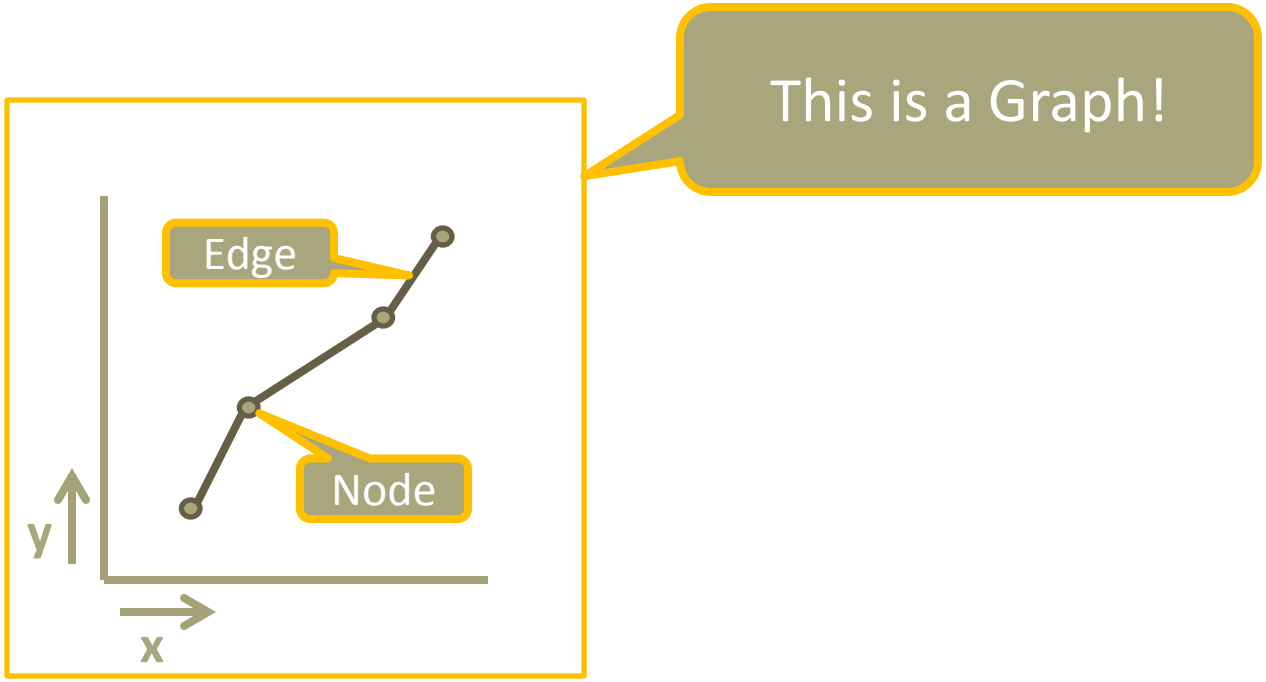
Graph Data: What is it? (1)



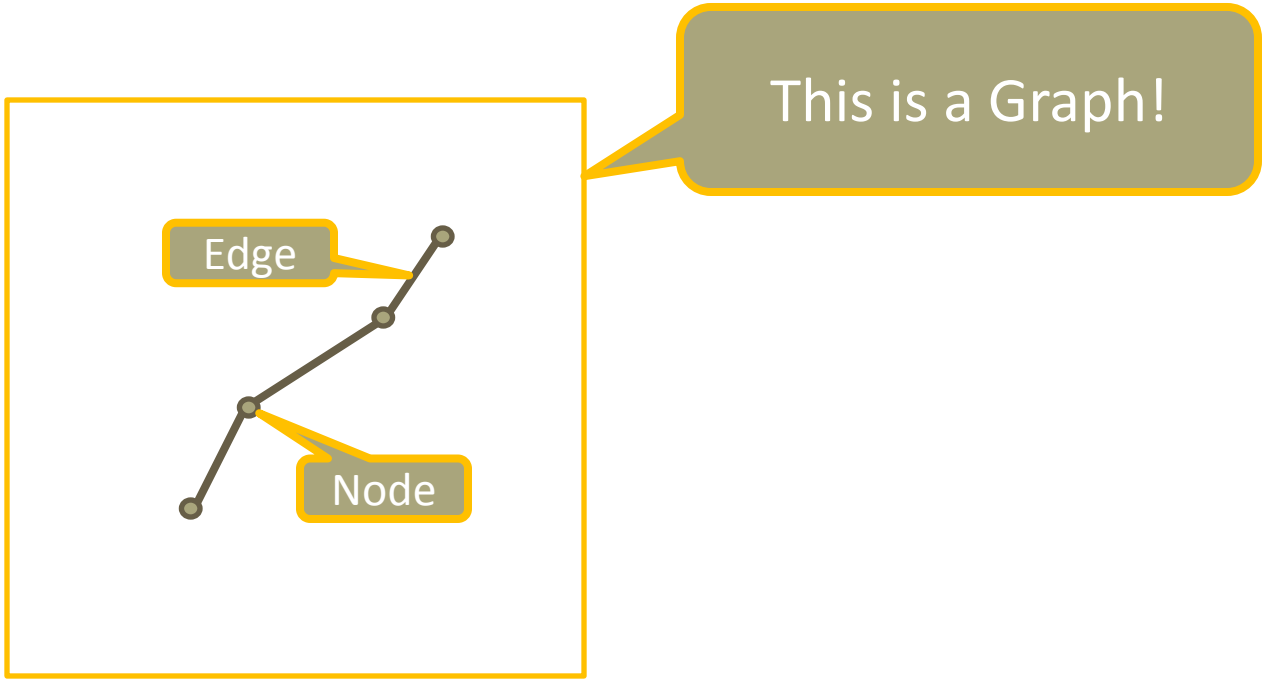
Graph Data: What is it? (2)



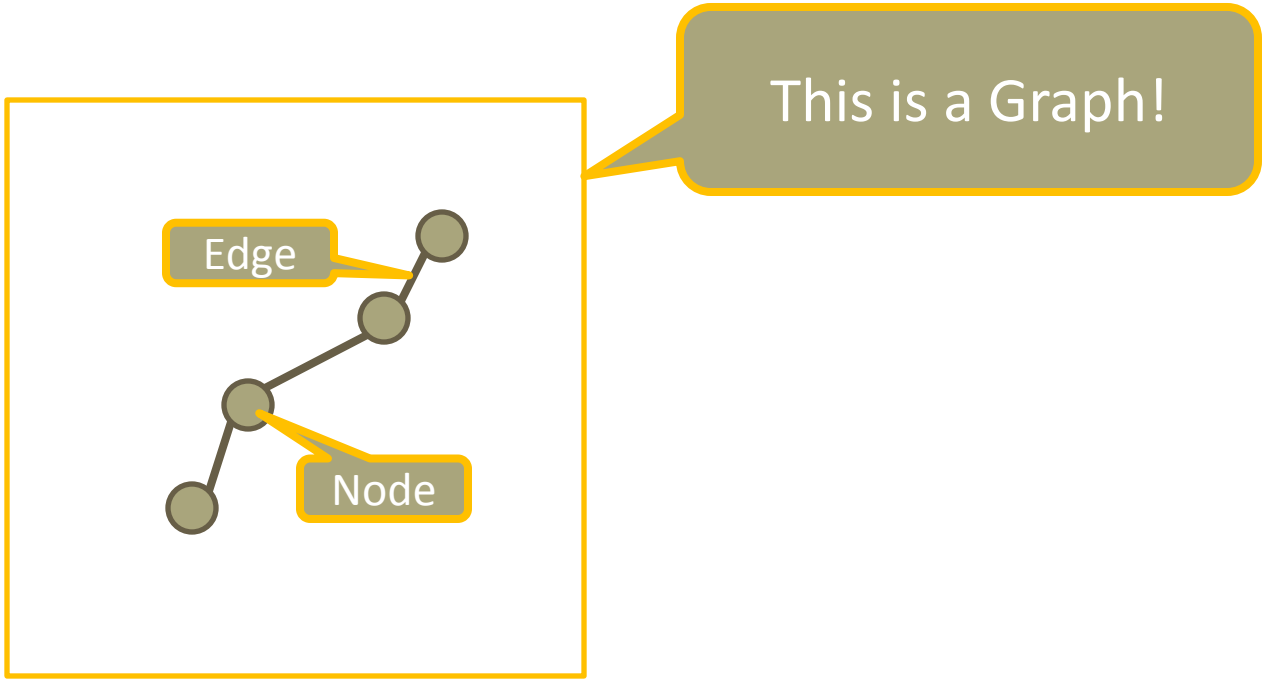
Graph Data: What is it? (3)



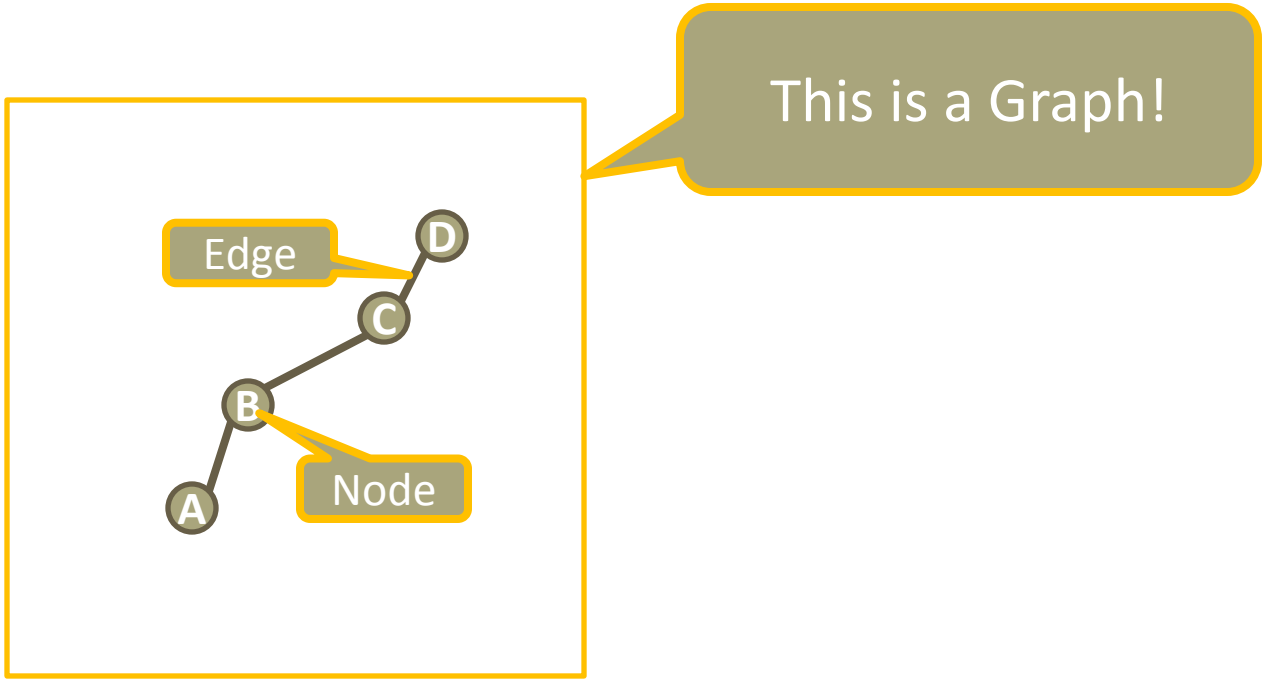
Graph Data: What is it? (4)



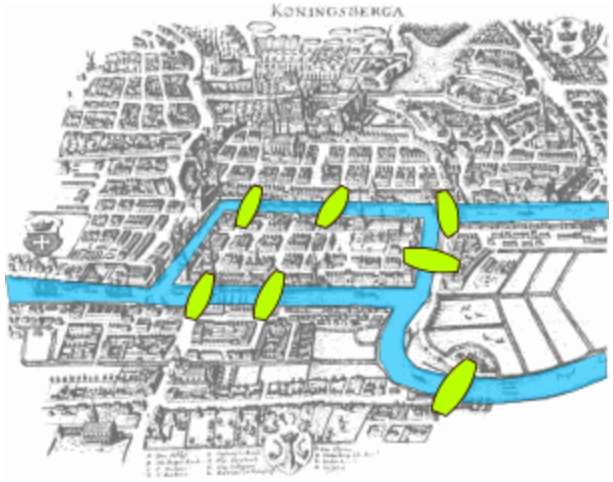
Graph Data: What is it? (5)



Graph Data: What is it? (6)



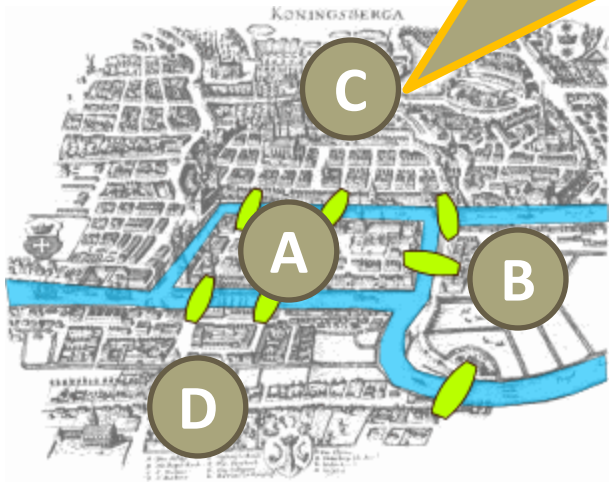
Graph Data: Euler's Seven Bridges of Königsberg (0)



"find a walk through the city that would cross each bridge once and only once"
http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

Graph Data: Euler's Seven Bridges of Königsberg (1)

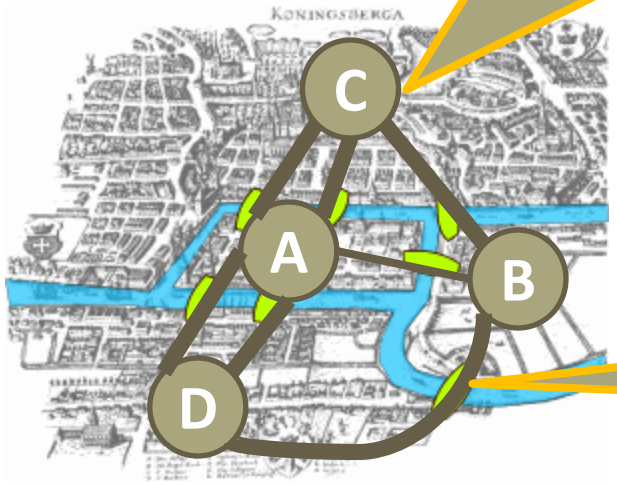
Each separated piece of land is a node



"find a walk through the city that would cross each bridge once and only once"
http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

Graph Data: Euler's Seven Bridges of Königsberg (2)

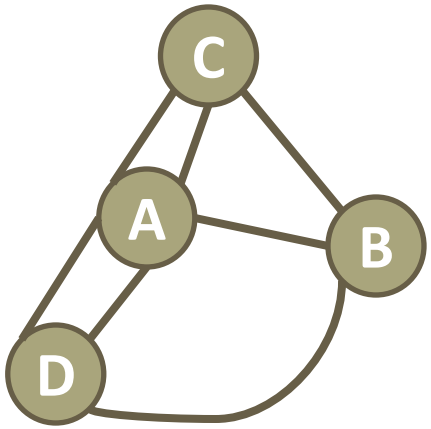
Each separated piece of land is a node



Each bridge is an edge

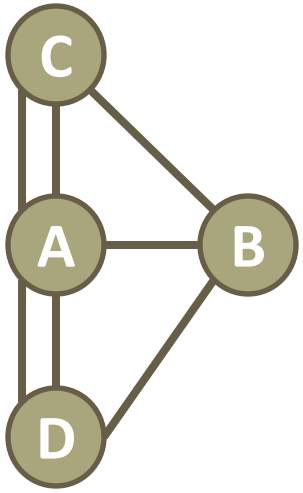
"find a walk through the city that would cross each bridge once and only once"
http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

Graph Data: Euler's Seven Bridges of Königsberg (3)



"find a walk through the city that would cross each bridge once and only once"
http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

Graph Data: Euler's Seven Bridges of Königsberg (4)

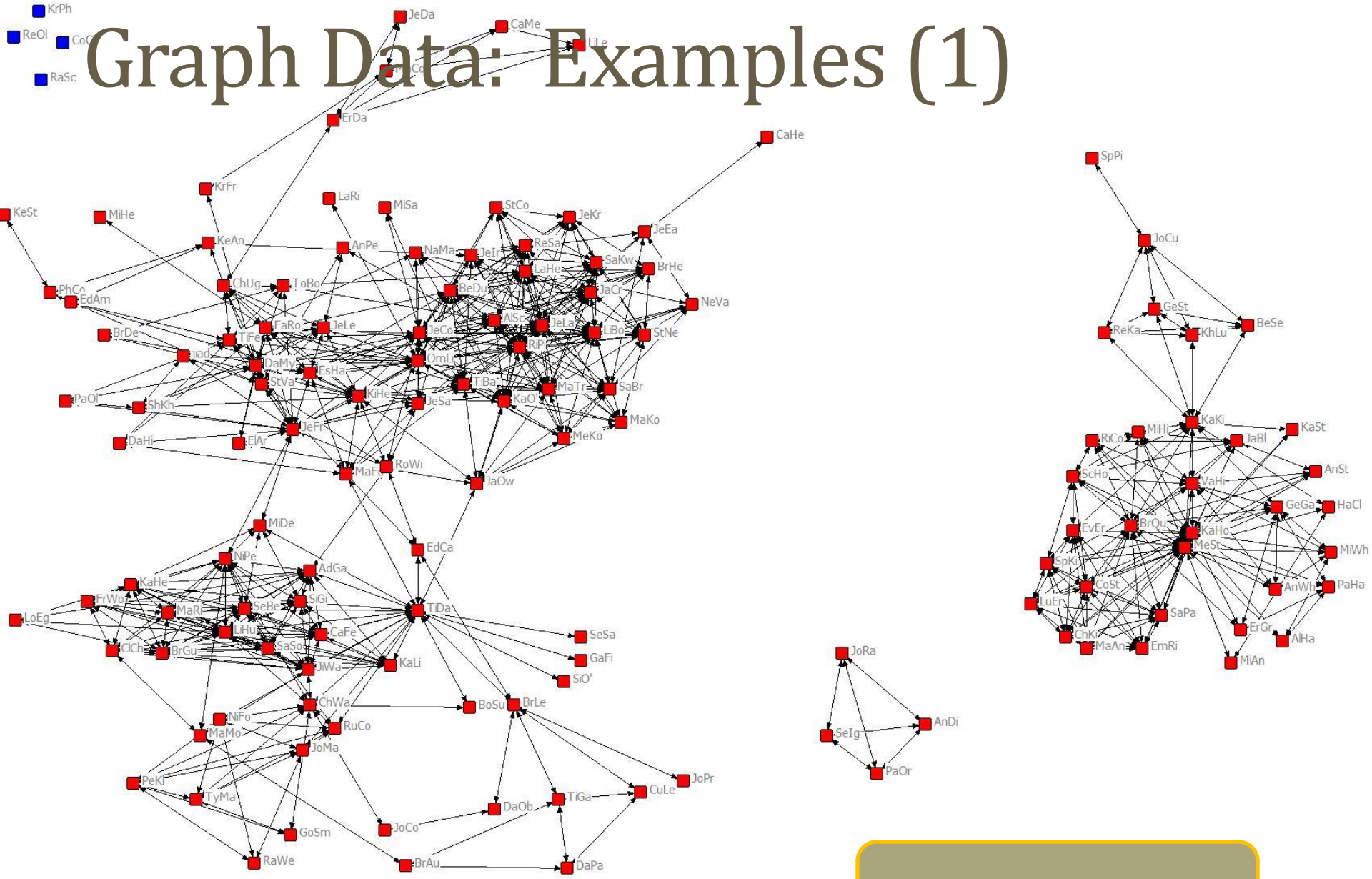


"find a walk through the city that would cross each bridge once and only once"
http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

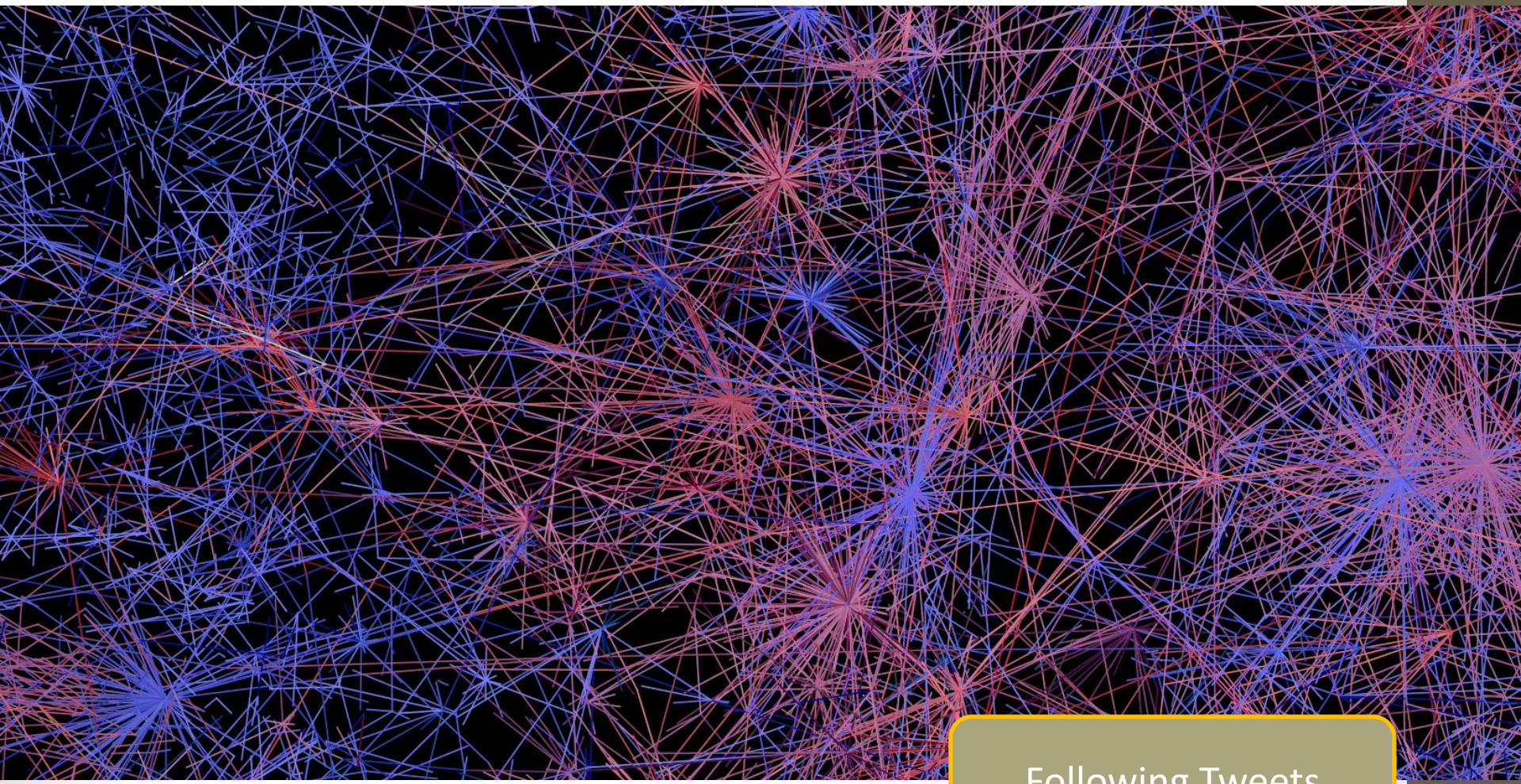
Graph Data: Examples (0)

- Graphs represent many things like Associations and Networks.
- Graphs represent the world wide web
- Graphs represent Facebook networks
- Graphs represent metabolic pathways

Graph Data: Examples (1)

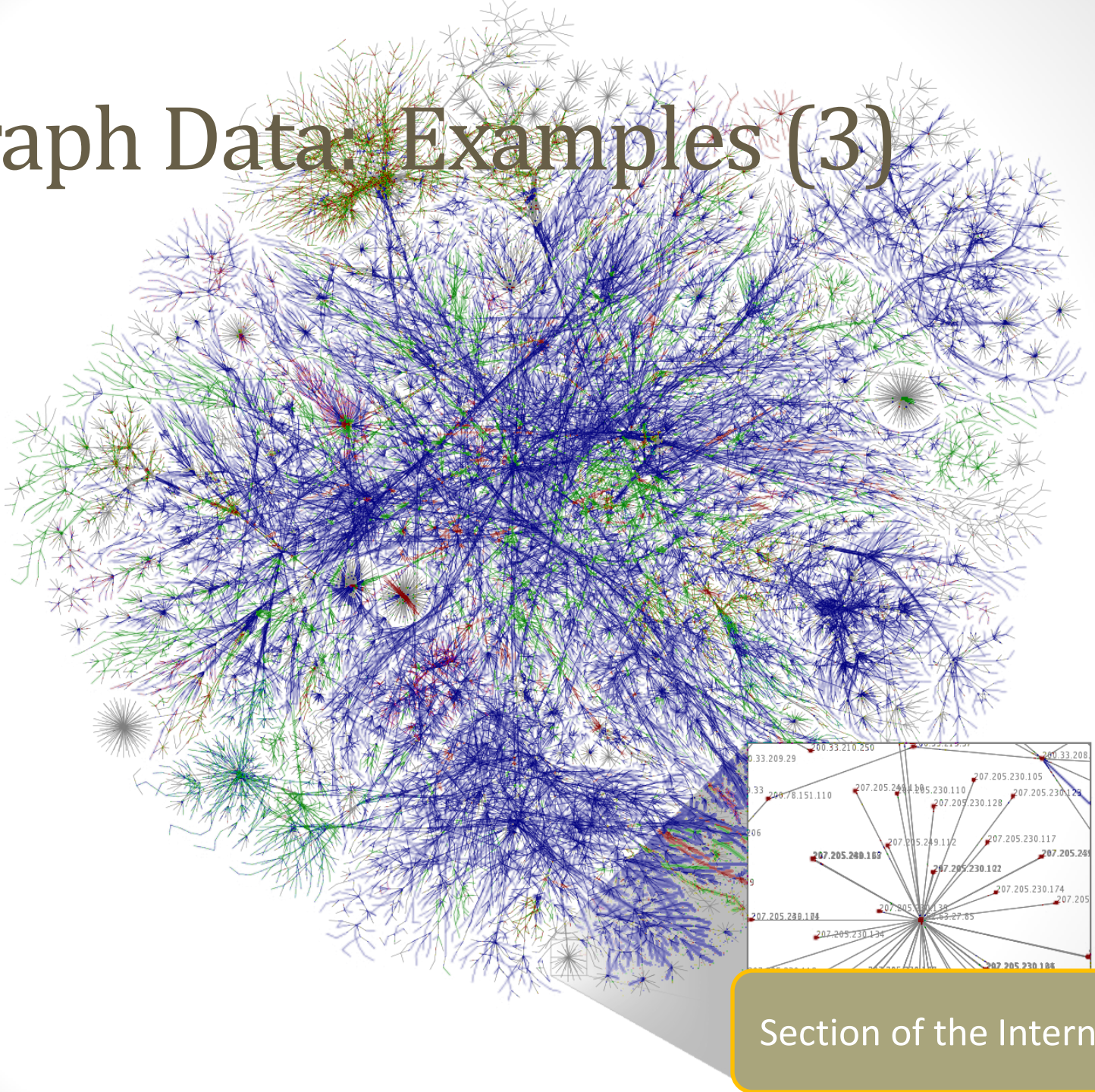


Graph Data: Examples (2)

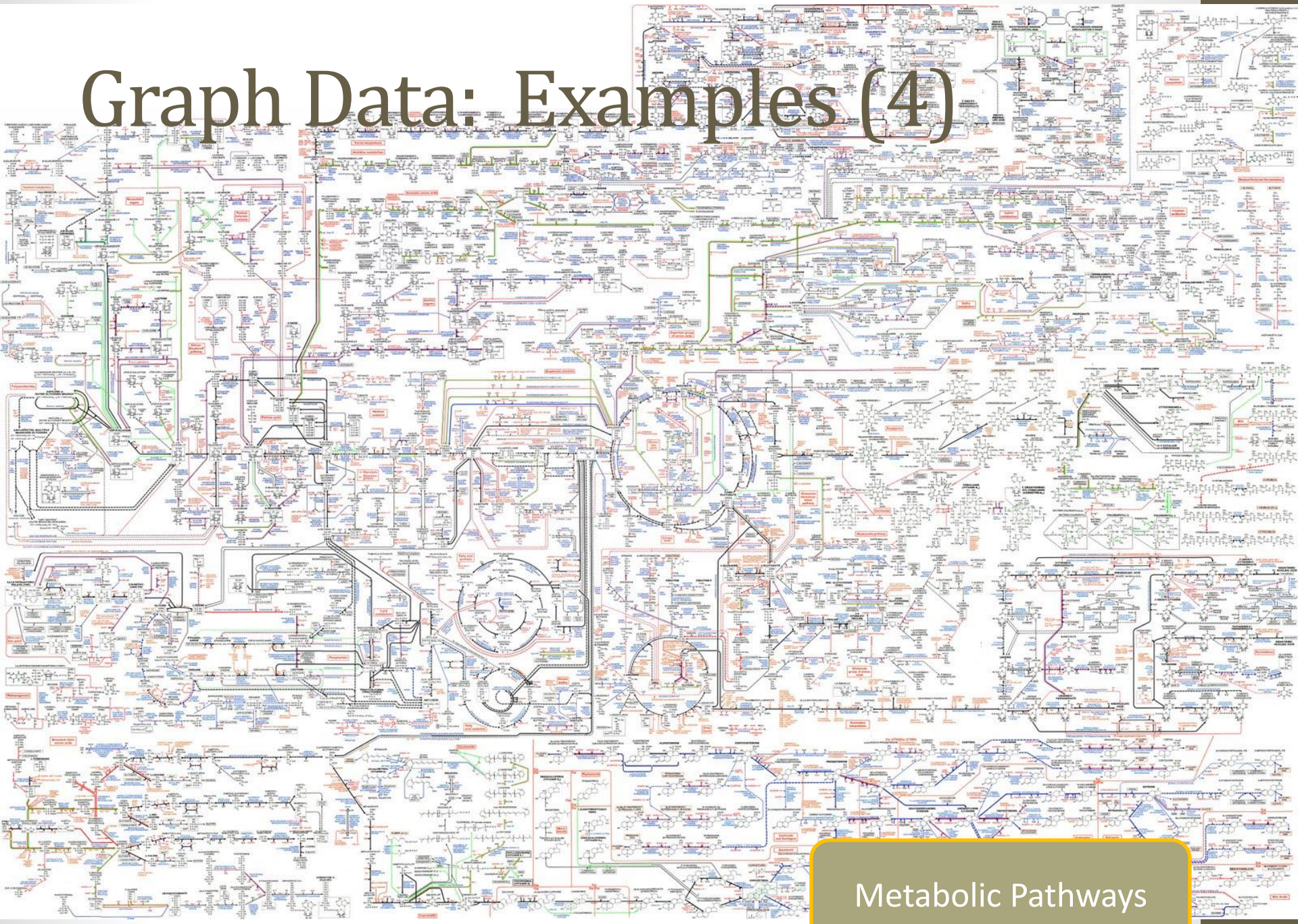


Following Tweets

Graph Data: Examples (3)

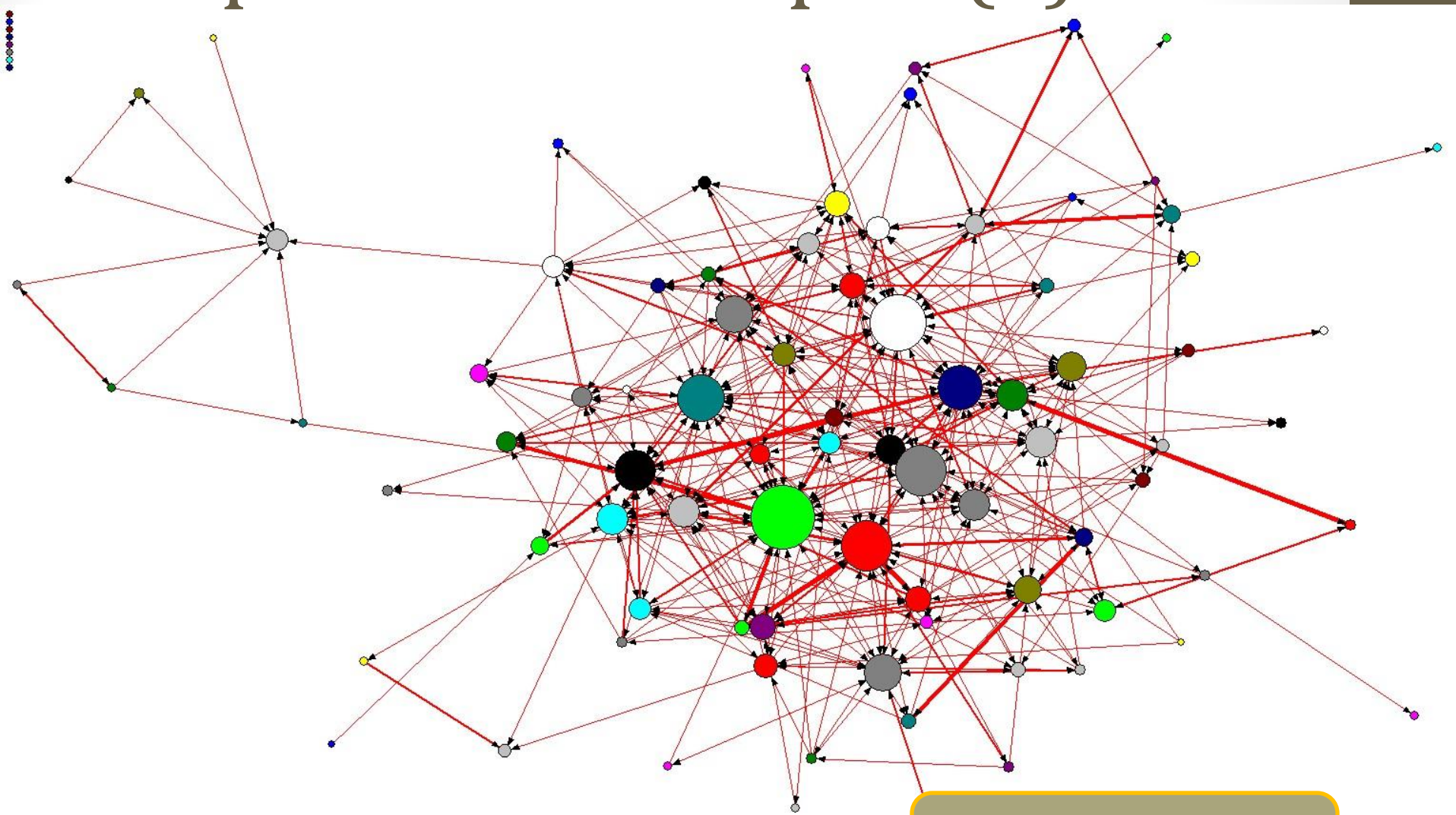


Graph Data: Examples (4)



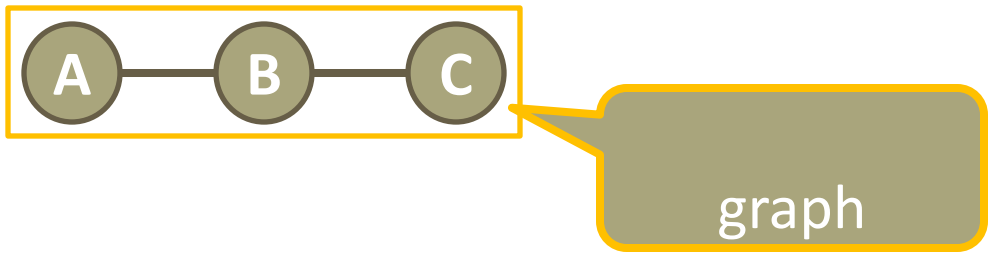
Metabolic Pathways

Graph Data: Examples (5)

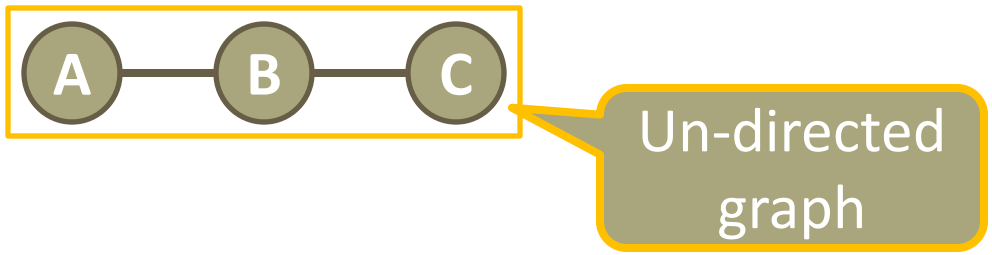


Ranked Web Pages

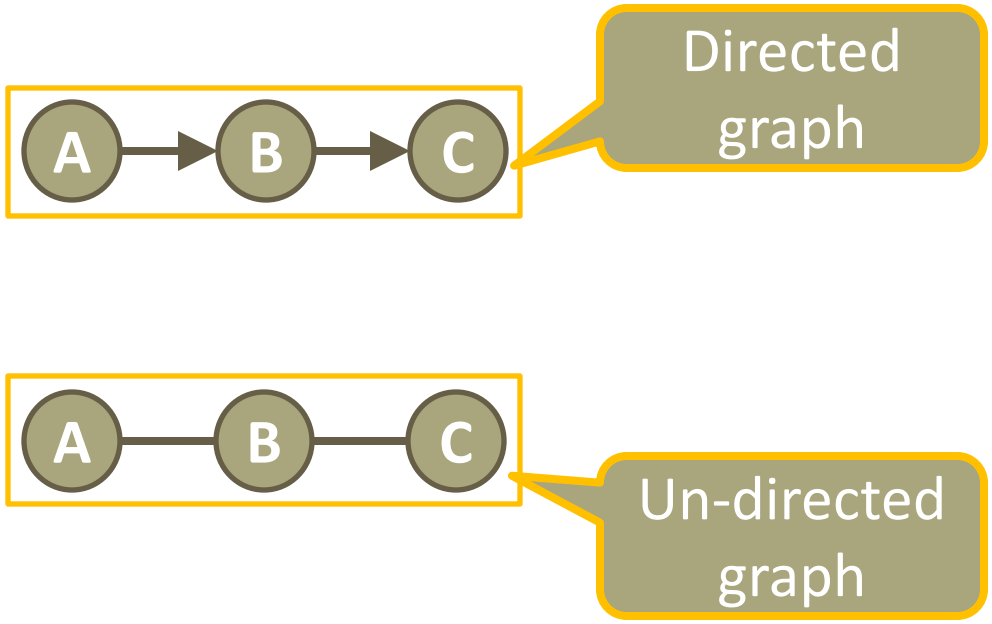
Graph Data: Directed and Un-directed Graphs (0)



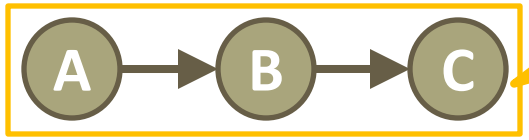
Graph Data: Directed and Un-directed Graphs (1)



Graph Data: Directed and Un-directed Graphs (2)

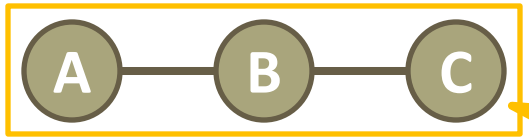


Graph Data: Directed and Un-directed Graphs (3)



Directed graph

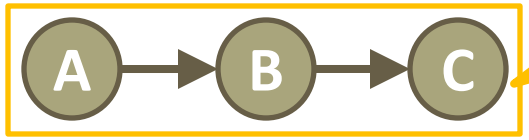
"The link goes from A to B"



Un-directed graph

"The link is between A and B"

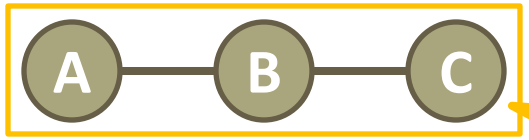
Graph Data: Directed and Un-directed Graphs (4)



Directed graph

"The link goes from A to B"

Example of directed graphs: World Wide Web

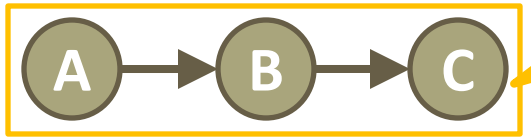


Un-directed graph

"The link is between A and B"

Examples of un-directed graphs: Facebook, LinkedIn

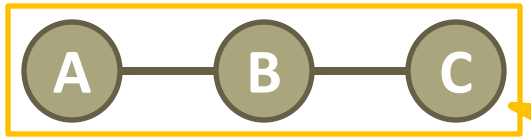
Graph Data: Directed and Un-directed Graphs (5)



Directed graph

"The link goes from A to B"

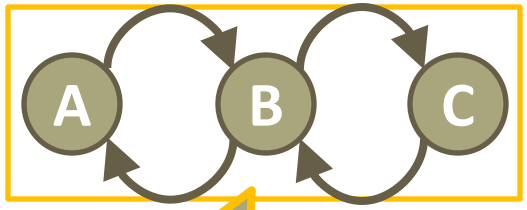
Example of directed graphs: World Wide Web



Un-directed graph

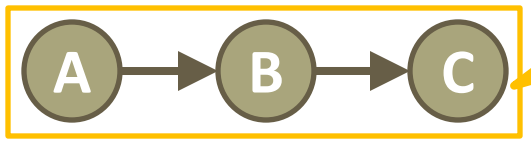
"The link is between A and B"

Examples of un-directed graphs: Facebook, LinkedIn



graph with bi-directional links

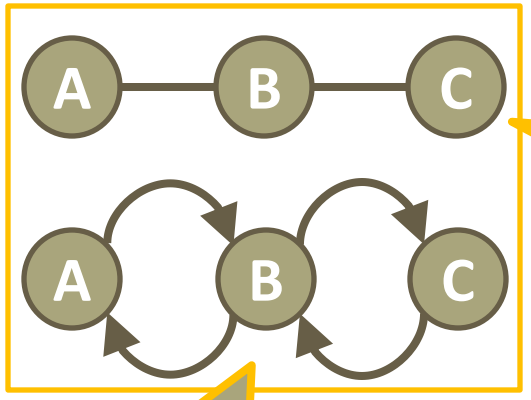
Graph Data: Directed and Un-directed Graphs (6)



Directed graph

"The link goes from A to B"

Example of directed graphs: World Wide Web



Un-directed graph

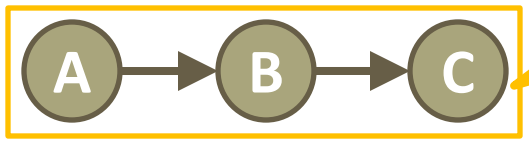
"The link is between A and B"

Examples of un-directed graphs: Facebook, LinkedIn

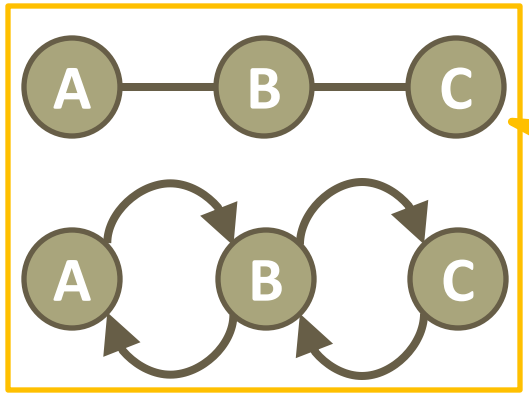
graph with bi-directional links

A graph with bi-directional links is like an undirected graph

Graph Data: Formalize as Rectangular Data (0)



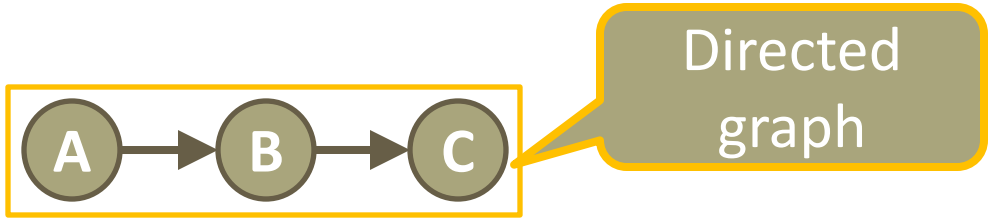
Directed graph



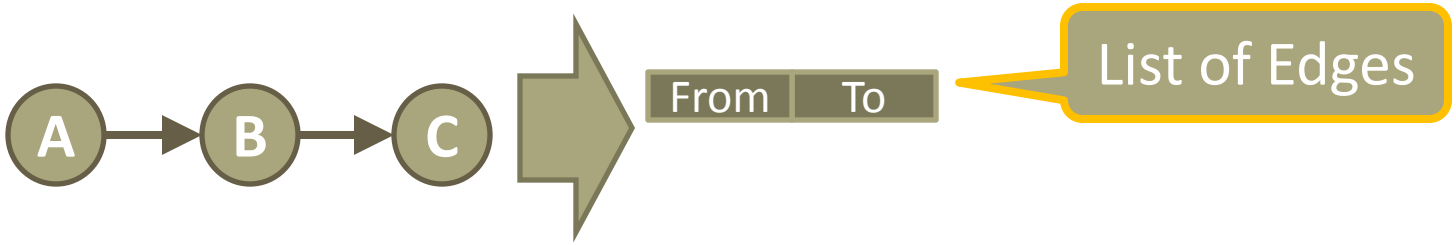
Un-directed graph

- How can we perform operations on graphs?
- How can we formalize graphs as rectangular data?

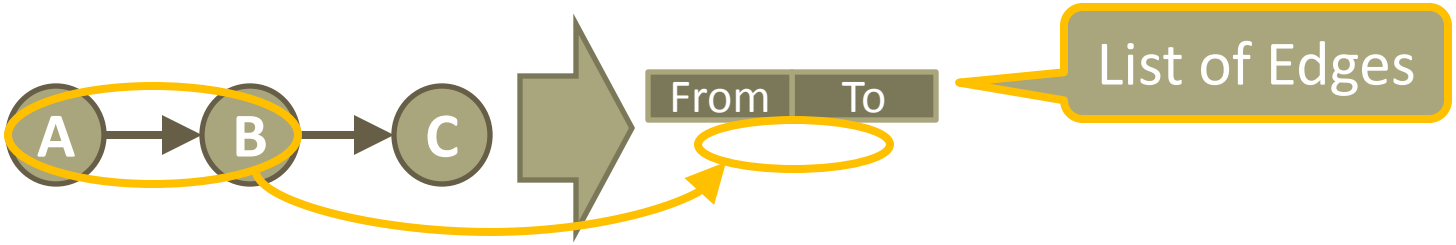
Graph Data: Formalize as Rectangular Data (1)



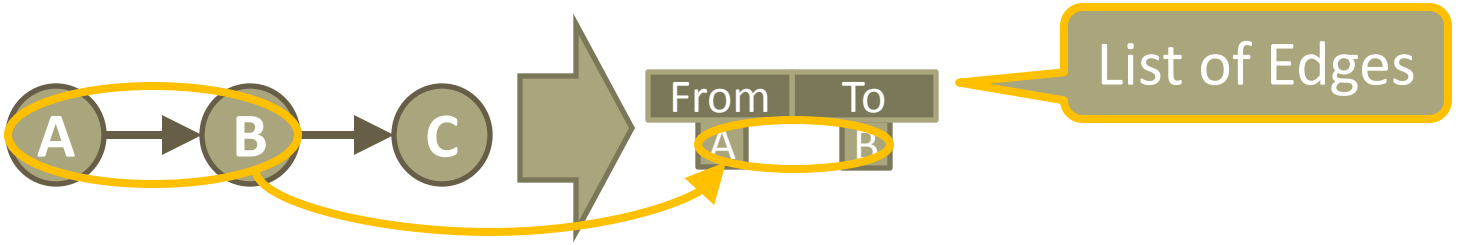
Graph Data: Formalize as Rectangular Data (2)



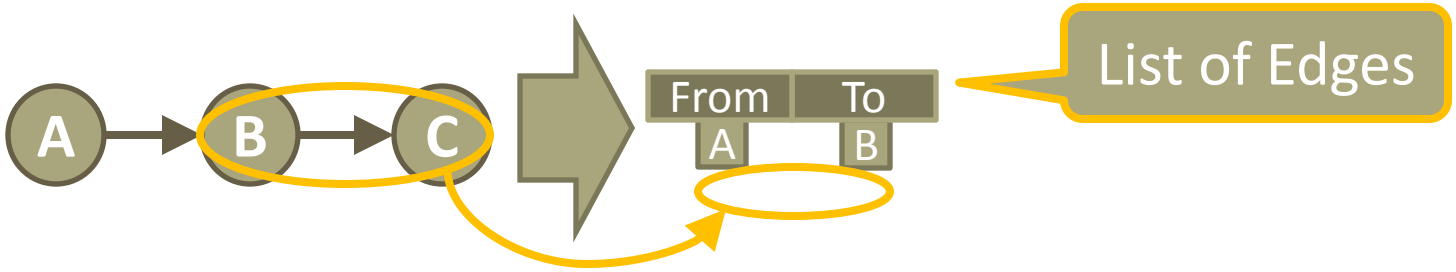
Graph Data: Formalize as Rectangular Data (3)



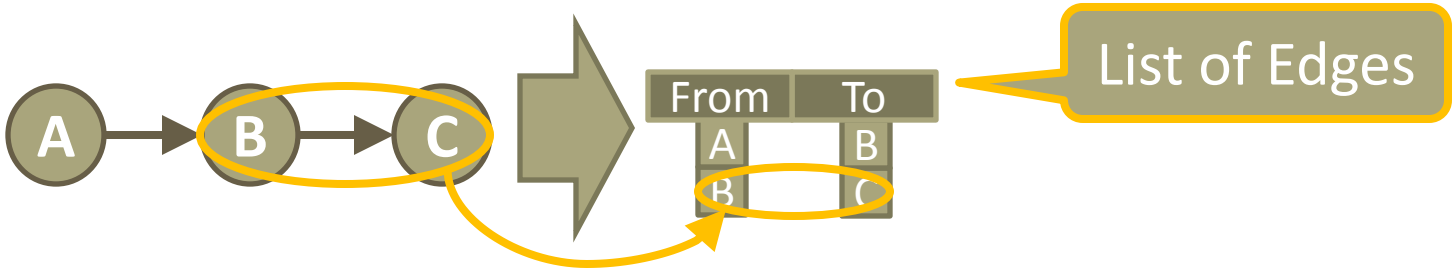
Graph Data: Formalize as Rectangular Data (4)



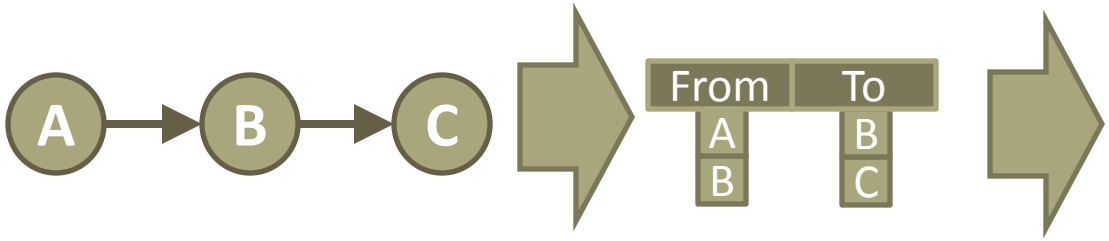
Graph Data: Formalize as Rectangular Data (5)



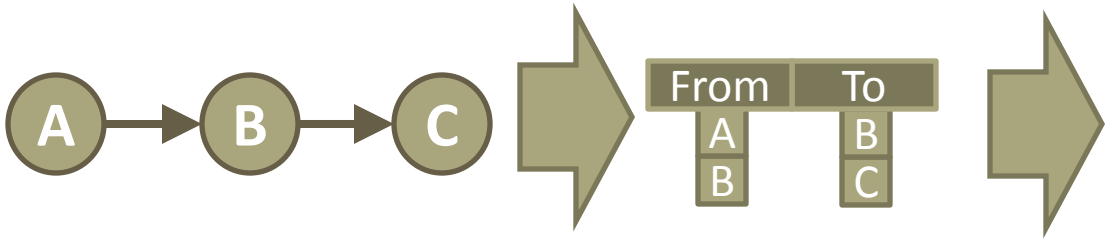
Graph Data: Formalize as Rectangular Data (6)



Graph Data: Formalize as Rectangular Data (7)



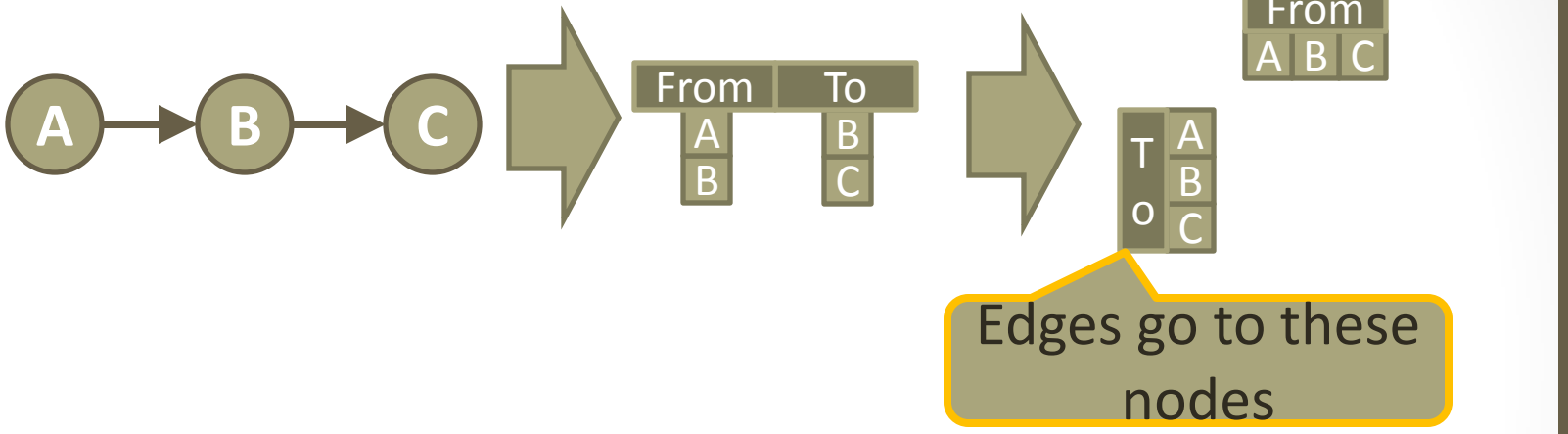
Graph Data: Formalize as Rectangular Data (8)



Edges start from these nodes

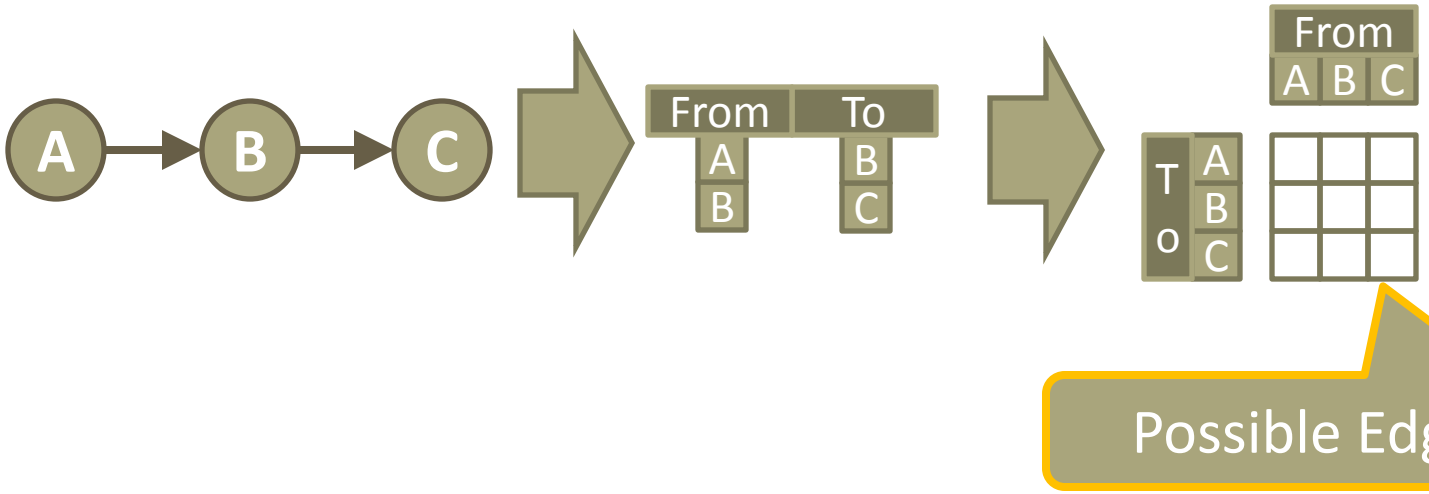
From
A
B
C

Graph Data: Formalize as Rectangular Data (9)



Every node could emanate and receive links.

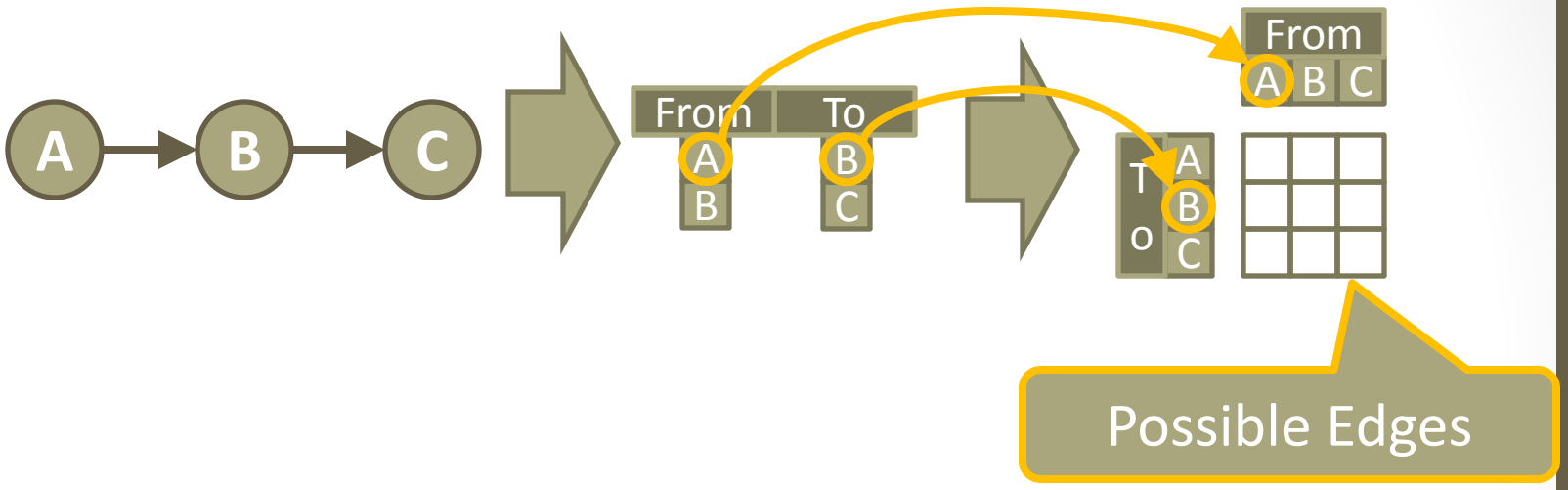
Graph Data: Formalize as Rectangular Data (10)



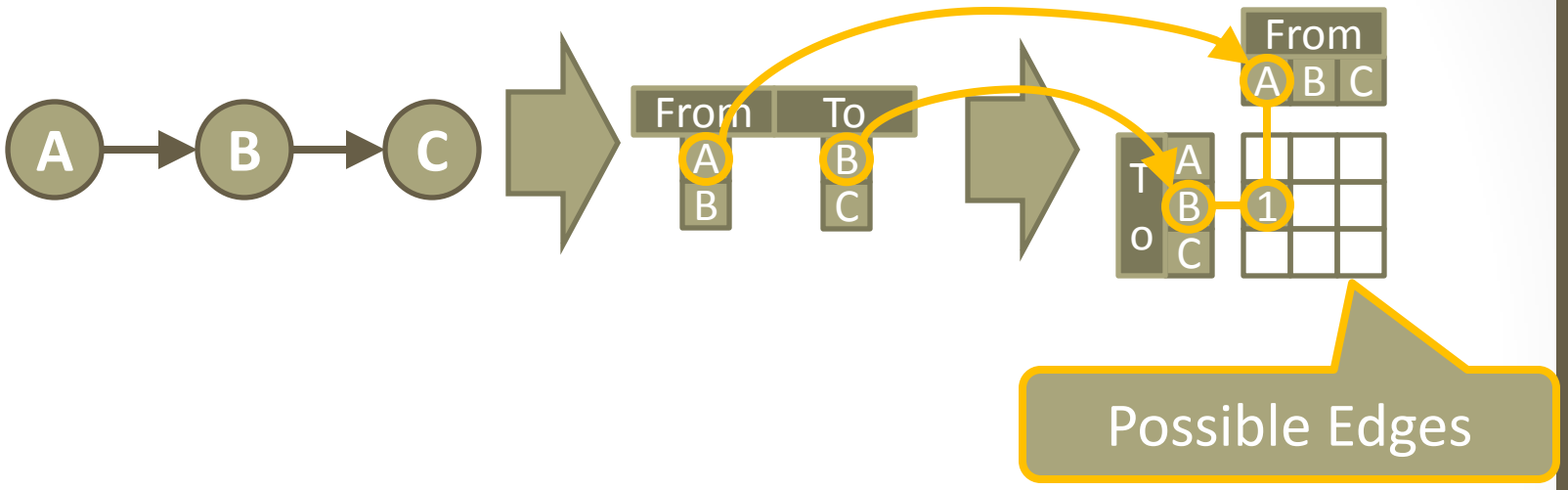
Possible Edges

Every node could emanate and receive links. Therefore the matrix is square.

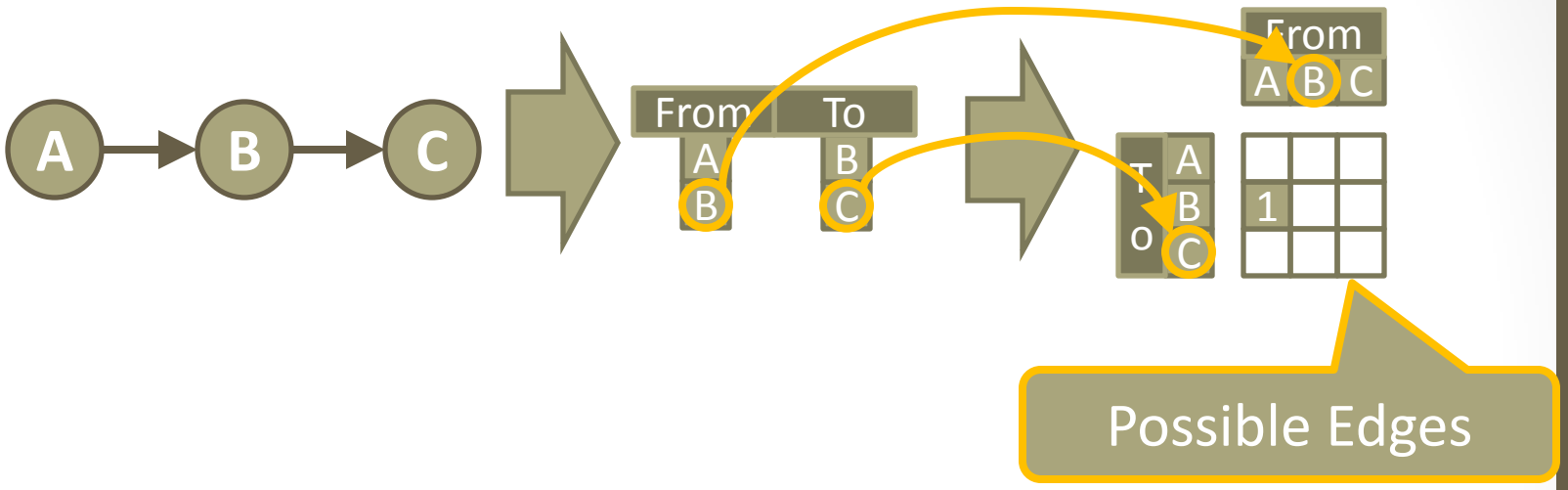
Graph Data: Formalize as Rectangular Data (11)



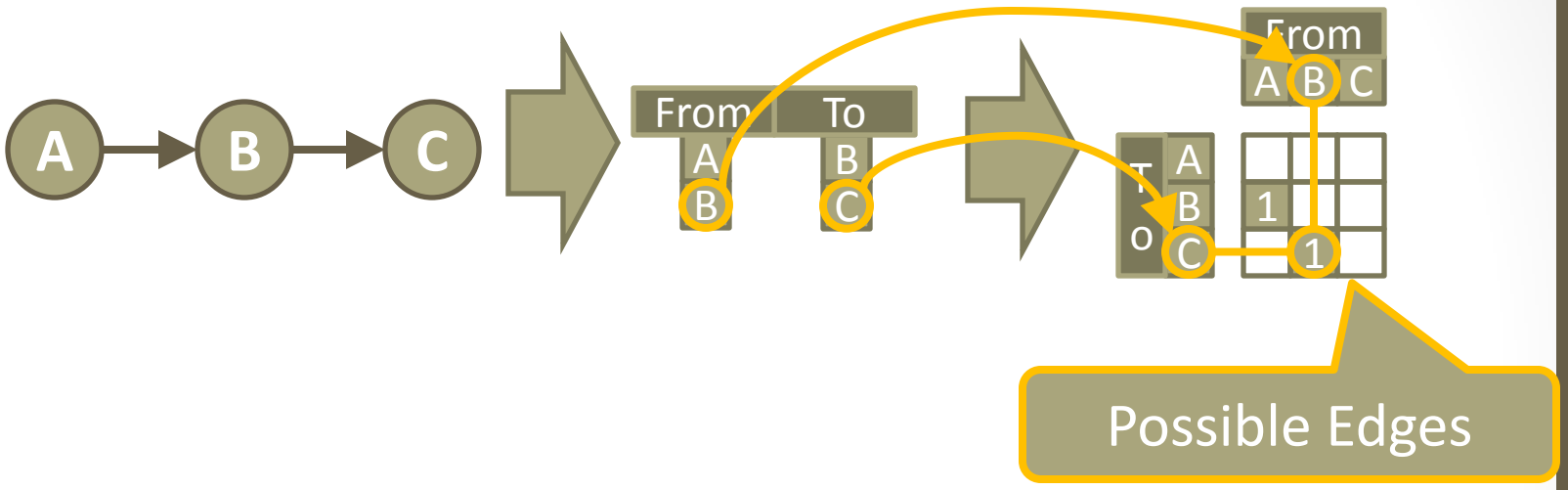
Graph Data: Formalize as Rectangular Data (12)



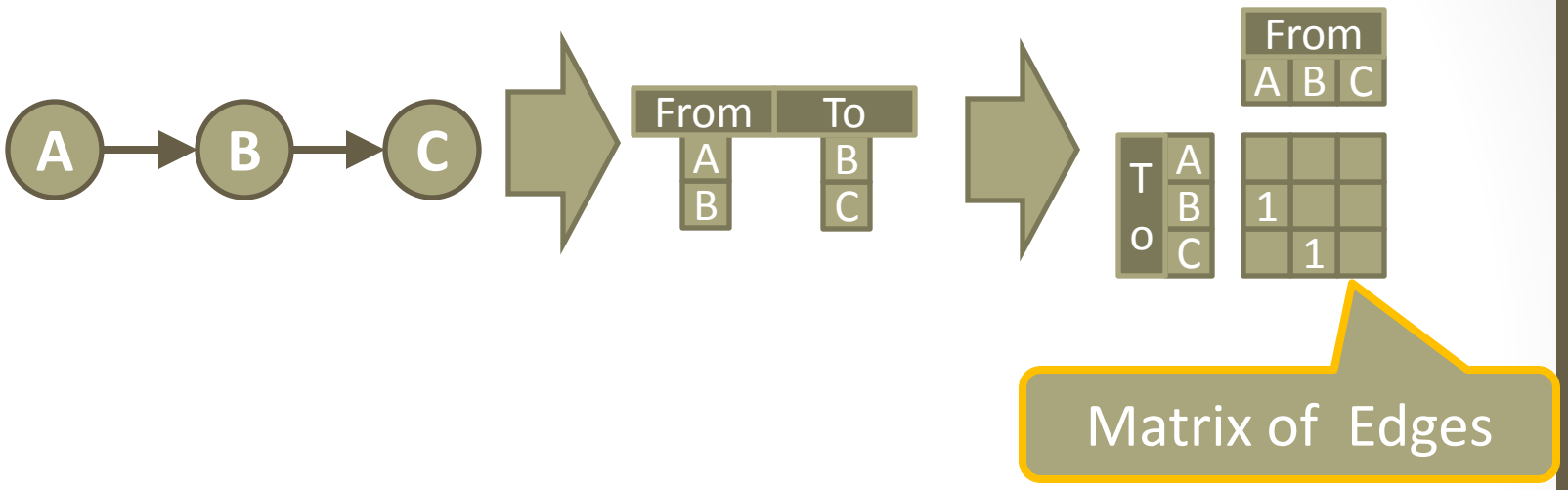
Graph Data: Formalize as Rectangular Data (13)



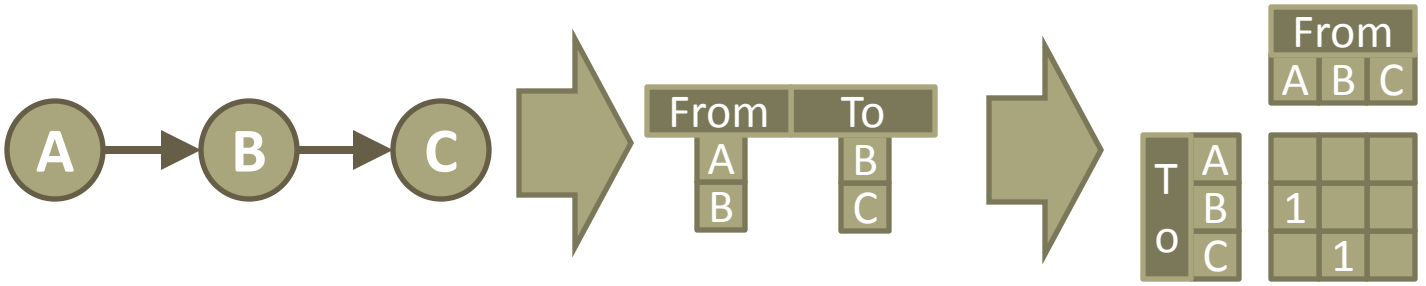
Graph Data: Formalize as Rectangular Data (14)

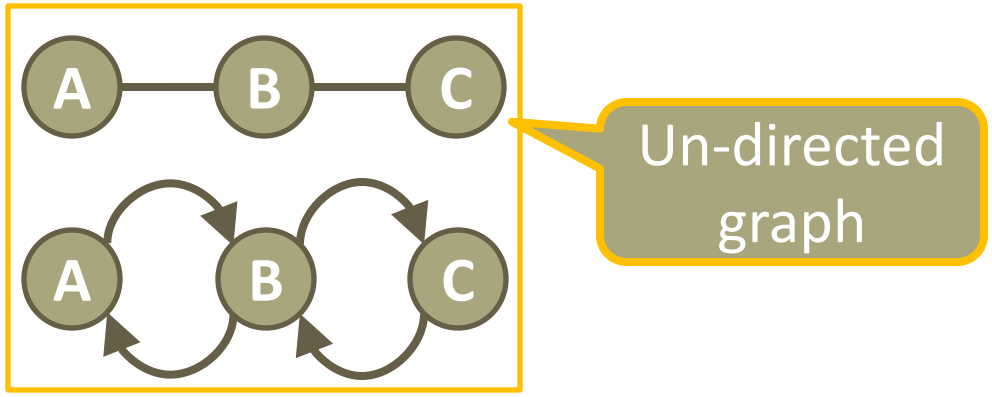


Graph Data: Formalize as Rectangular Data (15)

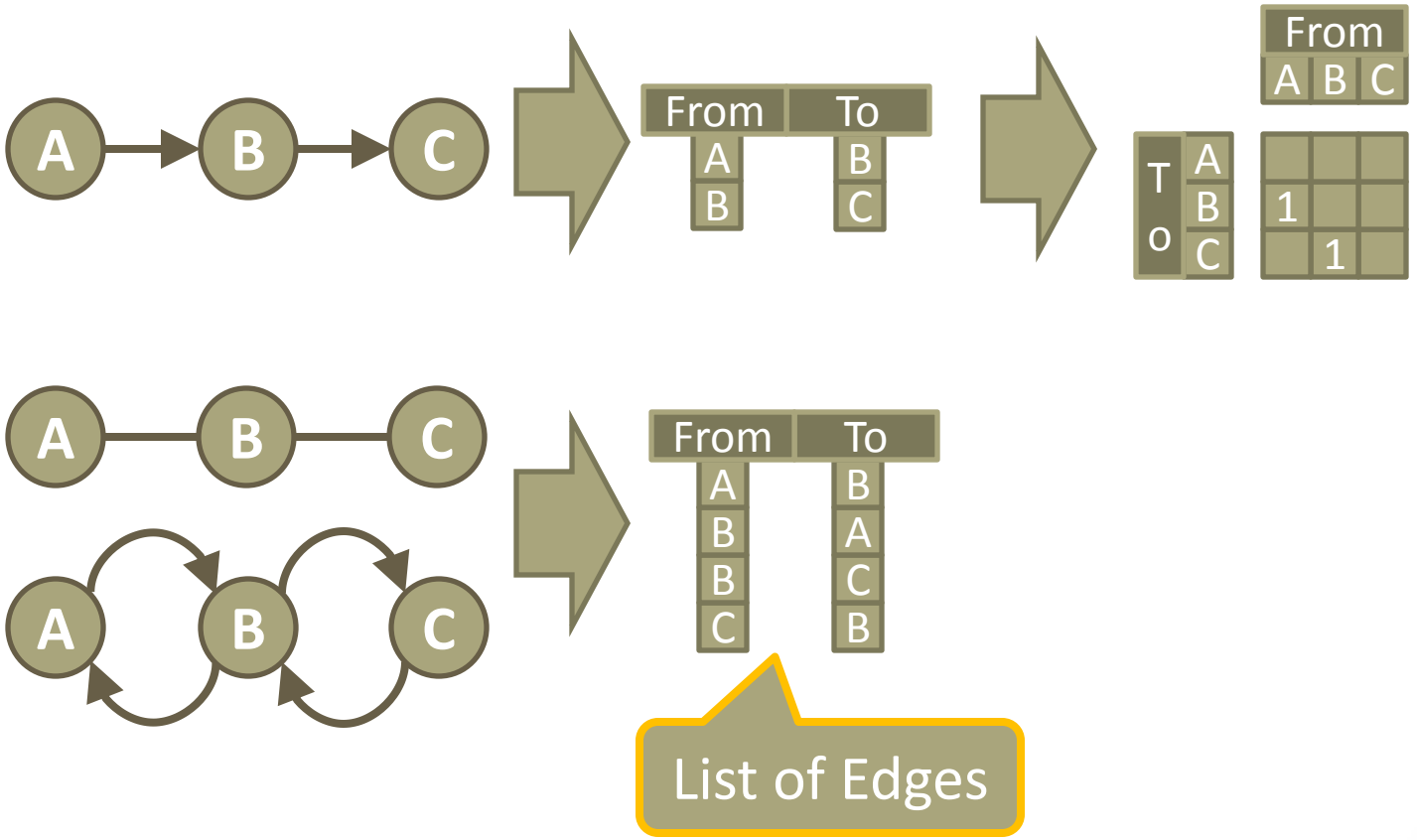


Graph Data: Formalize as Rectangular Data (16)

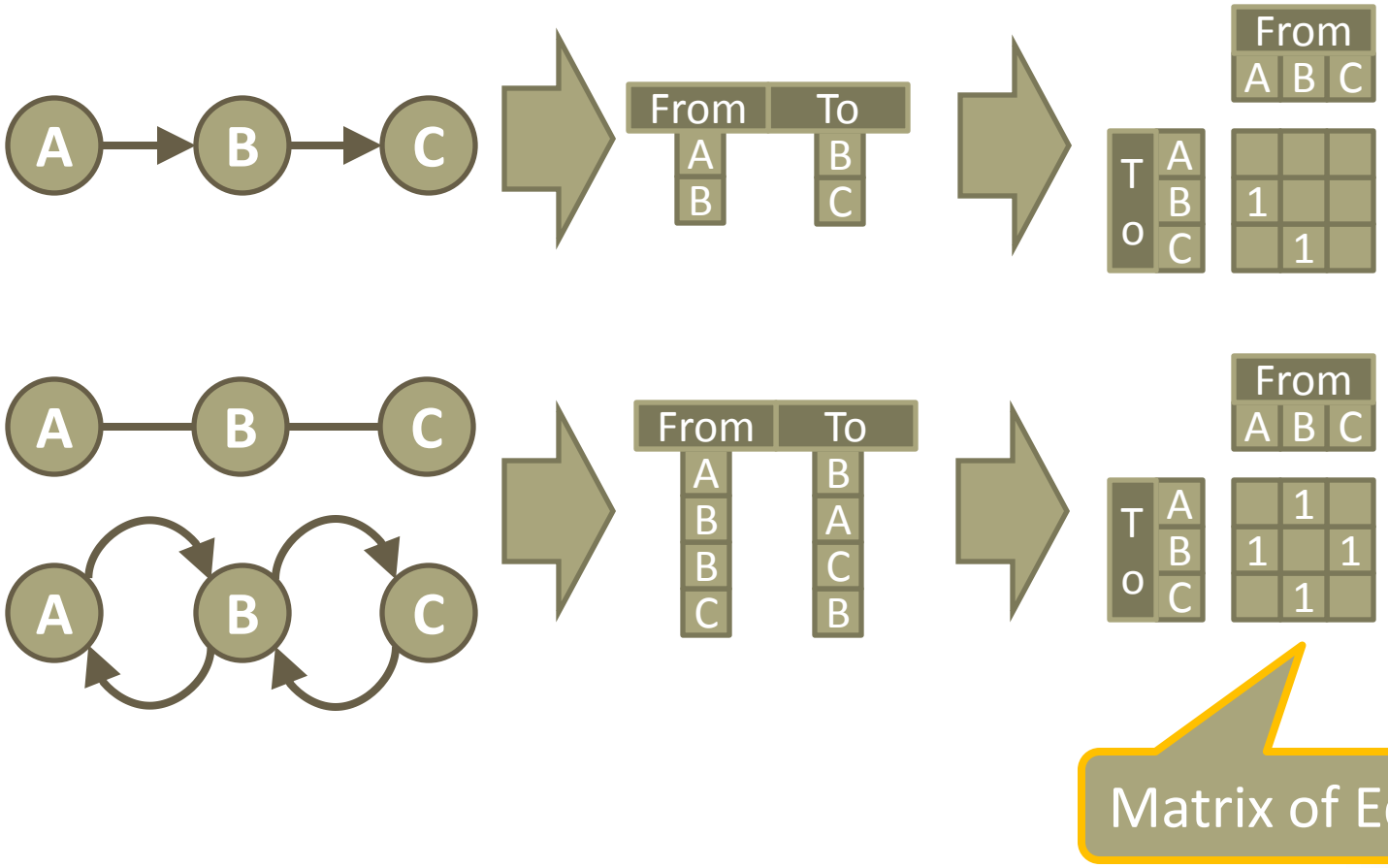




Graph Data: Formalize as Rectangular Data (17)

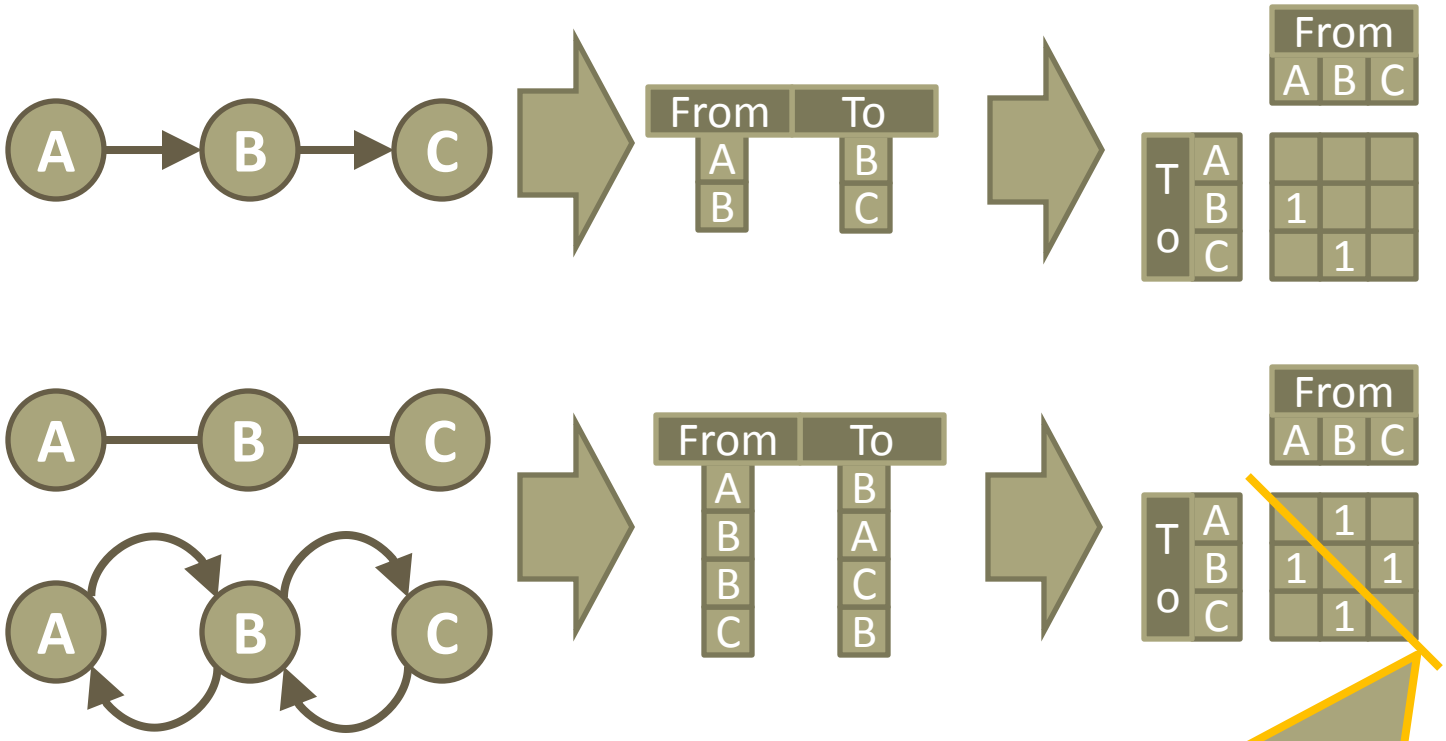


Graph Data: Formalize as Rectangular Data (18)



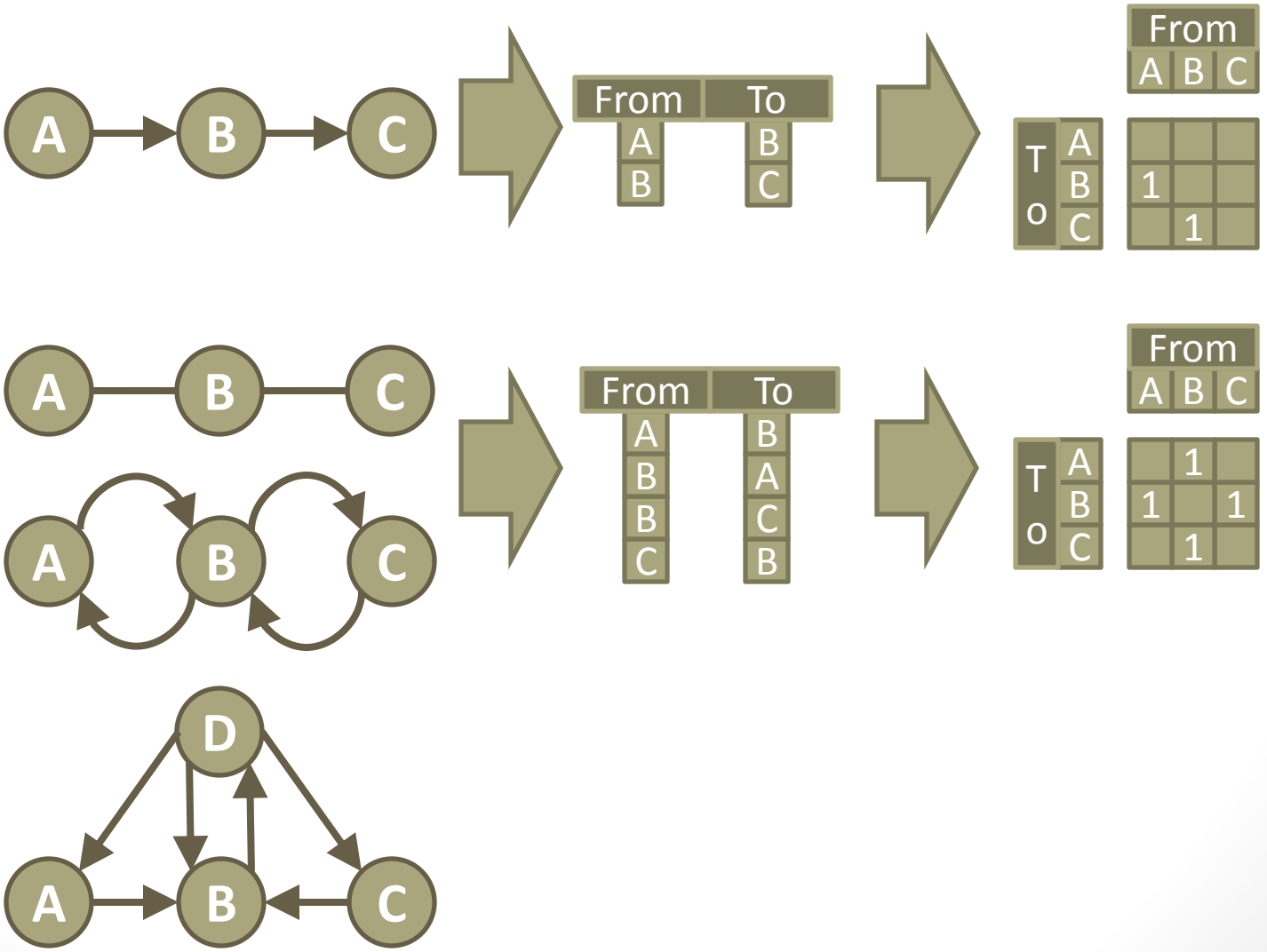
Matrix of Edges

Graph Data: Formalize as Rectangular Data (19)

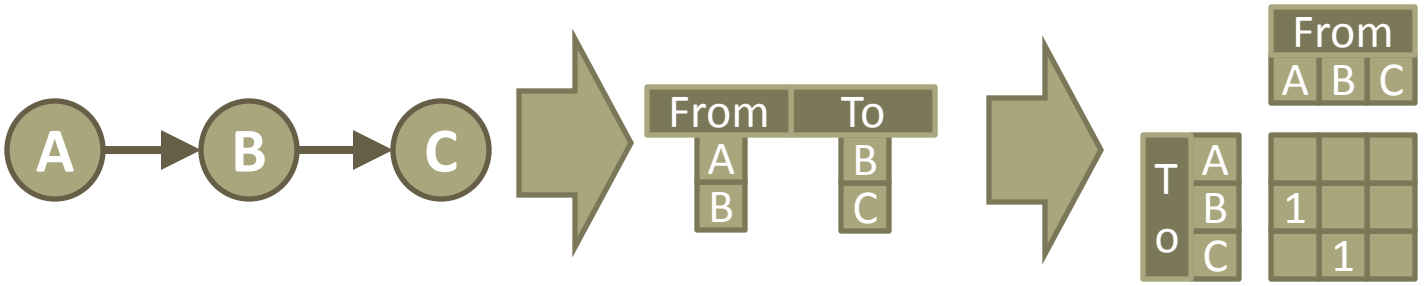


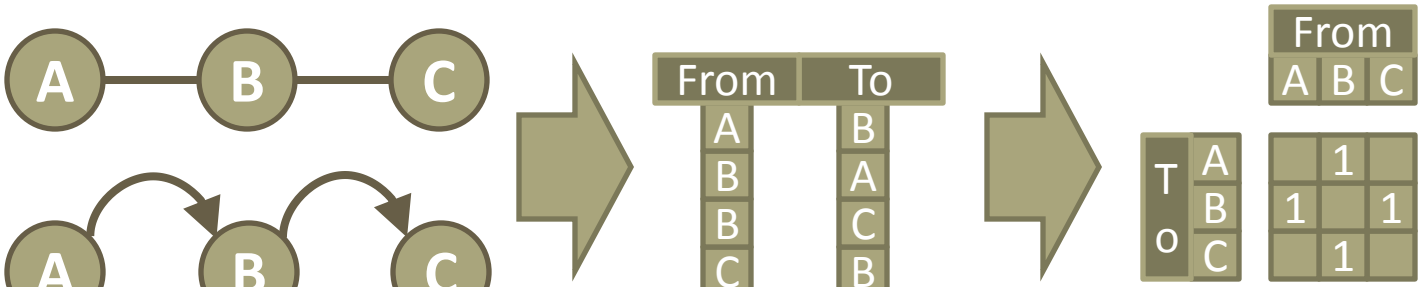
Note: undirected or bi-directional graphs Have symmetric matrices

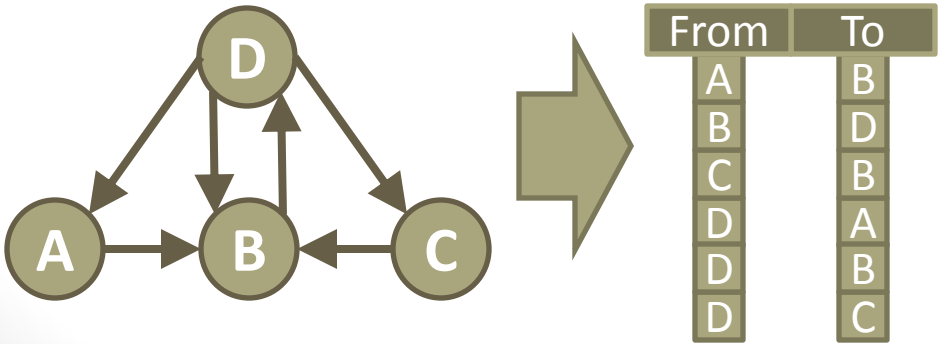
Graph Data: Formalize as Rectangular Data (20)



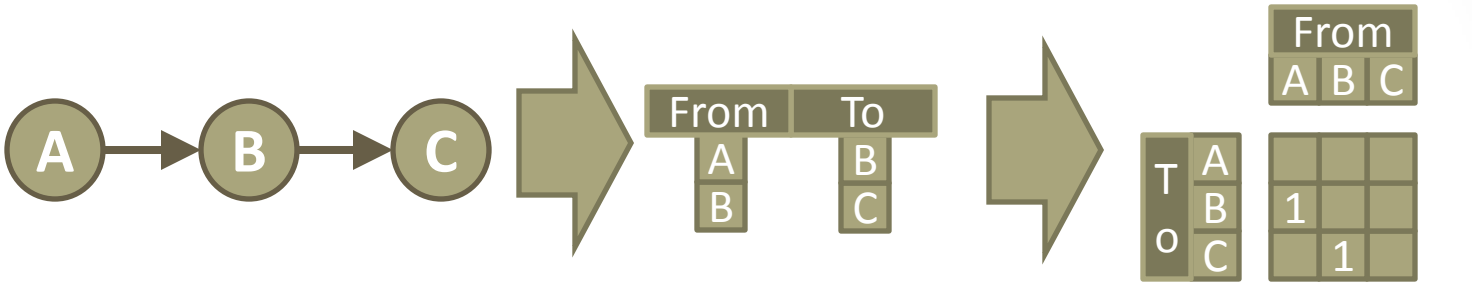
Graph Data: Formalize as Rectangular Data (21)

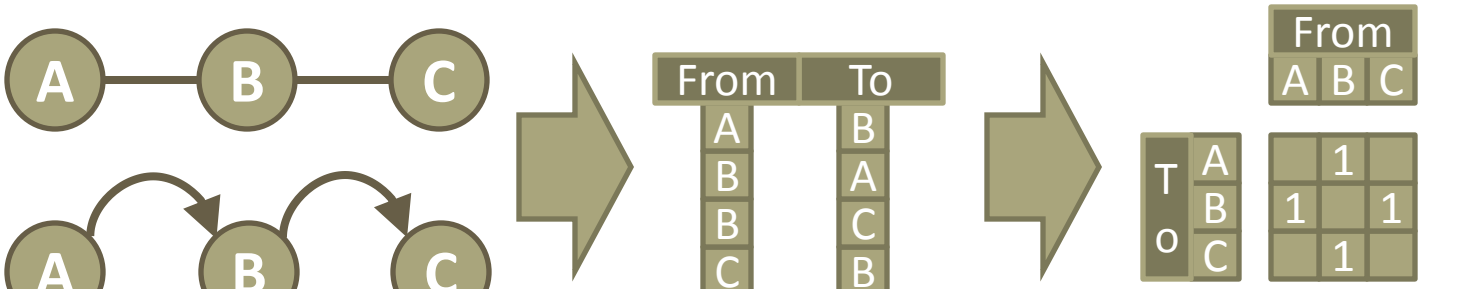


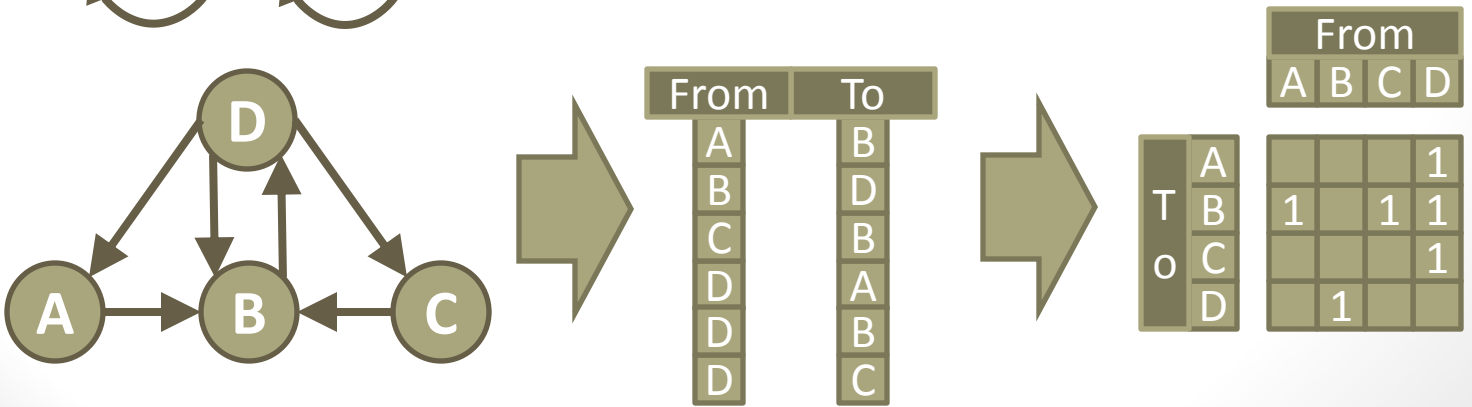




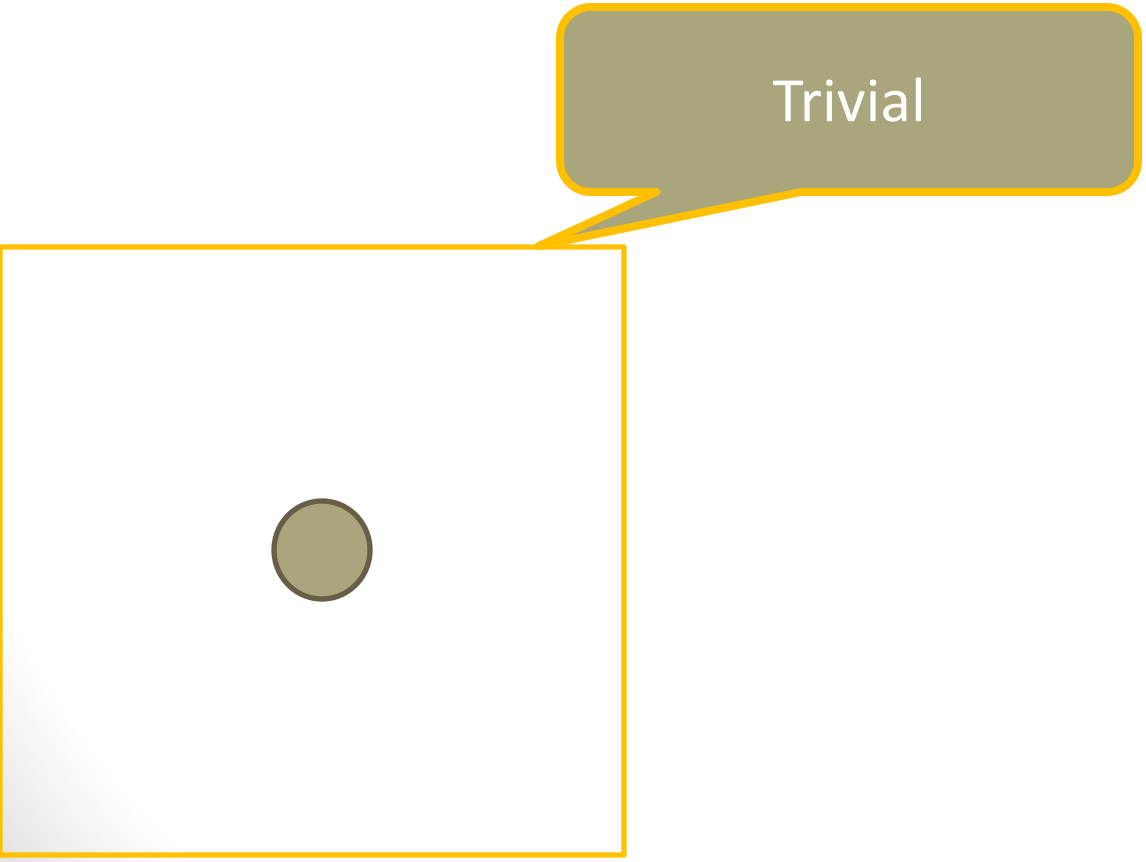
Graph Data: Formalize as Rectangular Data (22)



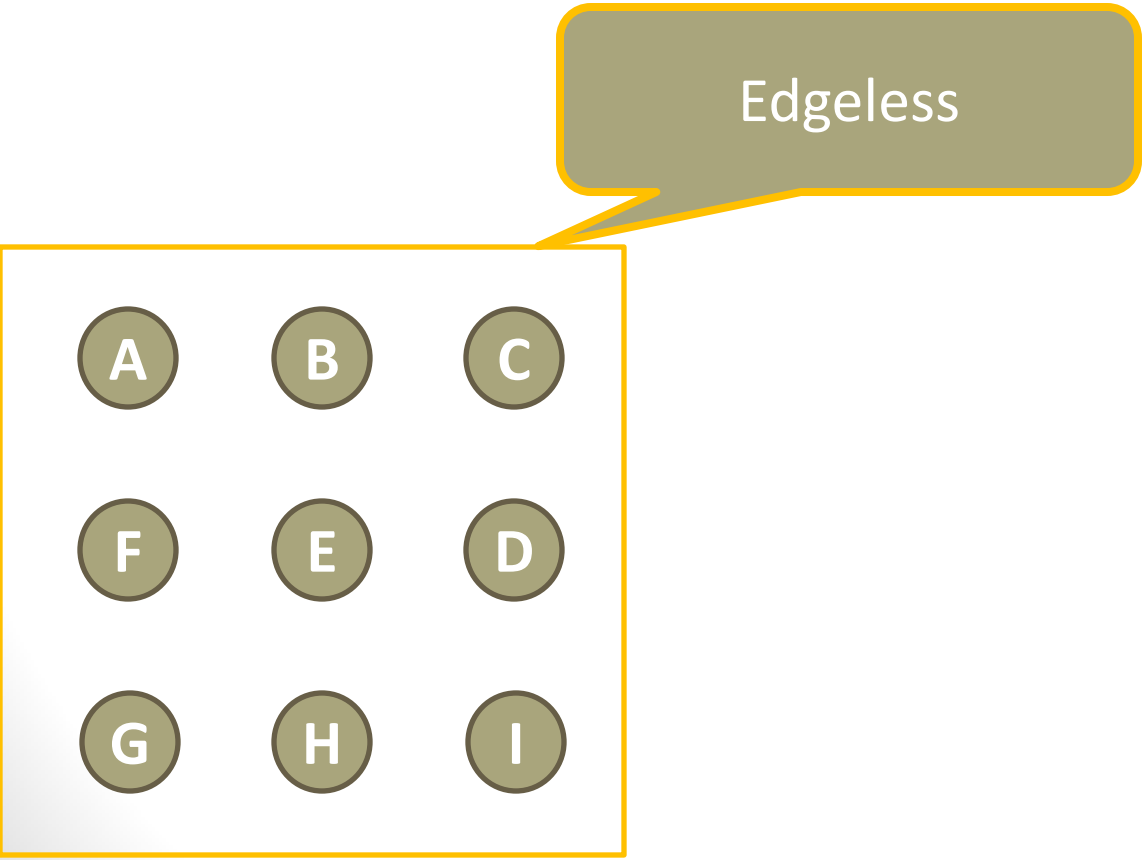




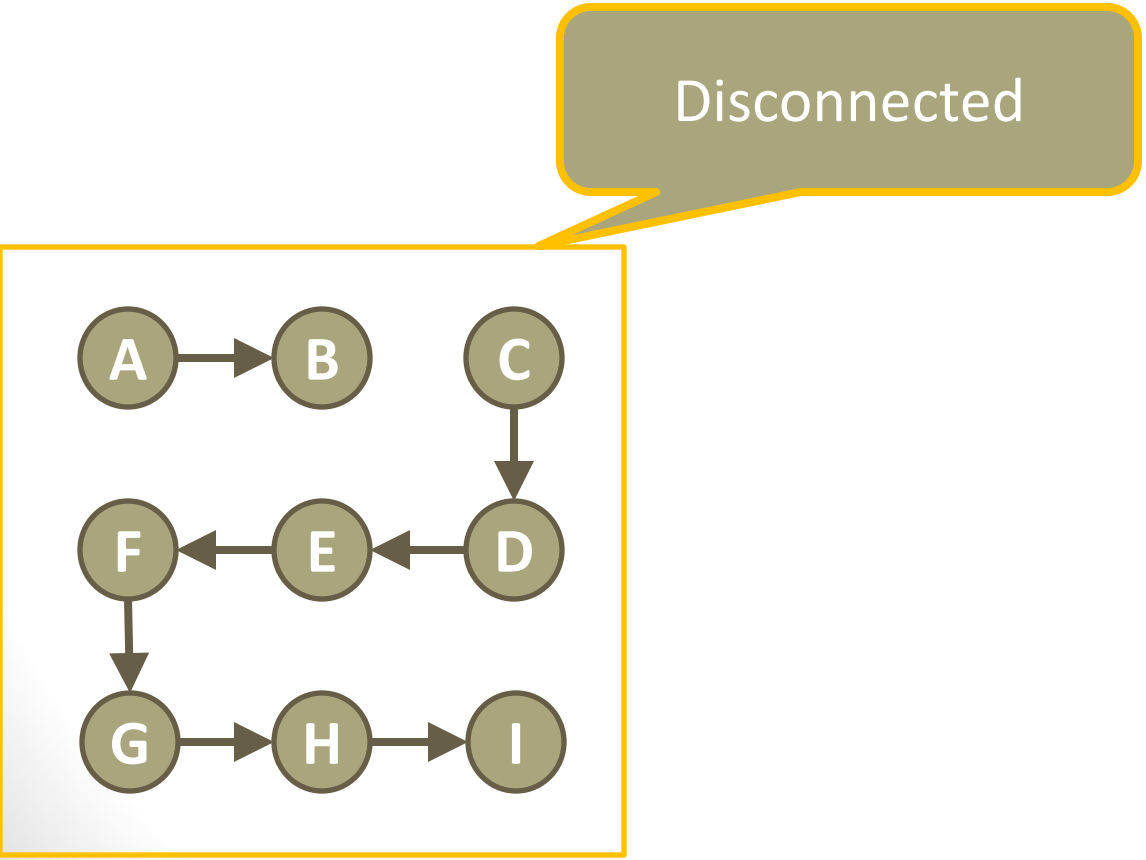
Graph Data: Connectedness and Density (0)



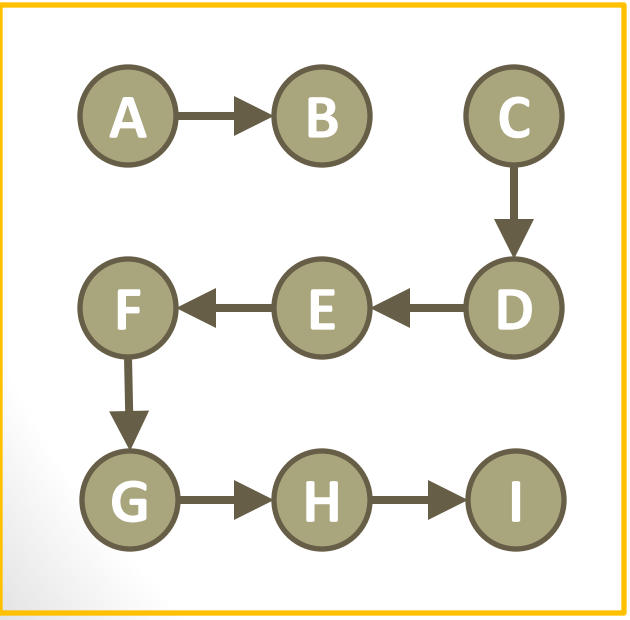
Graph Data: Connectedness and Density (1)

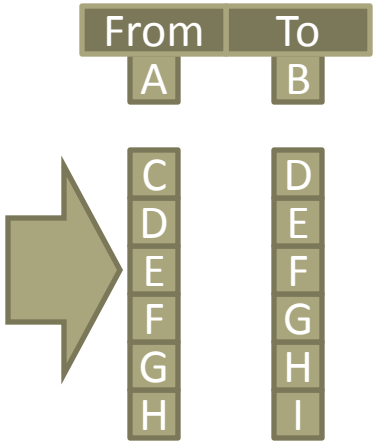


Graph Data: Connectedness and Density (2)

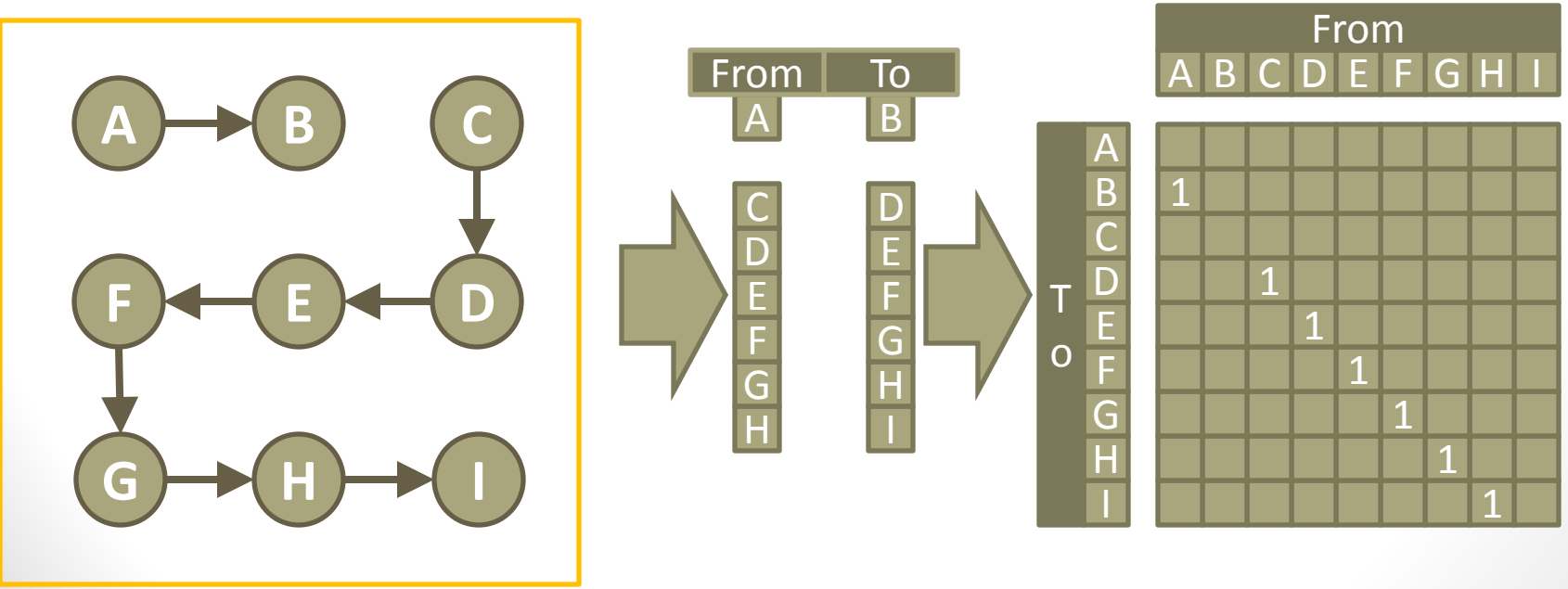


Graph Data: Connectedness and Density (3)

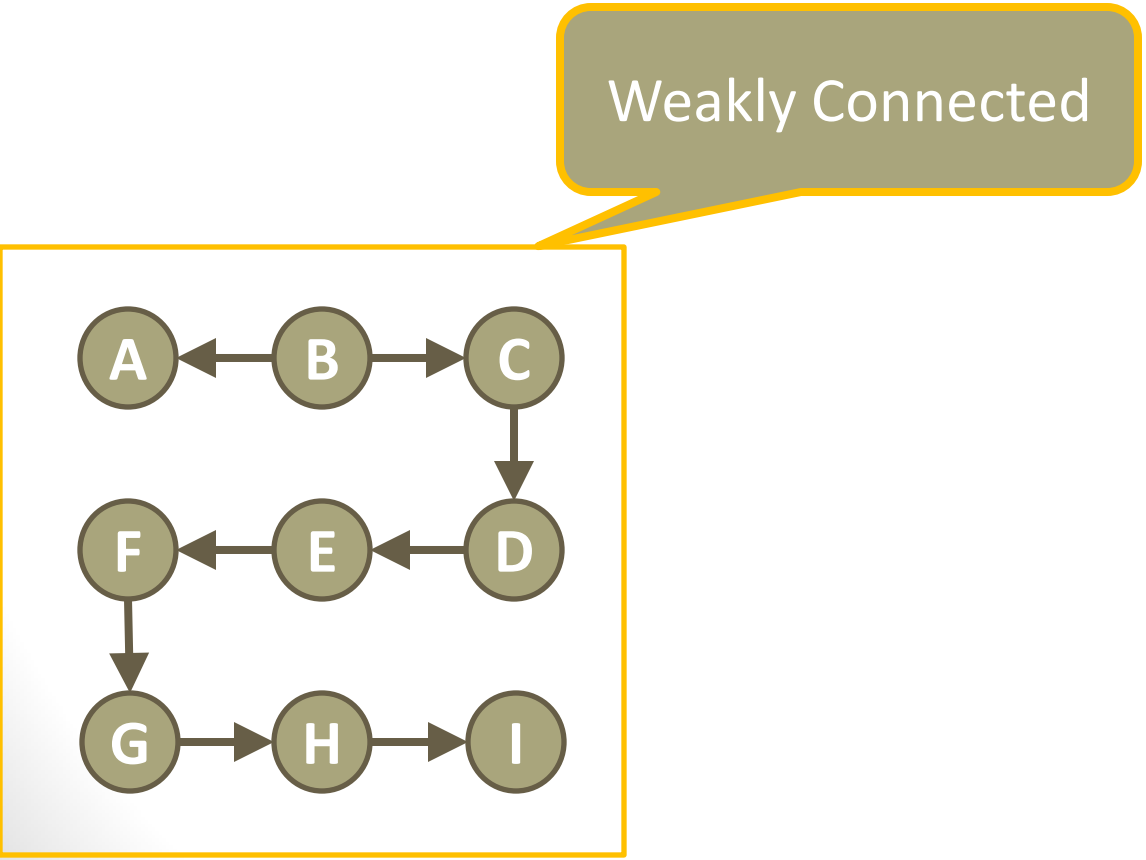




Graph Data: Connectedness and Density (4)

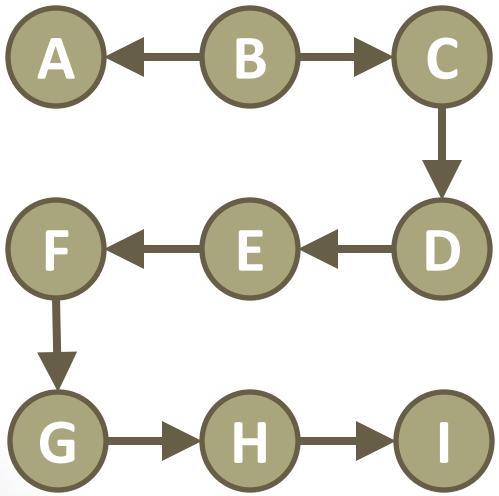


Graph Data: Connectedness and Density (5)



Graph Data: Connectedness and Density (6)

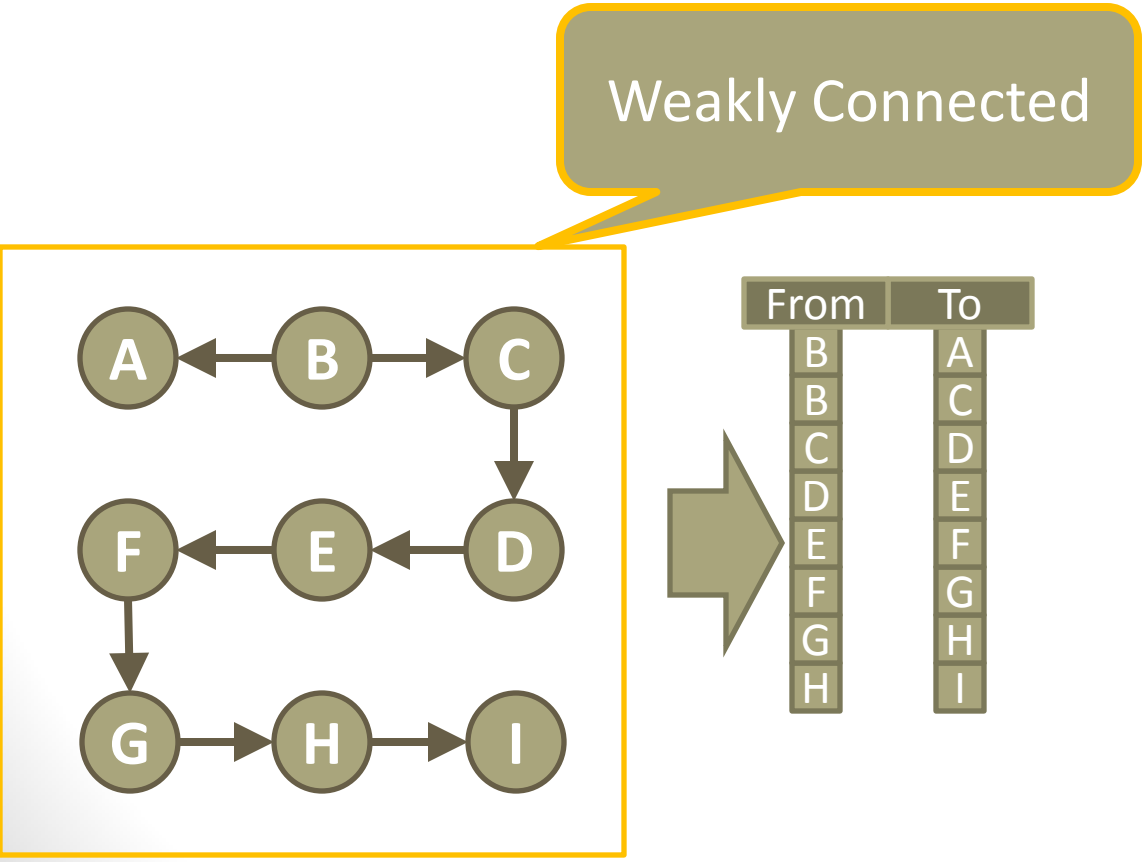
Weakly Connected



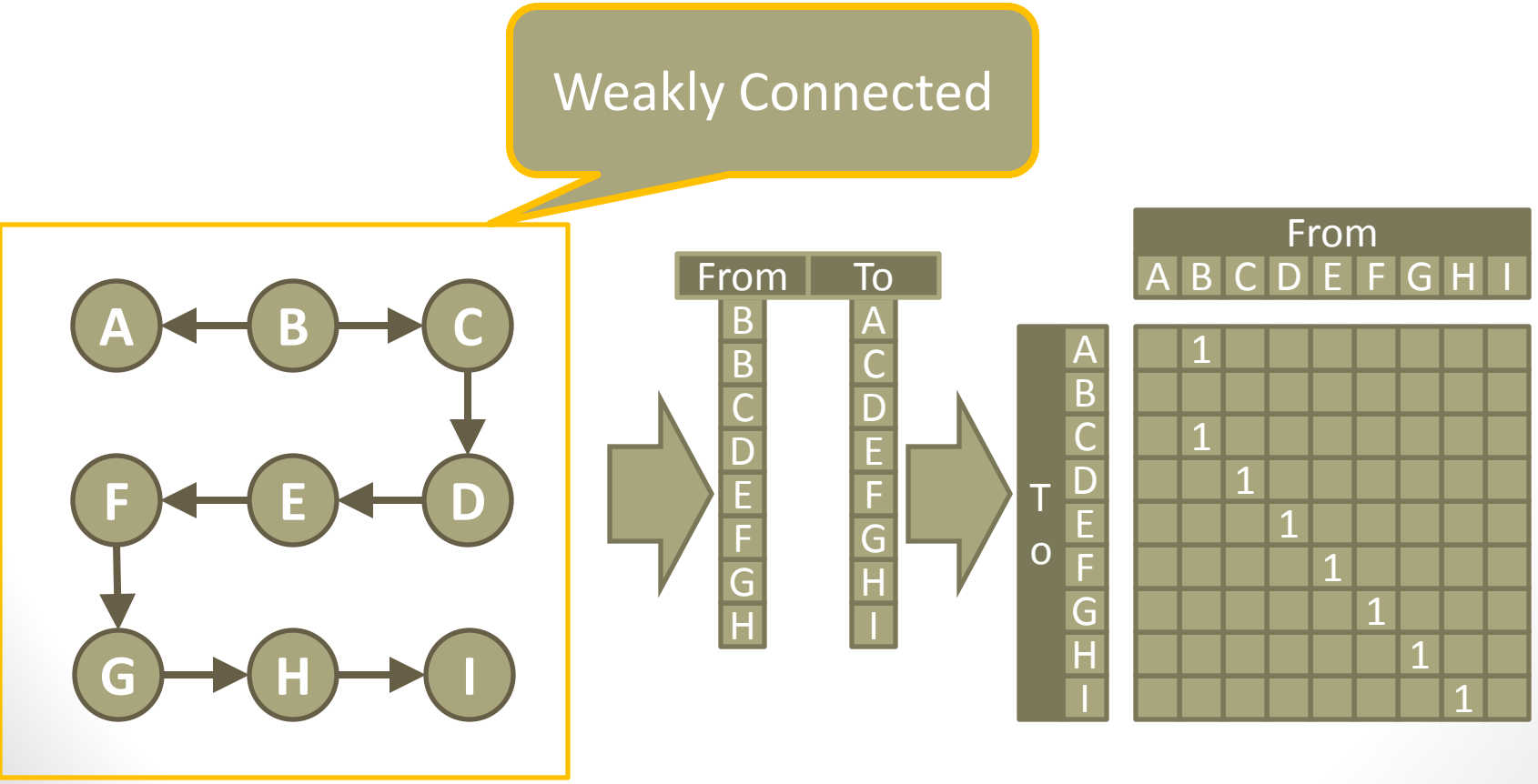
Weakly Connected:
For every pair of vertices, a path exists along the graph's edges regardless of the edge's direction. There exists at least one pair of vertices (u, v), where a path does not exist along the graph's directed edges from u to v an does not exist from v to u.

In this case: A path does not exist from A to C and does not exist from C to A.

Graph Data: Connectedness and Density (7)

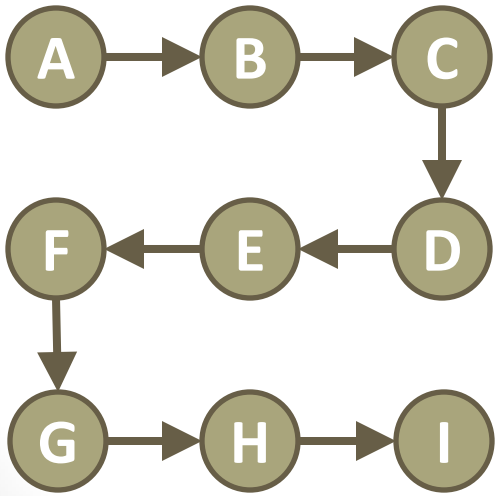


Graph Data: Connectedness and Density (8)



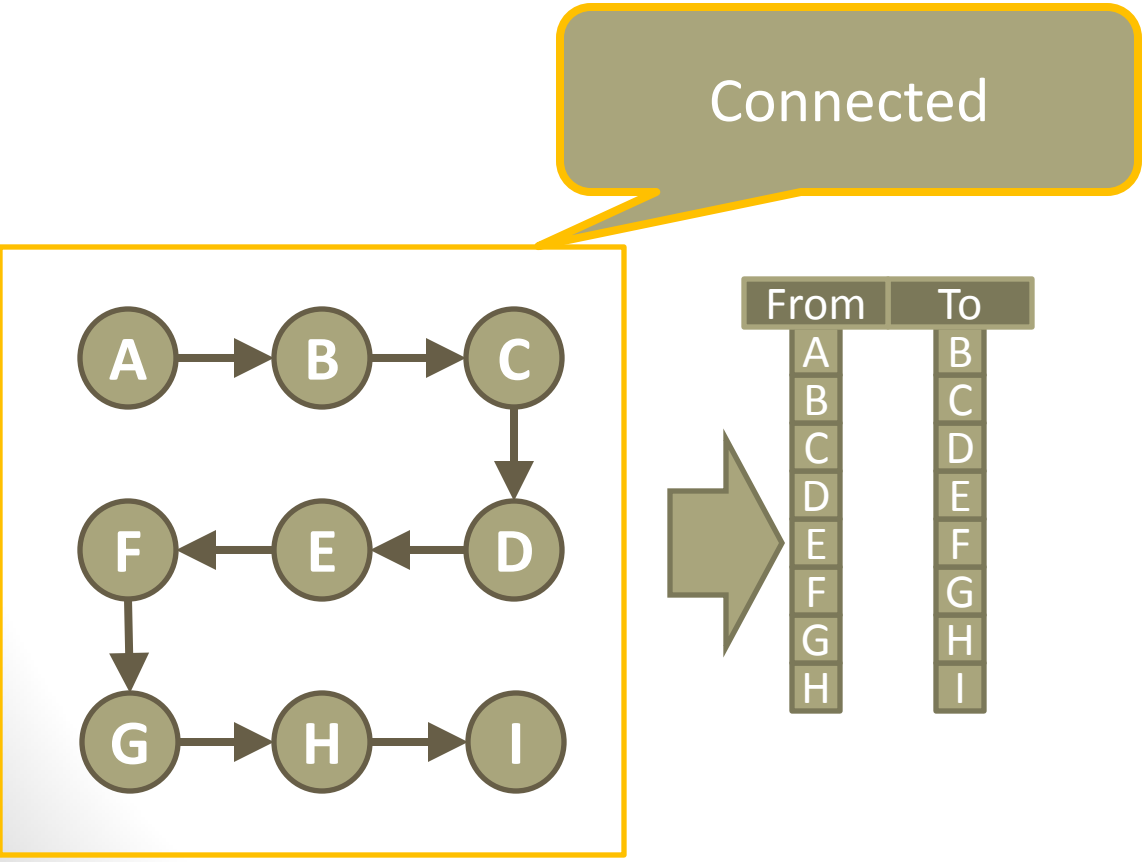
Graph Data: Connectedness and Density (9)

Connected

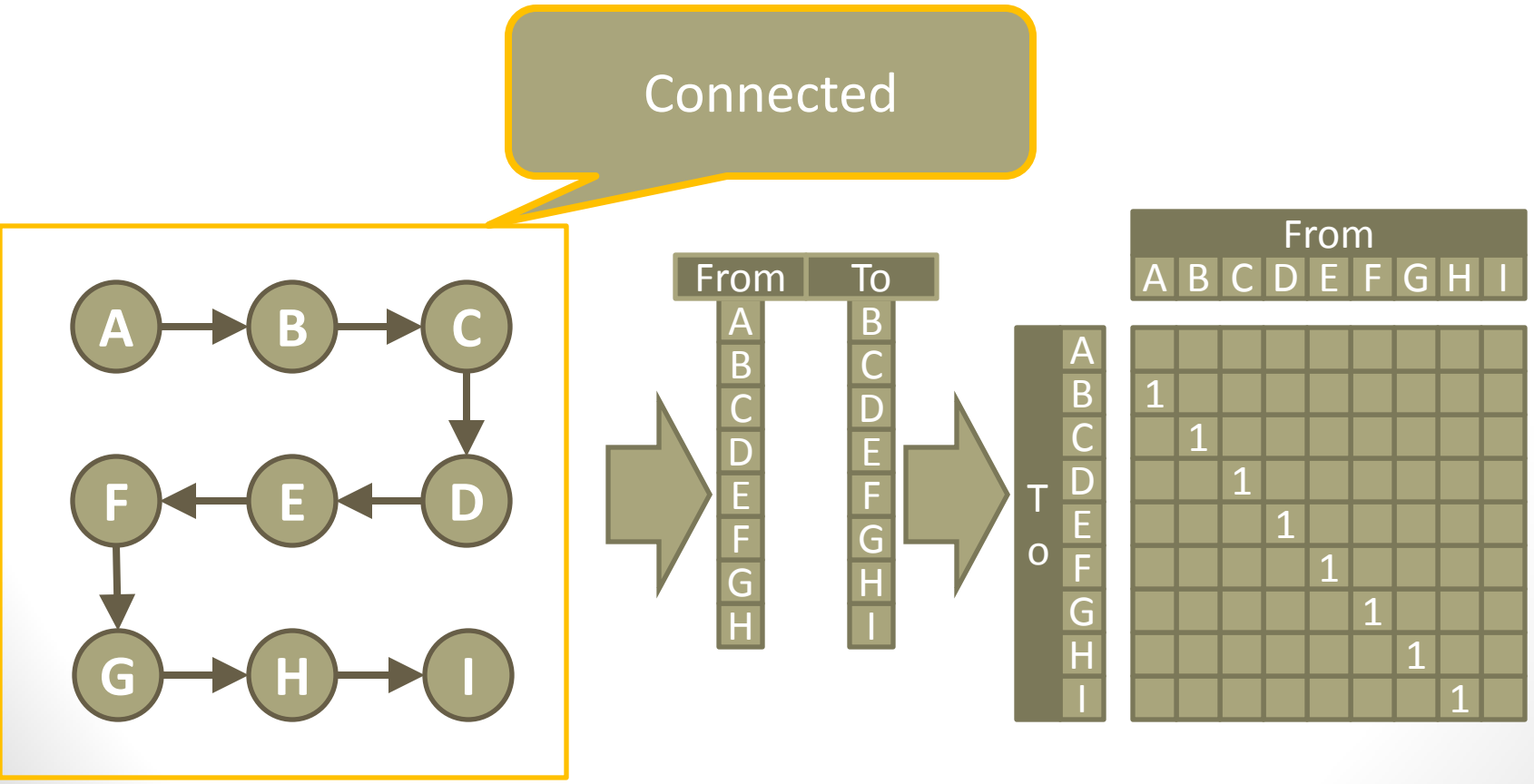


Connected:
For every pair of vertices (u, v), a path exists along the graph's directed edges from u to v or from v to u. There exists at least one pair of vertices (u, v), where a path does not exist along the graph's directed edges from either u to v or from v to u.

Graph Data: Connectedness and Density (10)

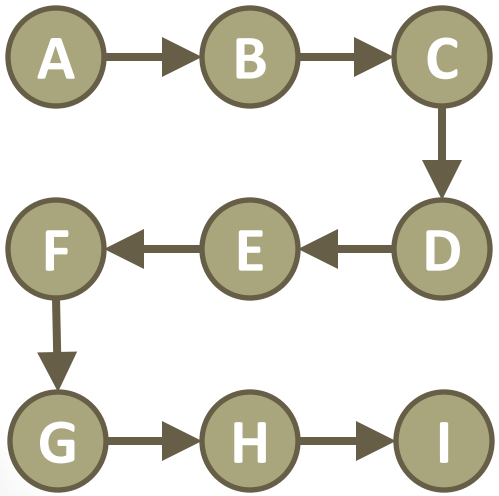


Graph Data: Connectedness and Density (11)



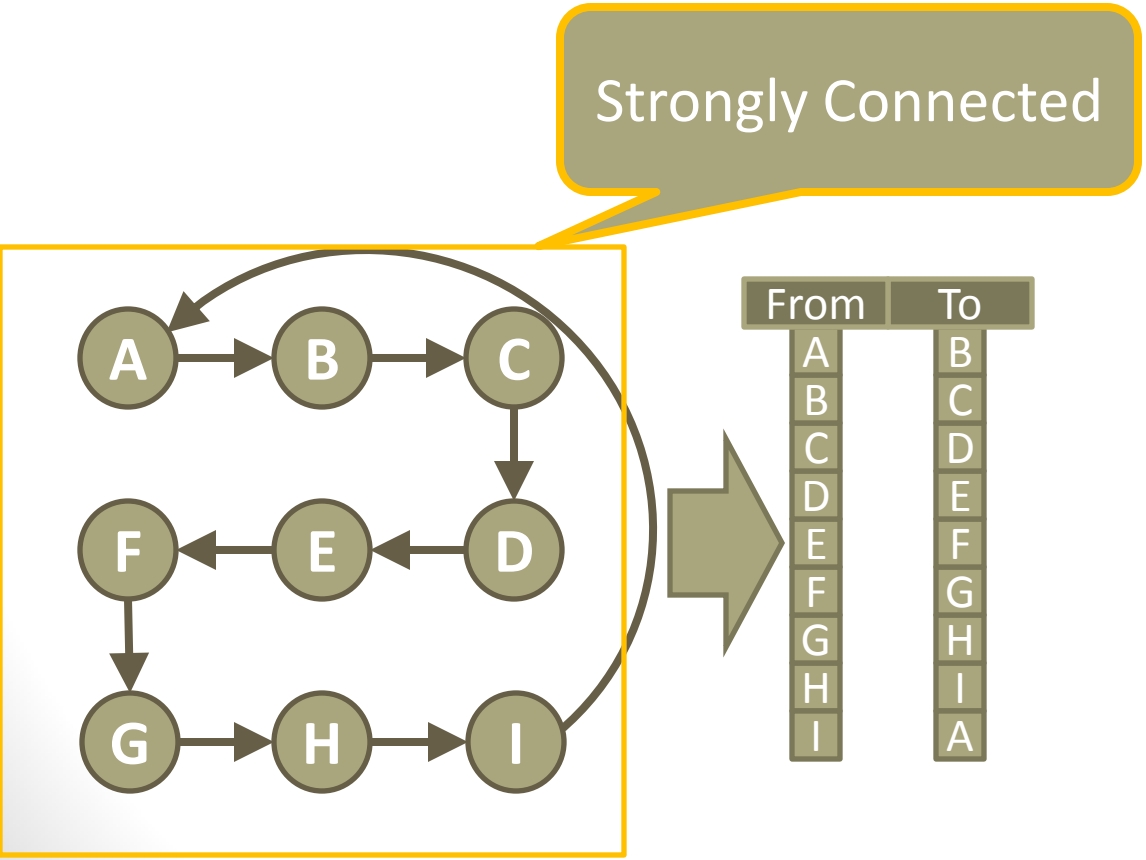
Graph Data: Connectedness and Density (12)

Connected

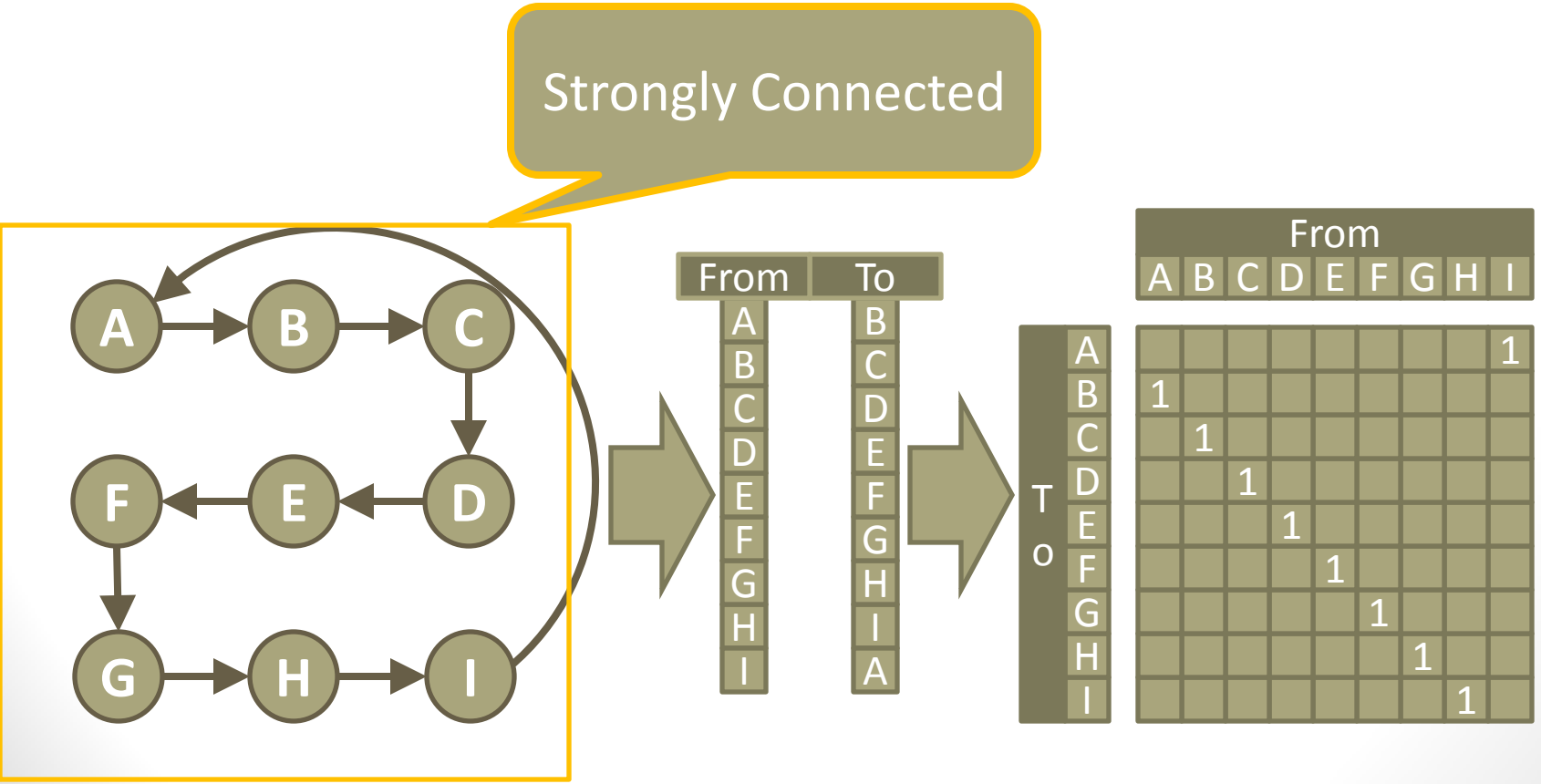


Connected:
A path exists along the graph's directed edges from every node to every node

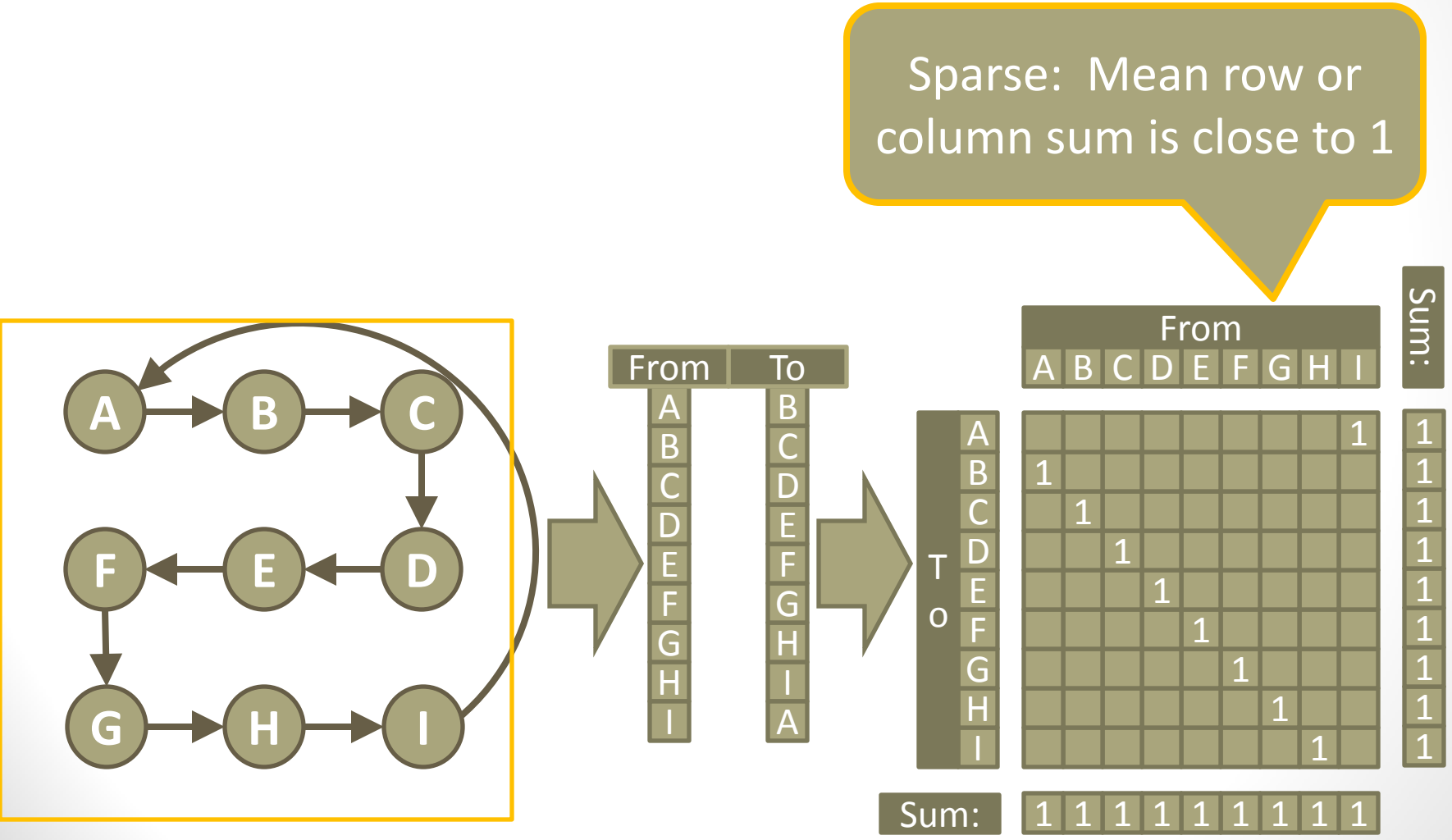
Graph Data: Connectedness and Density (13)



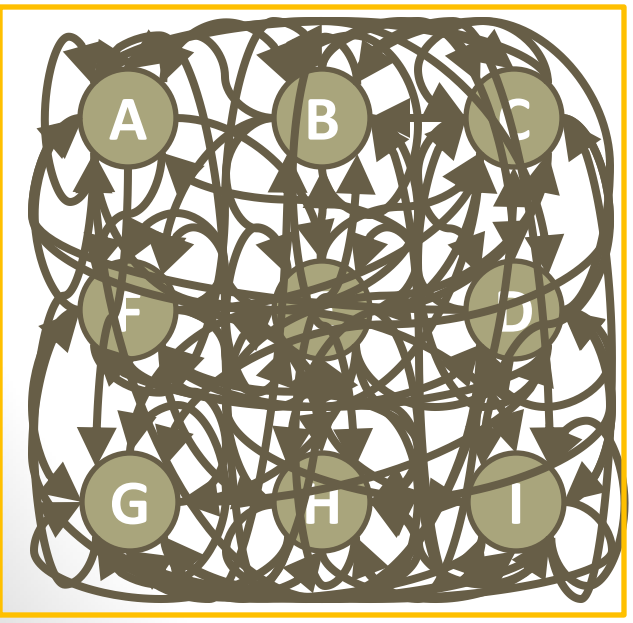
Graph Data: Connectedness and Density (14)



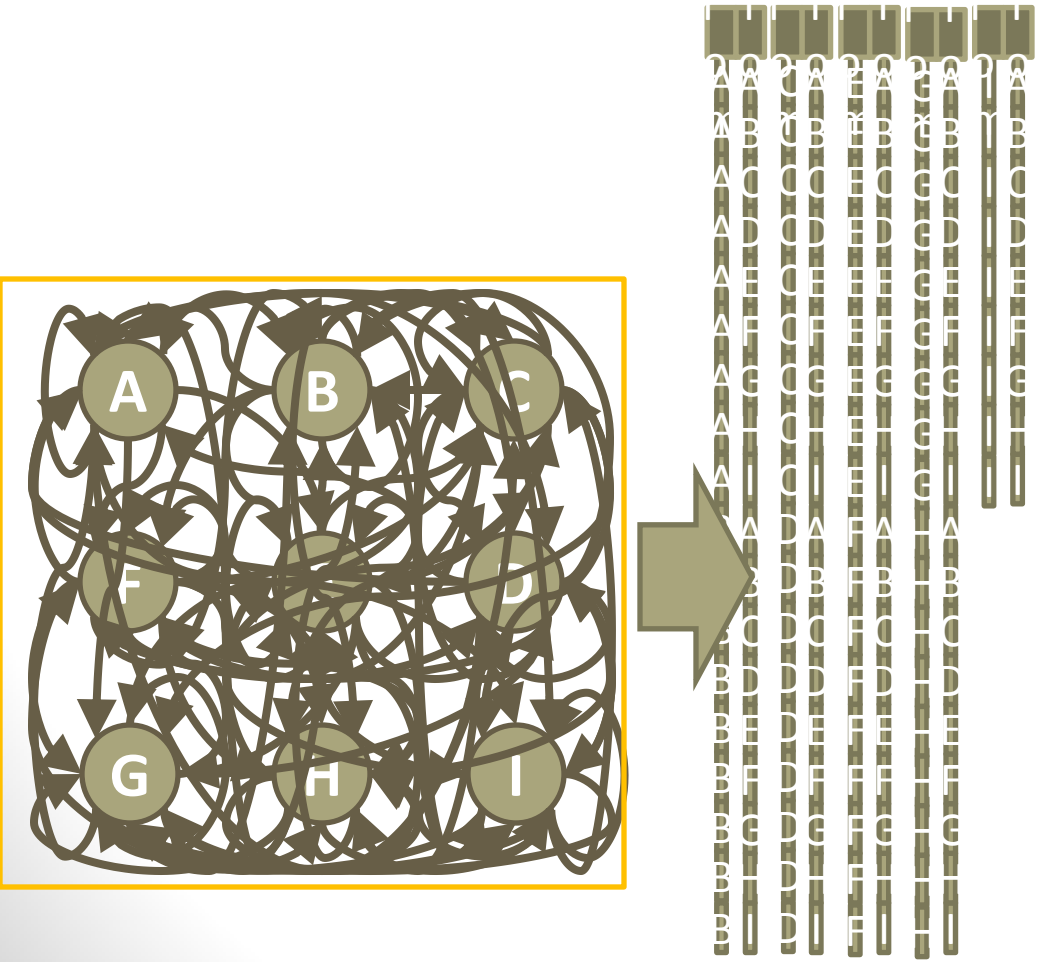
Graph Data: Connectedness and Density (15)



Graph Data: Connectedness and Density (16)



Graph Data: Connectedness and Density (17)



Graph Data: Connectedness and Density (18)

From	To
A	A
A	B
A	C
A	D
A	E
A	F
A	G
A	H
A	I
B	A
B	B
B	C
B	D
B	E
B	F
B	G
B	H
B	I

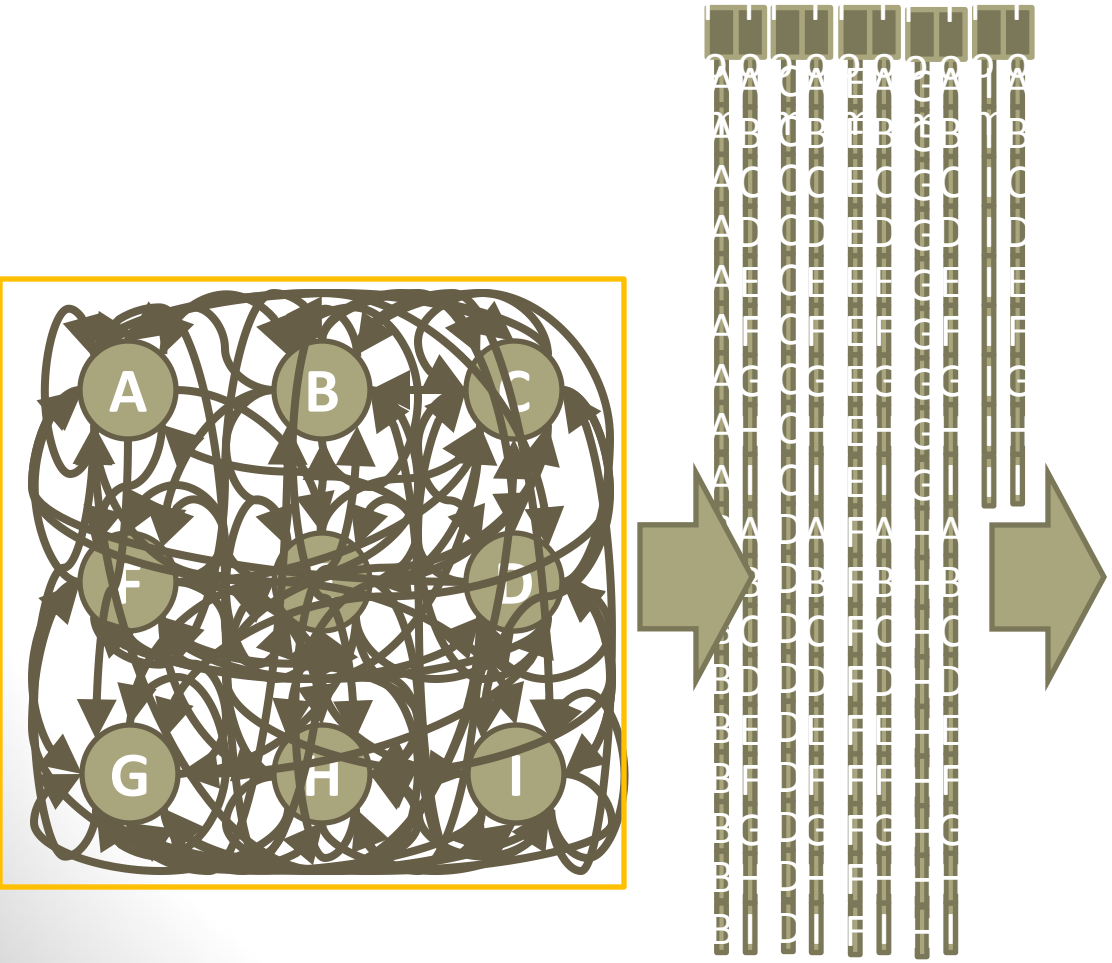
From	To
C	A
C	B
C	C
C	D
C	E
C	F
C	G
C	H
C	I
D	A
D	B
D	C
D	D
D	E
D	F
D	G
D	H
D	I

From	To
E	A
E	B
E	C
E	D
E	E
E	F
E	G
E	H
E	I
F	A
F	B
F	C
F	D
F	E
F	F
F	G
F	H
F	I

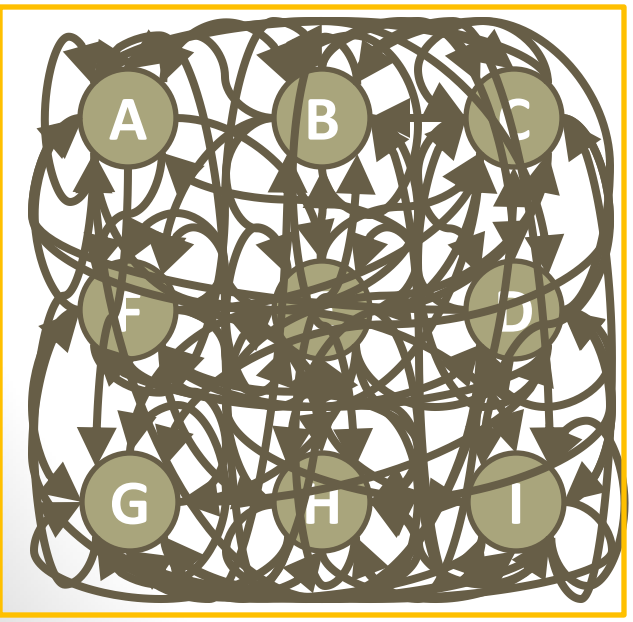
From	To
G	A
G	B
G	C
G	D
G	E
G	F
G	G
G	H
G	I
H	A
H	B
H	C
H	D
H	E
H	F
H	G
H	H
H	I

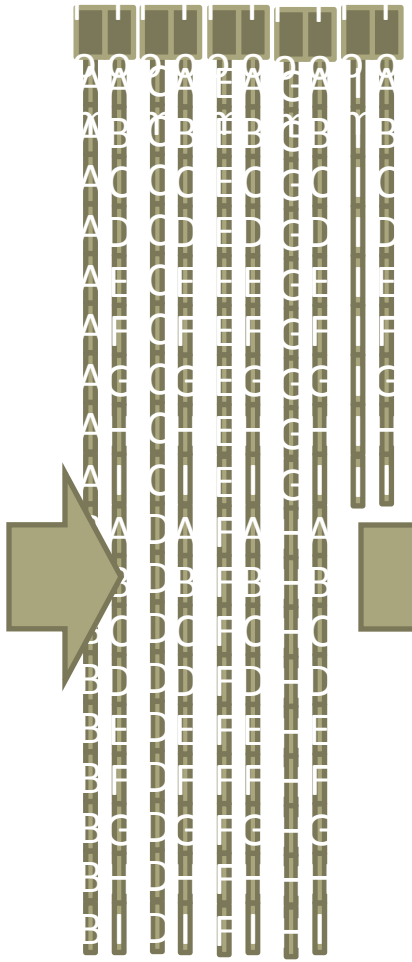
From	To
I	A
I	B
I	C
I	D
I	E
I	F
I	G
I	H
I	I

Graph Data: Connectedness and Density (19)



Graph Data: Connectedness and Density (20)

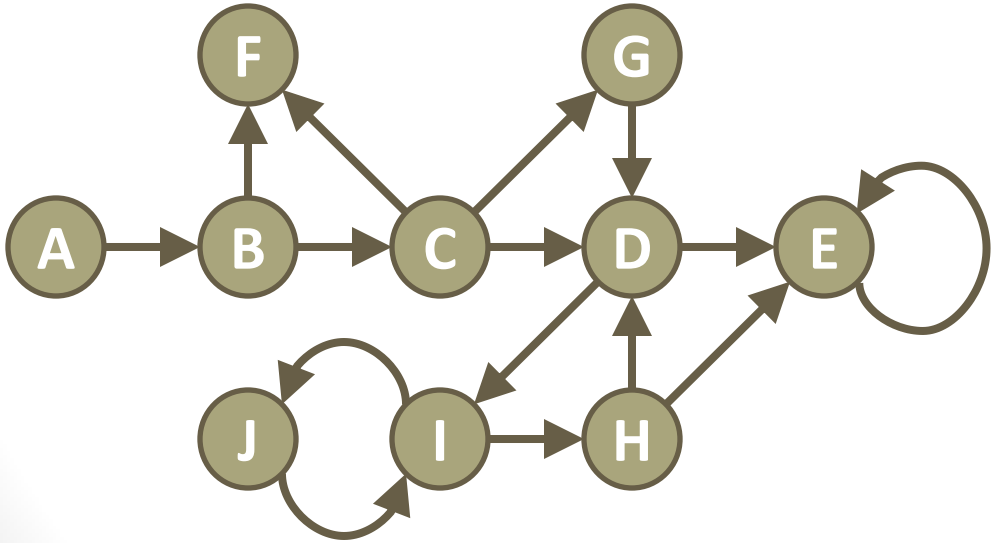




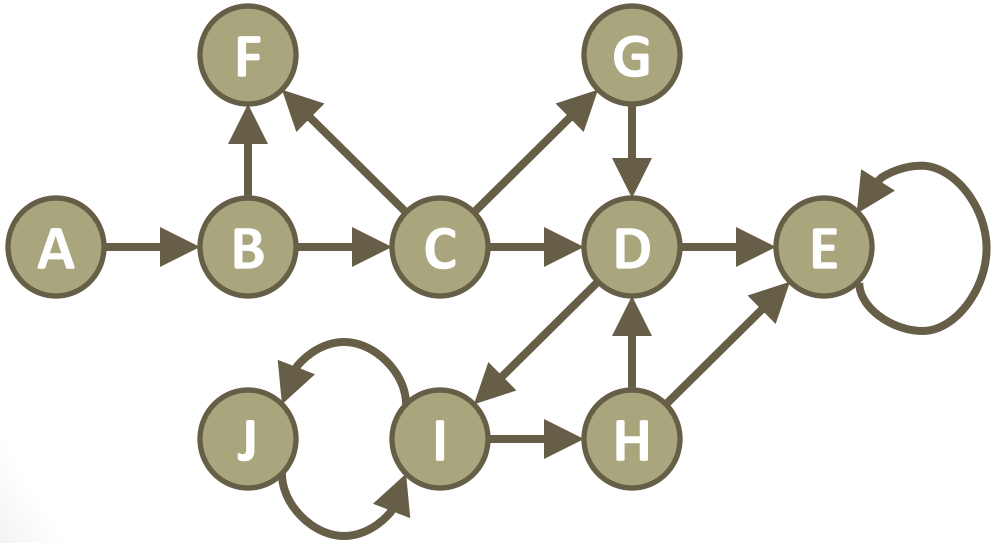
Dense: Mean row or column sum is close to number of nodes

To \ From	A	B	C	D	E	F	G	H	I	Sum:
A	1	1	1	1	1	1	1	1	1	9
B	1	1	1	1	1	1	1	1	1	9
C	1	1	1	1	1	1	1	1	1	9
D	1	1	1	1	1	1	1	1	1	9
E	1	1	1	1	1	1	1	1	1	9
F	1	1	1	1	1	1	1	1	1	9
G	1	1	1	1	1	1	1	1	1	9
H	1	1	1	1	1	1	1	1	1	9
I	1	1	1	1	1	1	1	1	1	9
Sum:	9	9	9	9	9	9	9	9	9	

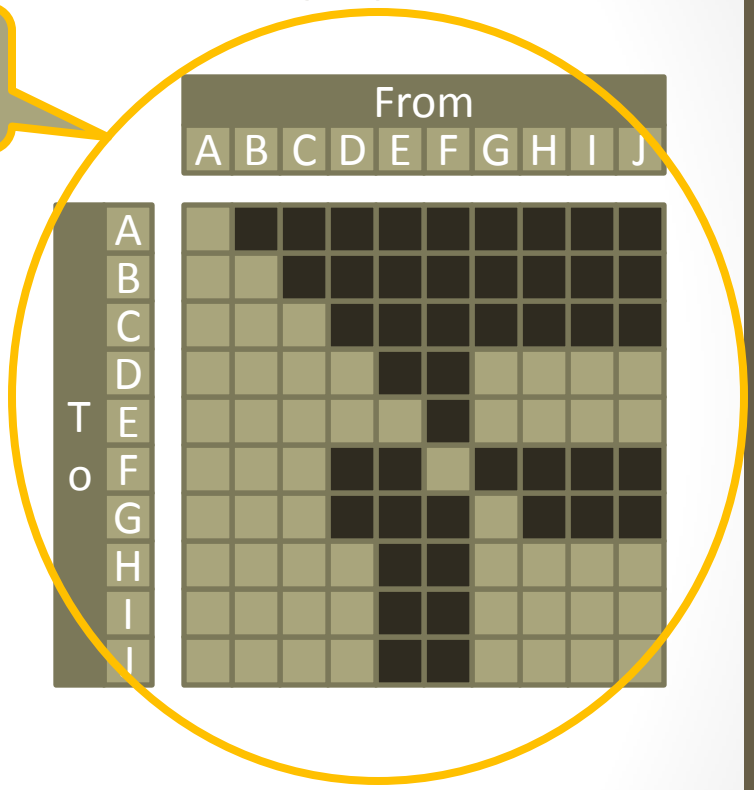
Graph Data: Allowed Paths, Distance, and Diameter (0)



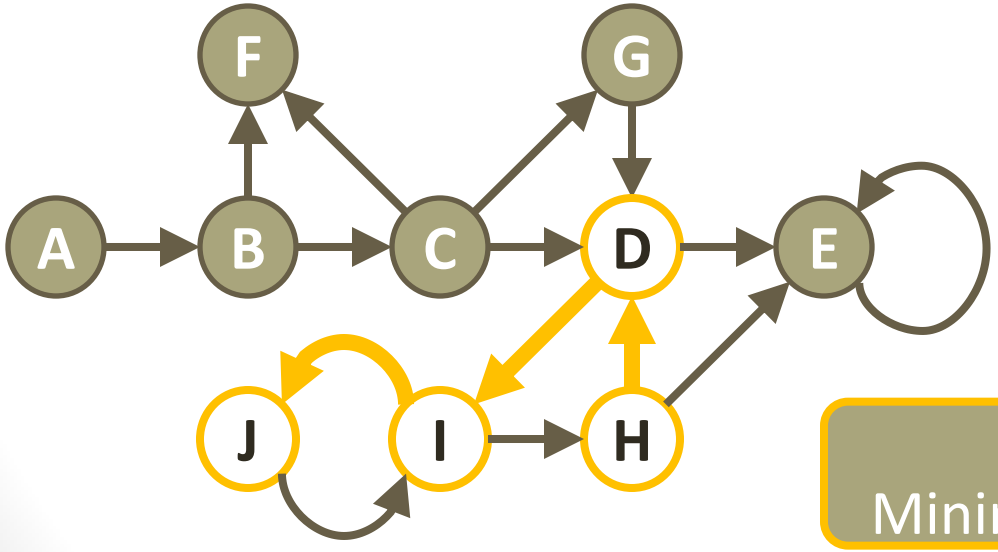
Graph Data: Allowed Paths, Distance, and Diameter (1)

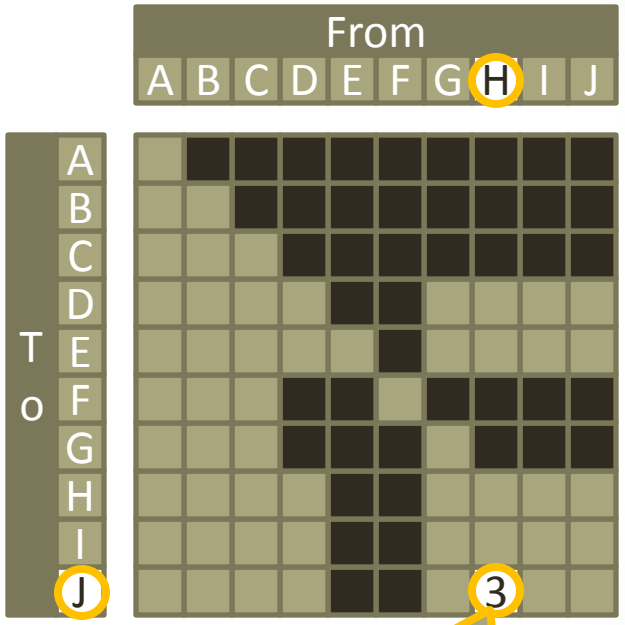


Allowed Paths



Graph Data: Allowed Paths, Distance, and Diameter (2)

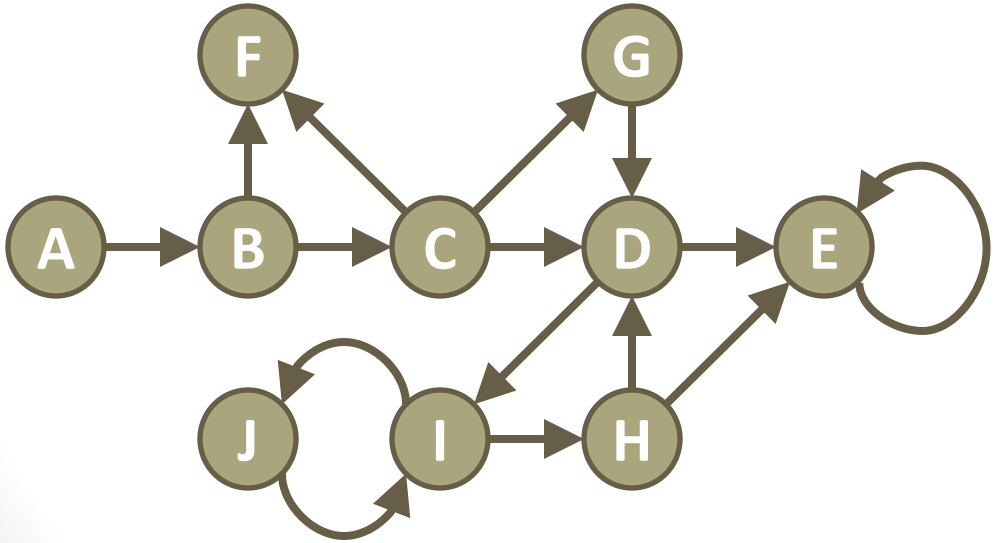




Path Distance:
Minimum Distance from H to J

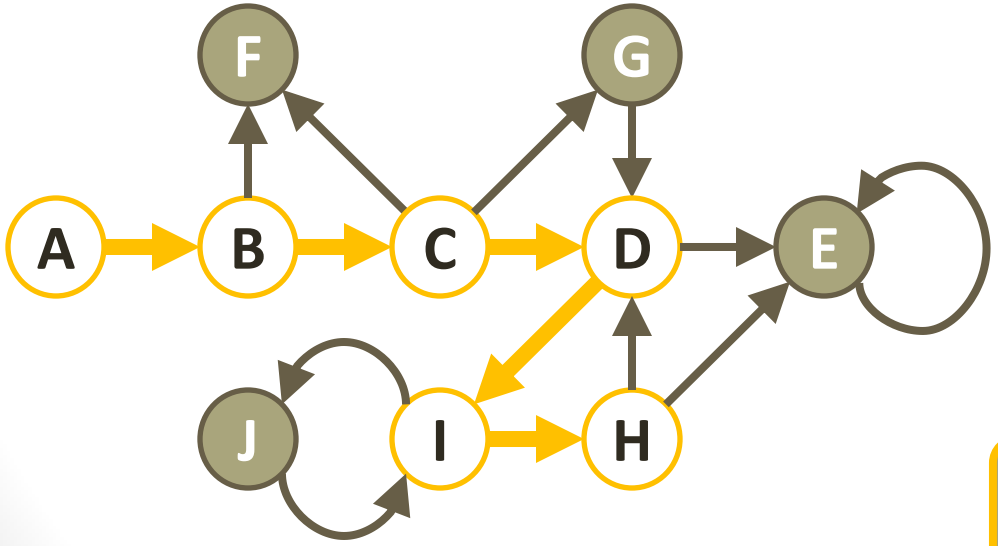
Graph Data: Allowed Paths, Distance, and Diameter (3)

Distance Matrix:
Minimum distance between any two vertices along a directed edge



To \ From	A	B	C	D	E	F	G	H	I	J
A	0									
B	1	0								
C	2	1	0							
D	3	2	1	0			1	1	2	3
E	4	3	2	1	0		2	1	2	3
F	2	1	1			0				
G	4	2	1				0			
H	5	4	3	2			3	0	1	2
I	4	3	2	1			2	2	0	1
J	5	4	3	2			3	3	1	0

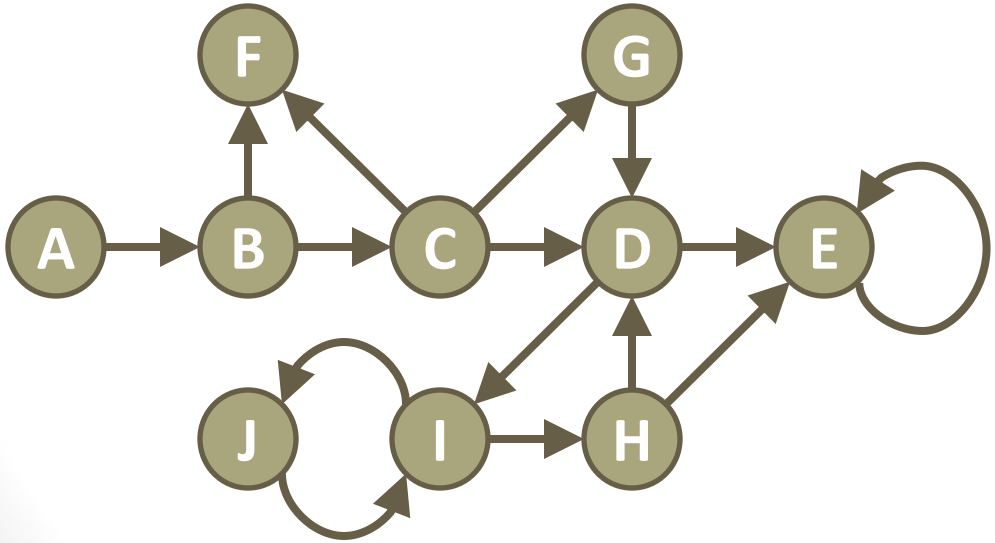
Graph Data: Allowed Paths, Distance, and Diameter (4)



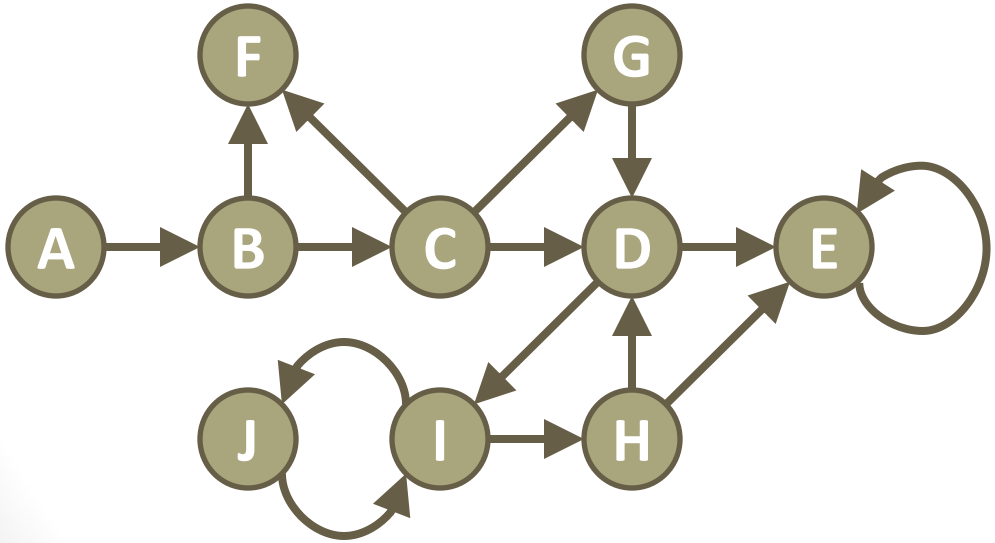
To \ From	A	B	C	D	E	F	G	H	I	J
A	0									
B	1	0								
C	2	1	0							
D	3	2	1	0			1	1	2	3
E	4	3	2	1	0		2	1	2	3
F	2	1	1			0				
G	4	2	1				0			
H	5	4	3	2			3	0	1	2
I	4	3	2	1			2	2	0	1
J	5	4	3	2			3	3	1	0

Diameter:
Max Path Distance

Graph Data: Order and Size (0)



Graph Data: Order and Size (1)

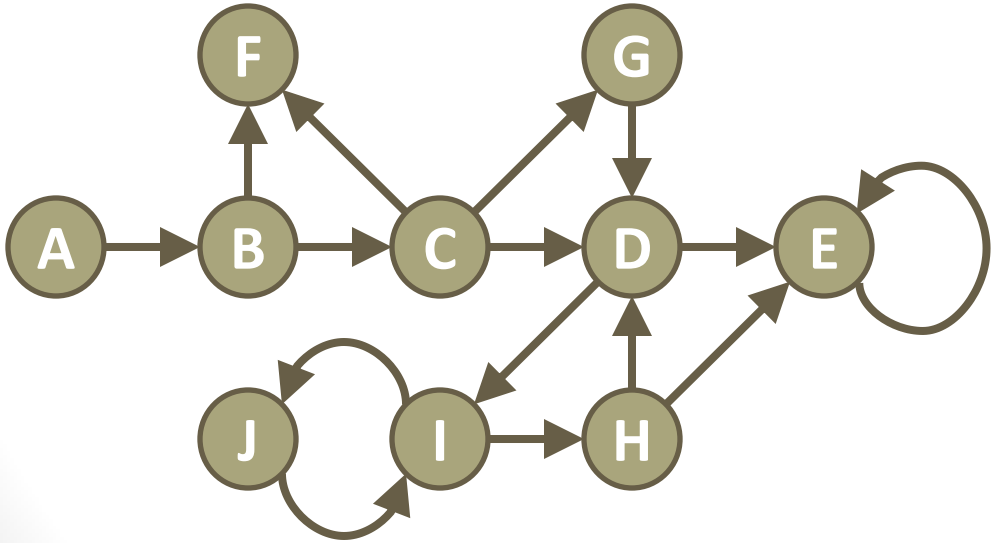


Matrix of Edges

To \ From	A	B	C	D	E	F	G	H	I	J
A										
B	1									
C		1								
D			1				1	1		
E				1	1			1		
F		1	1							
G			1							
H									1	
I				1						1
J									1	

Graph Data: Order and Size (2)

Order of Graph: 10

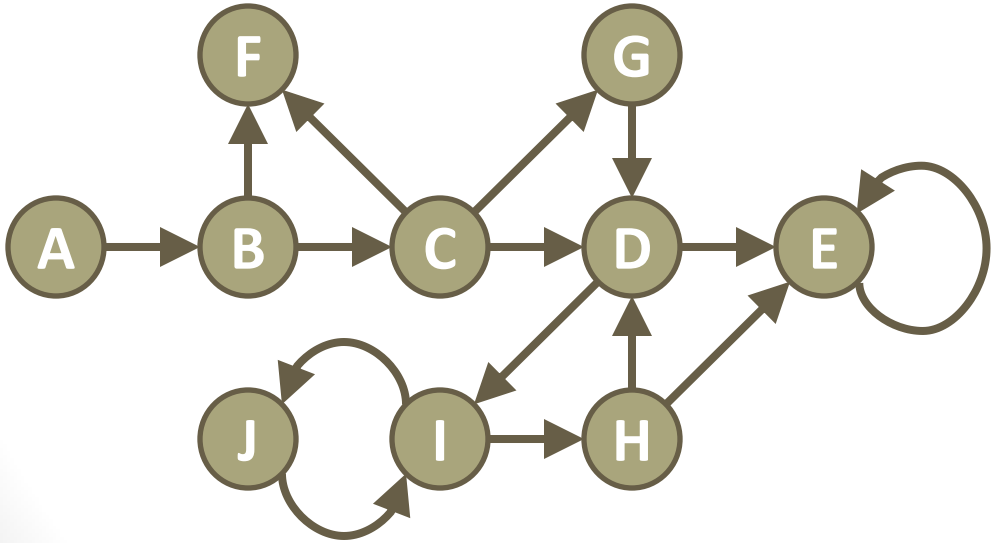


To \ From	A	B	C	D	E	F	G	H	I	J
A										
B	1									
C		1								
D			1				1	1		
E				1	1			1		
F		1	1							
G			1							
H									1	
I				1						1
J									1	

Order of Graph: Number of Nodes

Graph Data: Order and Size (3)

Order of Graph: 10
Size of Graph: 15



To \ From	A	B	C	D	E	F	G	H	I	J
A										
B	1									
C		1								
D			1				1	1		
E				1	1			1		
F		1	1							
G			1							
H									1	
I				1						1
J									1	

Size of Graph:
Number of Edges

Graph Data: Links (0)

- Graph Analytics:
  - http://en.wikipedia.org/wiki/Graph_theory
- Graph Diameter
  - http://people.hofstra.edu/geotrans/eng/methods/diameter1.html
- Distance
  - http://en.wikipedia.org/wiki/Distance_(graph_theory)
- Resistance
  - http://en.wikipedia.org/wiki/Resistance_distance
- Centrality
  - http://en.wikipedia.org/wiki/Betweenness#Betweenness_centrality

Graph Data: Links (1)

- http://en.wikipedia.org/wiki/Graph_(mathematics)
- http://en.wikipedia.org/wiki/Connectivity_(graph_theory)

Introduction to Graph Data

Quiz 10a

- Graph Theory

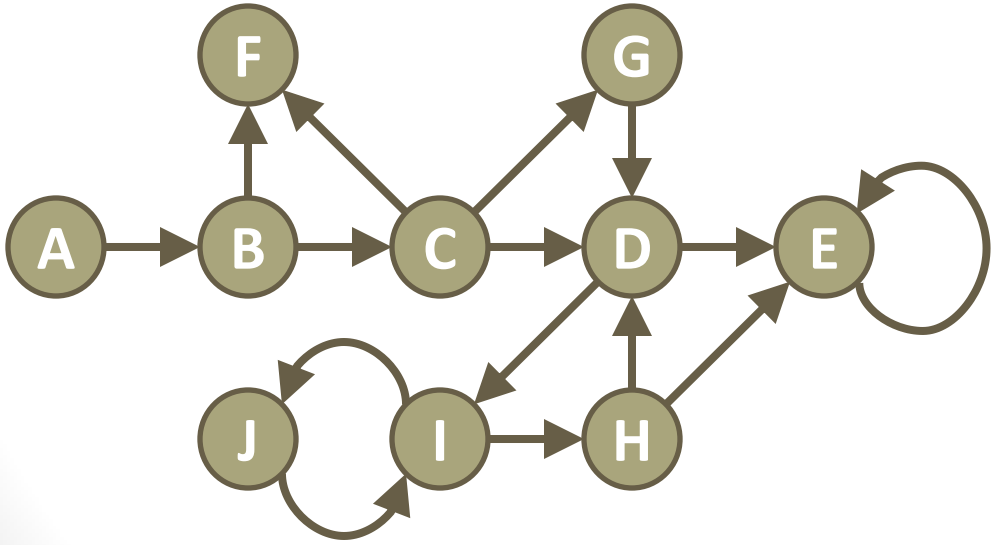
- You need to view the projected slide to answer questions 1 and 2
- https://catalyst.uw.edu/webq/survey/ernsthe/273783

Graph Data: Google Matrix

Google Matrix

- http://en.wikipedia.org/wiki/Google_matrix

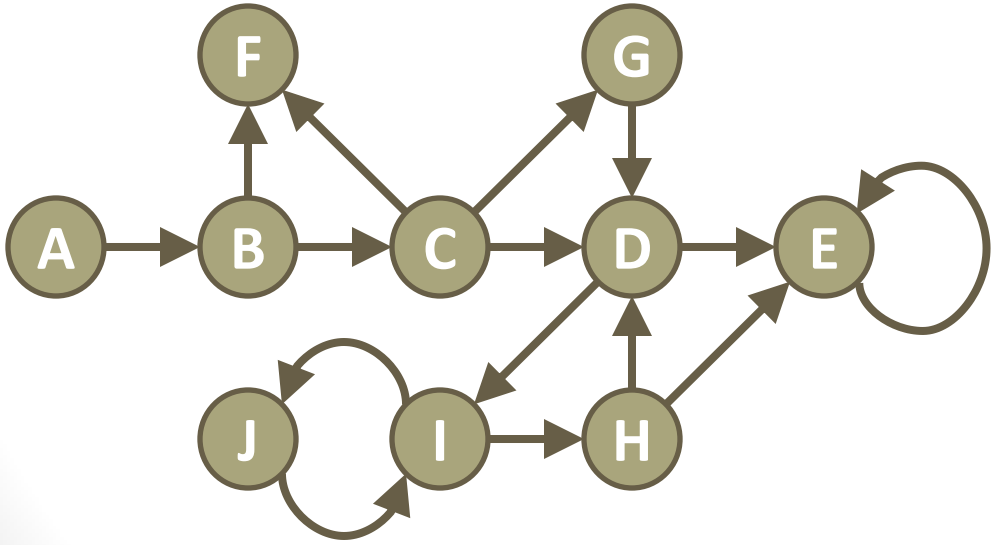
Graph Data: Popularity (0)



To \ From	A	B	C	D	E	F	G	H	I	J
A										
B	1									
C		1								
D			1				1	1		
E				1	1			1		
F		1	1							
G			1							
H									1	
I				1						1
J									1	

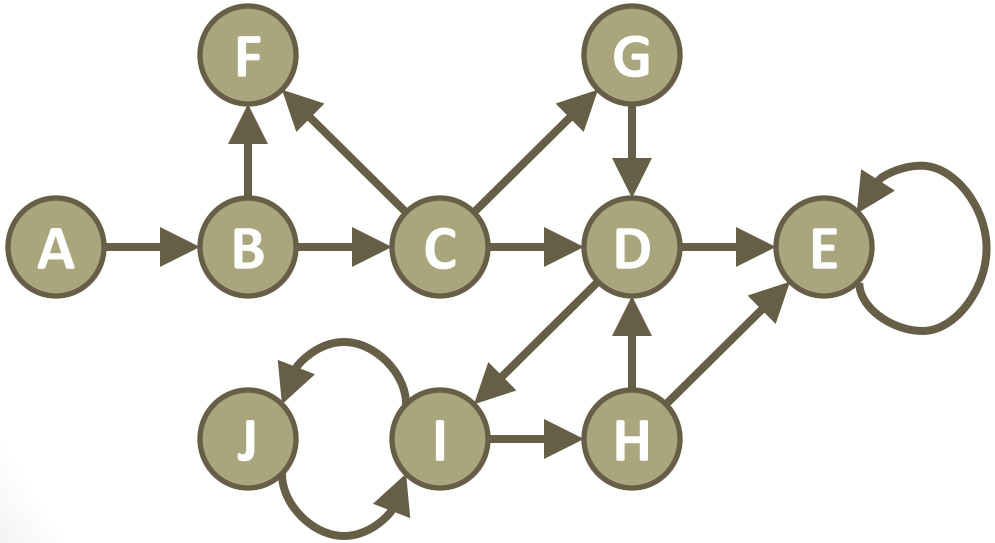
Graph Data: Popularity (1)

In-degree of a node: Popularity



To \ From	A	B	C	D	E	F	G	H	I	J	Sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1

Graph Data: Popularity (2)



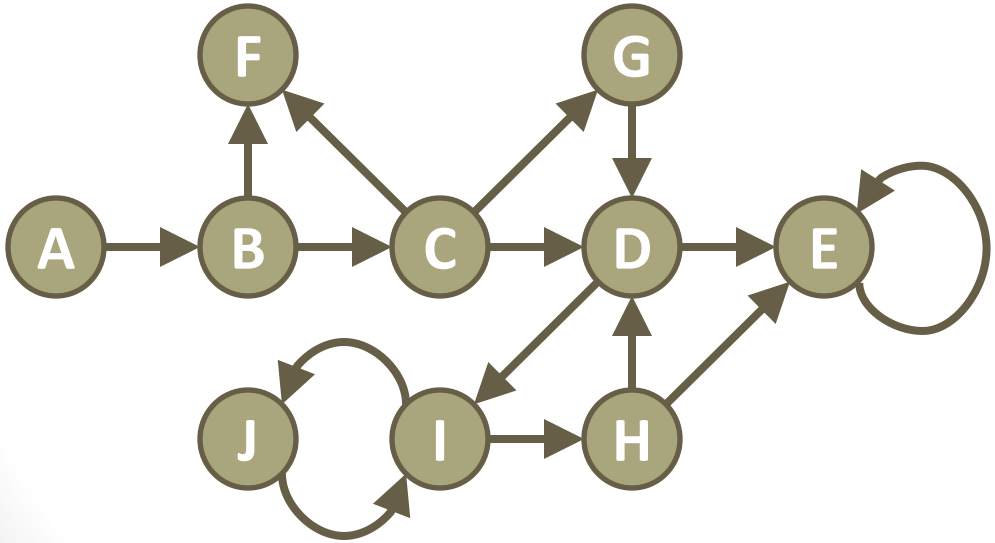
In-degree of a node: Popularity

To \ From	A	B	C	D	E	F	G	H	I	J	Sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1
Sum:	1	2	3	2	1	0	1	2	2	1	

Out-degree of a node: Gregariousness

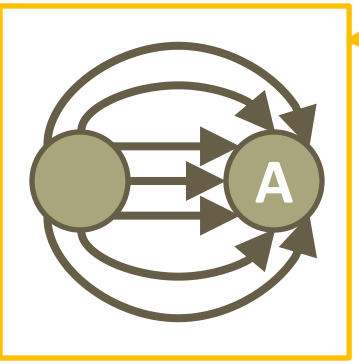
Graph Data: Popularity (3)

But, not every link is equally important. Otherwise, you could create a website with a thousand links that all point to your site.

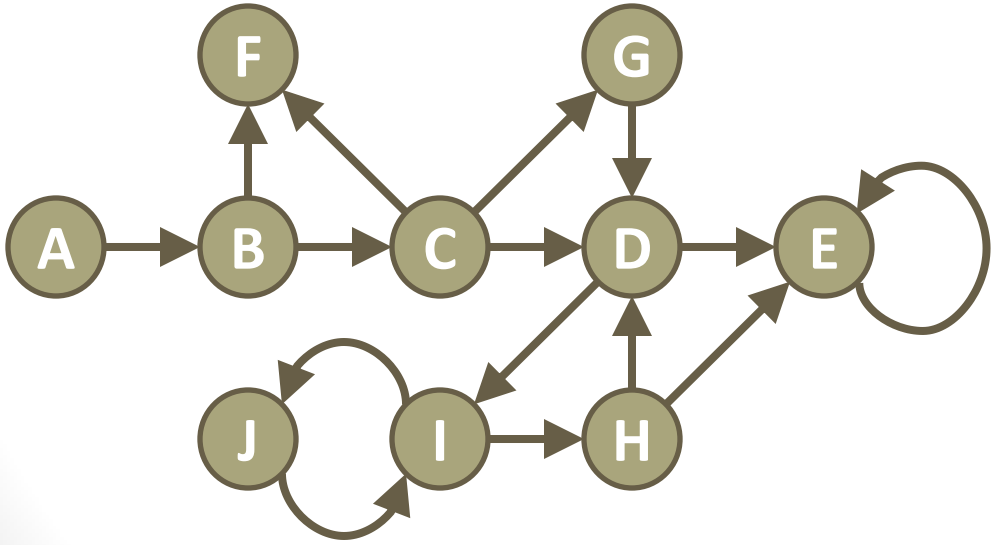


To \ From	A	B	C	D	E	F	G	H	I	J	Sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1
Sum:	1	2	3	2	1	0	1	2	2	1	

Graph Data: Popularity (4)

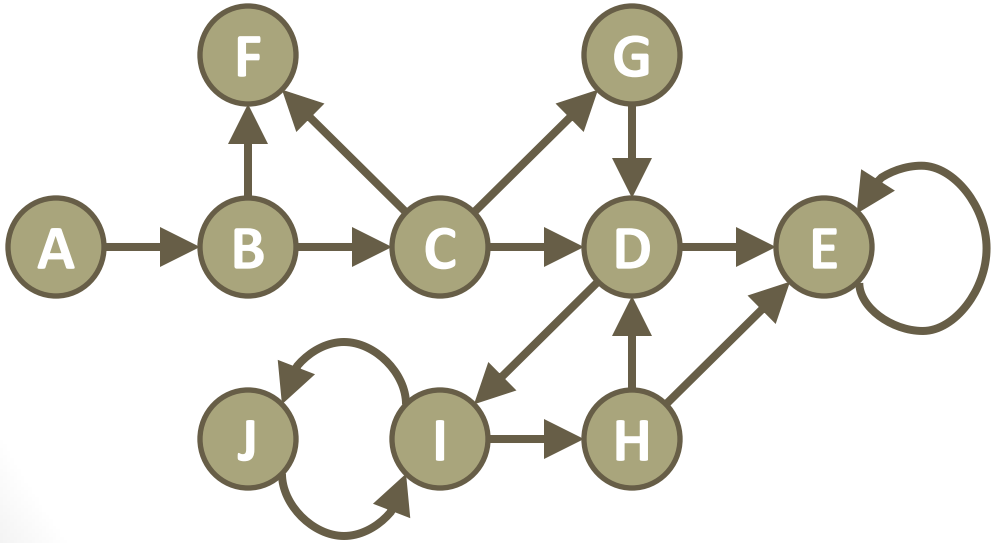


One very gregarious node could make another node very popular



To \ From	A	B	C	D	E	F	G	H	I	J	Sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1
Sum:	1	2	3	2	1	0	1	2	2	1	

Graph Data: Popularity (5)

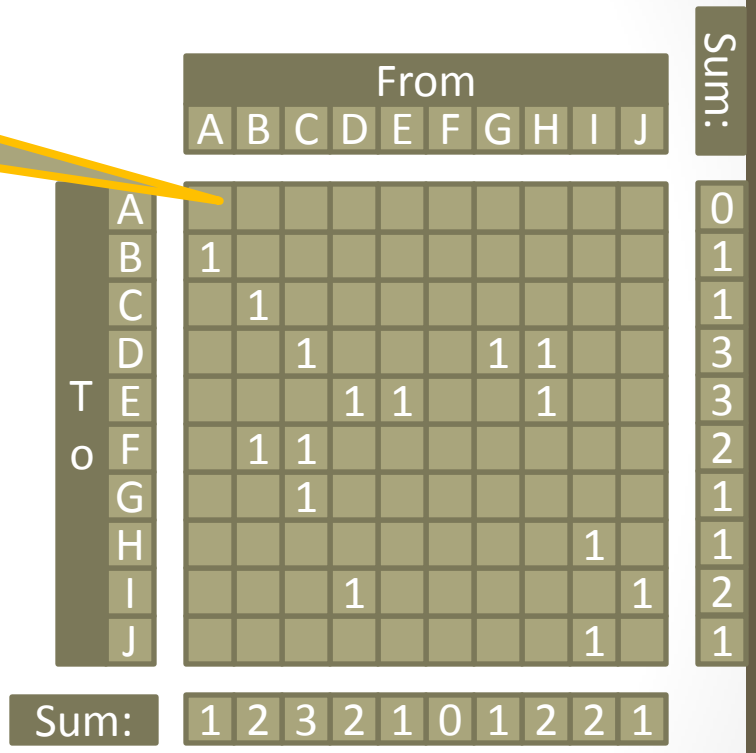


To \ From	A	B	C	D	E	F	G	H	I	J	Sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1
Sum:	1	2	3	2	1	0	1	2	2	1	

The amount of popularity that a single node can send out needs to be tempered by Gregariousness

Graph Data: Popularity (6)

This matrix needs some adjustments



Graph Data: Popularity (7)

To \ From	A	B	C	D	E	F	G	H	I	J	sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1
sum:	1	2	3	2	1	0	1	2	2	1	

Graph Data: Popularity (8)

To \ From	A	B	C	D	E	F	G	H	I	J	sum:
A											0
B	1										1.00
C		1/2									0.50
D			1/3				1	1/2			1.83
E				1/2	1			1/2			2.00
F		1/2	1/3								0.83
G			1/3								0.33
H									1/2		0.50
I				1/2						1	1.50
J									1/2		0.50
sum:	1	1	1	1	1	0	1	1	1	1	

Graph Data: Popularity (9)

To \ From	A	B	C	D	E	F	G	H	I	J	sum:	Popularity:
A											0	0
B	1										1.00	0.111
C		1/2									0.50	0.056
D			1/3				1	1/2			1.83	0.204
E				1/2	1			1/2			2.00	0.222
F		1/2	1/3								0.83	0.092
G			1/3								0.33	0.037
H									1/2		0.50	0.056
I				1/2						1	1.50	0.167
J									1/2		0.50	0.056
sum:	1	1	1	1	1	0	1	1	1	1		

Graph Data: Popularity (10)

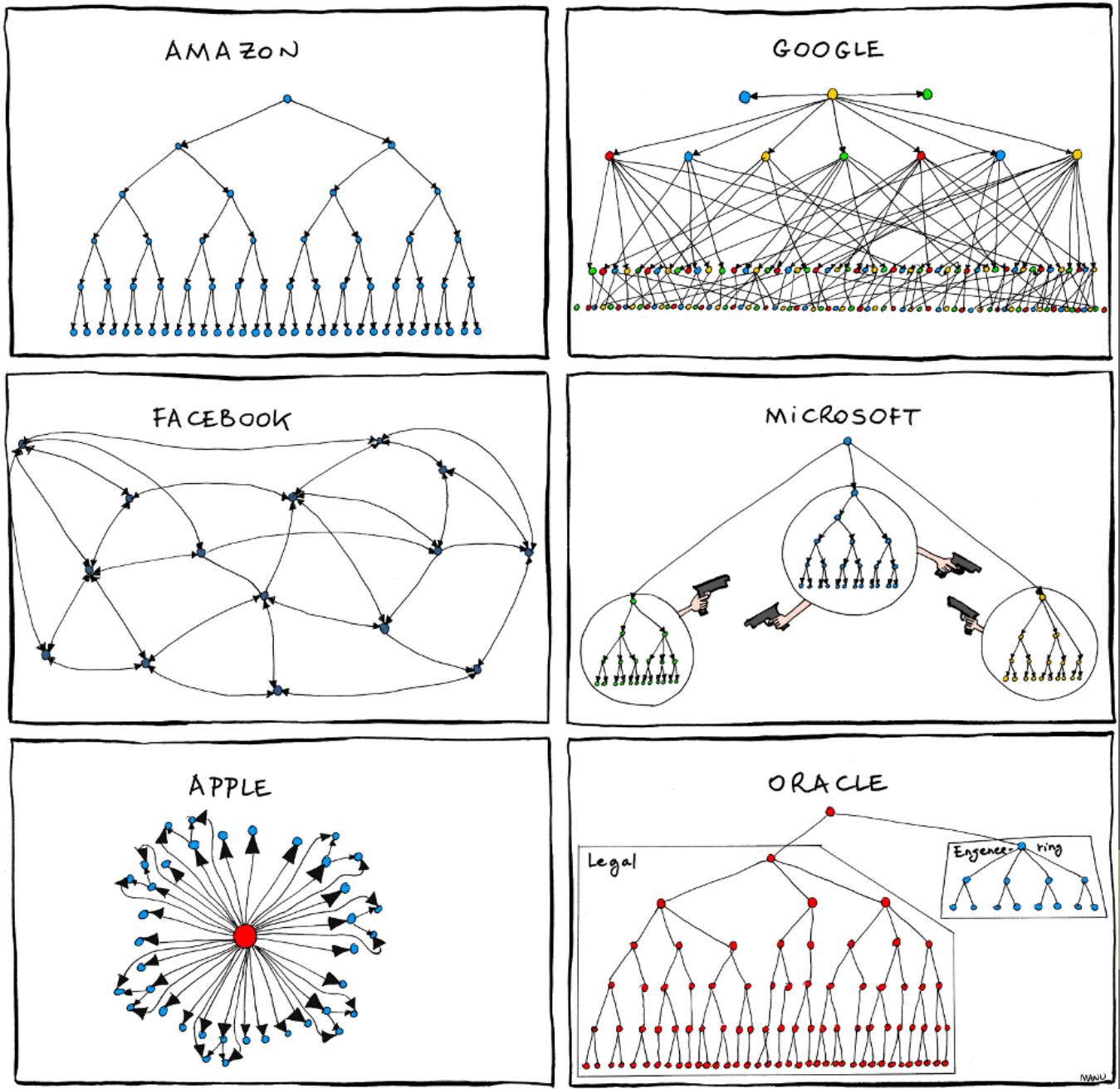
- Rules of a Popularity Network
  - Rule 0: Popularity flows with the arrow.
  - Rule 1: Popularity transmissions from multiple nodes to a single node are additive.
  - Rule 2: A more popular node transmits more popularity.
  - Rule 3: A node splits up its popularity among its recipients.

- Continue with EigenVectorOfGraphMatrix.R

- Solve (or Iterate until convergence)
  - Modified Popularity is the Popularity divided by Gregariousness (Out-Degree)
  - Popularity of a website is the sum of the Modified Popularities that point to the website.

Graph Data: Google Matrix

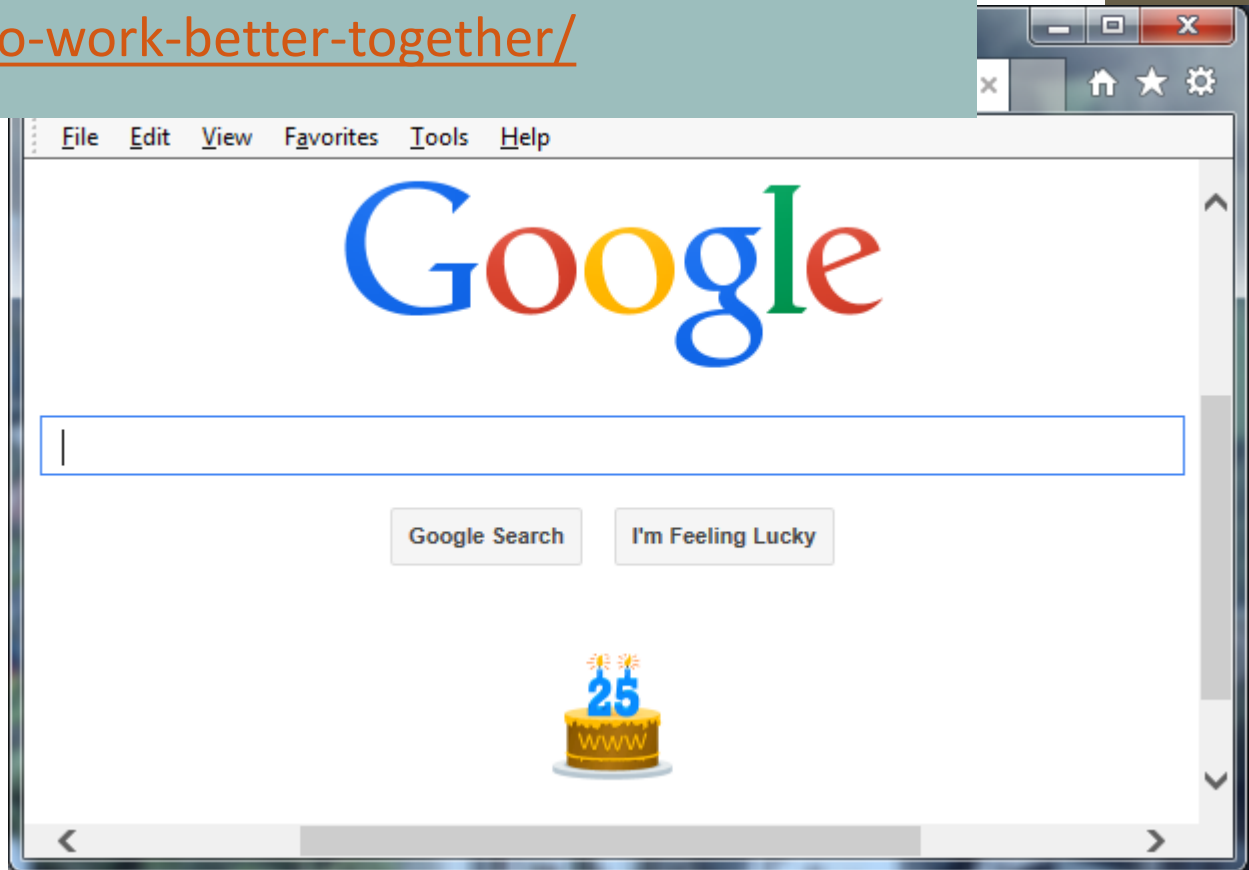
Break


AMAZON
GOOGLE
FACEBOOK
MICROSOFT
APPLE
ORACLE
Legal
Engineering
MANU

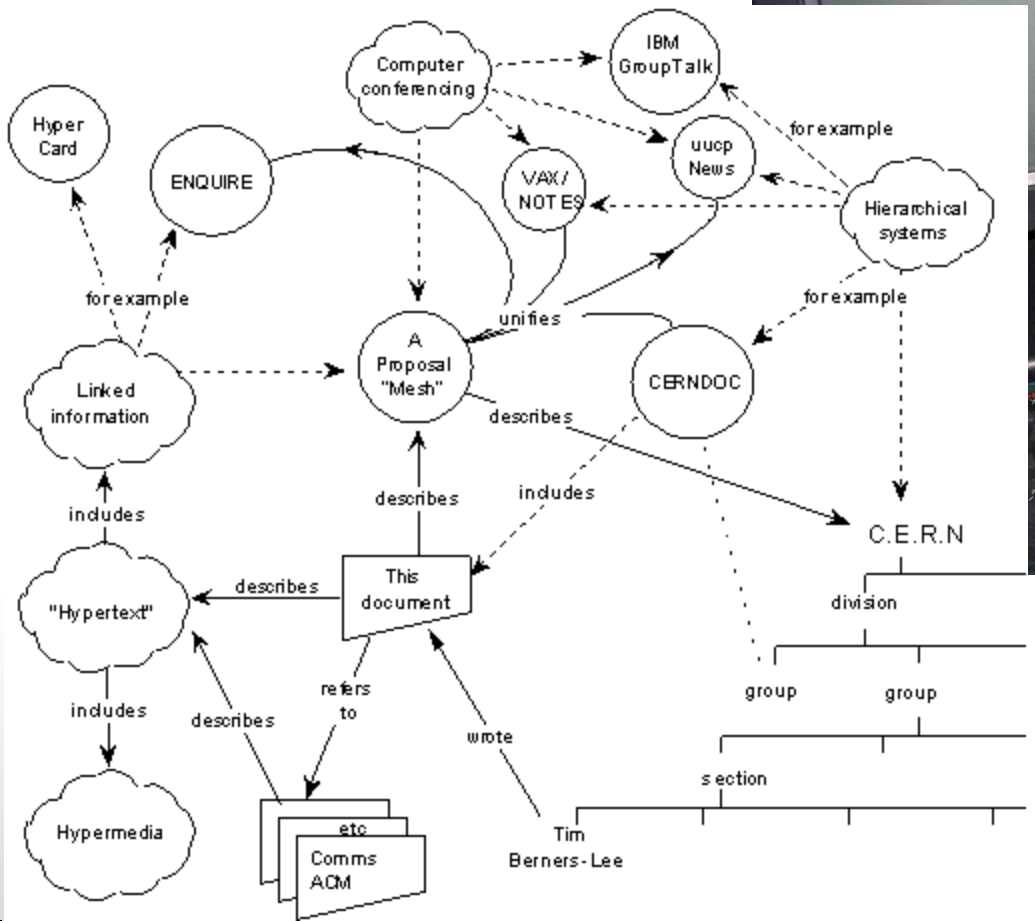
Introduction to SPARQL

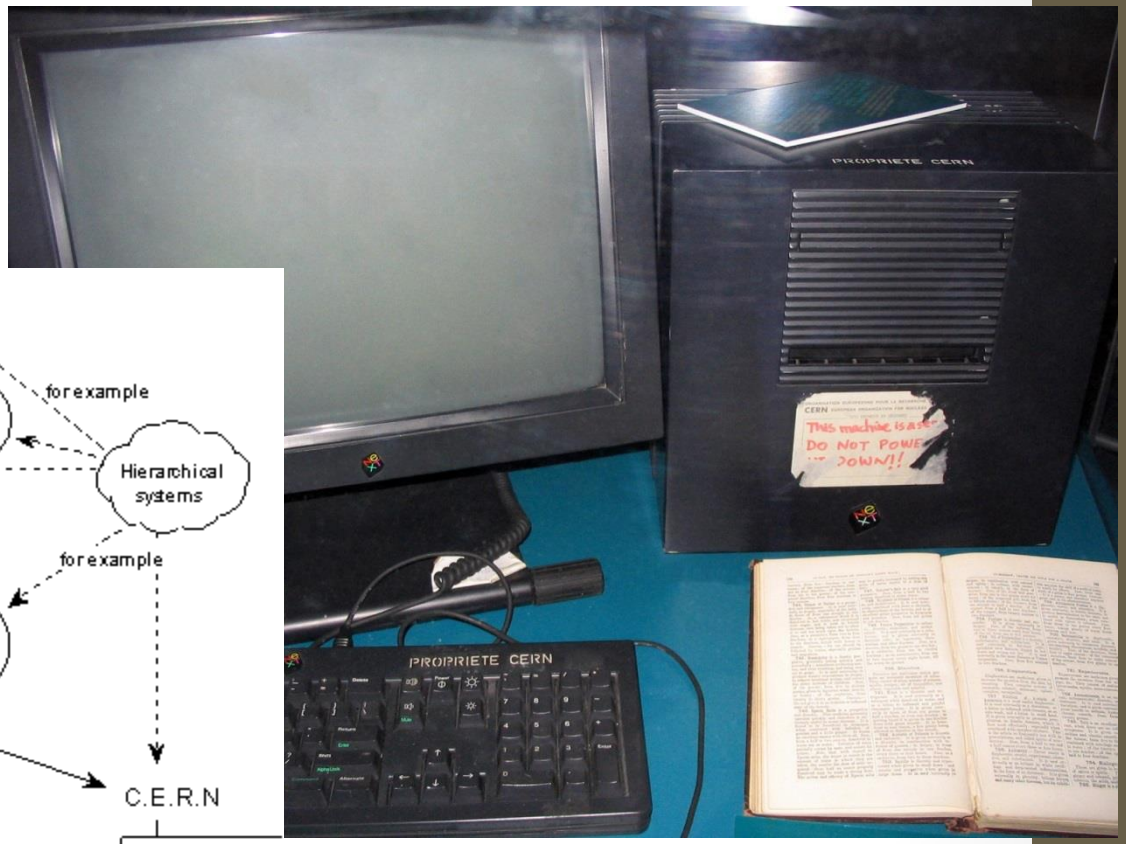
SPARQL

- ***S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage*
- "SPARQL will make a huge difference“ http://blog.semantic-web.at/2009/04/22/tim-berners-lee-we-need-data-on-the-web-to-work-better-together/



Tim Berners-Lee's Web Server



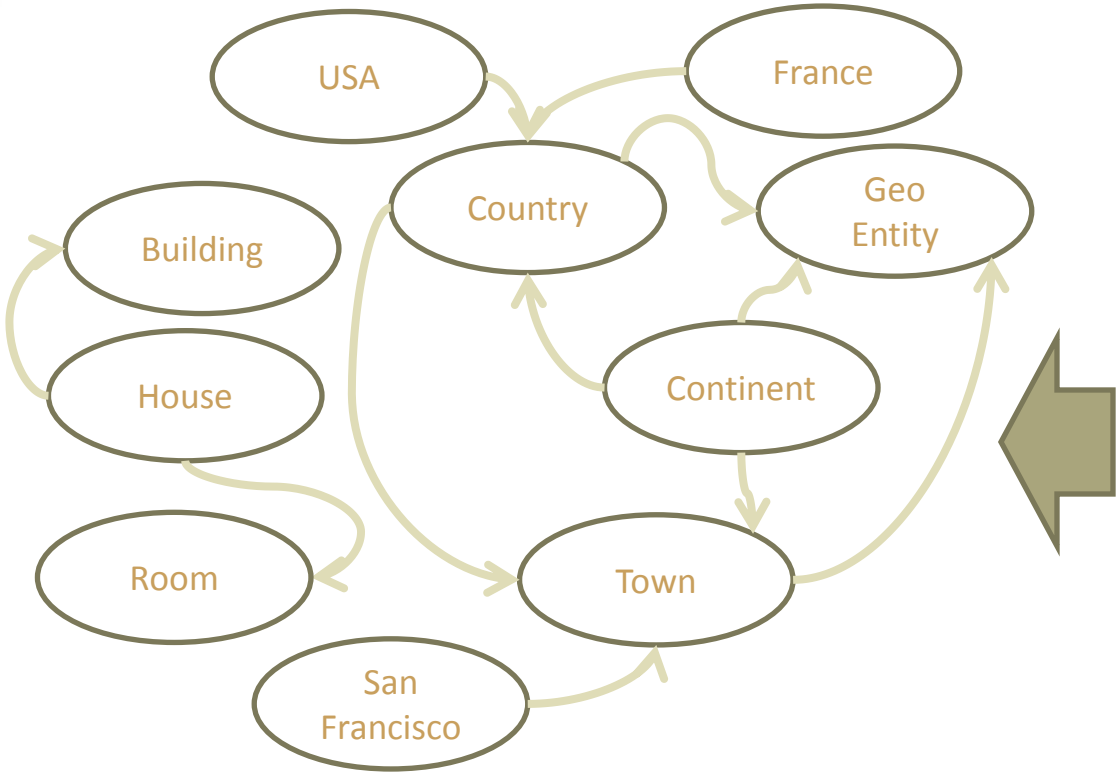


SPARQL: Triplestore

- http://en.wikipedia.org/wiki/Triplestore
- http://en.wikipedia.org/wiki/Sparql

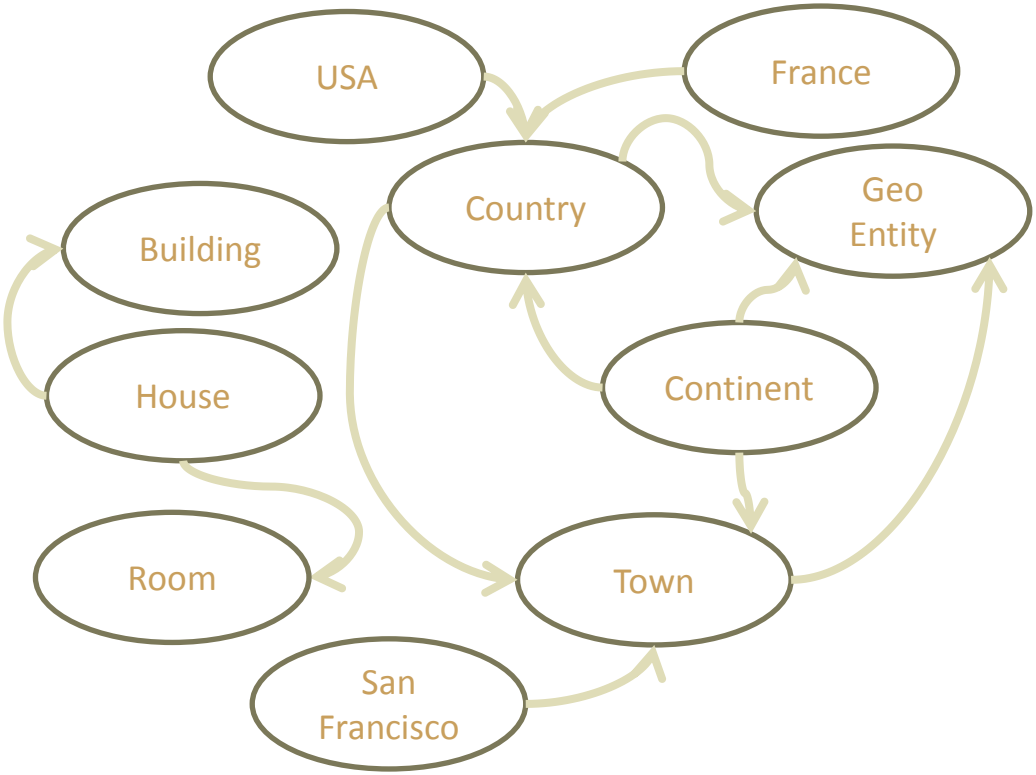
S	P	O
USA	is a	Country
France	is a	Country
Country	is a	Geo Entity
Country	contains	Town
Continent	contains	Country
Continent	is a	Geo Entity
Continent	contains	Town
Town	is a	Geo Entity
San Francisco	is a	Town
House	contains	Room
House	is a	Building

SPARQL: Graph

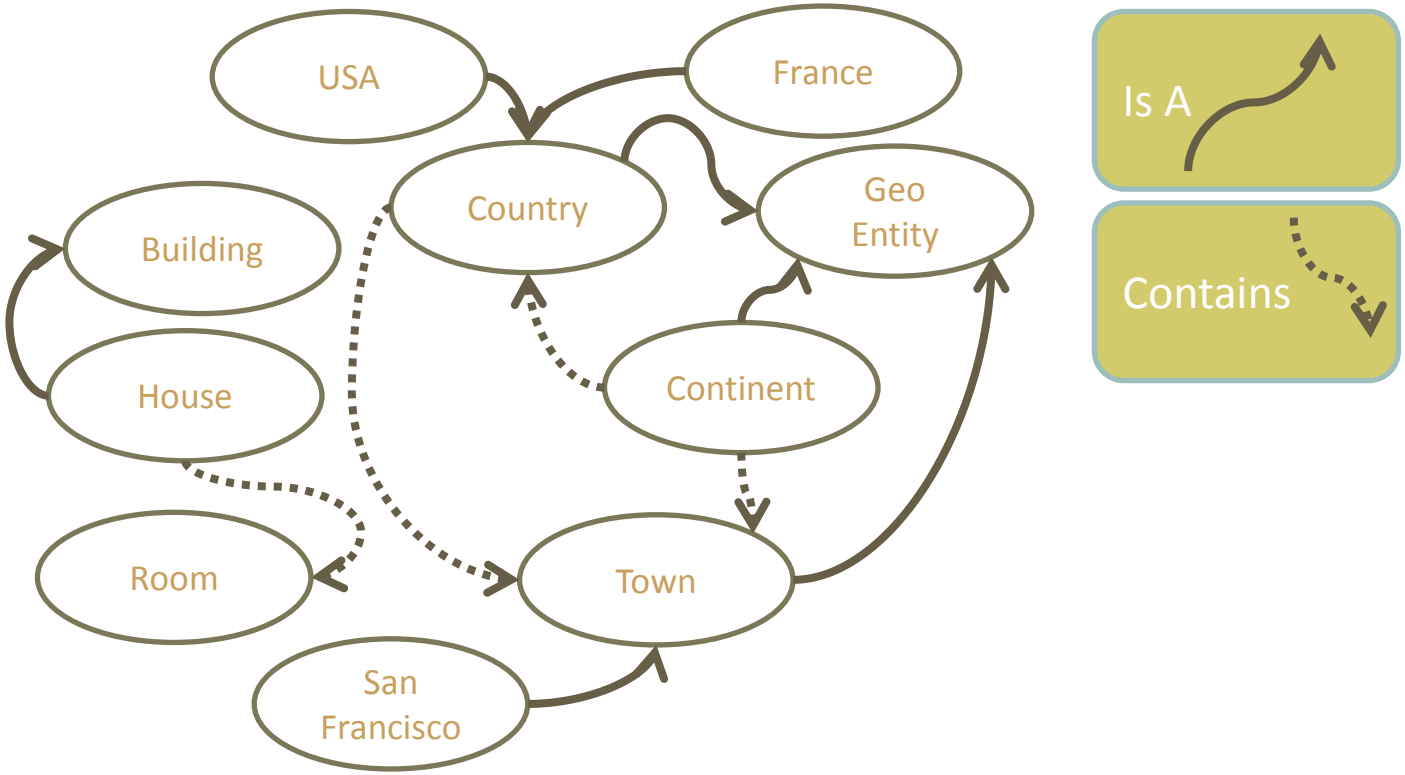


S	P	O
USA	is a	Country
France	is a	Country
Country	is a	Geo Entity
Country	contains	Town
Continent	contains	Country
Continent	is a	Geo Entity
Continent	contains	Town
Town	is a	Geo Entity
San Francisco	is a	Town
House	contains	Room
House	is a	Building

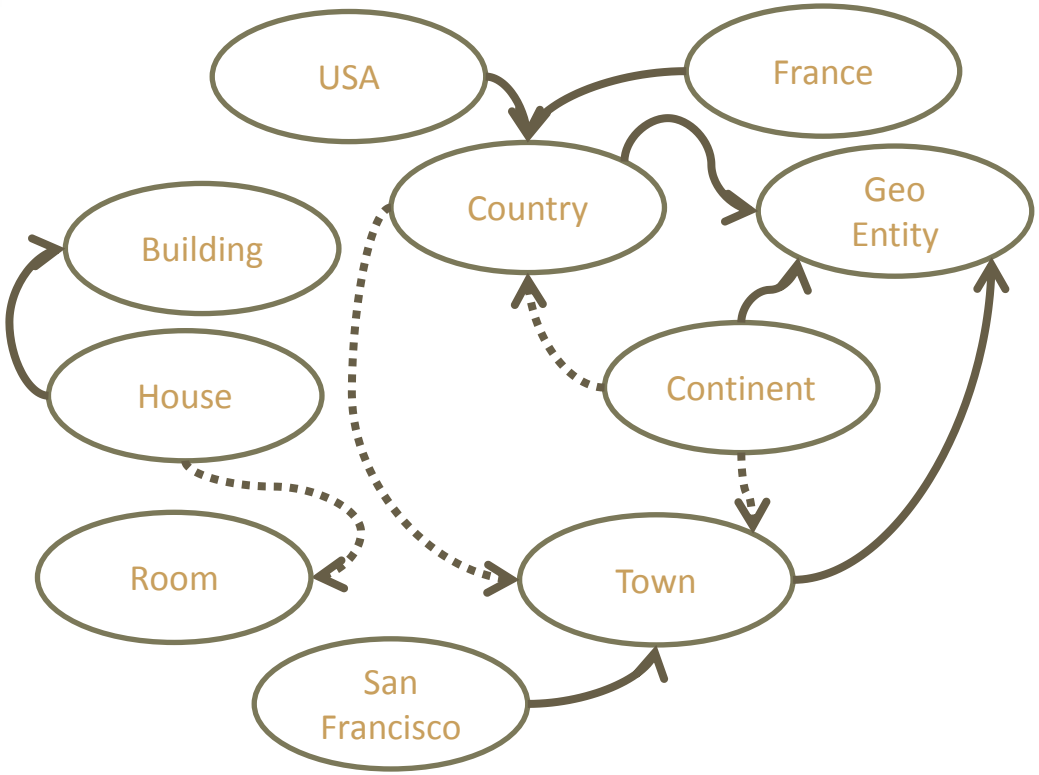
SPARQL: Graph

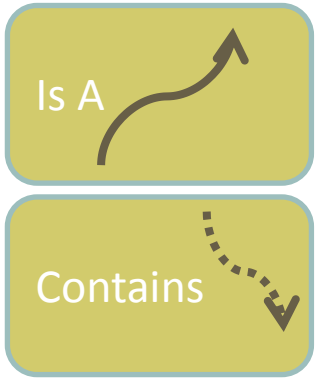


SPARQL: Graph



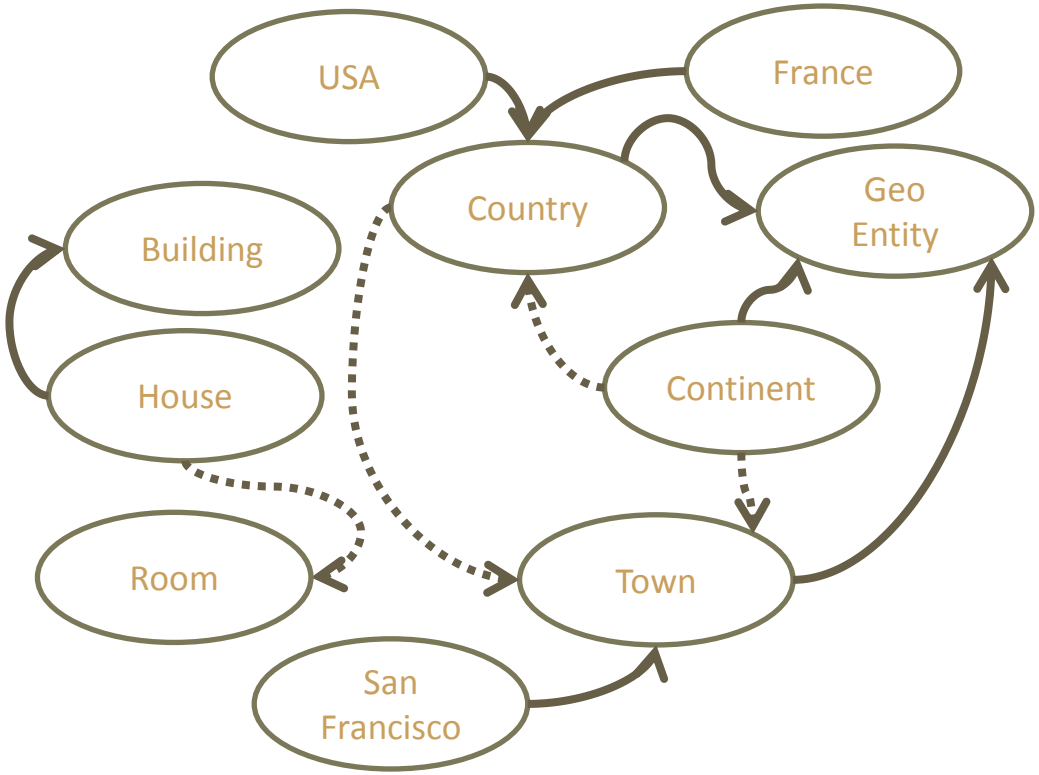
SPARQL: Subjects & Objects

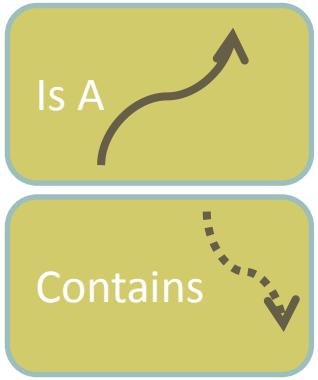




Show node (subject) that contains something
Show node (object) that is contained by something
Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects

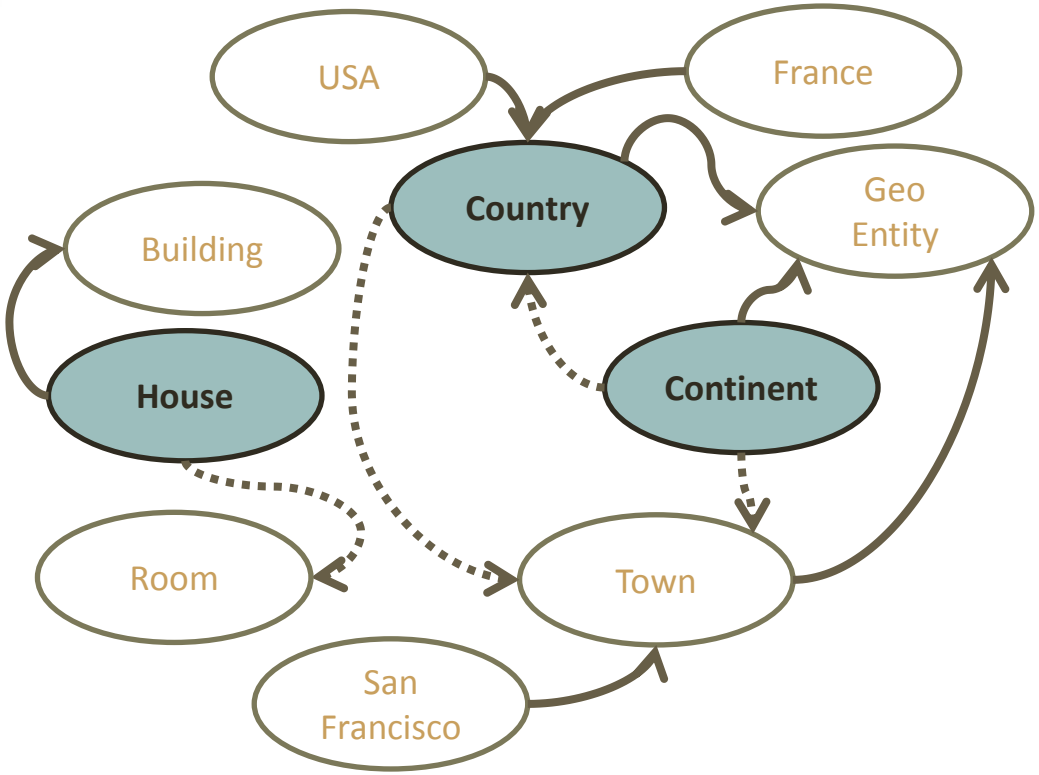


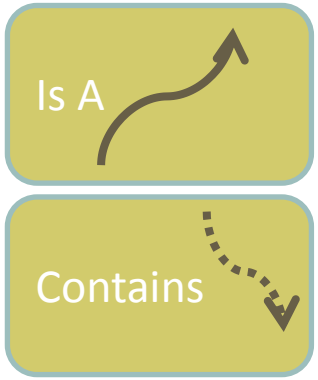



```
SELECT <Columns to be Presented>
FROM <Triplestore>
WHERE { <condition>}
```

- Show node (subject) that contains something
- Show node (object) that is contained by something
- Show node (subject) that is a Geo Entity

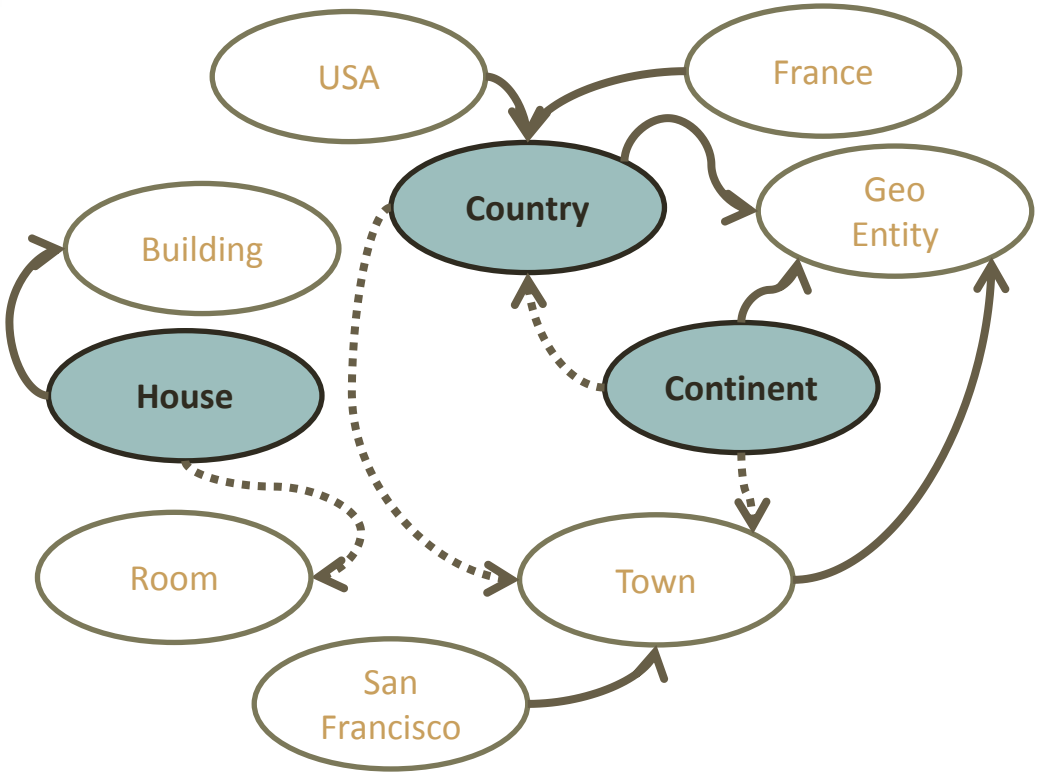
SPARQL: Subjects & Objects

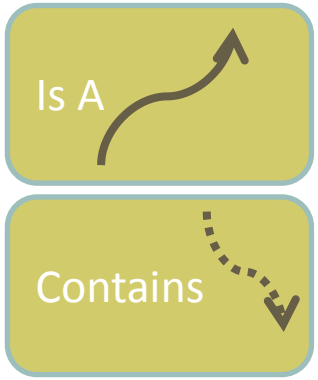




Show node (subject) that contains something
Show node (object) that is contained by something
Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects

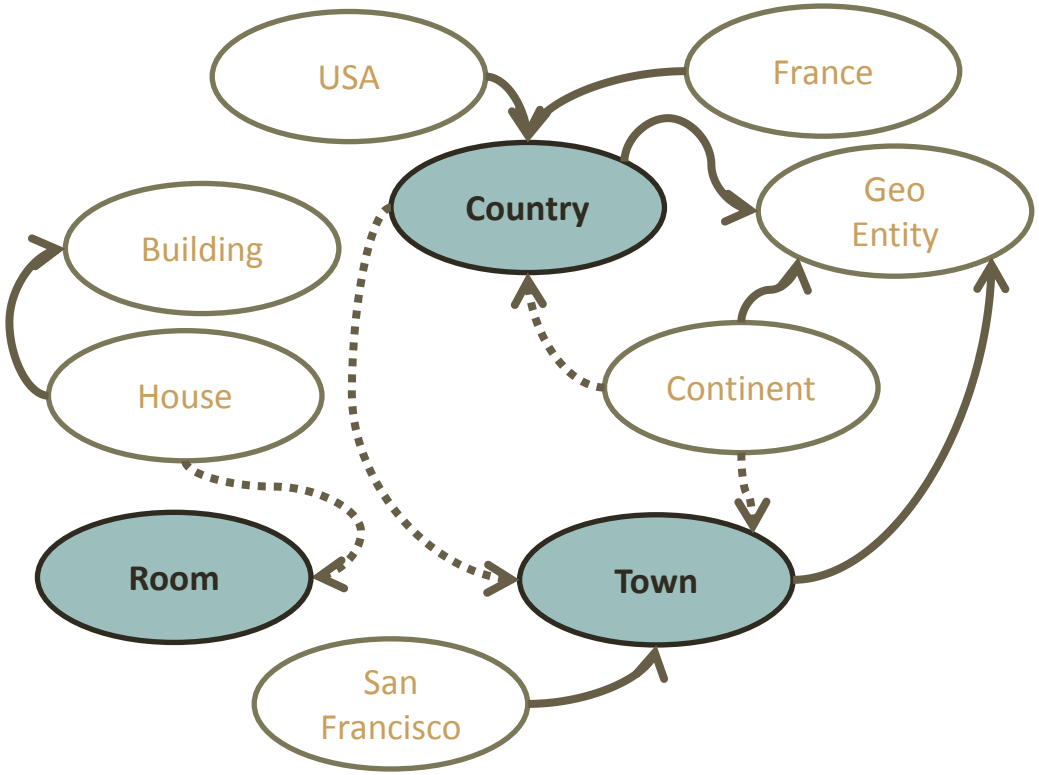


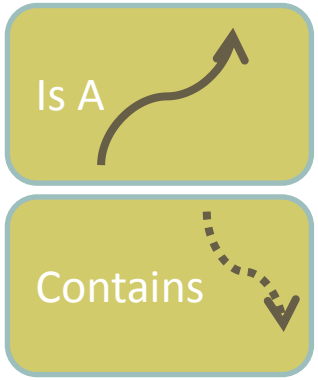



```
SELECT ?s
FROM <myTripleStore>
WHERE {
?s contains ?o
}
```

Show node (subject) that contains something
Show node (object) that is contained by something
Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects



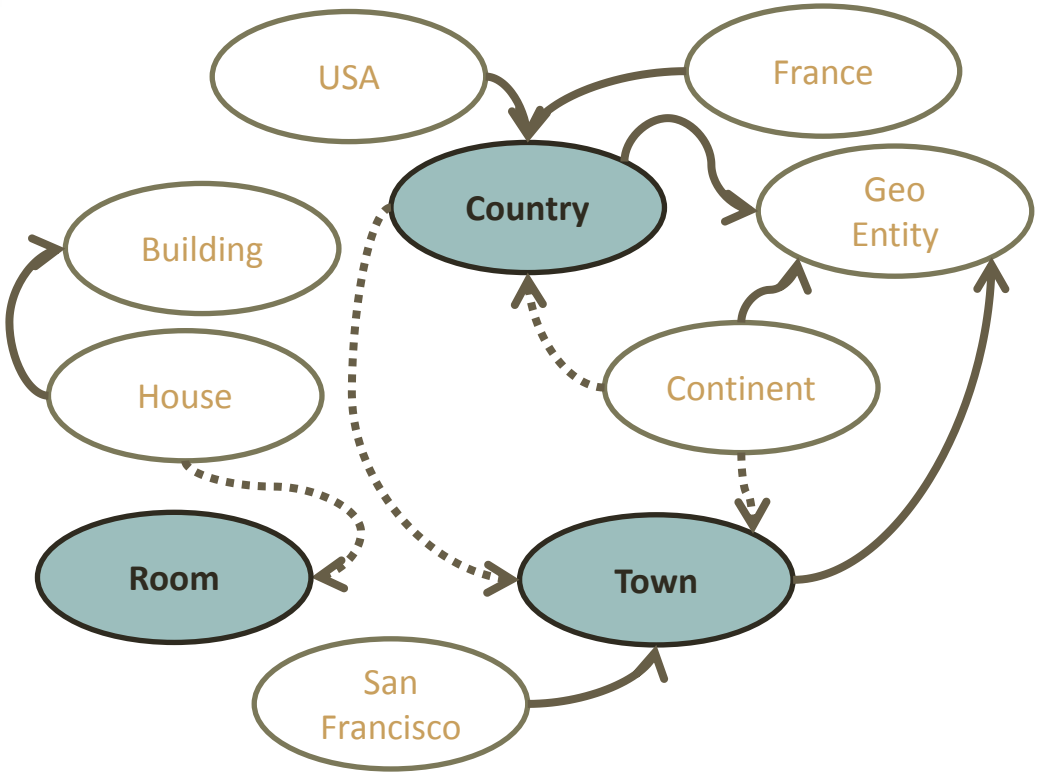


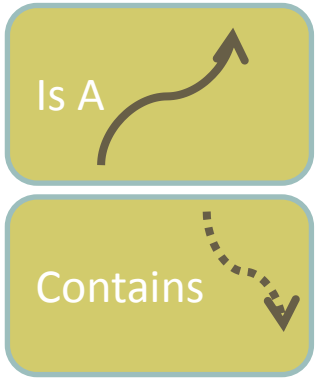
Show node (subject) that contains something

Show node (object) that is contained by something

Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects





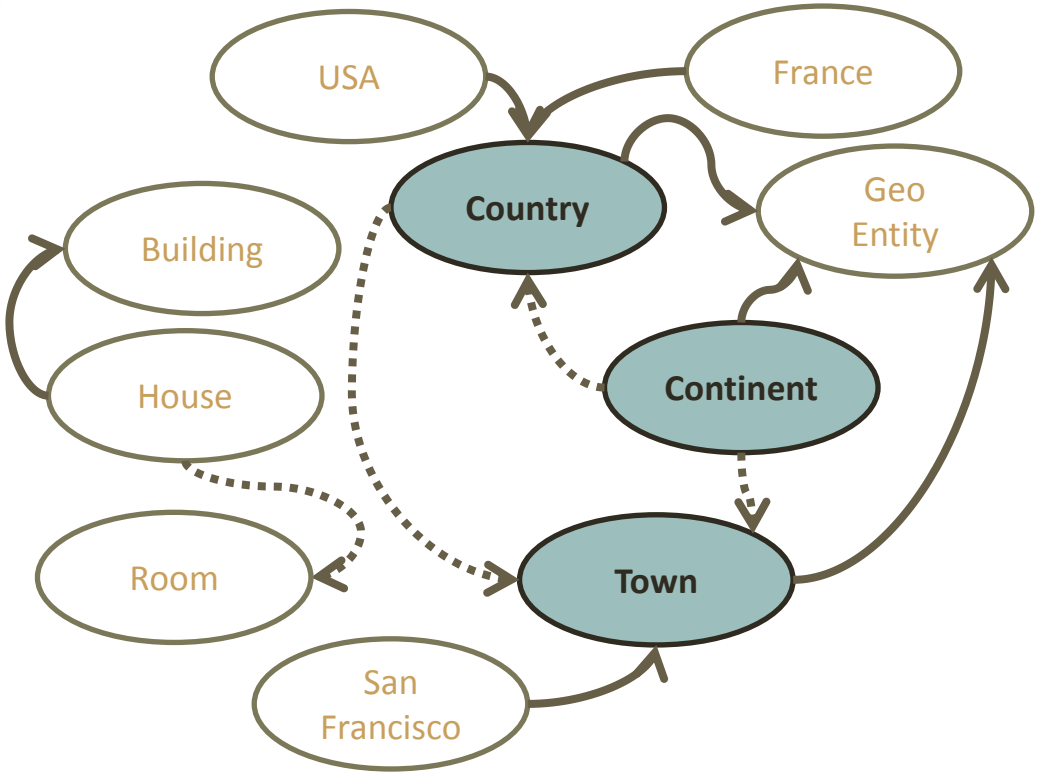

```
SELECT ?o
FROM <myTripleStore>
WHERE {
?s contains ?o}
```

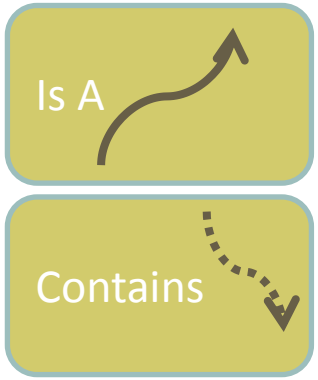
Show node (subject) that contains something

Show node (object) that is contained by something

Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects



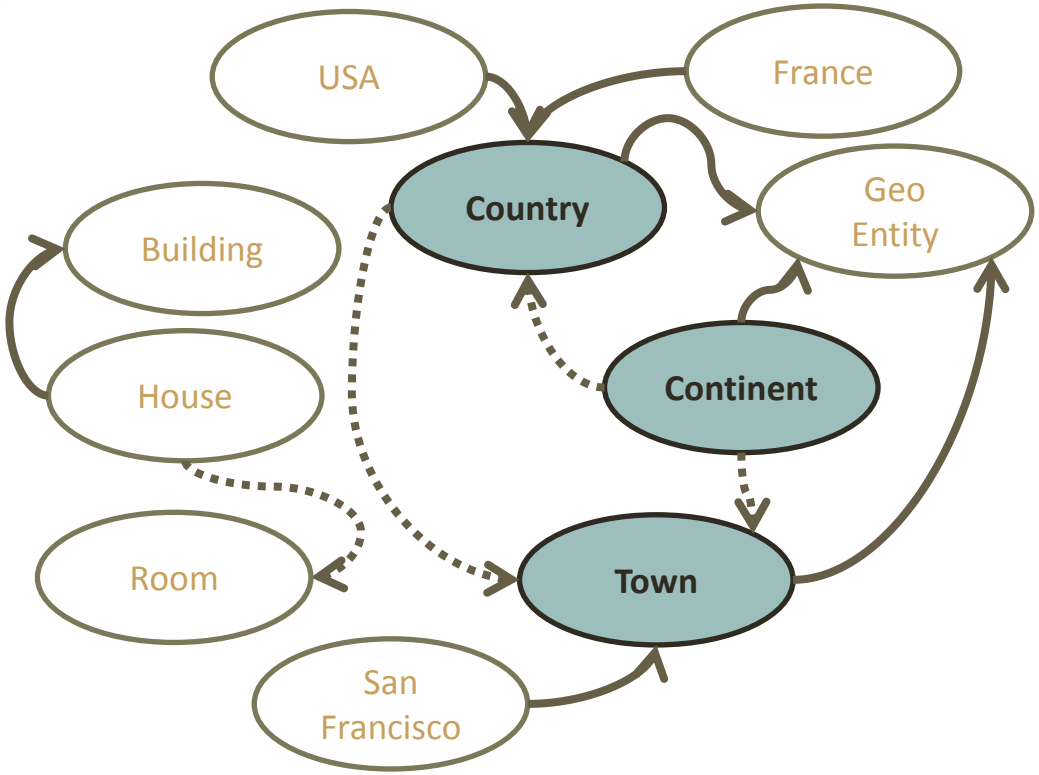


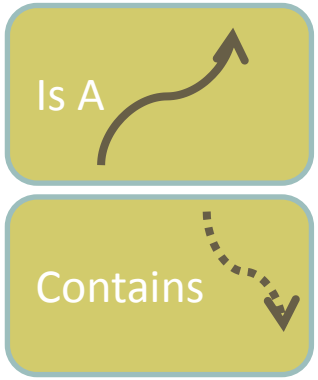
Show node (subject) that contains something

Show node (object) that is contained by something

Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects





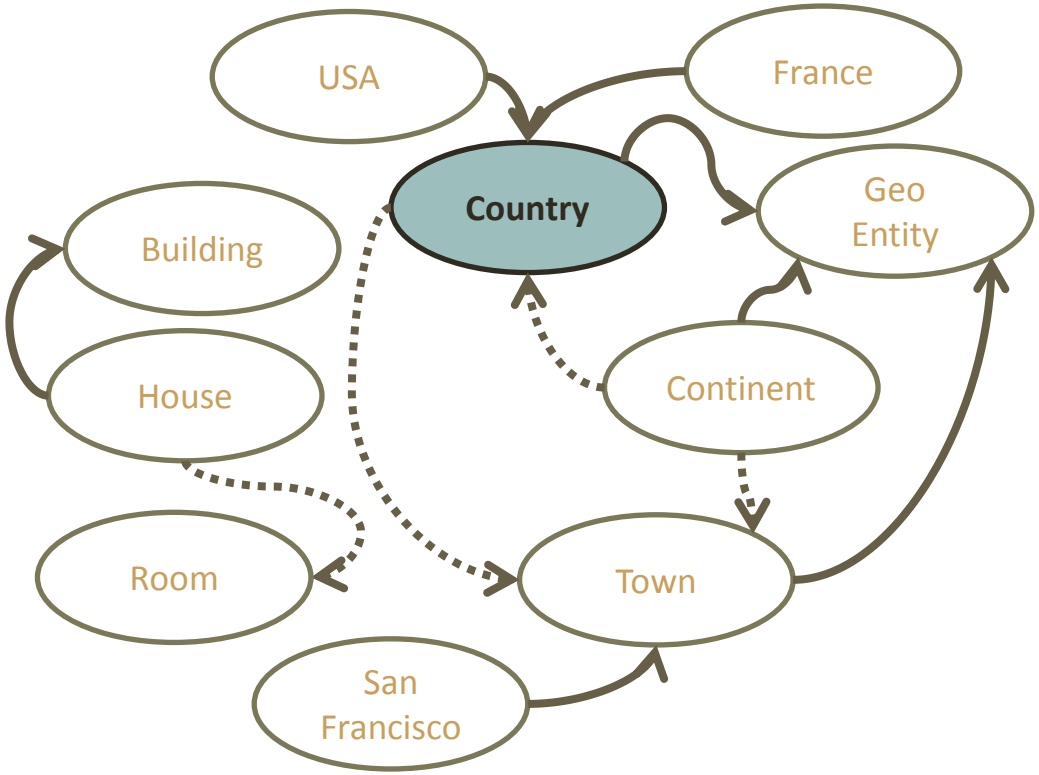

```
SELECT ?s
FROM <myTripleStore>
WHERE {
?s is a Geo Entity}
```

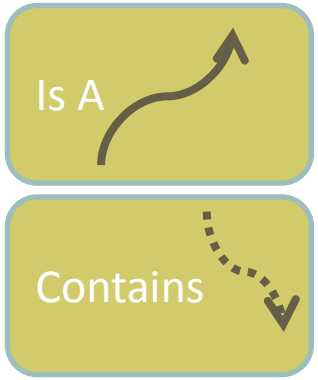
Show node (subject) that contains something

Show node (object) that is contained by something

Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects



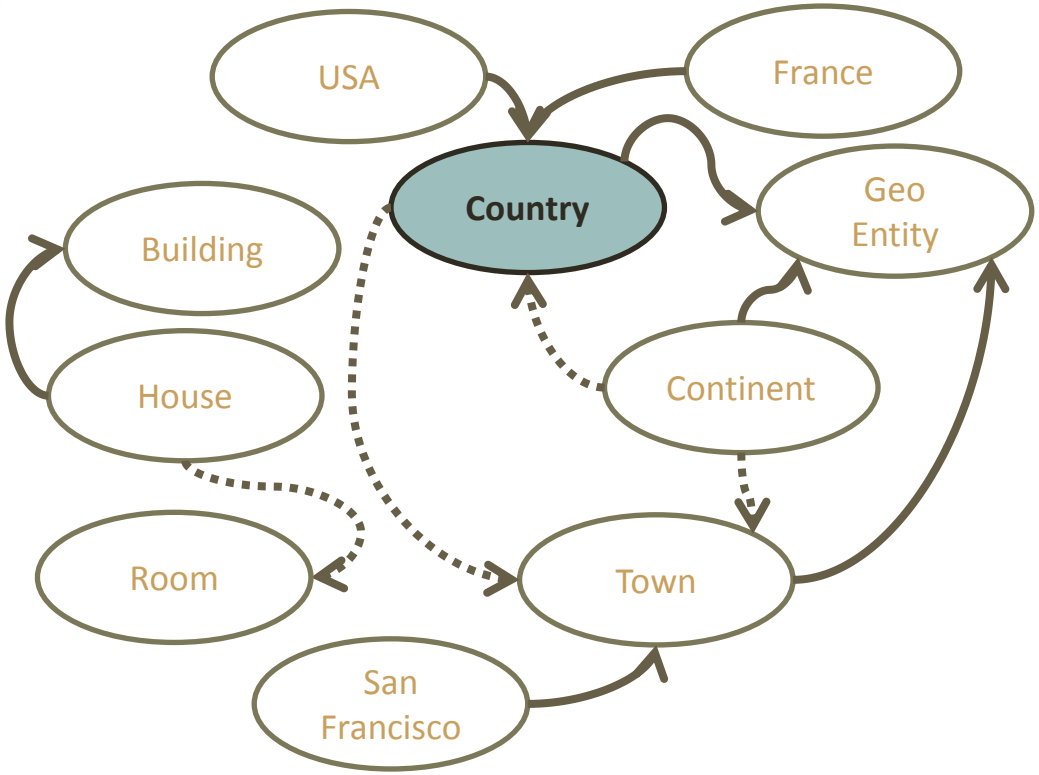


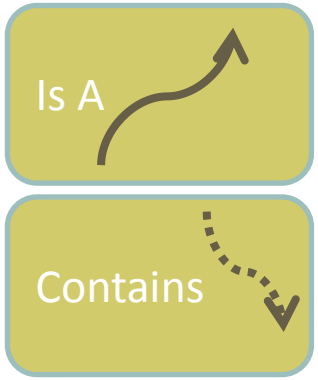
Show node (subject) that contains something

Show node (object) that is contained by something

Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects





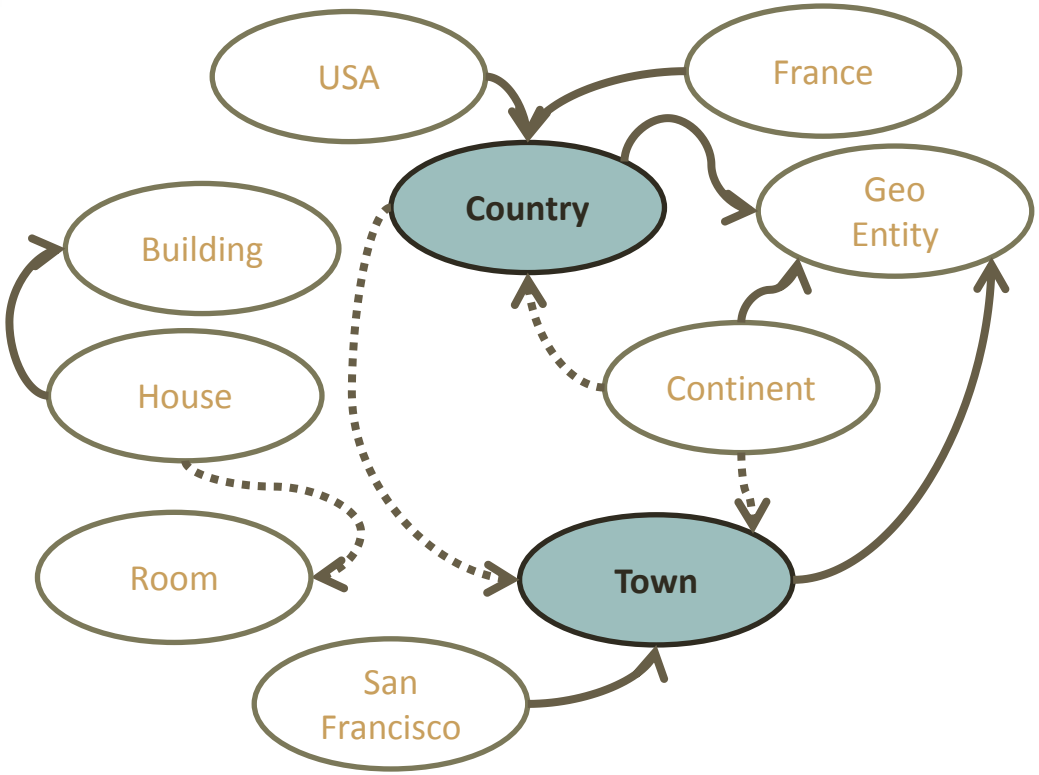

```
SELECT ?s
FROM <myTripleStore>
WHERE {
?s Contains ?o .
?x Contains ?s}
```

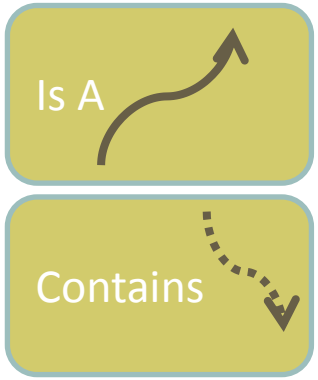
Show node (subject) that contains something

Show node (object) that is contained by something

Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects



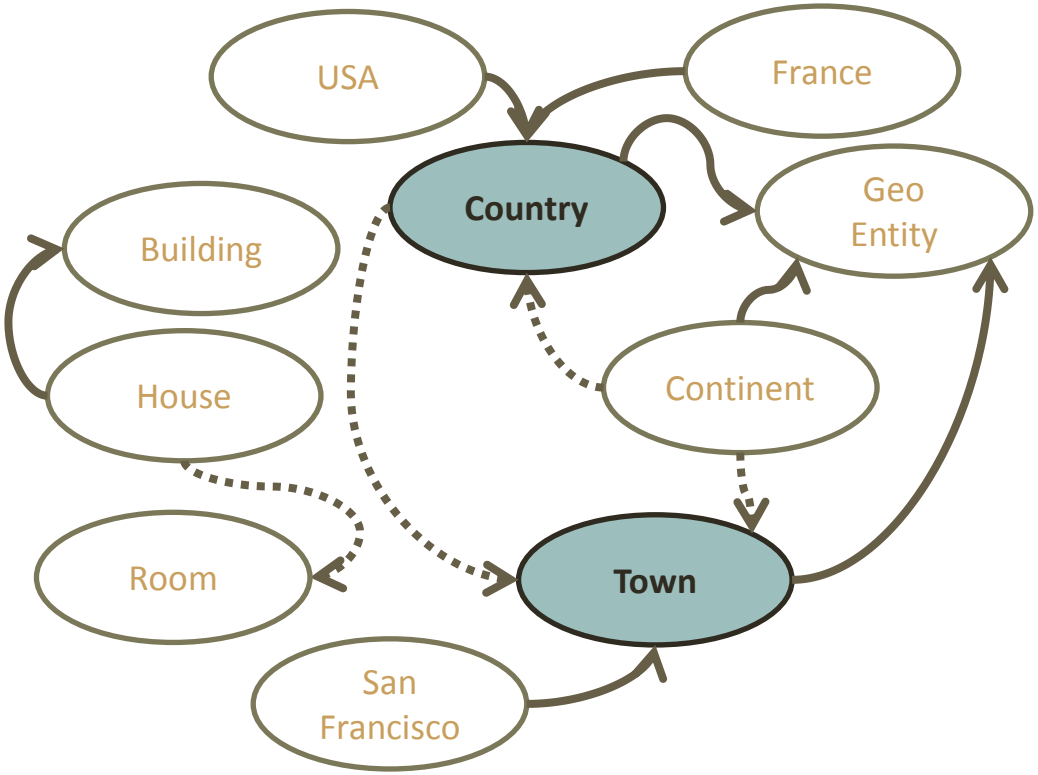


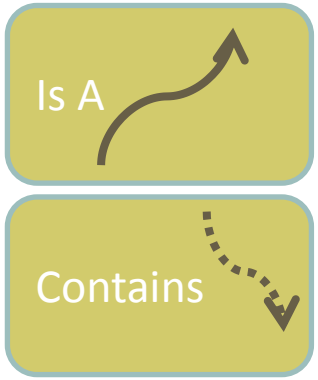
Show node (subject) that contains something

Show node (object) that is contained by something

Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects





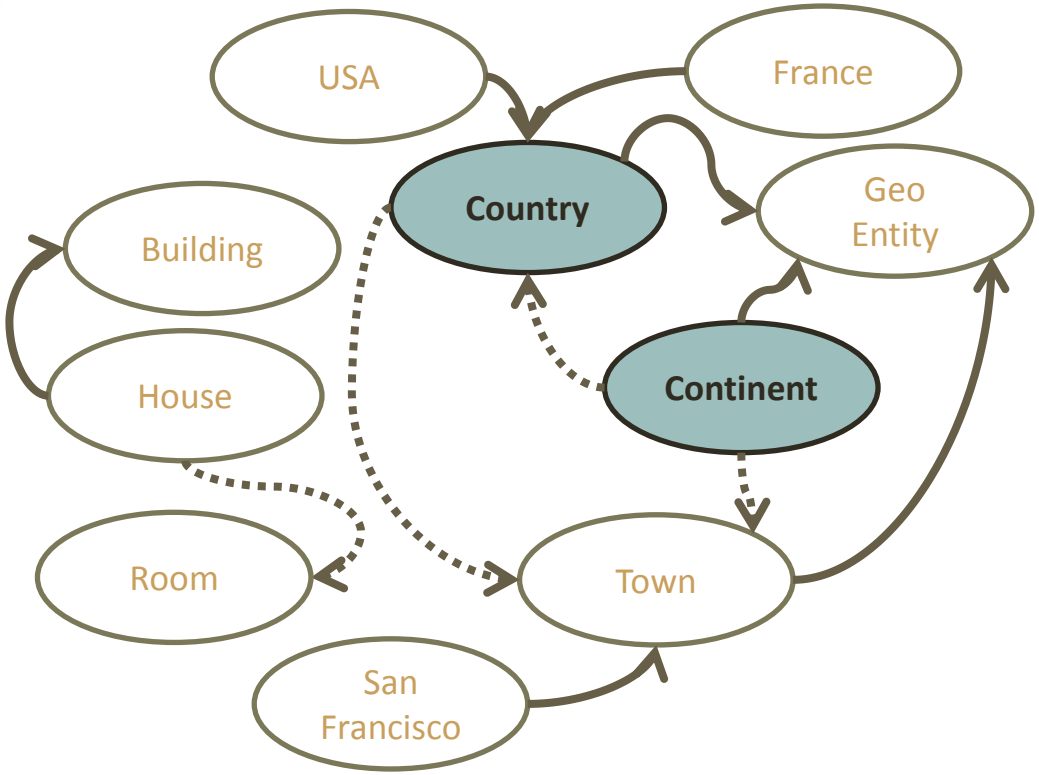

```
SELECT ?o
FROM <myTripleStore>
WHERE {
?x Contains ?o .
?o is a Geo Entity}
```

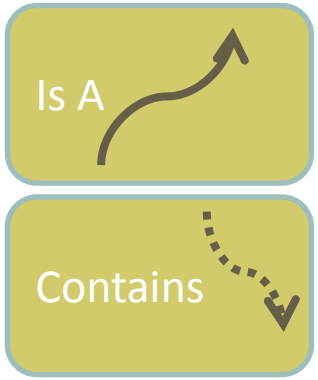
Show node (subject) that contains something

Show node (object) that is contained by something

Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects



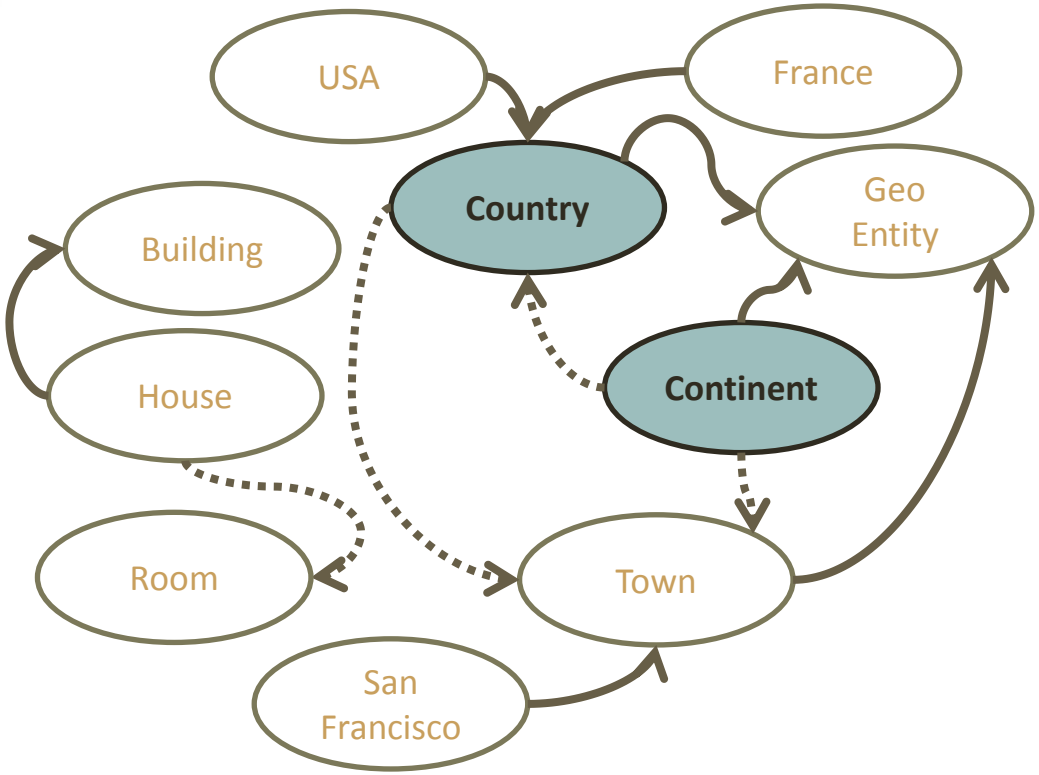


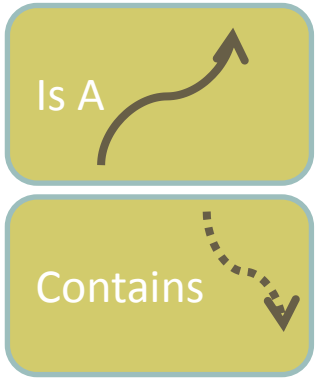
Show node (subject) that contains something

Show node (object) that is contained by something

Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects





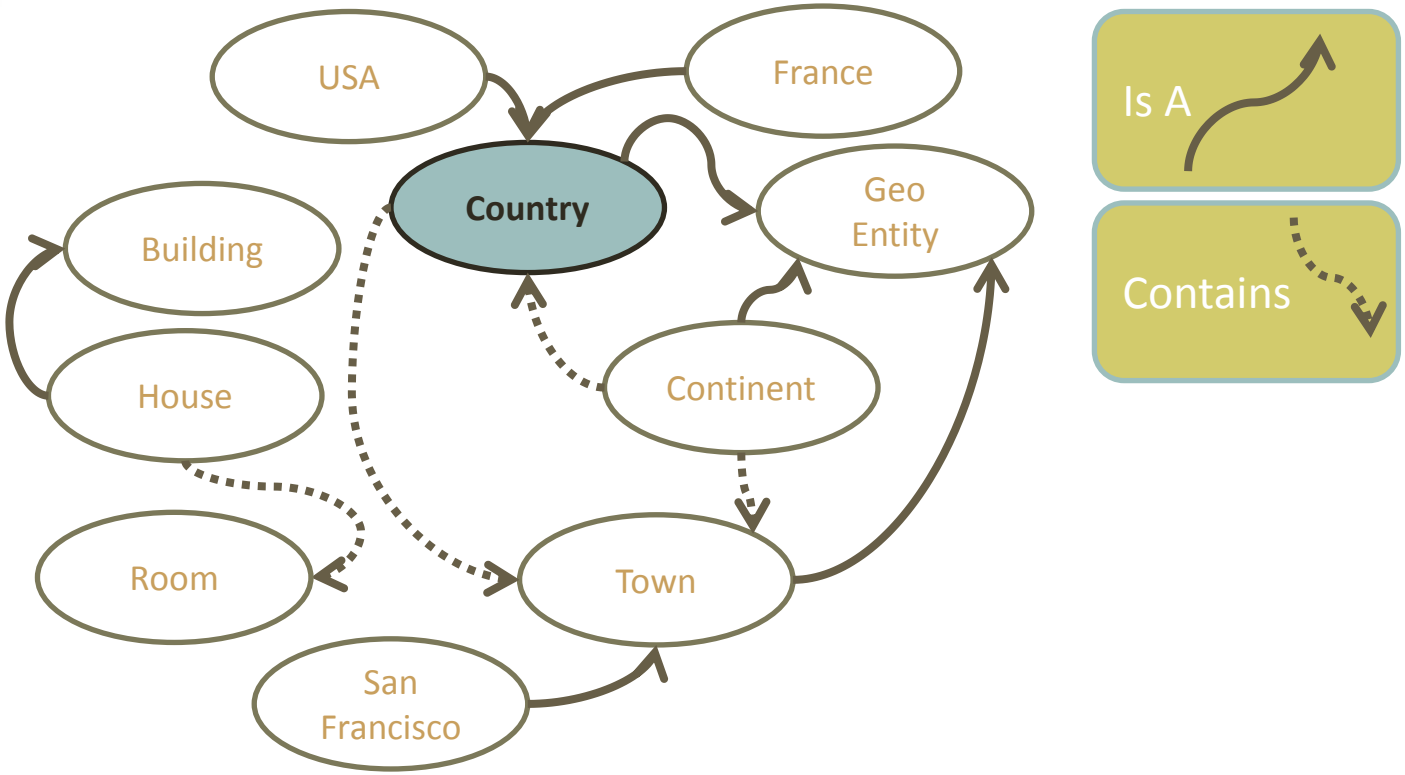

```
SELECT ?o
FROM <myTripleStore>
WHERE {
?s Contains ?o .
?s is a Geo Entity}
```

Show node (subject) that contains something

Show node (object) that is contained by something

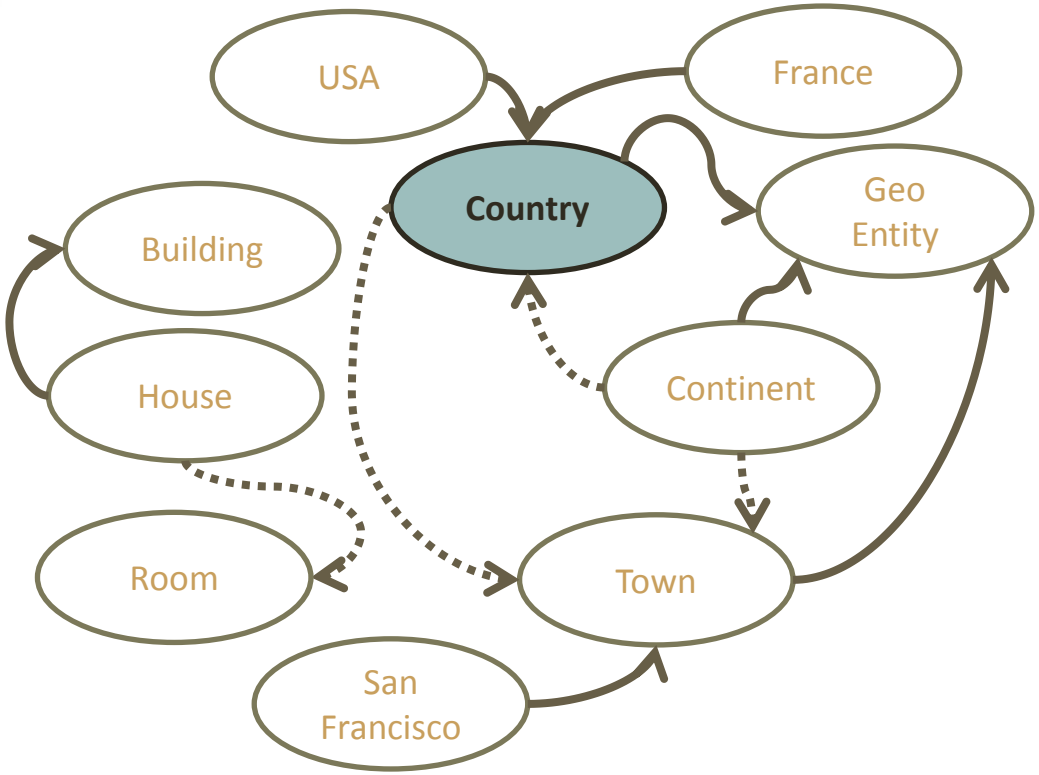
Show node (subject) that is a Geo Entity

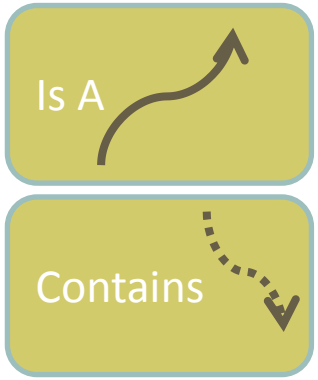
SPARQL: Subjects & Objects



Show node (subject) that contains something
Show node (object) that is contained by something
Show node (subject) that is a Geo Entity

SPARQL: Subjects & Objects






```
SELECT ?o
FROM <myTripleStore>
WHERE {
?s Contains ?o .
?x Contains ?s .
?s is a Geo Entity}
```

Show node (subject) that contains something

Show node (object) that is contained by something

Show node (subject) that is a Geo Entity

Quiz 10b

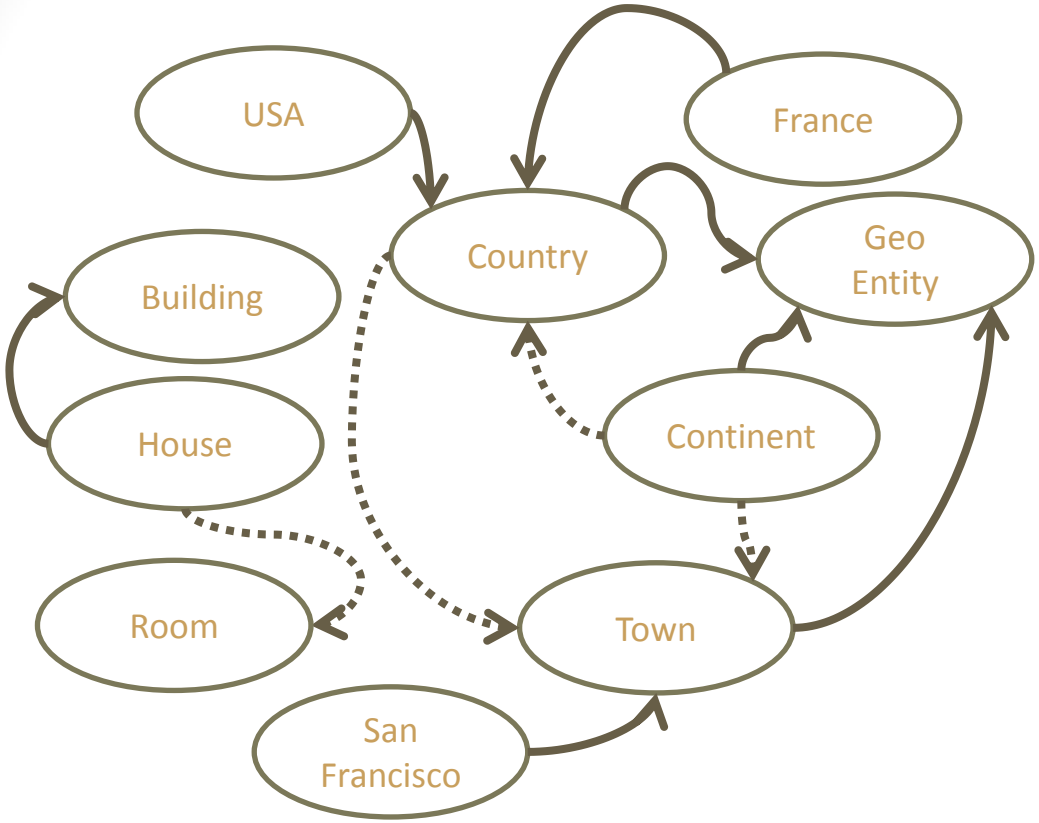
- SPARQL
- https://catalyst.uw.edu/webq/survey/ernsthe/273784

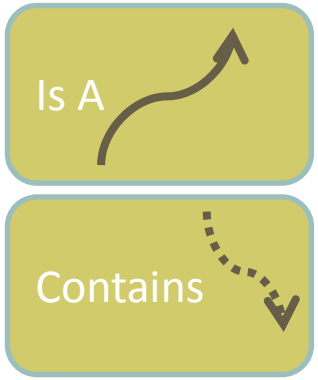
- The question and answer are shown during the quiz. See the projected slide.

SPARQL: Predicates

- Search for predicates

SPARQL: Predicates

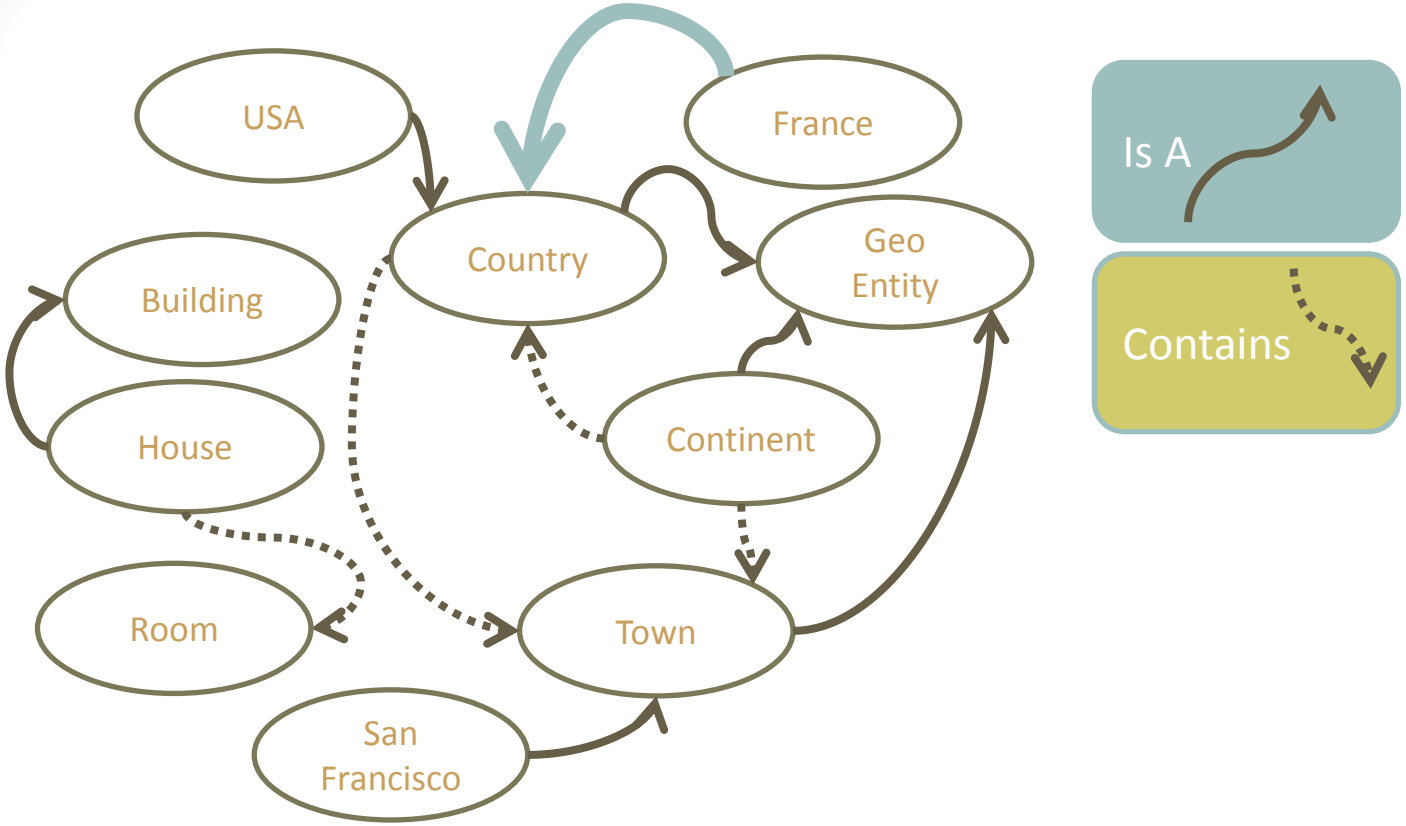




Show relationships (Predicate) from France to Anything

Show relationships from Country to Town

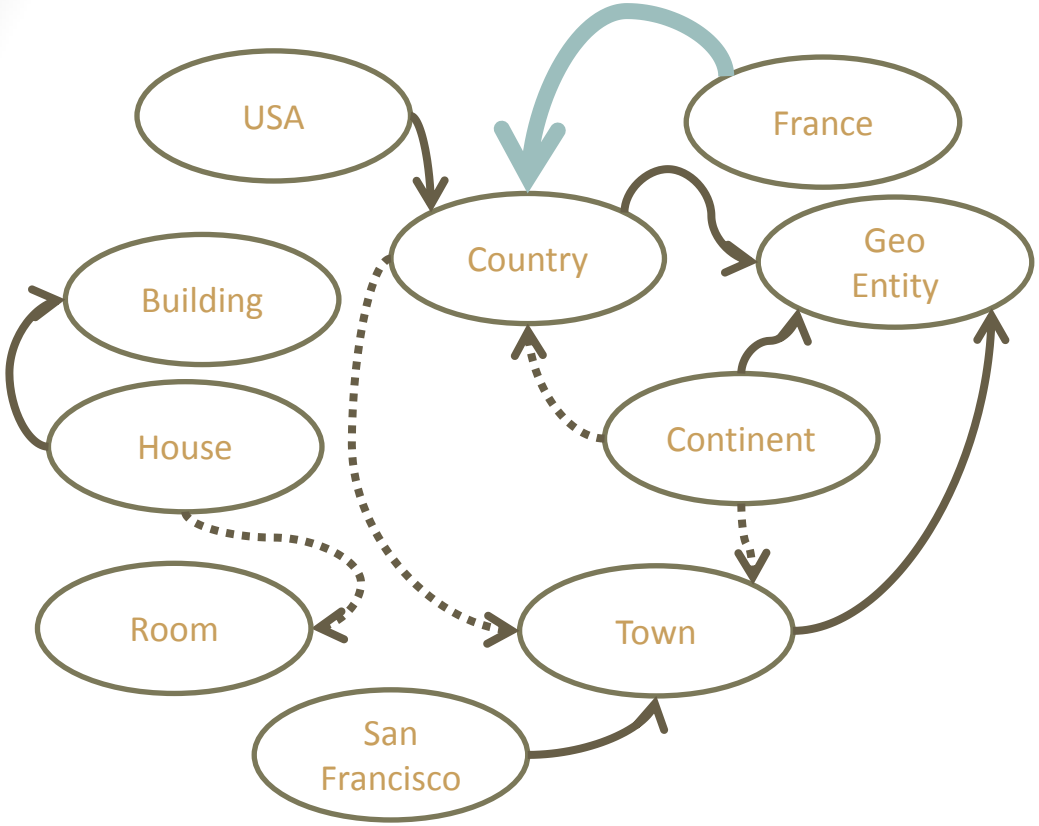
SPARQL: Predicates

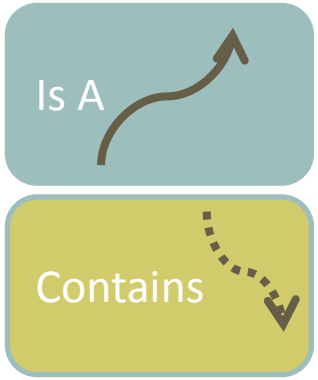


Show relationships (Predicate) from France to Anything

Show relationships from Country to Town

SPARQL: Predicates



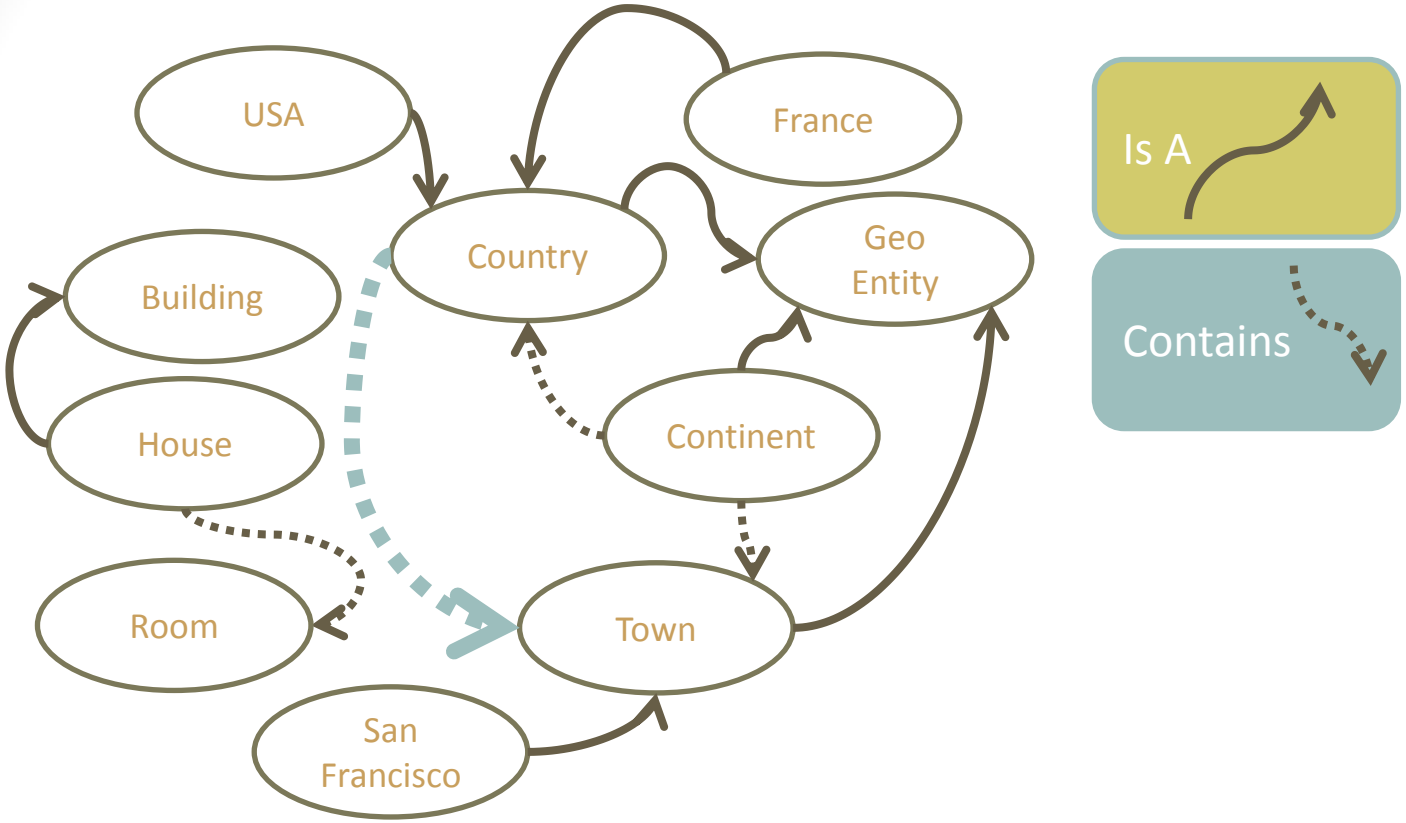



```
SELECT ?p
FROM <myTripleStore>
WHERE {
France ?p ?o}
```

Show relationships (Predicate) from France to Anything

Show relationships from Country to Town

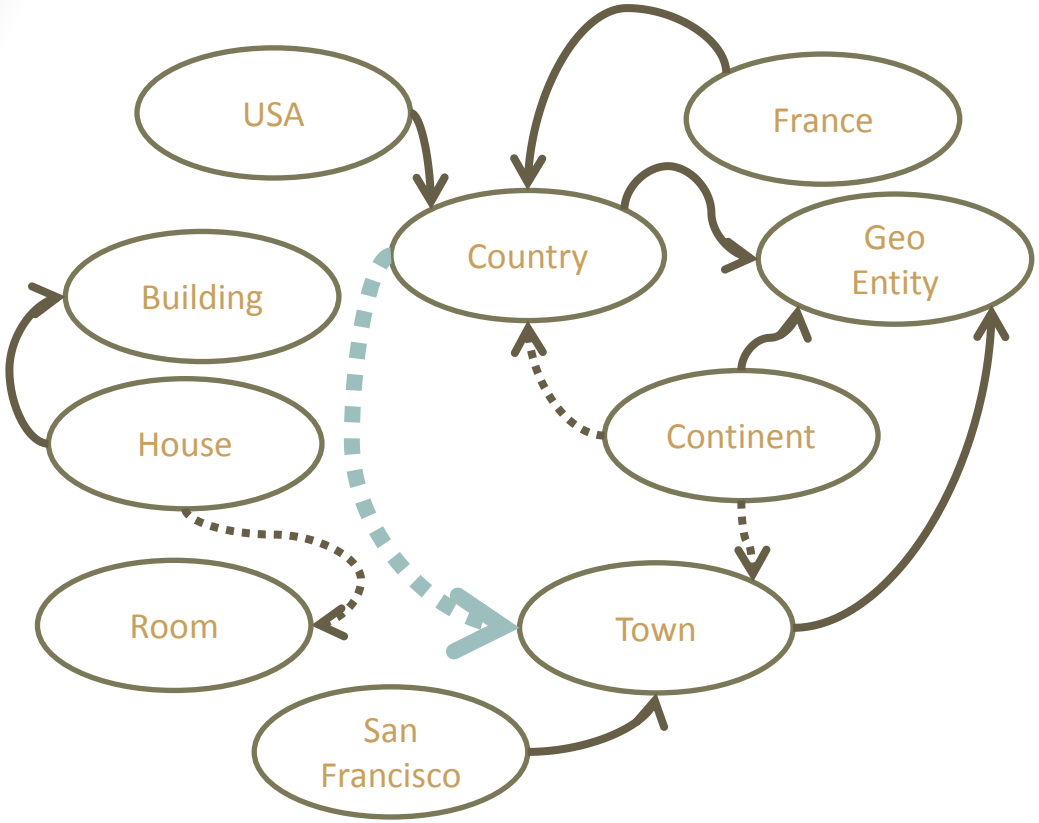
SPARQL: Predicates

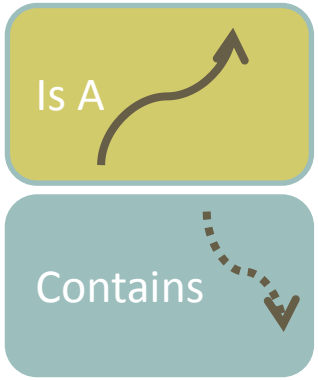


Show relationships (Predicate) from France to Anything

Show relationships from Country to Town

SPARQL: Predicates



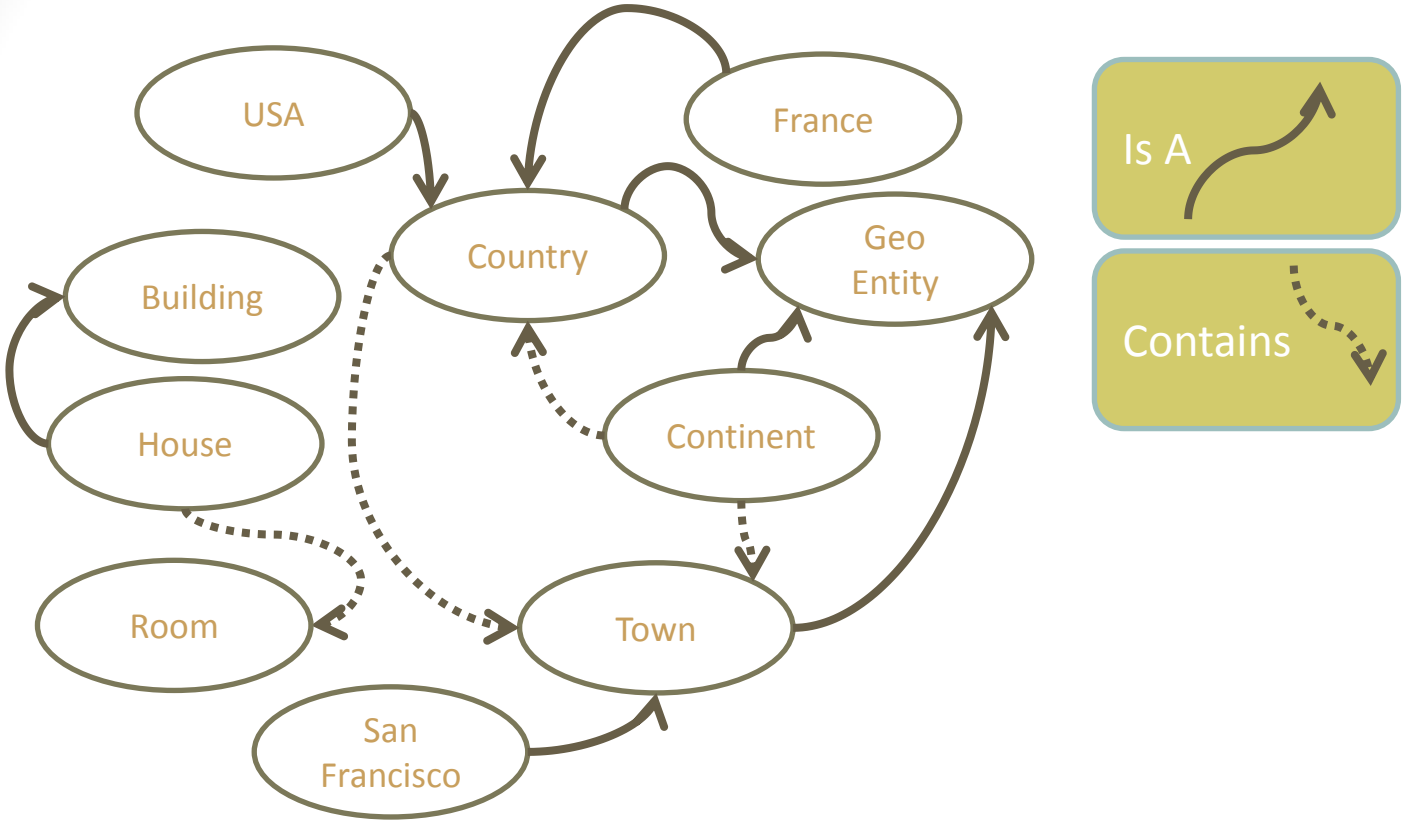



```
SELECT ?p
FROM <myTripleStore>
WHERE {
Country ?p Town}
```

Show relationships (Predicate) from France to Anything

Show relationships from Country to Town

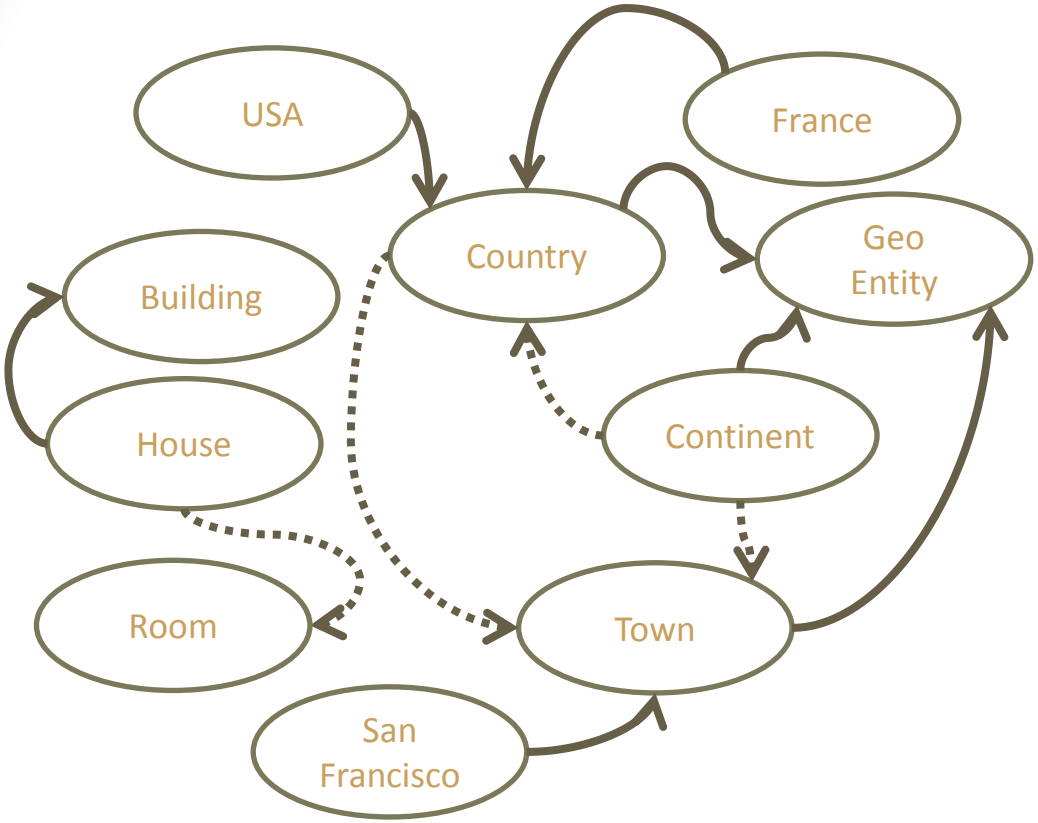
SPARQL: Predicates

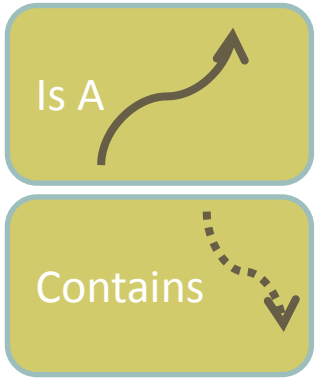


Show relationships (Predicate) from France to Anything

Show relationships from Country to Town

SPARQL: Predicates



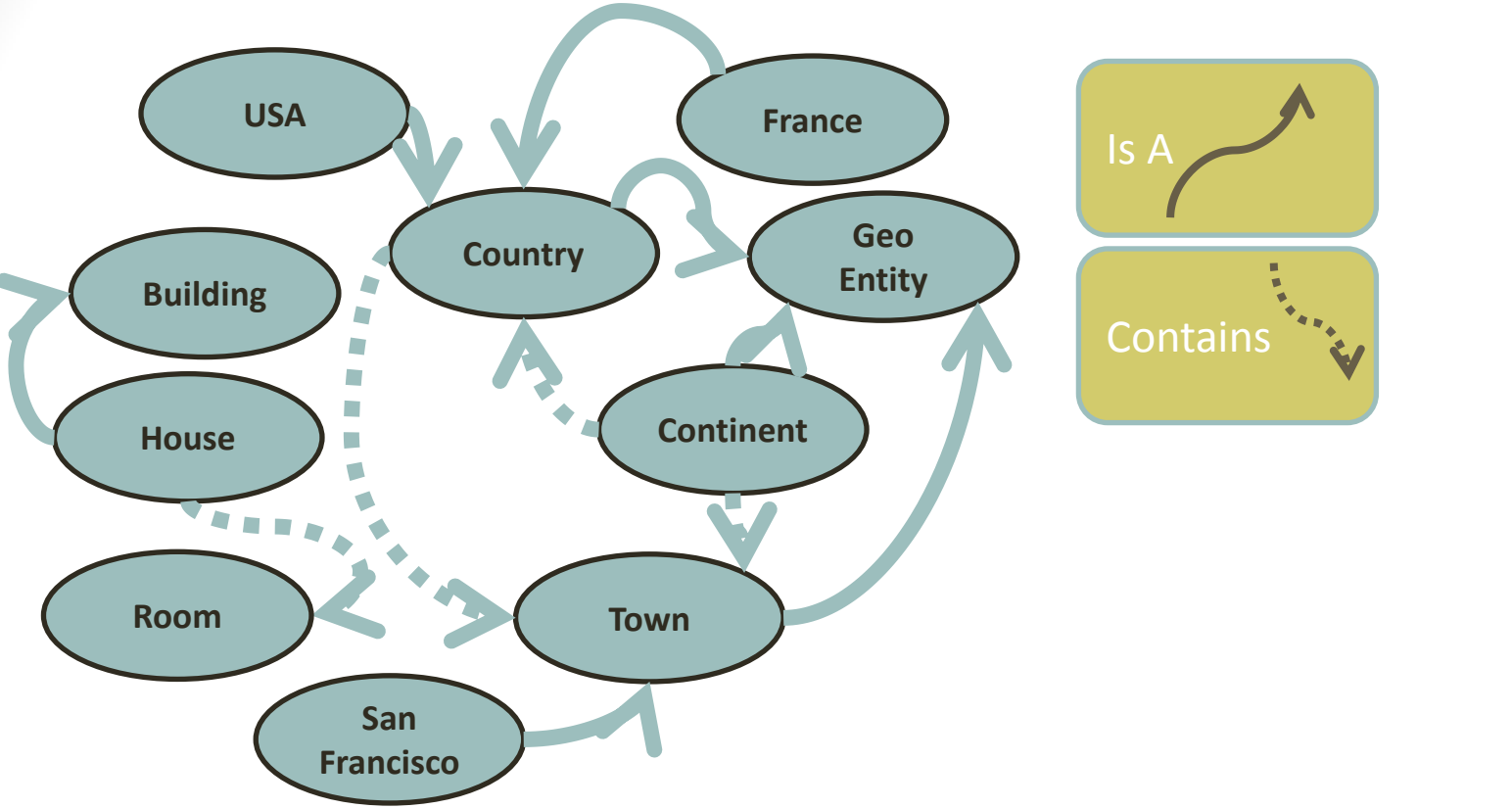



```
SELECT ?p
FROM <myTripleStore>
WHERE {
France ?p ?o .
Country ?p Town}
```

Show relationships (Predicate) from France to Anything

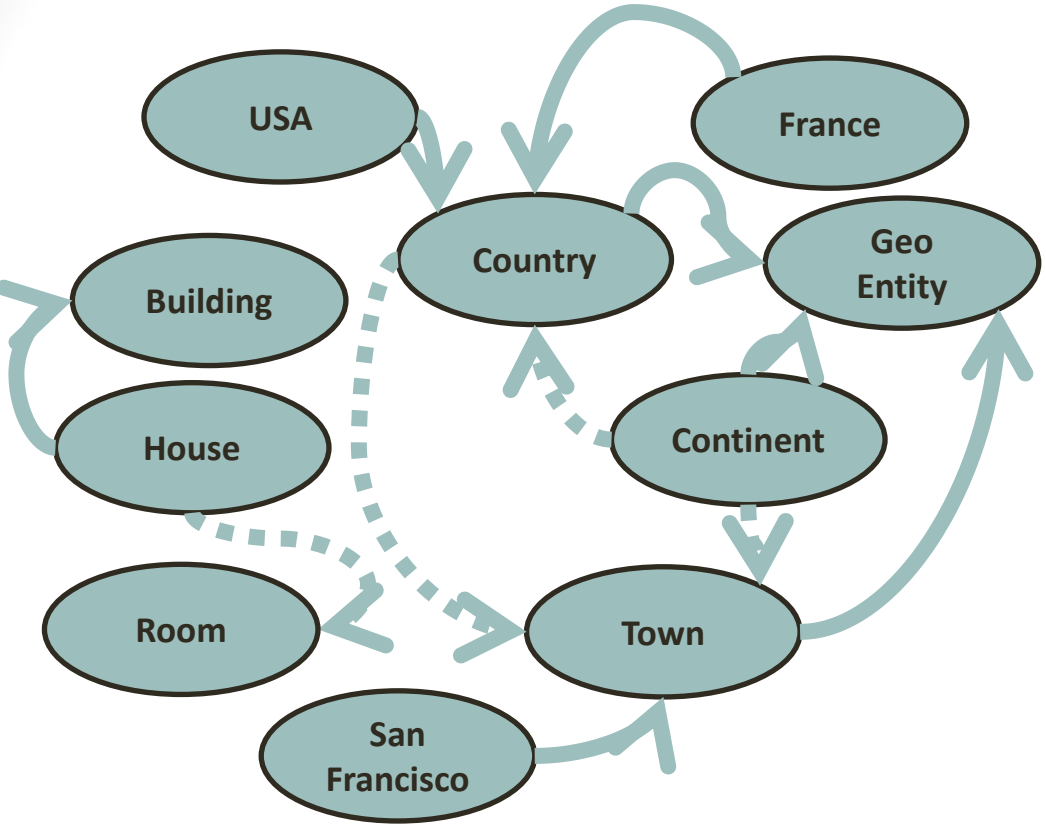
Show relationships from Country to Town

SPARQL: Predicates



Show any node and relationship

SPARQL: Predicates




```
SELECT ?s ?p ?o
FROM <myTripleStore>
WHERE {
?s ?p ?o}
```

Show any node and relationship

SPARQL Examples (0)

- http://dbpedia.org/sparql
- http://lod.openlinksw.com/sparql/

```
#Total number of triples:
SELECT (COUNT(*) AS ?NumberOfTriples)
FROM <http://dbpedia.org>
WHERE
{?s ?p ?o}
```

```
PREFIX property:
<http://dbpedia.org/property/>
SELECT ?s ?club (max(?o) as ?maxGoals)
FROM <http://dbpedia.org>
WHERE
{
?s property:totalgoals ?o .
?s property:currentclub ?club
filter(?o > 10 && ?o < 1000)
}
ORDER BY DESC(?maxGoals) limit 10
```

SPARQL Examples (1)

- http://dbpedia.org/sparql
- http://lod.openlinksw.com/sparql/

```
PREFIX property:
<http://dbpedia.org/property/>
SELECT DISTINCT ?p
FROM <http://dbpedia.org>
WHERE
{
?s property:years  ?o.
?s ?p ?o2
} limit 10
```

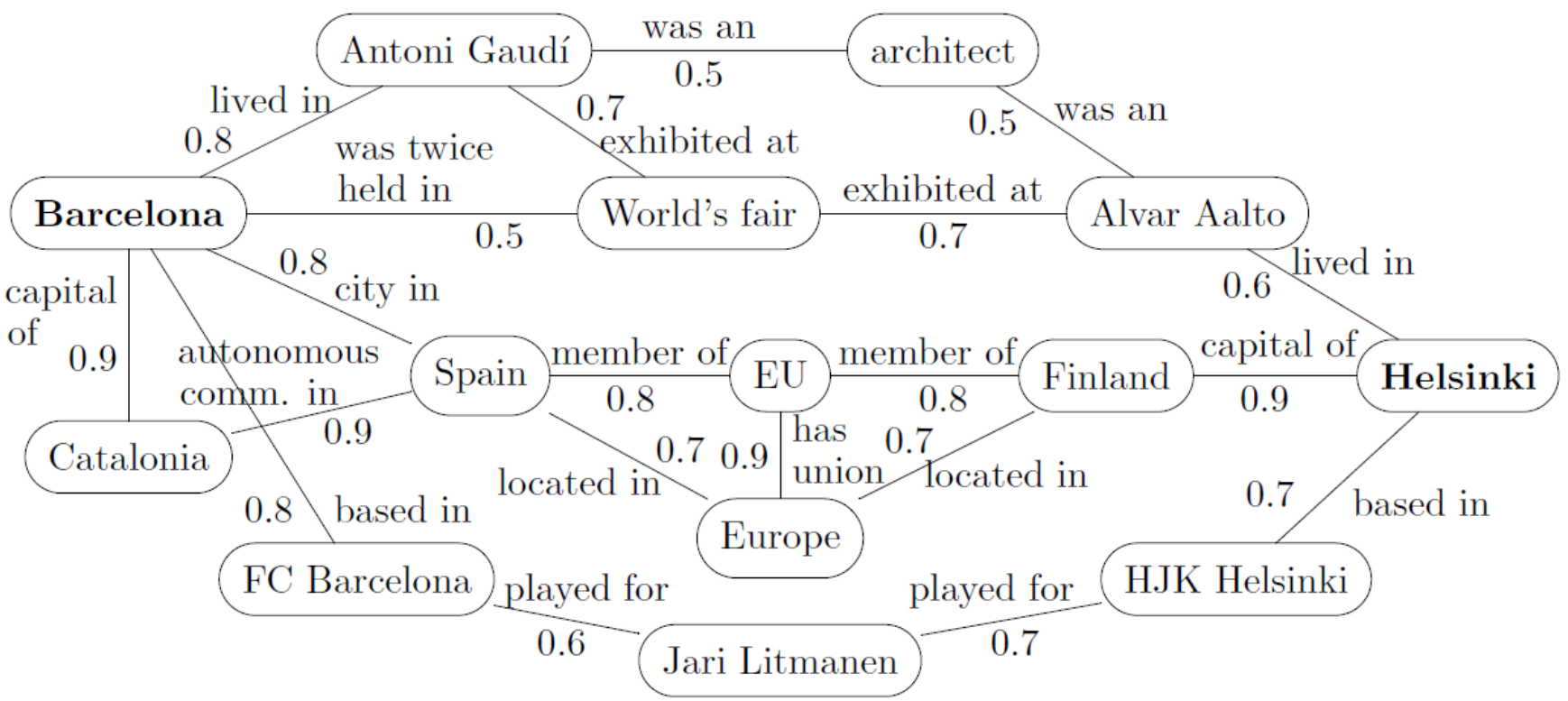
```
PREFIX property:
<http://dbpedia.org/property/>
SELECT DISTINCT ?p
FROM <http://dbpedia.org>
WHERE
{
?s property:years  ?o .
?s ?p ?o
} limit 10
```

SPARQL Examples (2)

- http://dbpedia.org/sparql
- http://lod.openlinksw.com/sparql/

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name ?email
WHERE {
?person a foaf:Person .
?person foaf:name ?name .
?person foaf:mbox ?email . }
```

Future of SPARQL: Semantic Web with Weighted Edges?



SPARQL Exercise 1

1. You have the following triple store <http://ImaginaryTripleStore.com>:
A X B
C X B
B X D
B Y E
F Y B
F X D
F Y E
E X D
G X E

2. You have the following SPARQL statement:
SELECT ?x
FROM
<http://ImaginaryTripleStore.com>
WHERE
{F ?p ?o .
?o ?p ?x}

3. Solution

3.1 List all items that fulfil "F ?p ?o". Place them below the where clause.

```
F  ?p  ?o  .  ?o  ?p   ?x
F   Y   B
F   Y   D
F   Y   E
```

3.2 List all items that fulfil "?o ?p ?x" where ?o is B, D, or E and ?p is Y. Place them below the where clause.

```
F  ?p  ?o  .  ?o  ?p   ?x
F   Y   B  .  B   Y    E
F   Y   D
F   Y   E
```

3.3 List all items that were placed under "?x"

SPARQL Exercise 2

1. You have the following triple store <http://ImaginaryTripleStore.com>:

A X B
C X B
B X D
B Y E
F Y B
F X D
F Y E
E X D
G X E

2. You have the following SPARQL statement:

```
SELECT ?s
FROM
<http://ImaginaryTripleStore.com>
WHERE
{?s Y ?o .
?o X D}
```

3. Solution

3.1 Look at the 2nd triple from the where clause first. Find all triplets that use X as a predicate where the object is D (fulfils "?o X D"). Place them below the where clause:

```
?s  Y  ?o  .  ?o  X  D
              B   X  D
              F   X  D
              E   X  D
```

3.2 The subject of the above triplets are: B, F, and E. Now, look at the first triplet of the where clause. Look for all triplets that have an object that is B, F, or E and that have a predicate Y (fulfils "?s Y ?o"). Place those triplets below the where clause:

```
?s  Y  ?o  .  ?o  X  D
F   Y   B     B   X  D
              F   X  D
F   Y   E     E   X  D
B   Y   E     E   X  D
```

3.3 List all items that were placed under "?s"

Some Links

- http://dbpedia.org/sparql
- http://lod.openlinksw.com/sparql/
- http://librdf.org/query

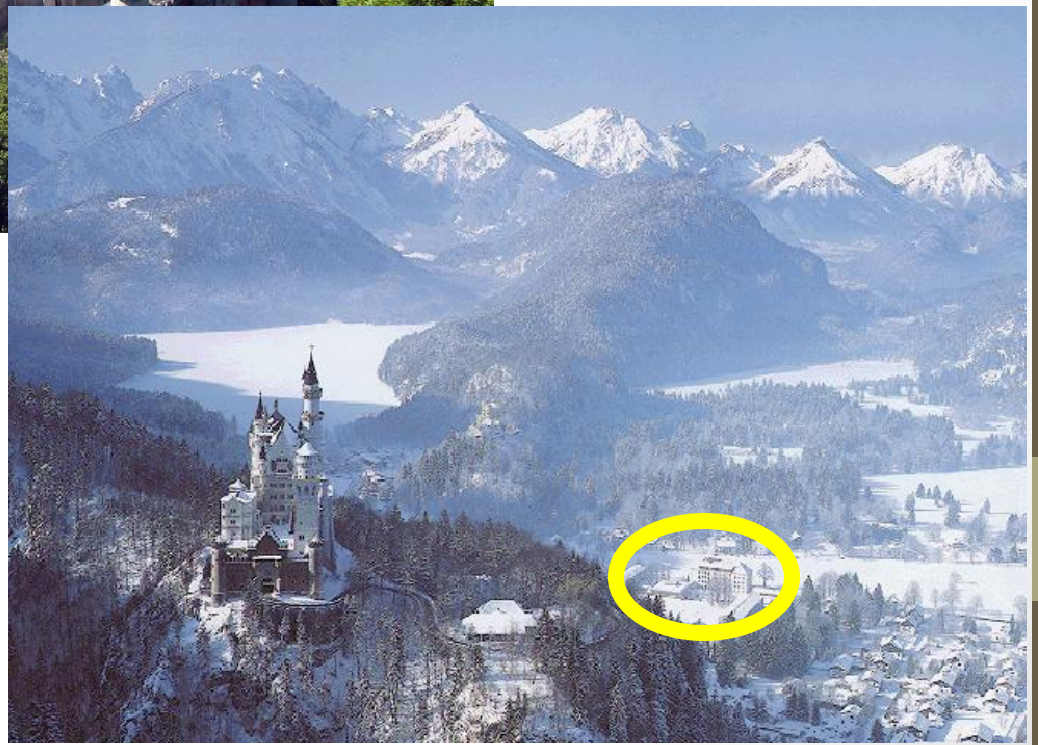
- Online tutorial: http://www.cambridgesemantics.com/semantic-university/sparql-by-example
-
- http://sparql.org/sparql.html
- http://demo.openlinksw.com/sparql

Introduction to SPARQL

Assignment

- Take Quiz 10c. This is the real Graph Quiz: Opens after 6:30 PM closes on **Wednesday** at 11:00 PM.
  - htt
- Take Quiz 10d. This is the real SPARQL Quiz: Opens after 6:30 PM closes on **Wednesday** at 11:00 PM. Use SPARQLExercises.pdf as a guide for answering the more difficult questions.
  - htt
- Review Statistics in Preparation of the 2nd Quarter
  - Ponder the pdf titled: Predictive Anecdotes
  - Read these papers posted on the Catalyst site:
    - orange cars.pdf
    - SSRN-id1045281.pdf
    - dataminejune_2000.pdf
  - Read some of this book on R and statistics and do some of the exercises:
    - http://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf
- There is nothing to submit!

This Summer's Vacation Spot



Thank You!

After hours with Matt!

Introduction to Data Science