Application Design Using Java

Lecture 22

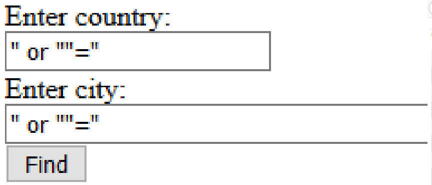
SQL Injection

- Code injection technique that might destroy your database
- One of the most common Web hacking techniques
- Malicious code placed in SQL statements via Web page input
- Prevention
  - Limit permissions of the MySQL user used by the application to access the database
  - Input validation, don’t trust any user input
    - Regular expressions as whitelists for structured data
    - For fixed sets of values (drop-down lists, radio buttons, etc.), determine which value is returned. The input data should match one of the offered options exactly.
  - SQL parameters
  - Stored procedures
  - Use character-escaping functions for user-supplied input

SQL Injection Examples

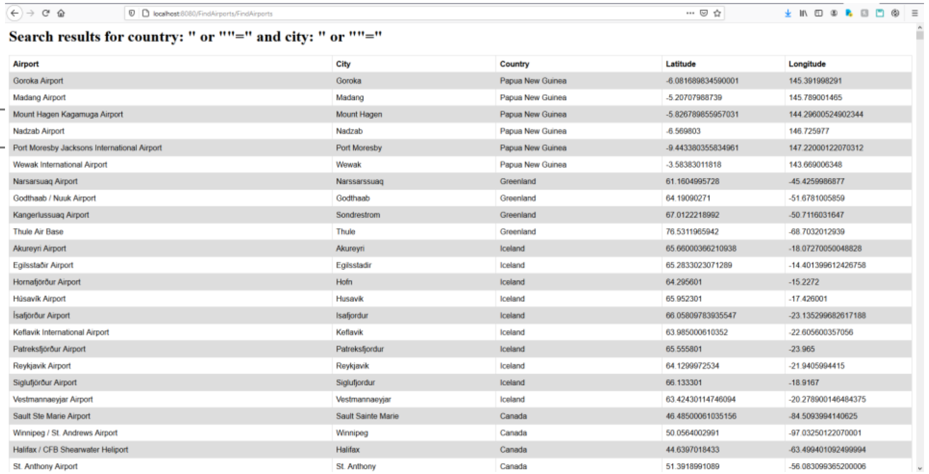
- Getting access to the entire table, all rows and columns. Imagine it were a table with all users, their passwords, etc.

Executed the following SQL statement:
SELECT airport, city, country, latitude, longitude FROM advjava.airports WHERE country = "" or ""="" AND city = "" or ""=""



Enter country:
" or ""="

Enter city:
" or ""="

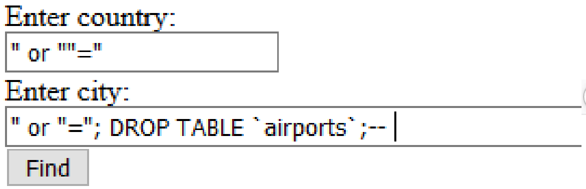


Search results for country: " or ""=" and city: " or ""="

Airport	City	Country	Latitude	Longitude
Goroka Airport	Goroka	Papua New Guinea	-6.081689834590001	145.391998291
Madang Airport	Madang	Papua New Guinea	-5.20707988739	145.789001465
Mount Hagen Kagamuga Airport	Mount Hagen	Papua New Guinea	-5.826789855957031	144.29600524902344
Nadzab Airport	Nadzab	Papua New Guinea	-6.569803	146.725977
Port Moresby Jacksons International Airport	Port Moresby	Papua New Guinea	-9.443380355834961	147.22000122070312
Wewak International Airport	Wewak	Papua New Guinea	-3.58383011818	143.669006348
Narsarsuaq Airport	Narssarssuaq	Greenland	61.1604995728	-45.4259986877
Godthaab / Nuuk Airport	Godthaab	Greenland	64.19090271	-51.6781005859
Kangerlussuaq Airport	Sondrestrom	Greenland	67.0122218992	-50.7116031647
Thule Air Base	Thule	Greenland	76.5311965942	-68.7032012939
Akureyri Airport	Akureyri	Iceland	65.66000366210938	-18.07270050048828
Egilsstaðir Airport	Egilsstadir	Iceland	65.2833023071289	-14.401399612426758
Hornafjörður Airport	Hofn	Iceland	64.295601	-15.2272
Húsavík Airport	Husavik	Iceland	65.952301	-17.426001
Ísafjörður Airport	Isafjordur	Iceland	66.05809783935547	-23.135299682617188
Keflavik International Airport	Keflavik	Iceland	63.985000610352	-22.605600357056
Patreksfjörður Airport	Patreksfjordur	Iceland	65.555801	-23.965
Reykjavik Airport	Reykjavik	Iceland	64.1299972534	-21.9405994415
Siglufjörður Airport	Siglufjordur	Iceland	66.133301	-18.9167
Vestmannaeyjar Airport	Vestmannaeyjar	Iceland	63.42430114746094	-20.27890014648437 5
Sault Ste Marie Airport	Sault Sainte Marie	Canada	46.48500061035156	-84.5093994140625
Winnipeg / St. Andrews Airport	Winnipeg	Canada	50.0564002991	-97.03250122070001
Halifax / CFB Shearwater Heliport	Halifax	Canada	44.6397018433	-63.499401092499994
St. Anthony Airport	St. Anthony	Canada	51.3918991089	-56.083099365200006

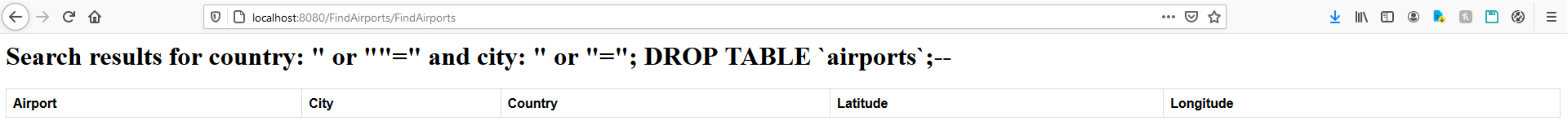
- Deleting the table.

Executed the following SQL statement:
SELECT airport, city, country, latitude, longitude FROM advjava.airports WHERE country = "" or ""="" AND city = "" or "=";
DROP TABLE `airports`;-- "



Enter country:
" or ""="

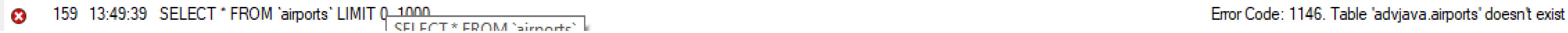
Enter city:
" or "="; DROP TABLE `airports`;--



Search results for country: " or ""=" and city: " or "="; DROP TABLE `airports`;--

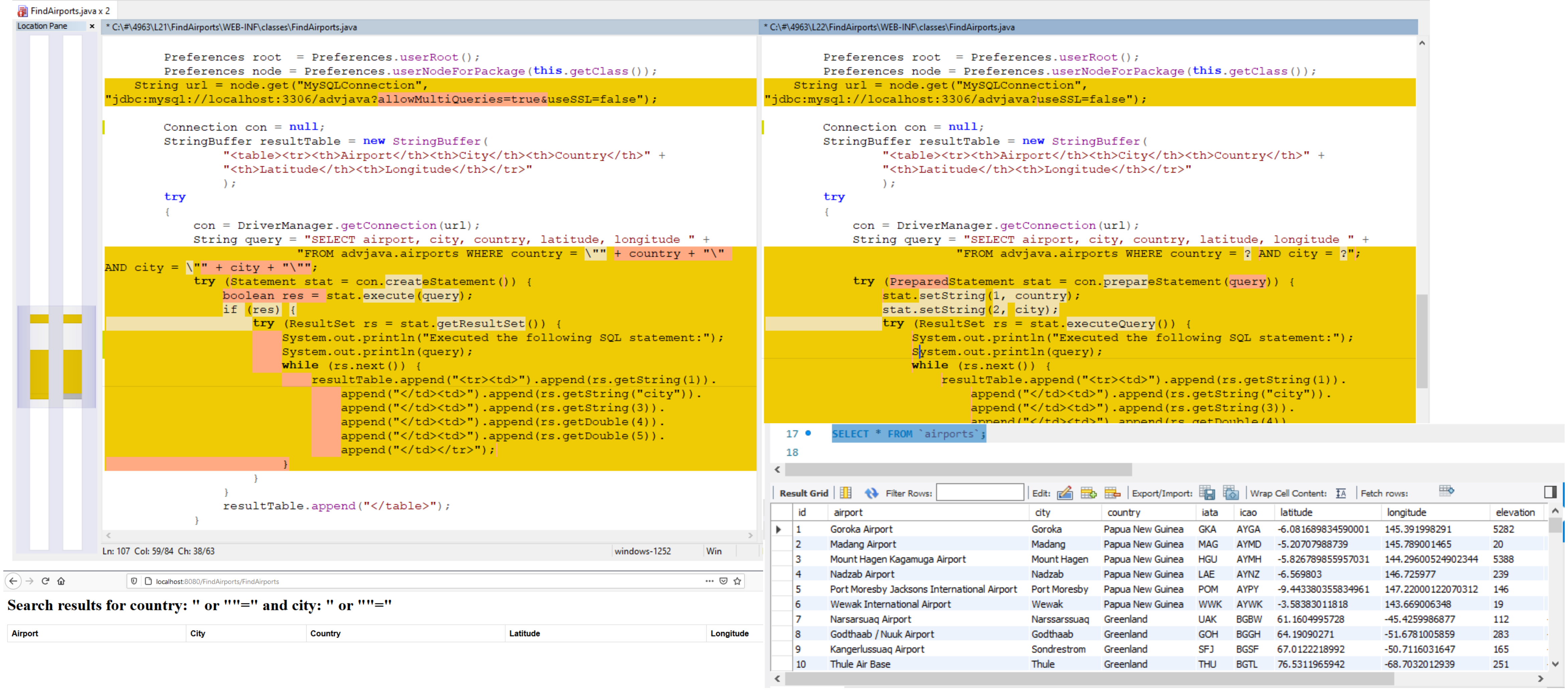
Airport	City	Country	Latitude	Longitude

SELECT * FROM `airports`;



```
159   13:49:39   SELECT * FROM `airports` LIMIT 0, 1000        Error Code: 1146. Table 'advjava.airports' doesn't exist
```

SQL Injection Eliminated



* C:\#\4963\L21\FindAirports\WEB-INF\classes\FindAirports.java

```
        Preferences root  = Preferences.userRoot();
        Preferences node = Preferences.userNodeForPackage(this.getClass());
    String url = node.get("MySQLConnection",
"jdbc:mysql://localhost:3306/advjava?allowMultiQueries=true&useSSL=false");

        Connection con = null;
        StringBuffer resultTable = new StringBuffer(
                "<table><tr><th>Airport</th><th>City</th><th>Country</th>" +
                "<th>Latitude</th><th>Longitude</th></tr>"
                );
        try
        {
            con = DriverManager.getConnection(url);
            String query = "SELECT airport, city, country, latitude, longitude " +
                          "FROM advjava.airports WHERE country = \"" + country + "\"
AND city = \"" + city + "\"";
            try (Statement stat = con.createStatement()) {
                boolean res = stat.execute(query);
                if (res) {
                    try (ResultSet rs = stat.getResultSet()) {
                    System.out.println("Executed the following SQL statement:");
                    System.out.println(query);
                    while (rs.next()) {
                        resultTable.append("<tr><td>").append(rs.getString(1)).
                            append("</td><td>").append(rs.getString("city")).
                            append("</td><td>").append(rs.getString(3)).
                            append("</td><td>").append(rs.getDouble(4)).
                            append("</td><td>").append(rs.getDouble(5)).
                            append("</td></tr>");
                    }
                }
            }
            resultTable.append("</table>");
        }
```

Ln: 107 Col: 59/84 Ch: 38/63 windows-1252 Win

* C:\#\4963\L22\FindAirports\WEB-INF\classes\FindAirports.java

```
        Preferences root  = Preferences.userRoot();
        Preferences node = Preferences.userNodeForPackage(this.getClass());
    String url = node.get("MySQLConnection",
"jdbc:mysql://localhost:3306/advjava?useSSL=false");

        Connection con = null;
        StringBuffer resultTable = new StringBuffer(
                "<table><tr><th>Airport</th><th>City</th><th>Country</th>" +
                "<th>Latitude</th><th>Longitude</th></tr>"
                );
        try
        {
            con = DriverManager.getConnection(url);
            String query = "SELECT airport, city, country, latitude, longitude " +
                          "FROM advjava.airports WHERE country = ? AND city = ?";

            try (PreparedStatement stat = con.prepareStatement(query)) {
                stat.setString(1, country);
                stat.setString(2, city);
                try (ResultSet rs = stat.executeQuery()) {
                    System.out.println("Executed the following SQL statement:");
                    System.out.println(query);
                    while (rs.next()) {
                        resultTable.append("<tr><td>").append(rs.getString(1)).
                            append("</td><td>").append(rs.getString("city")).
                            append("</td><td>").append(rs.getString(3)).
                            append("</td><td>").append(rs.getDouble(4)).
```

localhost:8080/FindAirports/FindAirports

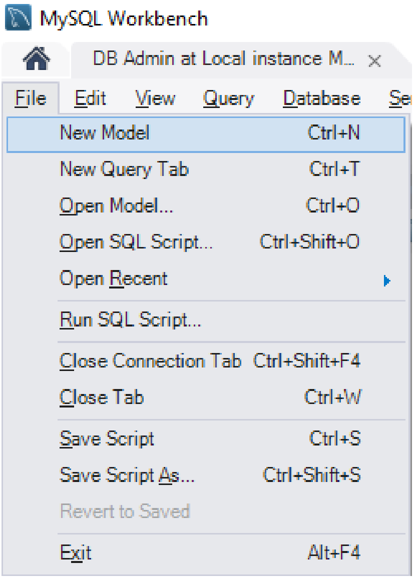
Search results for country: " or ""=" and city: " or ""="

Airport	City	Country	Latitude	Longitude

```
17 • SELECT * FROM `airports`;
18
```

id	airport	city	country	iata	icao	latitude	longitude	elevation
1	Goroka Airport	Goroka	Papua New Guinea	GKA	AYGA	-6.081689834590001	145.391998291	5282
2	Madang Airport	Madang	Papua New Guinea	MAG	AYMD	-5.20707988739	145.789001465	20
3	Mount Hagen Kagamuga Airport	Mount Hagen	Papua New Guinea	HGU	AYMH	-5.826789855957031	144.29600524902344	5388
4	Nadzab Airport	Nadzab	Papua New Guinea	LAE	AYNZ	-6.569803	146.725977	239
5	Port Moresby Jacksons International Airport	Port Moresby	Papua New Guinea	POM	AYPY	-9.443380355834961	147.22000122070312	146
6	Wewak International Airport	Wewak	Papua New Guinea	WWK	AYWK	-3.58383011818	143.669006348	19
7	Narsarsuaq Airport	Narssarssuaq	Greenland	UAK	BGBW	61.1604995728	-45.4259986877	112
8	Godthaab / Nuuk Airport	Godthaab	Greenland	GOH	BGGH	64.19090271	-51.6781005859	283
9	Kangerlussuaq Airport	Sondrestrom	Greenland	SFJ	BGSF	67.0122218992	-50.7116031647	165
10	Thule Air Base	Thule	Greenland	THU	BGTL	76.5311965942	-68.7032012939	251

Designing Database Schema



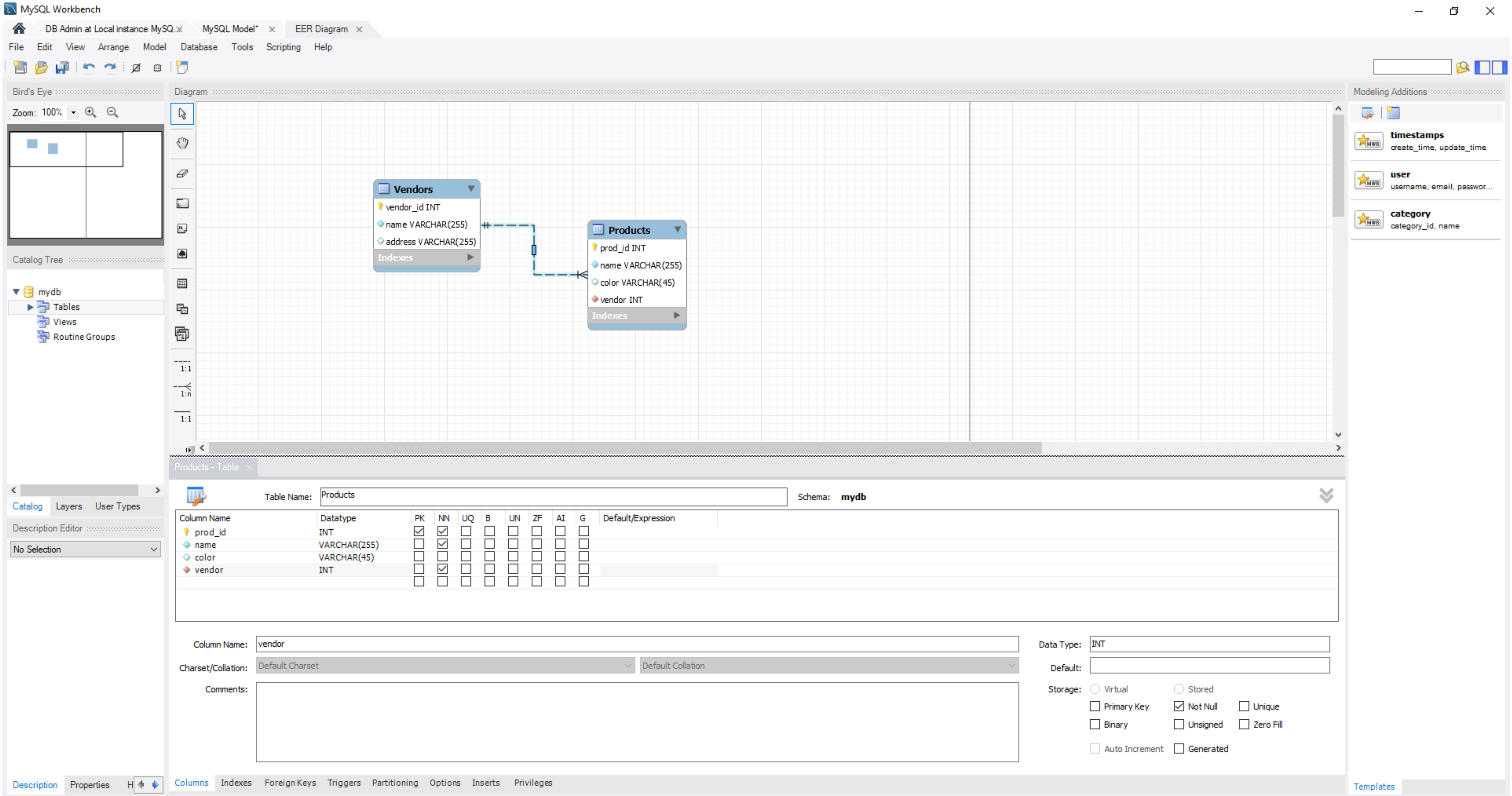


Table Name: Products

Schema: mydb

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
prod_id	INT	☑	☑	□	□	□	□	□	□	
name	VARCHAR(255)	□	☑	□	□	□	□	□	□	
color	VARCHAR(45)	□	□	□	□	□	□	□	□	
vendor	INT	□	☑	□	□	□	□	□	□	
		□	□	□	□	□	□	□	□	

Column Name: vendor

Data Type: INT

Charset/Collation: Default Charset Default Collation

Default:

Comments:

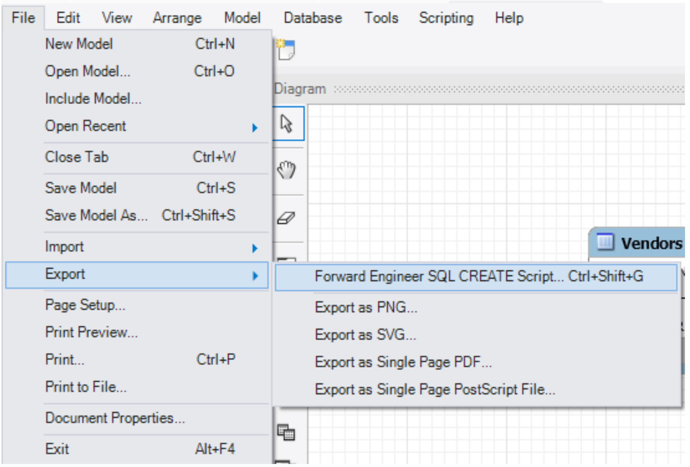
Storage: ○ Virtual ○ Stored

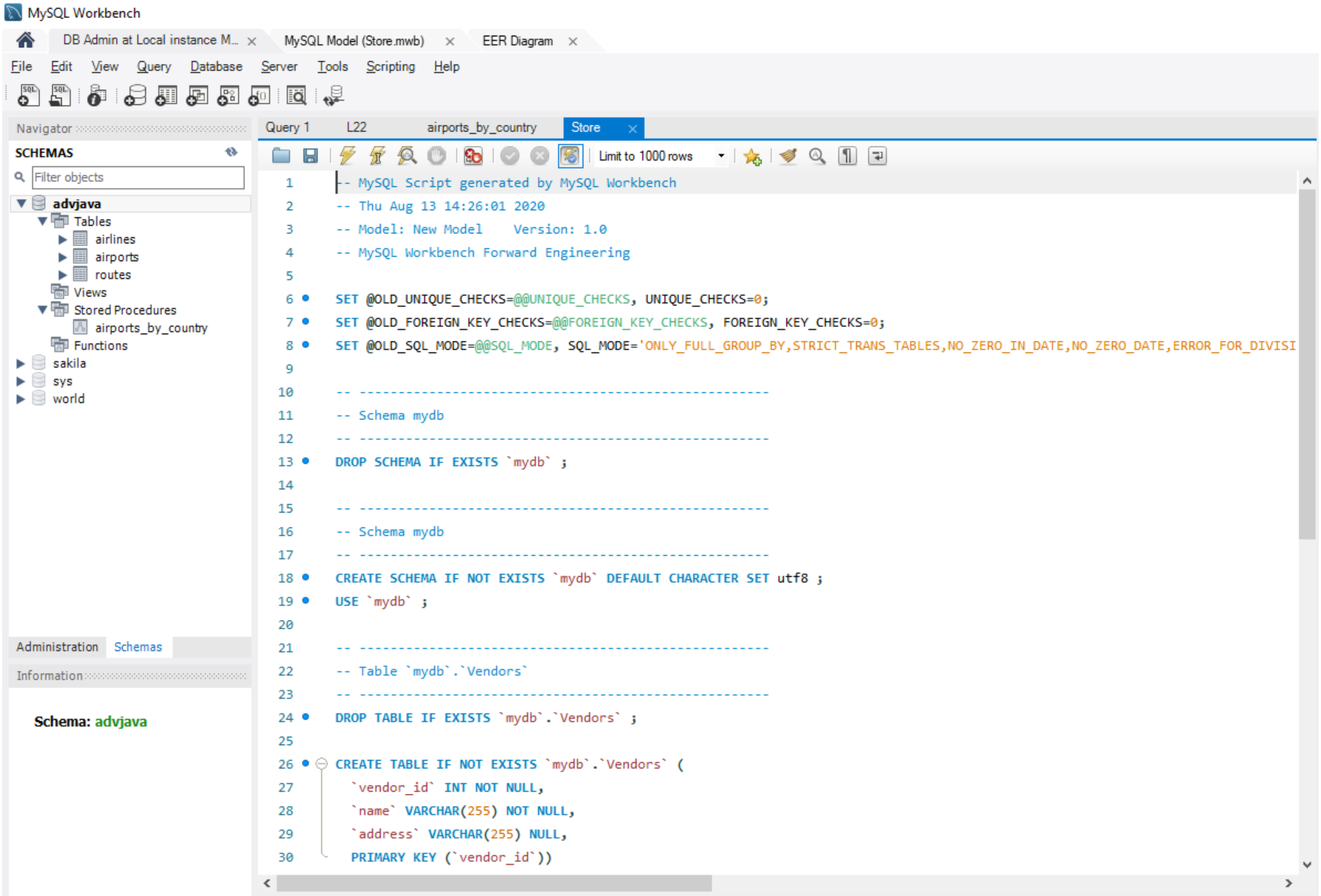
□ Primary Key ☑ Not Null □ Unique

□ Binary □ Unsigned □ Zero Fill

□ Auto Increment □ Generated

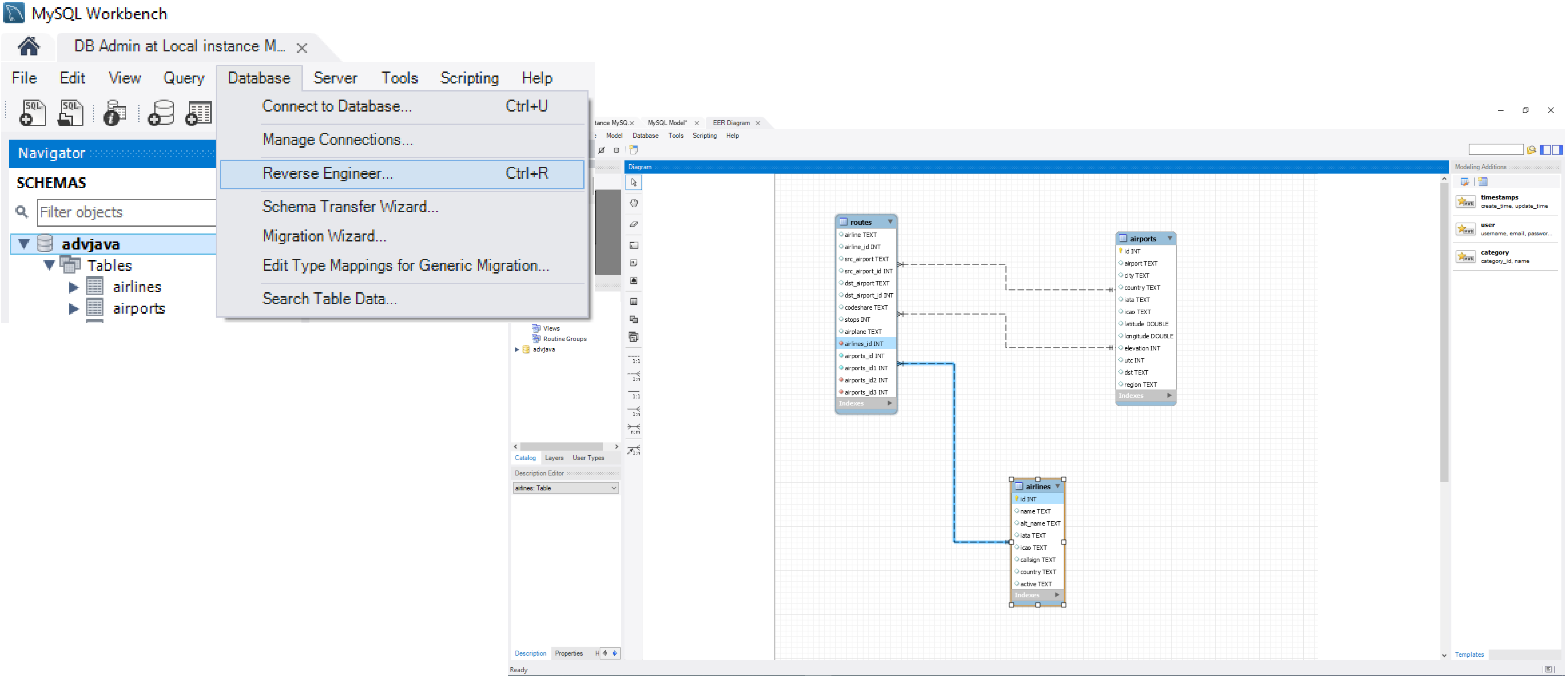
Creating Database from Model





```
-- MySQL Script generated by MySQL Workbench
-- Thu Aug 13 14:26:01 2020
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISI

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `mydb` ;

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`Vendors`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Vendors` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Vendors` (
  `vendor_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `address` VARCHAR(255) NULL,
  PRIMARY KEY (`vendor_id`))
```

Reverse Engineering a Database

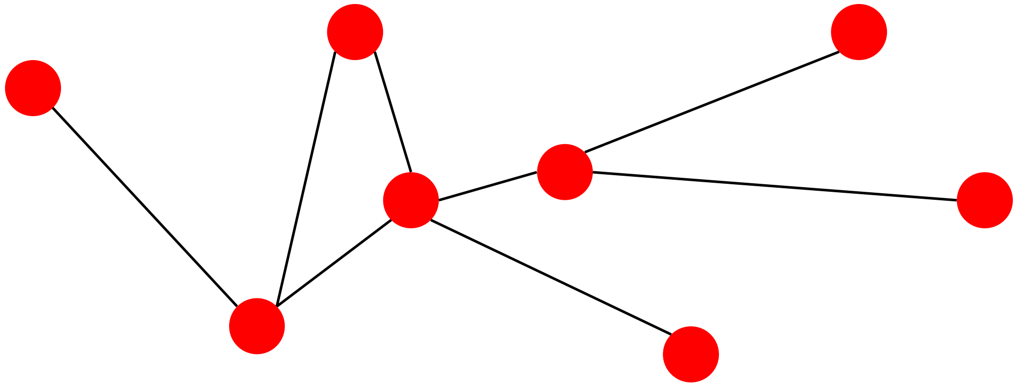


Section 2

CSCI-4250 Frontiers of Network Science

Networks and graphs

COMPONENTS OF A COMPLEX SYSTEM



▪ **components**:	nodes, vertices	N
▪ **interactions**:	links, edges	L
▪ **system**:	network, graph	(N,L)

NETWORKS OR GRAPHS?

network often refers to real systems

- www,
- social network
- metabolic network.

Language: (Network, node, link)

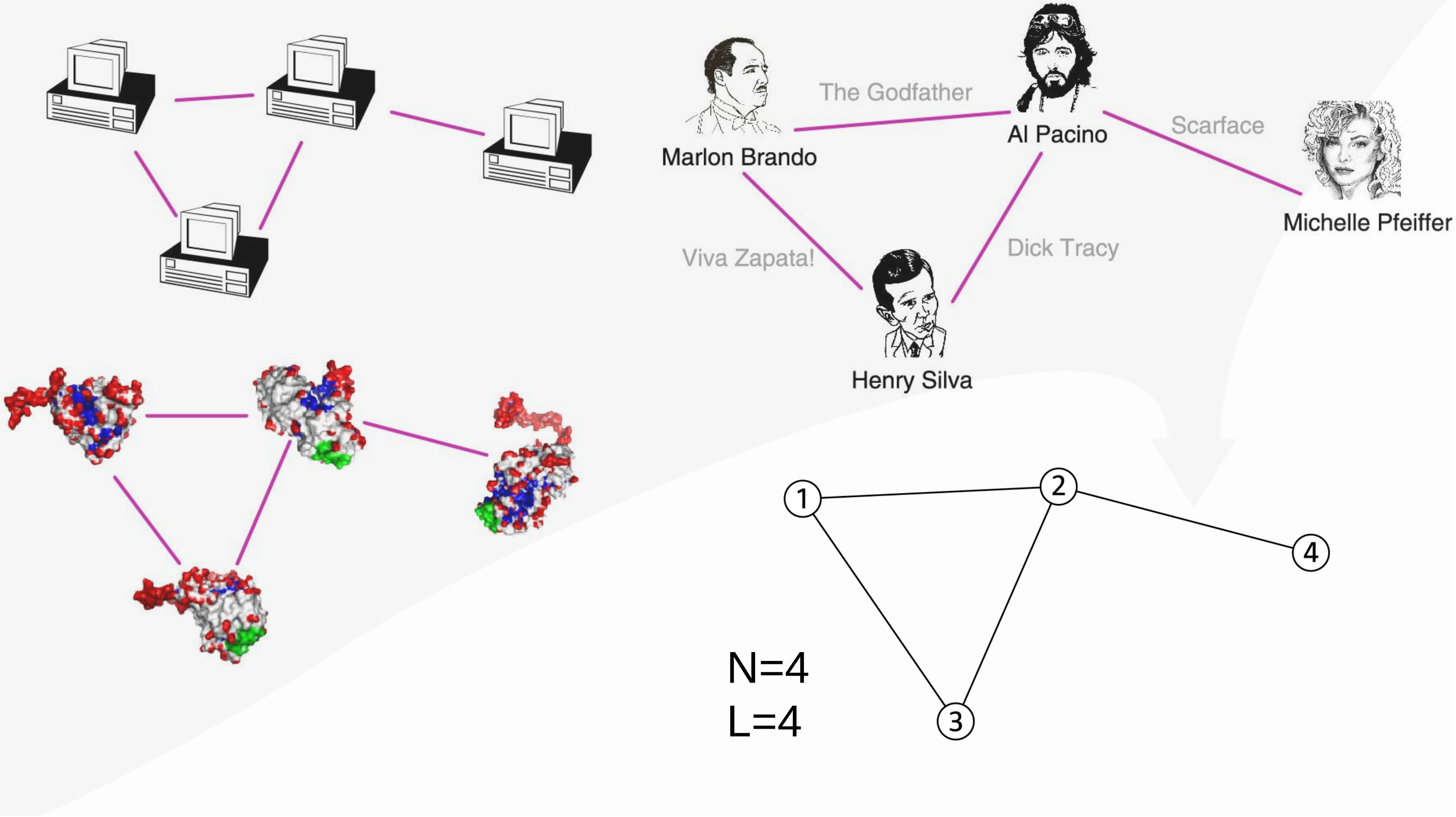
graph: mathematical representation of a network

- web graph,
- social graph (a Facebook term)

Language: (Graph, vertex, edge)

We will try to make this distinction whenever it is appropriate, but in most cases we will use the two terms interchangeably.

A COMMON LANGUAGE



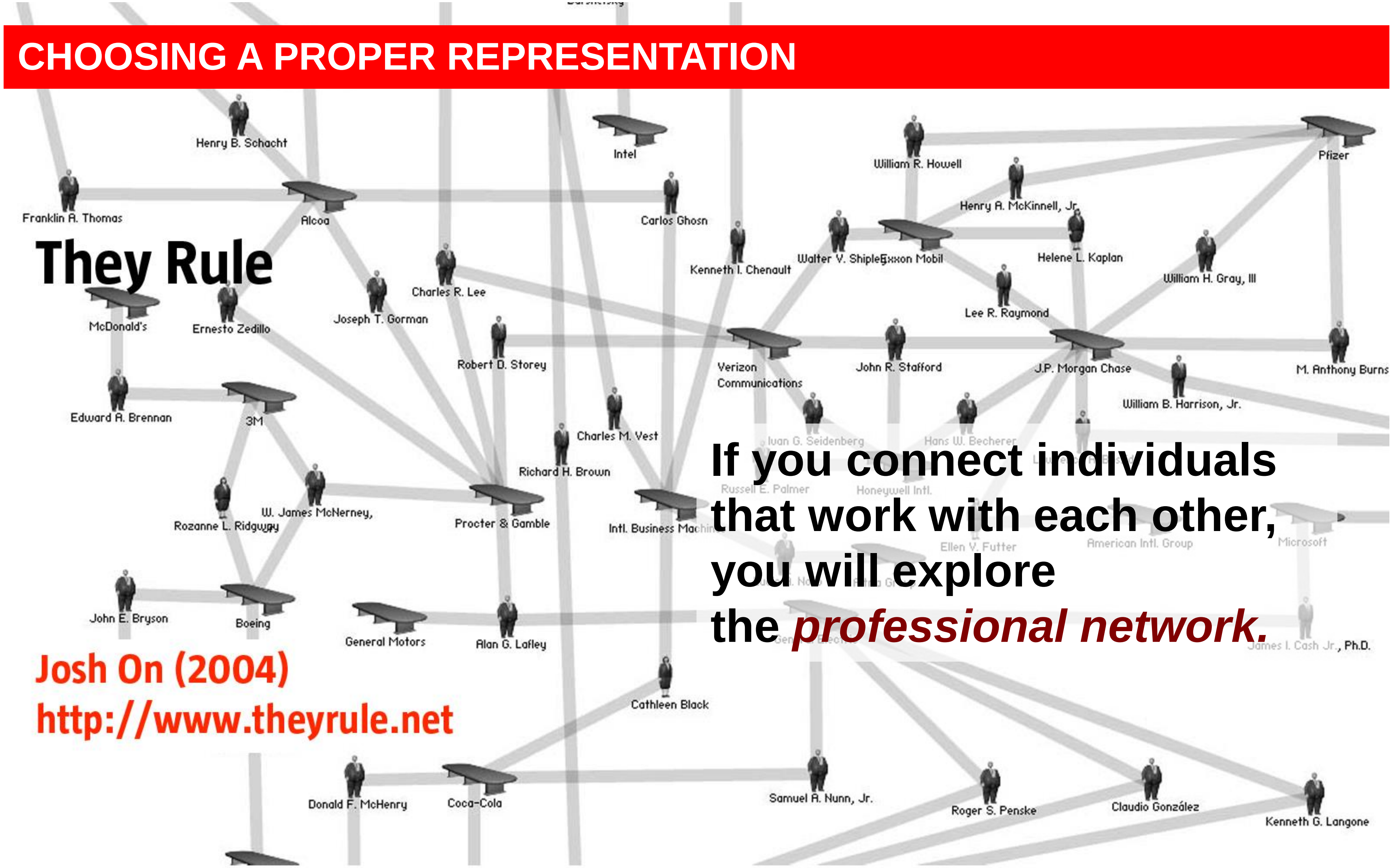
CHOOSING A PROPER REPRESENTATION

The choice of the proper network representation determines our ability to use network theory successfully.

In some cases there is a unique, unambiguous representation.
In other cases, the representation is by no means unique.

For example, the way we assign the links between a group of individuals will determine the nature of the question we can study.

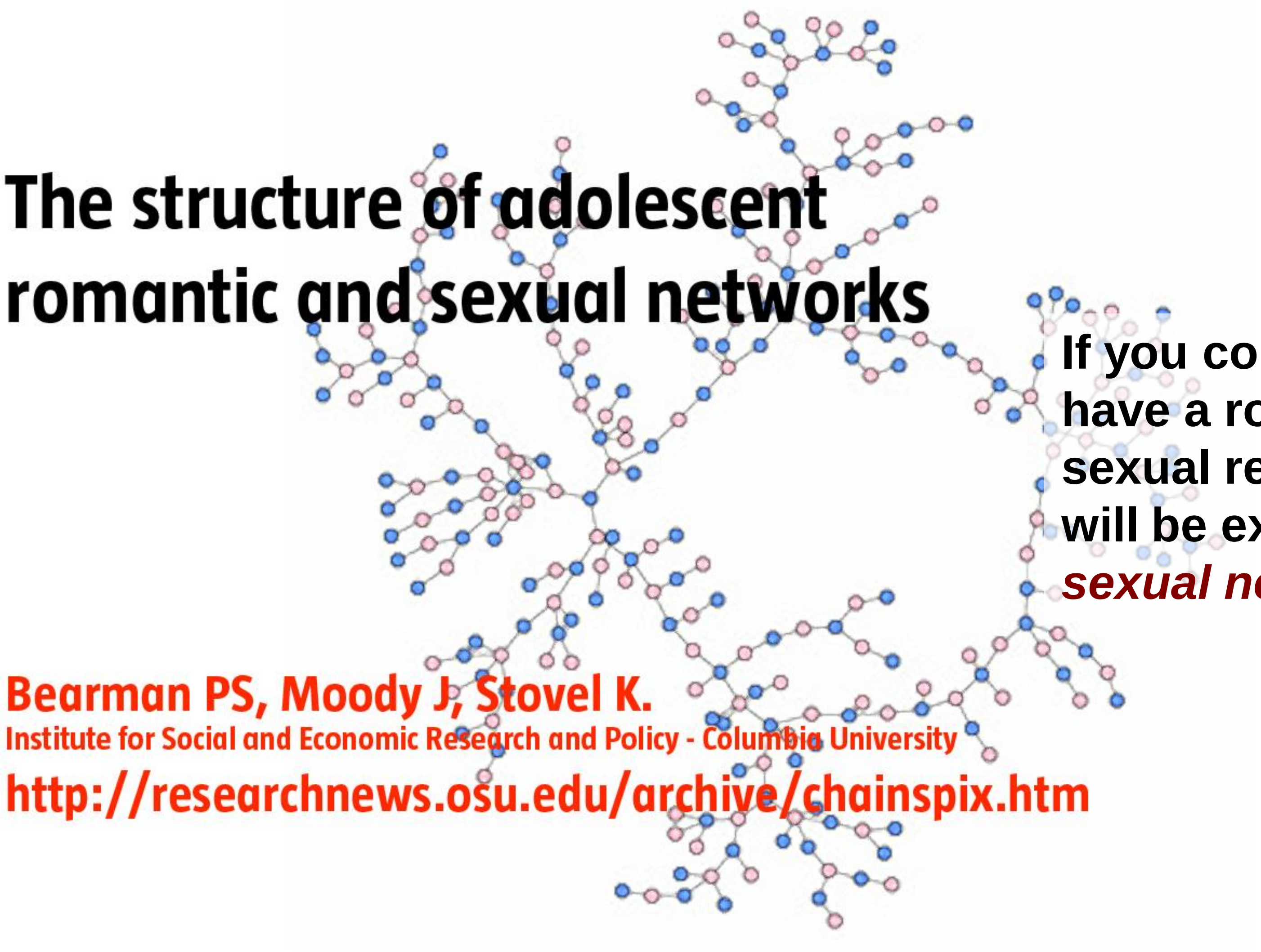
CHOOSING A PROPER REPRESENTATION



Josh On (2004)
http://www.theyrule.net

If you connect individuals that work with each other, you will explore the ***professional network.***

CHOOSING A PROPER REPRESENTATION



The structure of adolescent romantic and sexual networks

Bearman PS, Moody J, Stovel K.
Institute for Social and Economic Research and Policy - Columbia University
http://researchnews.osu.edu/archive/chainspix.htm

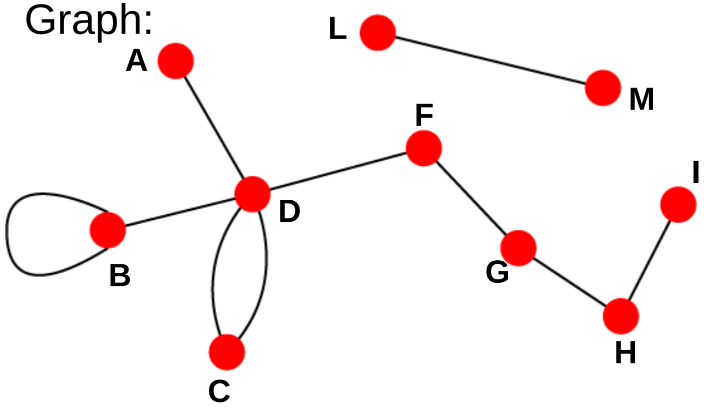
If you connect those that have a romantic and sexual relationship, you will be exploring the *sexual networks*.

UNDIRECTED VS. DIRECTED NETWORKS

Undirected

Links: undirected (*symmetrical*)

Graph:

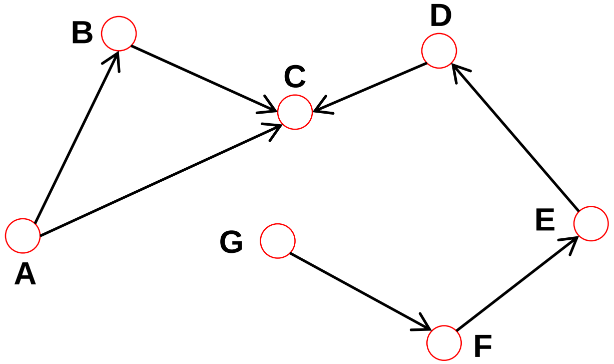


Undirected links :
coauthorship links
Actor network
protein interactions

Directed

Links: directed (*arcs*).

Digraph = directed graph:



An undirected link is the superposition of two opposite directed links.

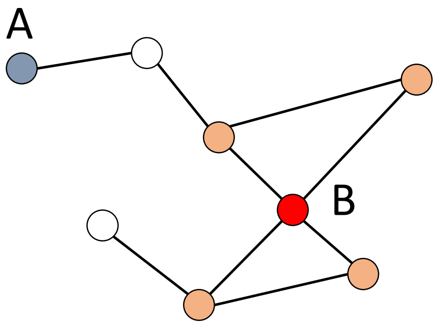
Directed links :
URLs on the www
phone calls
metabolic reactions

Section 2.3

Degree, Average Degree and Degree Distribution

NODE DEGREES

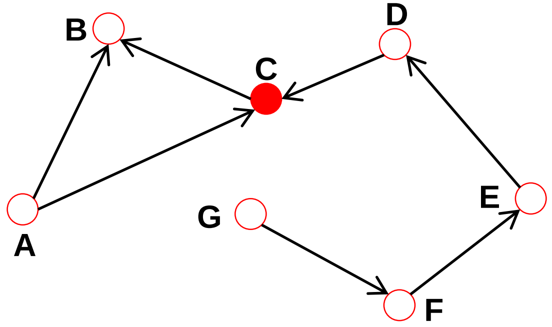
Undirected



Node degree: the number of links connected to the node.

$$k_A = 1 \qquad k_B = 4$$

Directed



In *directed networks* we can define an in-degree and out-degree.

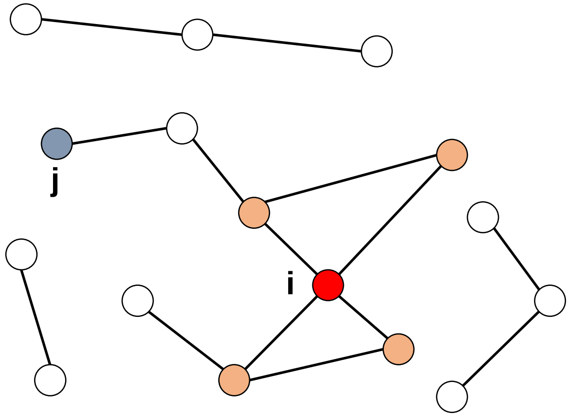
The (total) degree is the sum of in- and out-degree.

$$k_C^{in} = 2 \qquad k_C^{out} = 1 \qquad k_C = 3$$

Source: a node with $k^{in} = 0$; Sink: a node with $k^{out} = 0$.

AVERAGE DEGREE

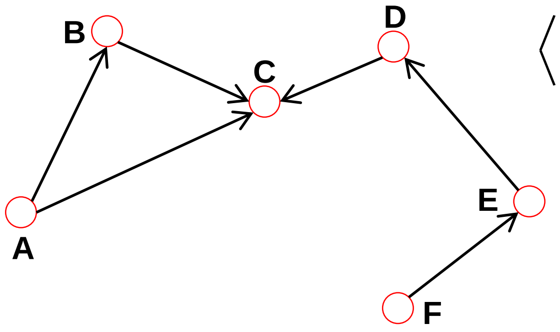
Undirected



$$\langle k \rangle \equiv \frac{1}{N}\sum_{i=1}^{N} k_i \qquad \langle k \rangle \,\mathrm{o}\, \frac{2L}{N}$$

N – the number of nodes in the graph

Directed



$$\langle k^{in} \rangle \equiv \frac{1}{N}\sum_{i=1}^{N} k_i^{in}, \quad \langle k^{out} \rangle \equiv \frac{1}{N}\sum_{i=1}^{N} k_i^{out}, \quad \langle k^{in} \rangle = \langle k^{out} \rangle$$

$$\langle k \rangle \,\mathrm{o}\, \frac{L}{N}$$

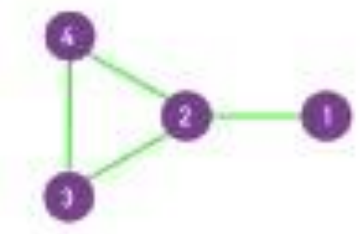
DEGREE DISTRIBUTION

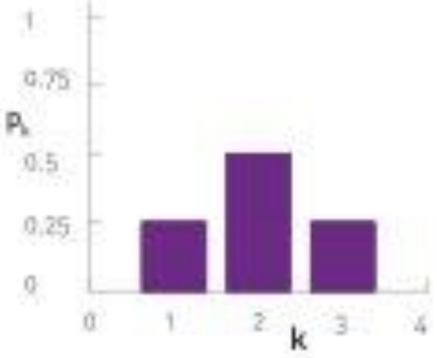
Degree distribution

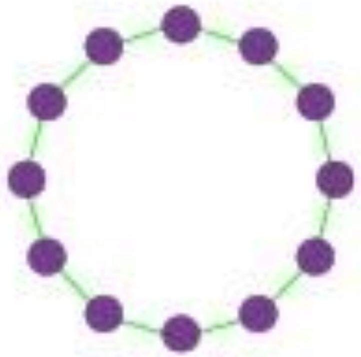
P(k): probability that a

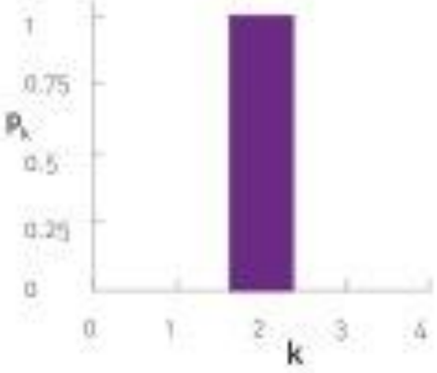
randomly chosen node

has degree *k*









N_k = # nodes with degree k

$P(k) = N_k / N$ ❾ plot

DEGREE DISTRIBUTION

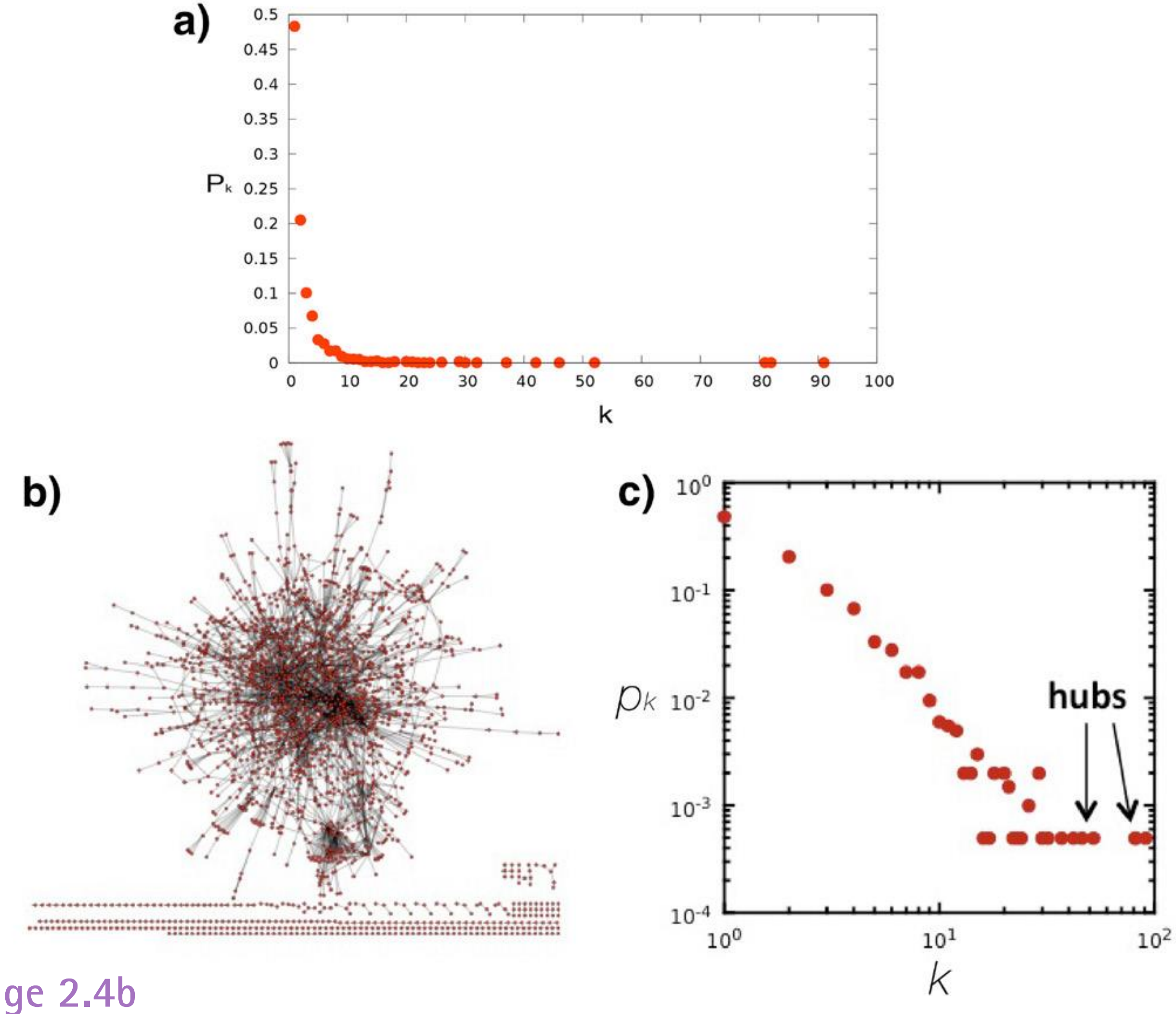
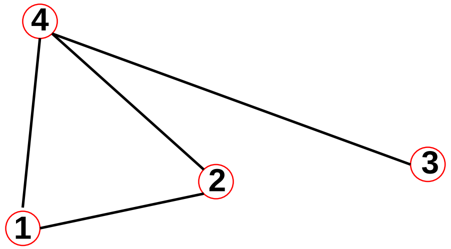


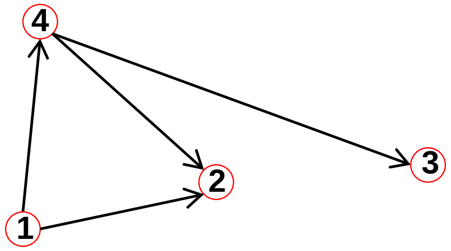
Image 2.4b

Section 2.4

Adjacency matrix

ADJACENCY MATRIX





A_{ij}=1 if there is a link between node *i* and *j*

A_{ij}=0 if nodes *i* and *j* are not connected to each other.

$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \qquad A_{ij} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

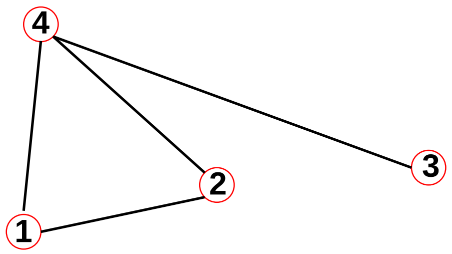
Note that for a directed graph (right) the matrix is not symmetric.

$A_{ij} = 1$ if there is a link pointing from node *j* and *i*

$A_{ij} = 0$ if there is no link pointing from *j* to *i*.

ADJACENCY MATRIX AND NODE DEGREES

Undirected



$$A_{ij}=\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

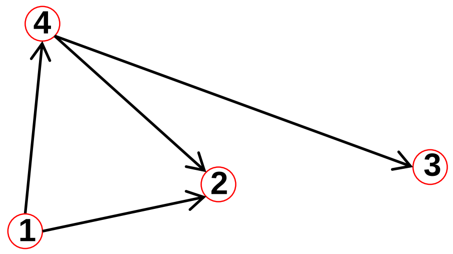
$$A_{ij}=A_{ji}$$
$$A_{ii}=0$$

$$k_i=\sum_{j=1}^{N}A_{ij}$$

$$k_j=\sum_{i=1}^{N}A_{ij}$$

$$L=\frac{1}{2}\sum_{i=1}^{N}k_i=\frac{1}{2}\sum_{ij}^{N}A_{ij}$$

Directed



$$A_{ij}=\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

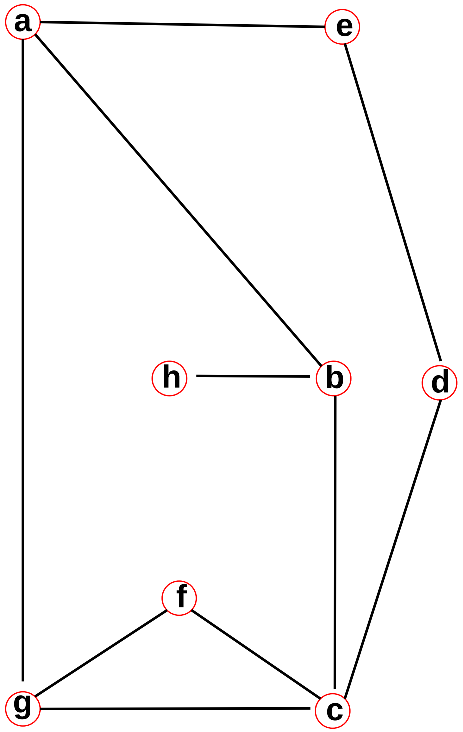
$$A_{ij}\neq A_{ji}$$
$$A_{ii}=0$$

$$k_i^{in}=\sum_{j=1}^{N}A_{ij}$$

$$k_j^{out}=\sum_{i=1}^{N}A_{ij}$$

$$L=\sum_{i=1}^{N}k_i^{in}=\sum_{j=1}^{N}k_j^{out}=\sum_{i,j}^{N}A_{ij}$$

ADJACENCY MATRIX



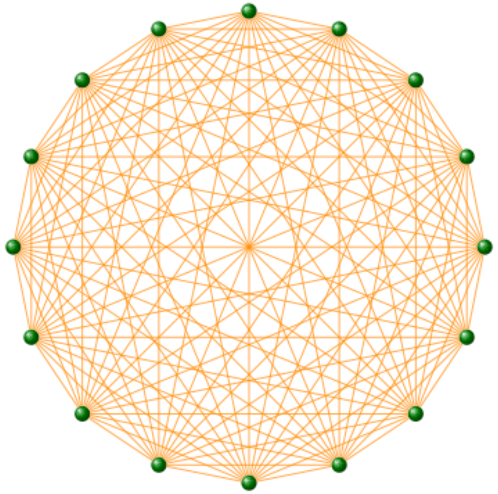
	a	b	c	d	e	f	g	h
a	0	1	0	0	1	0	1	0
b	1	0	1	0	0	0	0	1
c	0	1	0	1	0	1	1	0
d	0	0	1	0	1	0	0	0
e	1	0	0	1	0	0	0	0
f	0	0	1	0	0	0	1	0
g	1	0	1	0	0	0	0	0
h	0	1	0	0	0	0	0	0

Section 4

Real networks are sparse

COMPLETE GRAPH

The maximum number of links a network of N nodes can have is: $L_{\max} = \binom{N}{2} = \frac{N(N-1)}{2}$



A graph with degree L=L$_{max}$ is called a complete graph, and its average degree is **<k>=N-1**

REAL NETWORKS ARE SPARSE

Most networks observed in real systems are sparse:

$$L \ll L_{max}$$

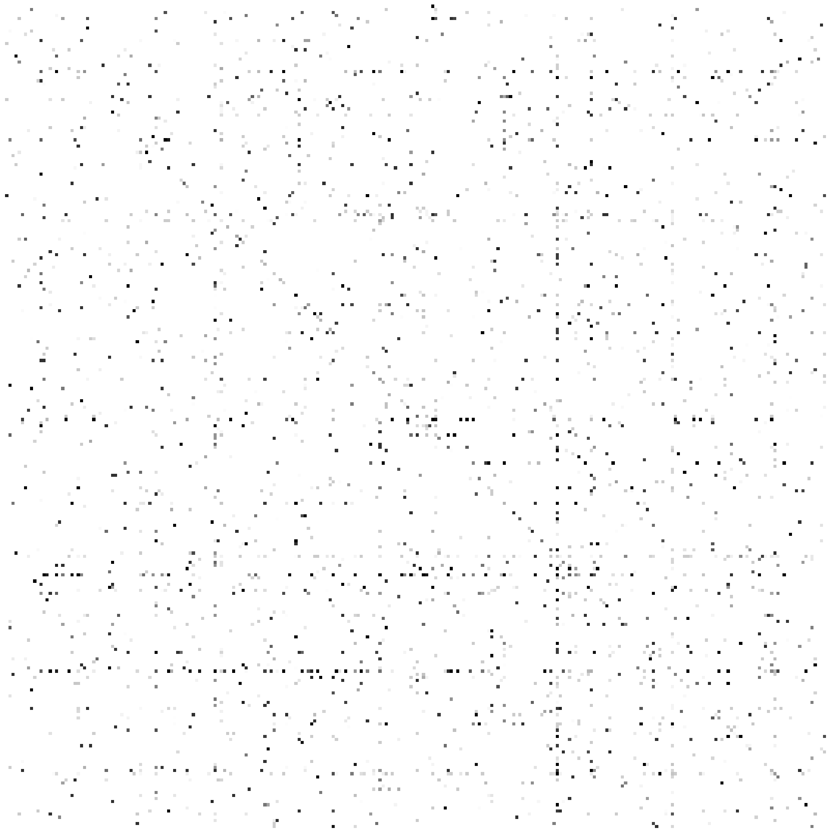
or

$$\langle k \rangle \ll N-1.$$

WWW (ND Sample):	N=325,729;	$L=1.4\ 10^6$	$L_{max}=10^{12}$	<k>=4.51
Protein (*S. Cerevisiae*):	N= 1,870;	L=4,470	$L_{max}=10^7$	<k>=2.39
Coauthorship (Math):	N= 70,975;	$L=2\ 10^5$	$L_{max}=3\ 10^{10}$	<k>=3.9
Movie Actors:	N=212,250;	$L=6\ 10^6$	$L_{max}=1.8\ 10^{13}$	<k>=28.78

(Source: Albert, Barabasi, RMP2002)

ADJACENCY MATRICES ARE SPARSE



Section 2.6

WEIGHTED AND UNWEIGHTED NETWORKS

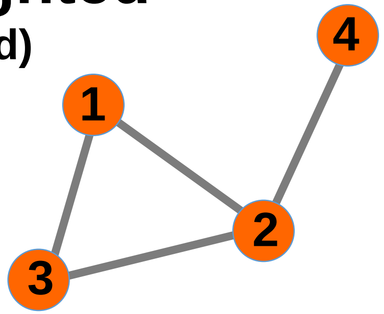
WEIGHTED AND UNWEIGHTED NETWORKS

$$A_{ij} = w_{ij}$$

GRAPHOLOGY *1*

Unweighted

(undirected)



$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

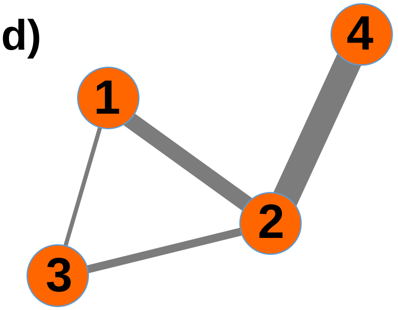
$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$L = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij} \qquad <k> = \frac{2L}{N}$$

protein-protein interactions, www

Weighted

(undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

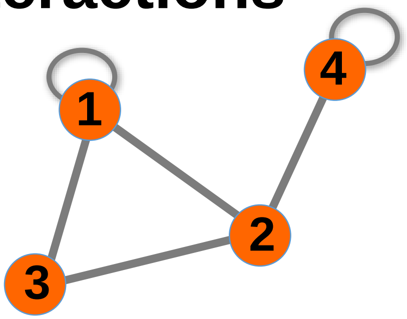
$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$L = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad <k> = \frac{2L}{N}$$

Call Graph, metabolic networks

GRAPHOLOGY *2*

Self-interactions



$$A_{ij} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

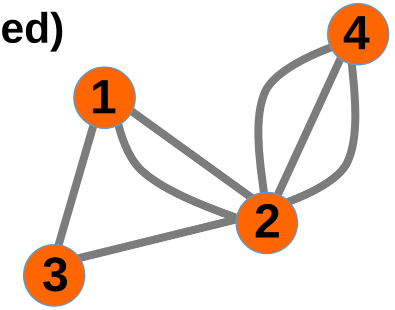
$$A_{ii} \neq 0 \qquad A_{ij} = A_{ji}$$

$$L = \frac{1}{2}\sum_{i,j=1,i\neq j}^{N} A_{ij} + \sum_{i=1}^{N} A_{ii} \qquad ?$$

Protein interaction network, www

Multigraph

(undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

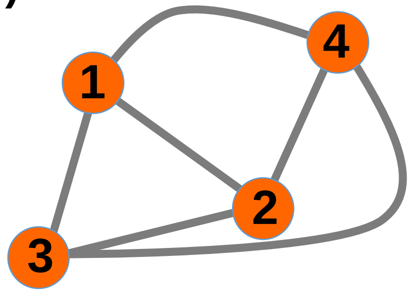
$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

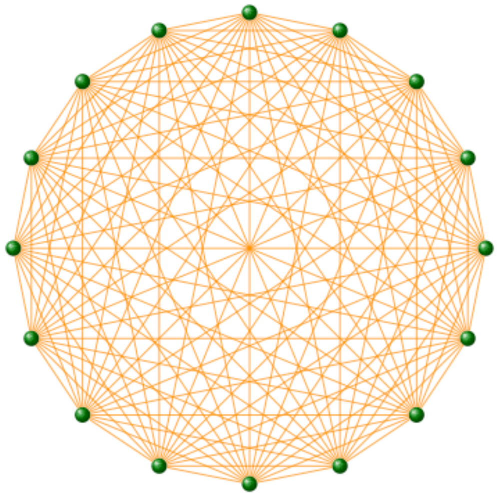
$$L = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad <k>= \frac{2L}{N}$$

Social networks, collaboration networks

Complete Graph

(undirected)





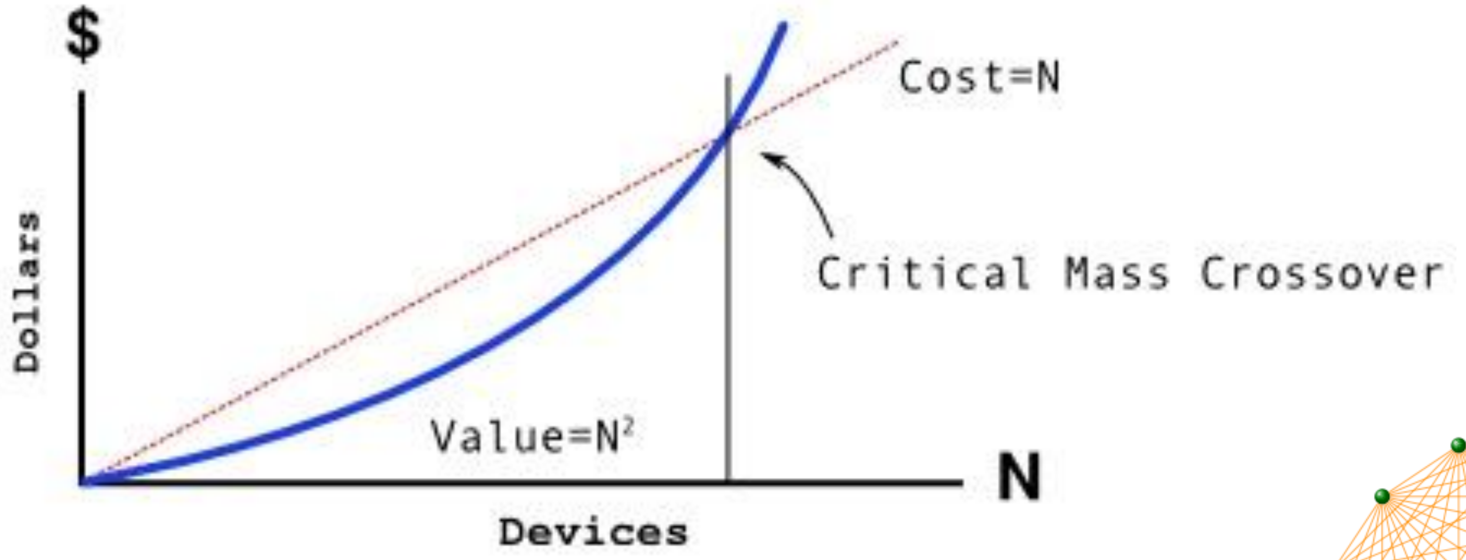
$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

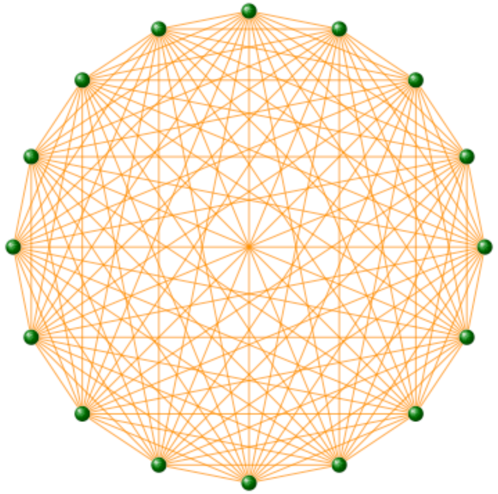
$$A_{ii} = 0 \qquad A_{i \neq j} = 1$$

$$L = L_{\max} = \frac{N(N-1)}{2} \qquad <k> = N-1$$

Actor network, protein-protein interactions

METCALFE’S LAW





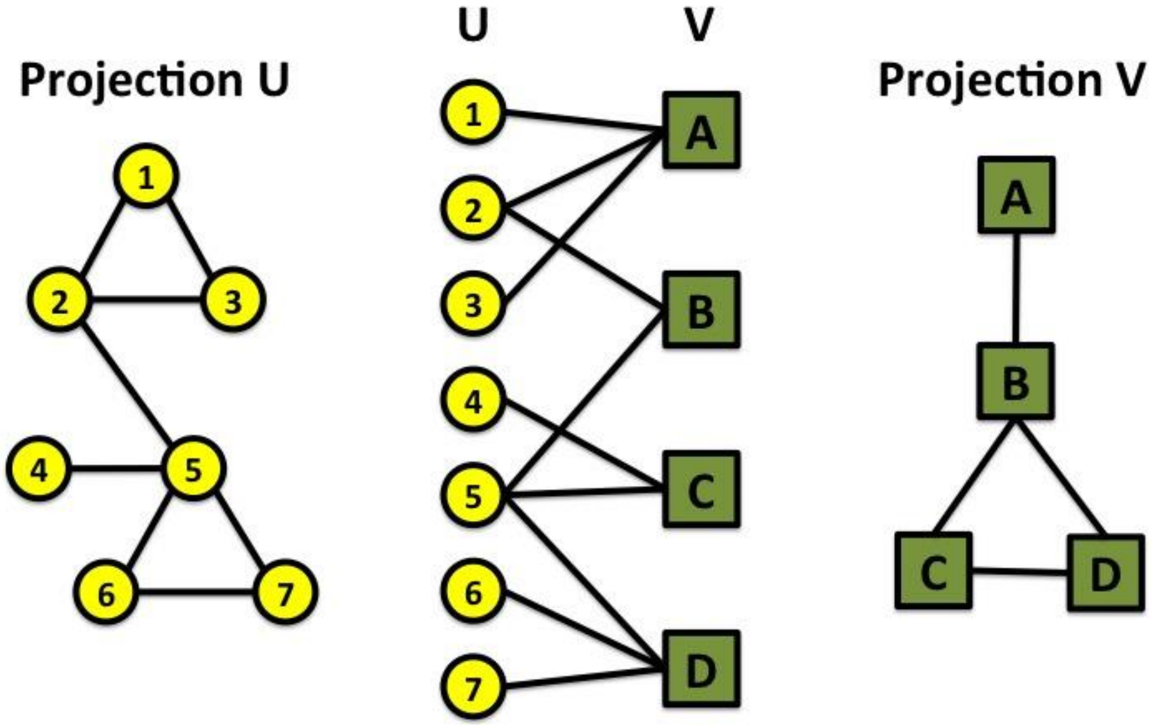
The maximum number of links a network of N nodes can have is: $L_{\max} = \binom{N}{2} = \frac{N(N-1)}{2}$

Section 2.7

BIPARTITE NETWORKS

BIPARTITE GRAPHS

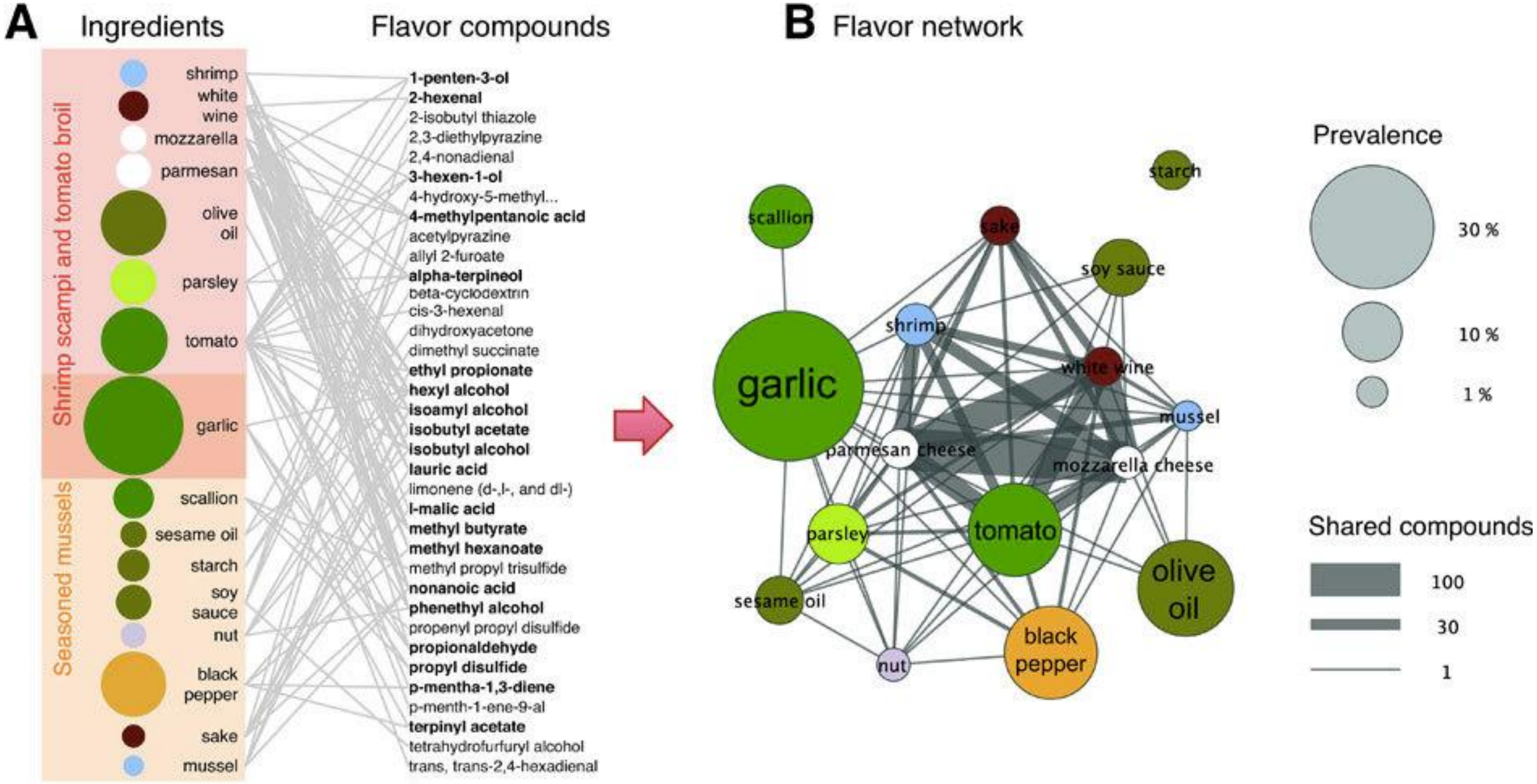
bipartite graph (or **bigraph**) is a graph whose nodes can be divided into two disjoint sets *U* and *V* such that every link connects a node in *U* to one in *V*; that is, *U* and *V* are independent sets.



Examples:

Hollywood actor network
Collaboration networks
Disease network (diseasome)

Ingredient-Flavor Bipartite Network



Y.-Y. Ahn, S. E. Ahnert, J. P. Bagrow, A.-L. Barabási Flavor network and the principles of food pairing , Scientific Reports 196, (2011).

Section 2.8

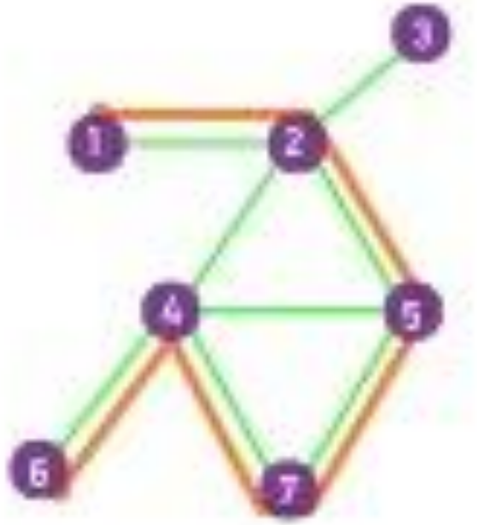
PATHOLOGY

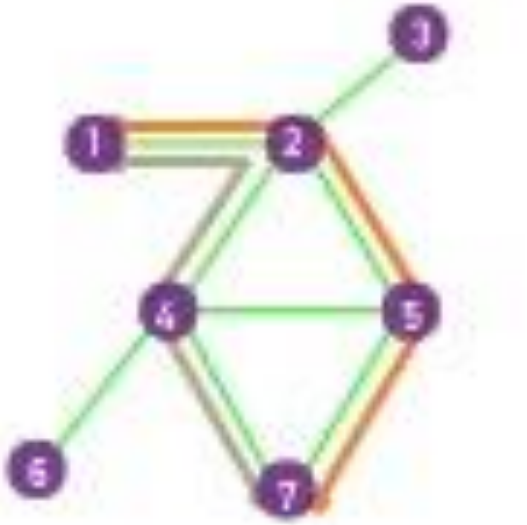
PATHS

A *path is* a sequence of nodes in which each node is adjacent to the next one

$P_{i0,in}$ of length *n* between nodes i_0 and i_n is an ordered collection of *n+1* nodes and *n* links

$$P_n = \{i_0, i_1, i_2, \ldots, i_n\} \qquad P_n = \{(i_0, i_1), (i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n)\}$$

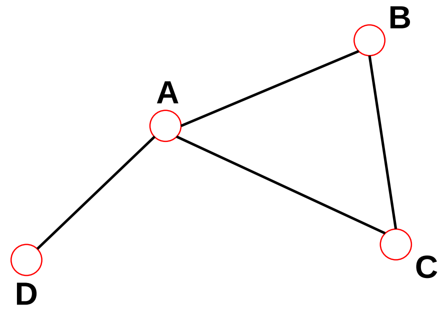




- In a directed network, the path can follow only the direction of an arrow.

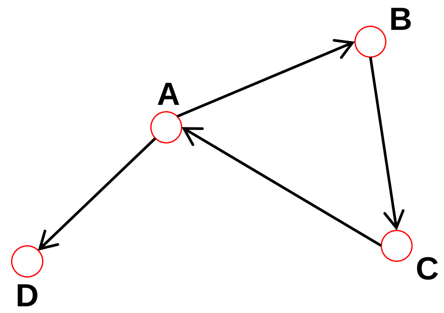
DISTANCE IN A GRAPH

Shortest Path, Geodesic Path



The *distance (shortest path, geodesic path)* between two nodes is defined as the number of edges along the shortest path connecting them.

*If the two nodes are disconnected, the distance is infinity.



In directed graphs each path needs to follow the direction of the arrows.

Thus in a digraph the distance from node A to B (on an AB path) is generally different from the distance from node B to A (on a BCA path).

NUMBER OF PATHS BETWEEN TWO NODES

Adjacency Matrix

N_{ij},number of paths between any two nodes *i* and *j*:

***Length n=1*:** If there is a link between *i* and *j*, then A_{ij}=1 and A_{ij}=0 otherwise.

Length n=2: If there is a path of length two between *i* and *j*, then $A_{ik}A_{kj}$=1, and $A_{ik}A_{kj}$=0 otherwise.

The number of paths of length 2:

$$N_{ij}^{(2)} = \sum_{k=1}^{N} A_{ik}A_{kj} = [A^2]_{ij}$$

Length n: In general, if there is a path of length *n* between *i* and *j*, then $A_{ik}\ldots A_{lj}$=1 and $A_{ik}\ldots A_{lj}$=0 otherwise.

The number of paths of length *n* between *i* and *j* is*

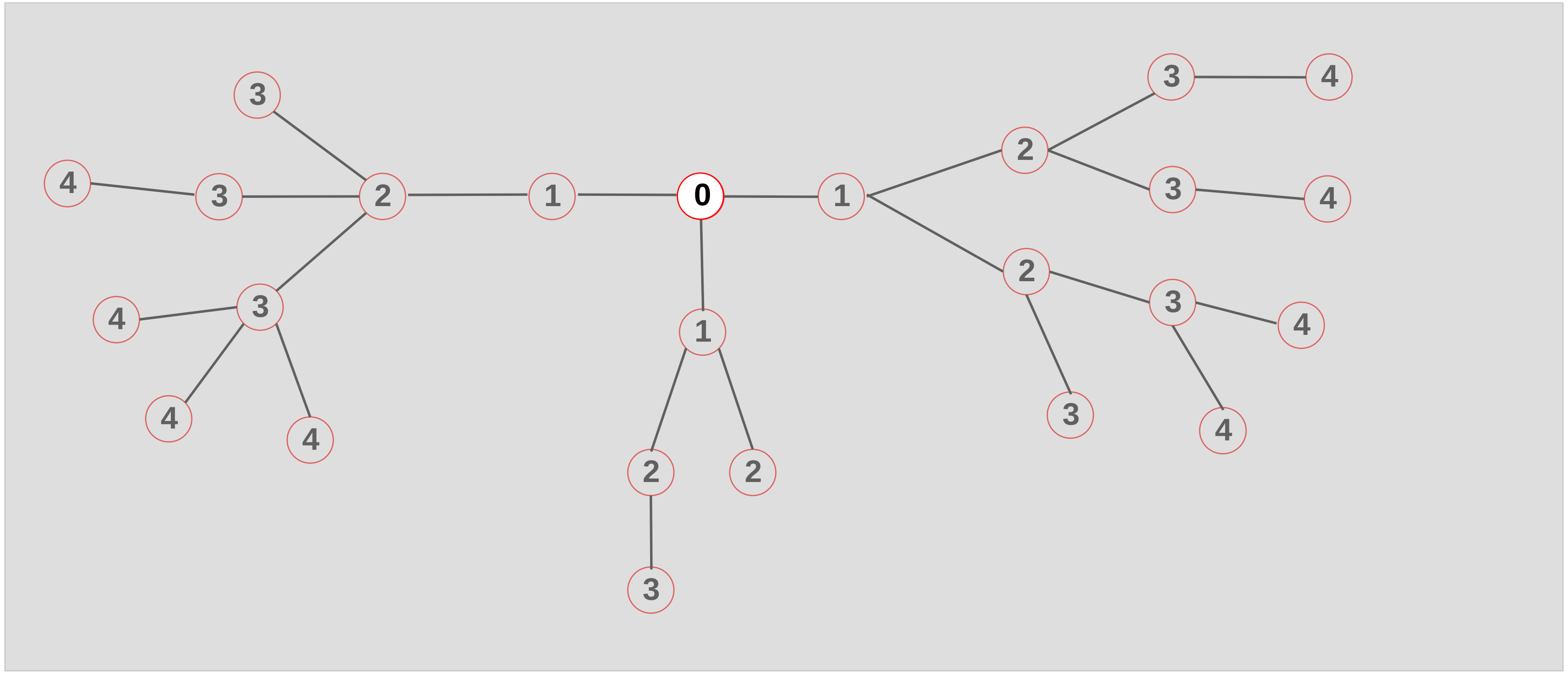
$$N_{ij}^{(n)} = [A^n]_{ij}$$

**holds for both directed and undirected networks.*

FINDING DISTANCES: BREADTH FIRST SEARCH

Distance between node 0 and node 4:

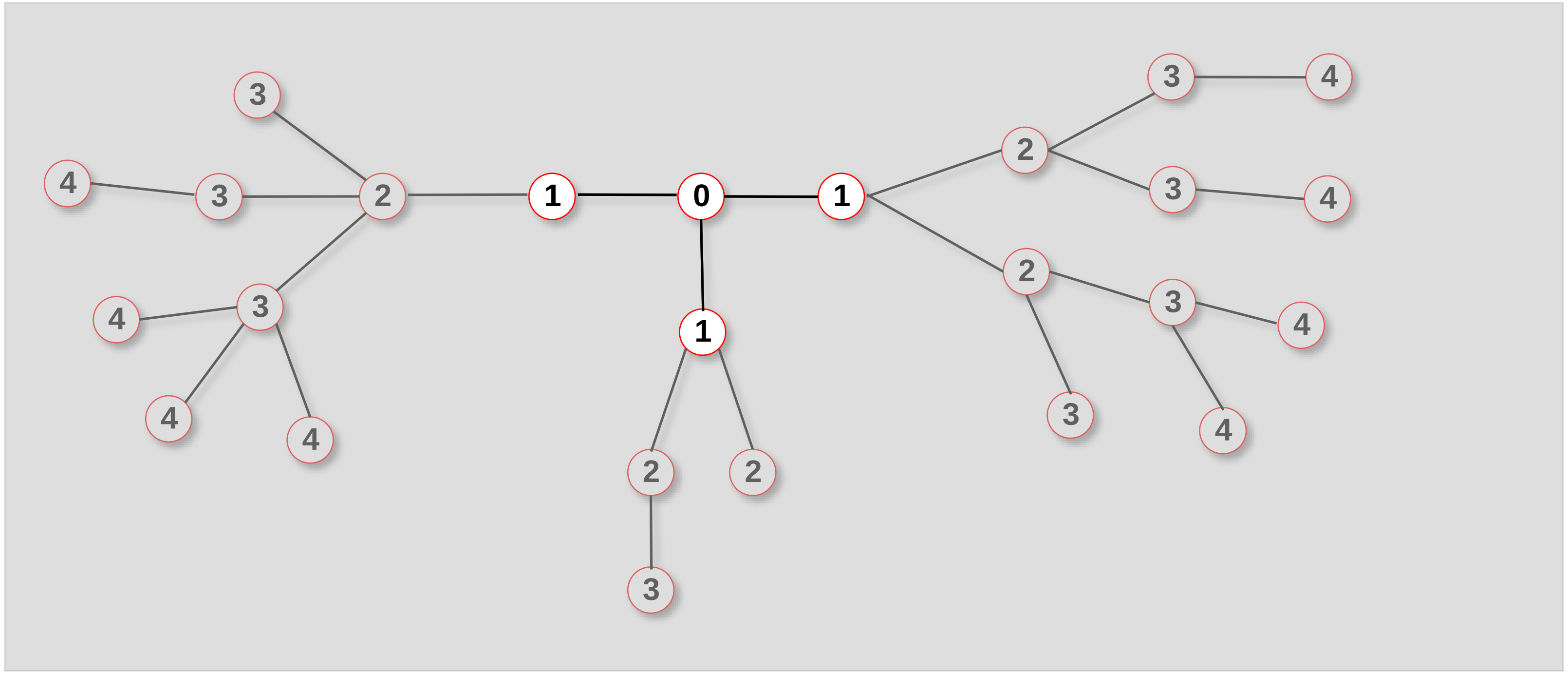
1.Start at 0.



FINDING DISTANCES: BREADTH FIRST SEARCH

Distance between node 0 and node 4:

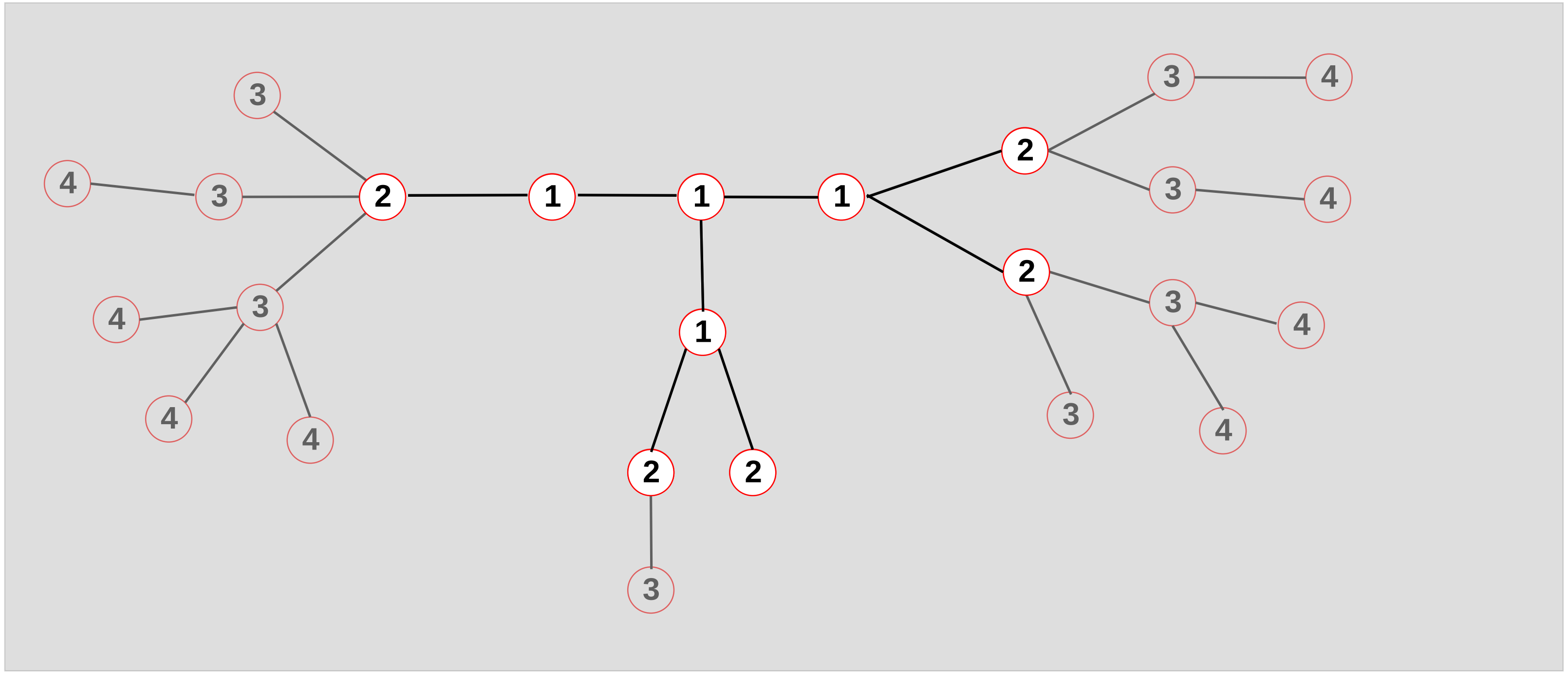
1.Start at 0.
2.Find the nodes adjacent to 1. Mark them as at distance 1. Put them in a queue.



FINDING DISTANCES: BREADTH FIRST SEARCH

Distance between node 0 and node 4:

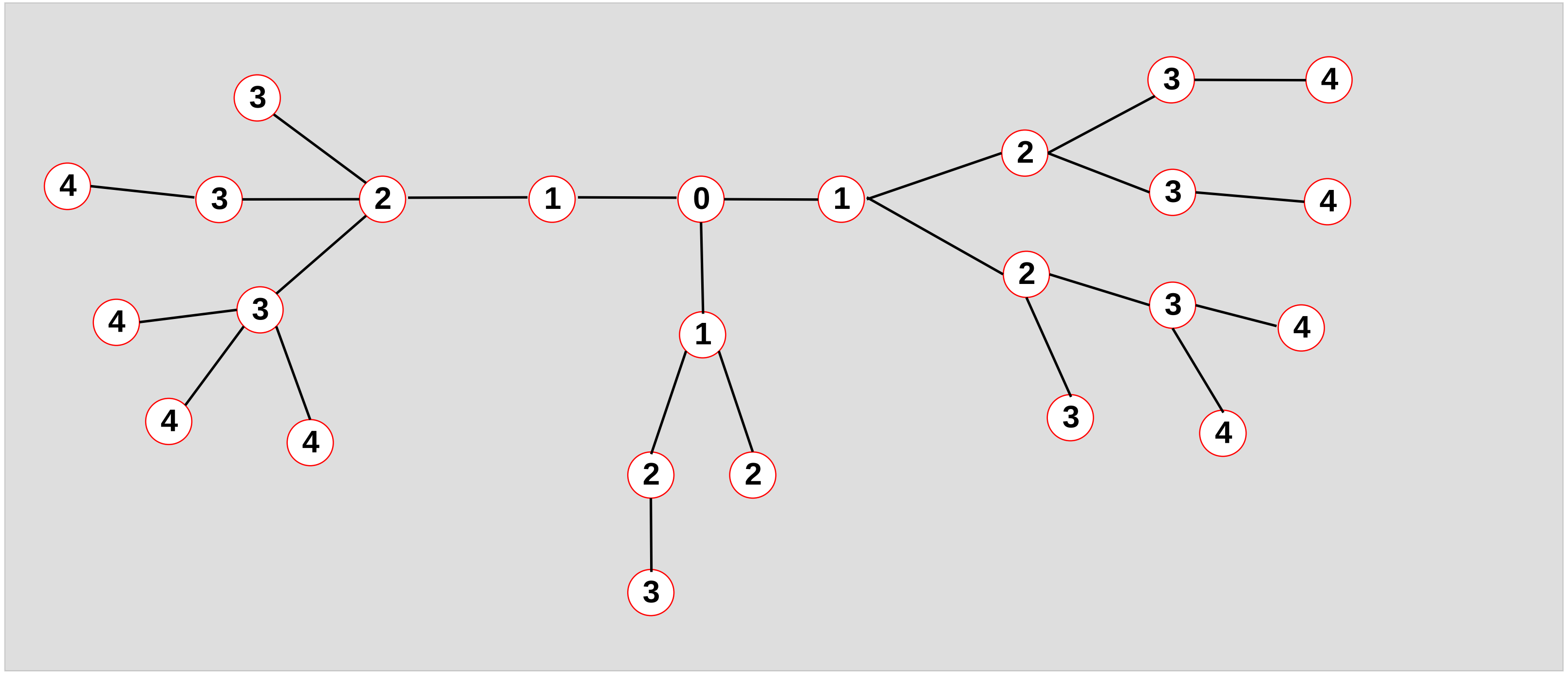
1. Start at 0.
2. Find the nodes adjacent to 0. Mark them as at distance 1. Put them in a queue.
3. Take the first node out of the queue. Find the unmarked nodes adjacent to it in the graph. Mark them with the label of 2. Put them in the queue.



FINDING DISTANCES: BREADTH FIRST SEARCH

Distance between node 0 and node 4:

1. Repeat until you find node 4 or there are no more nodes in the queue.
2. The distance between 0 and 4 is the label of 4 or, if 4 does not have a label, infinity.



NETWORK DIAMETER AND AVERAGE DISTANCE

Diameter: $\boldsymbol{d_{max}}$ the maximum distance between any pair of nodes in the graph.

Average path length/distance, <d>, for a connected graph:

$$\langle d \rangle \equiv \frac{1}{2L_{\max}} \sum_{i,j \neq i} d_{ij}$$

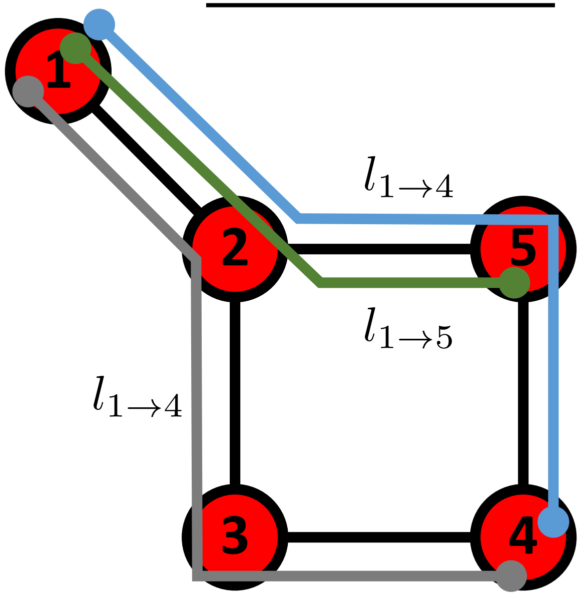
where d_{ij} is the distance from node *i* to node j

In an *undirected graph* $d_{ij} = d_{ji}$*, so* we only need to count them once:

$$\langle d \rangle \equiv \frac{1}{L_{\max}} \sum_{i,j>i} d_{ij}$$

PATHOLOGY: summary

Shortest Path



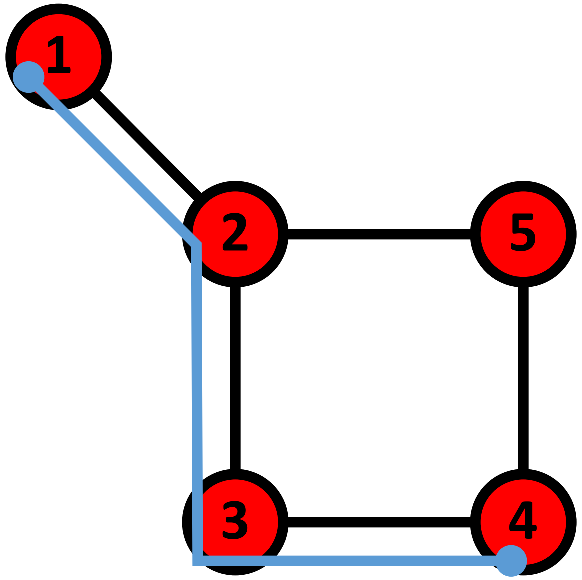
$l_{1\to4} = 3$

$l_{1\to5} = 2$

The path with the shortest length between two nodes (distance).

PATHOLOGY: summary

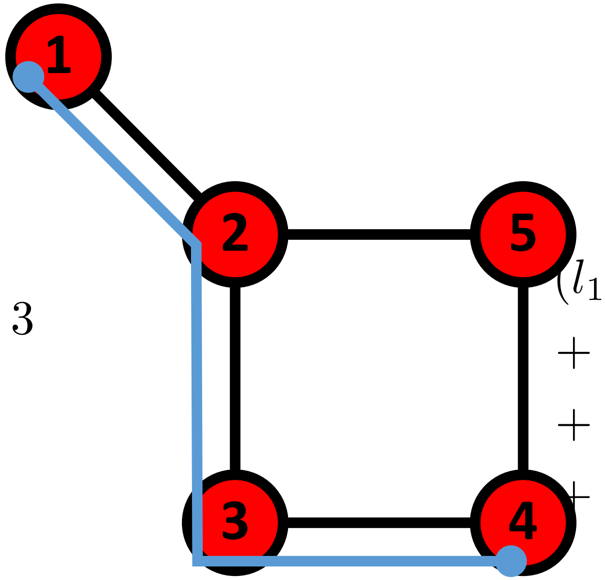
Diameter



$$l_{1\to 4} = 3$$

The longest shortest path in a graph

Average Path Length

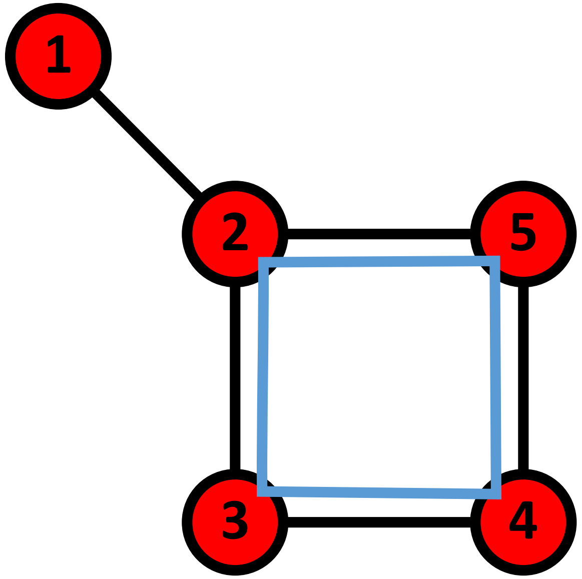


$$\begin{aligned}(l_{1\to 2} + l_{1\to 3} + l_{1\to 4} +\\ + l_{1\to 5} + l_{2\to 3} + l_{2\to 4} +\\ + l_{2\to 5} + l_{3\to 4} + l_{3\to 5} +\\ + l_{4\to 5})\,/10 = 1.6\end{aligned}$$

The average of the shortest paths for all pairs of nodes.

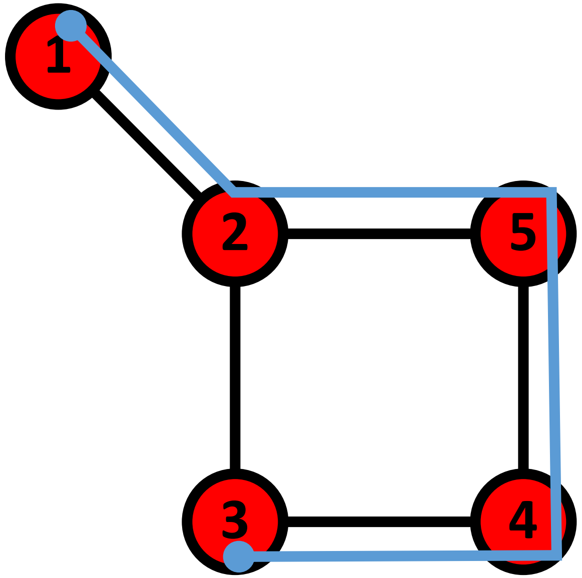
PATHOLOGY: summary

Cycle



A path with the same start and end node.

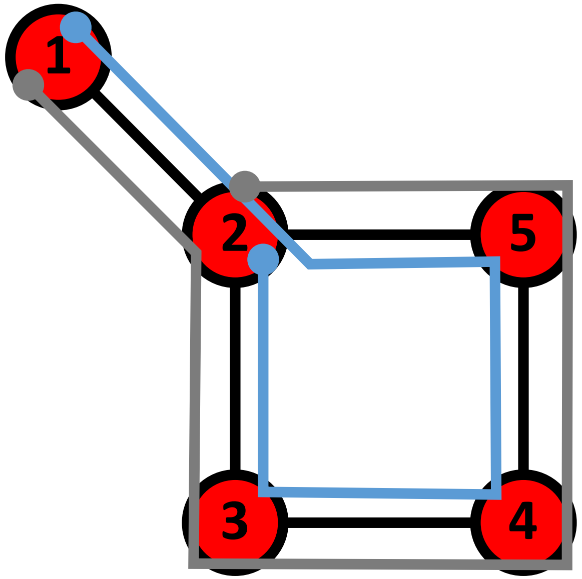
Self-avoiding Path



A path that does not intersect itself.

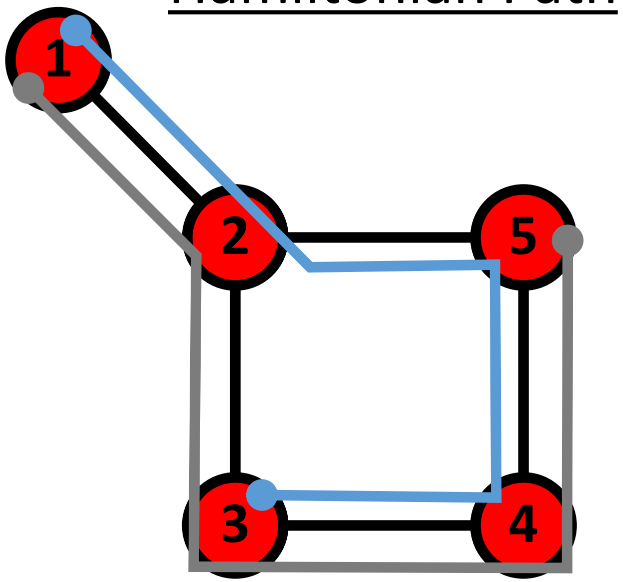
PATHOLOGY: summary

Eulerian Path



A path that traverses each link exactly once.

Hamiltonian Path



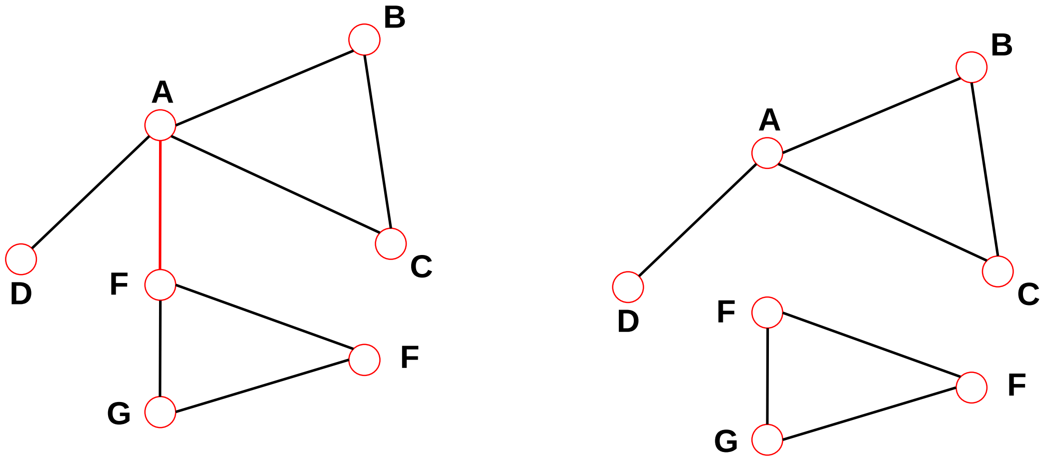
A path that visits each node exactly once.

Section 2.9

CONNECTEDNESS

CONNECTIVITY OF UNDIRECTED GRAPHS

Connected (undirected) graph: any two vertices can be joined by a path.
A disconnected graph is made up by two or more connected components.



Largest Component:
Giant Component

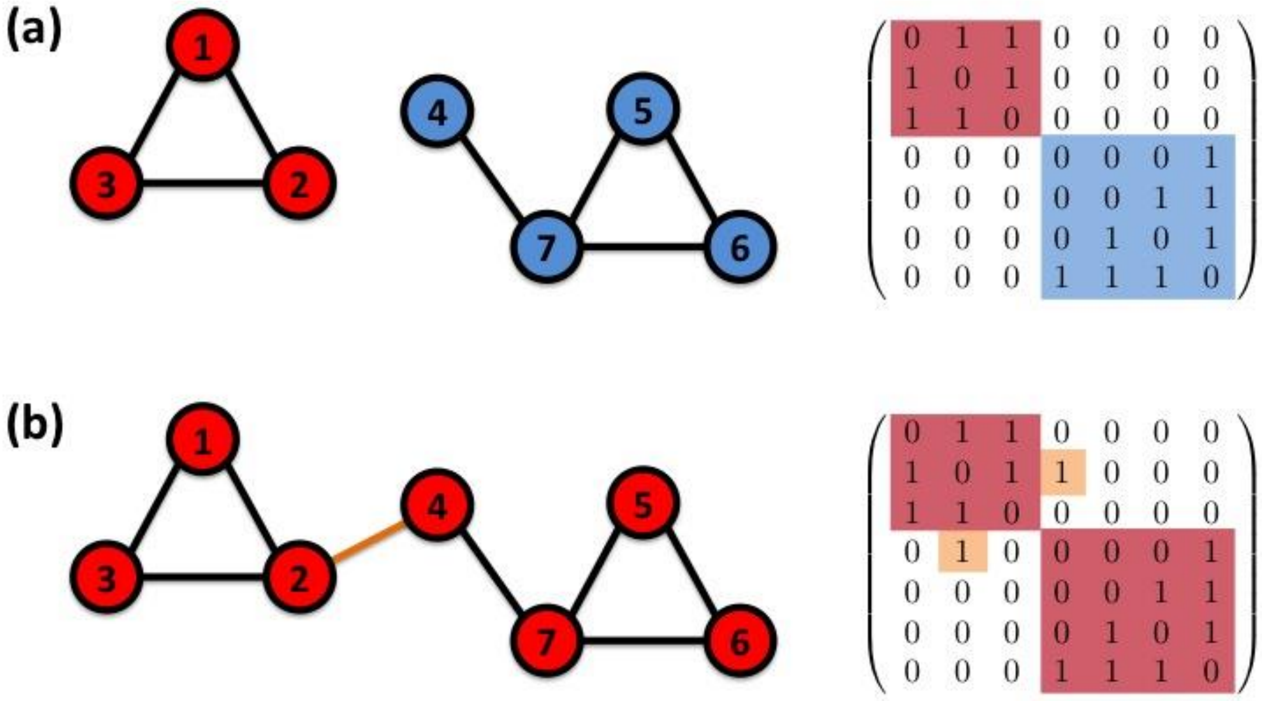
The rest: **Isolates**

Bridge: if we erase it, the graph becomes disconnected.

CONNECTIVITY OF UNDIRECTED GRAPHS

Adjacency Matrix

The adjacency matrix of a network with several components can be written in a block-diagonal form, so that nonzero elements are confined to squares, with all other elements being zero:



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

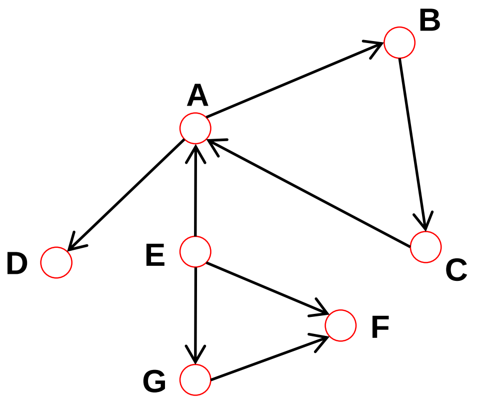
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

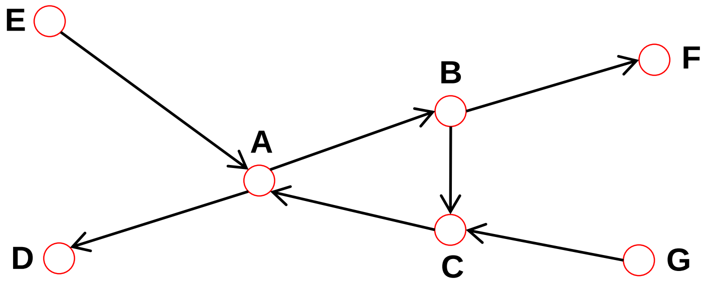
CONNECTIVITY OF DIRECTED GRAPHS

Strongly connected directed graph: has a path from each node to every other node and vice versa (e.g. AB path and BA path).

Weakly connected directed graph: it is connected if we disregard the edge directions.

Strongly connected components can be identified, but not every node is part of a nontrivial strongly connected component.

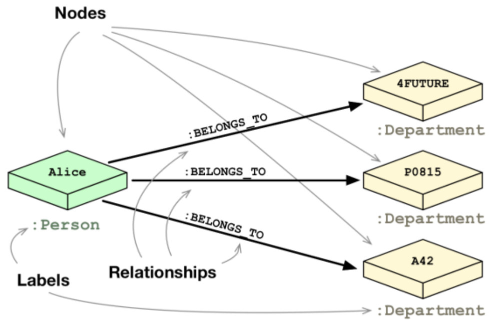




In-component: nodes that can reach the scc,

Out-component: nodes that can be reached from the scc.

Neo4j Graph Database

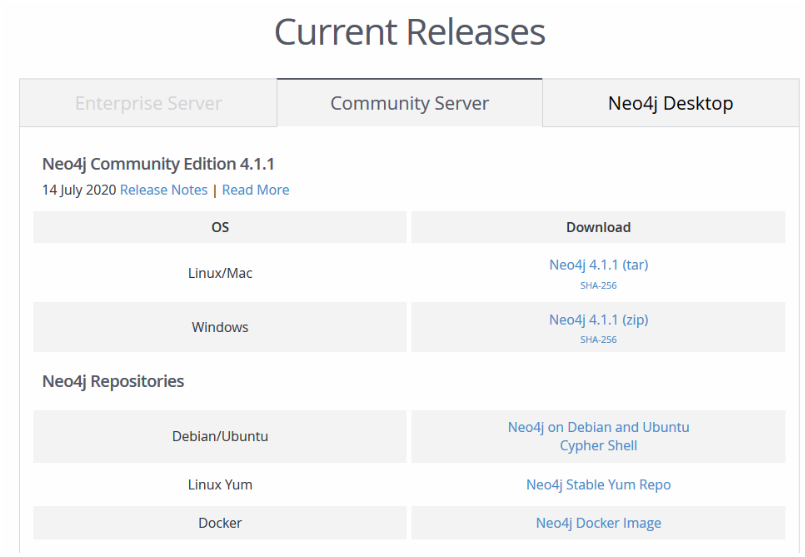


- First class support for nodes, relationships, and properties
- Efficient management of *semi-structured* and *network-oriented* data
- *Embedded* persistence engine – implemented as a small, light-weight, and non-intrusive Java library
- *Robust* – full support for distributed ACID transactions, configurable isolation levels, and transaction recovery
- *Highly scalable* – can handle large networks of data (no limits on the number of nodes, relationships, and properties that can be stored and indexed)
- *High-performance* – index-free adjacency, cost-based query optimizer, parallel indexes capability, binary protocol
- Open source, two editions – Community (GPL v3) and Enterprise (for commercial deployments with enterprise-grade availability, management, and scale-up and scale-out capabilities)

https://neo4j.com/developer/graph-db-vs-rdbms/

Neo4j Installation

- Community Edition
- Server, not Desktop
- Start the Neo4j service
- Default login is username **'neo4j'** and password **'neo4j'**; must be changed on the first login



Current Releases

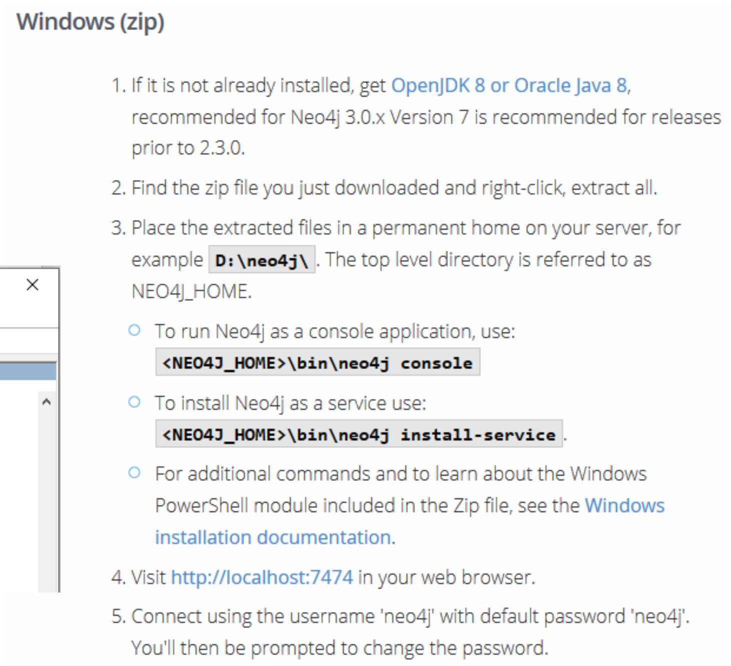
Neo4j Community Edition 4.1.1

14 July 2020 Release Notes | Read More

OS	Download
Linux/Mac	Neo4j 4.1.1 (tar) SHA-256
Windows	Neo4j 4.1.1 (zip) SHA-256

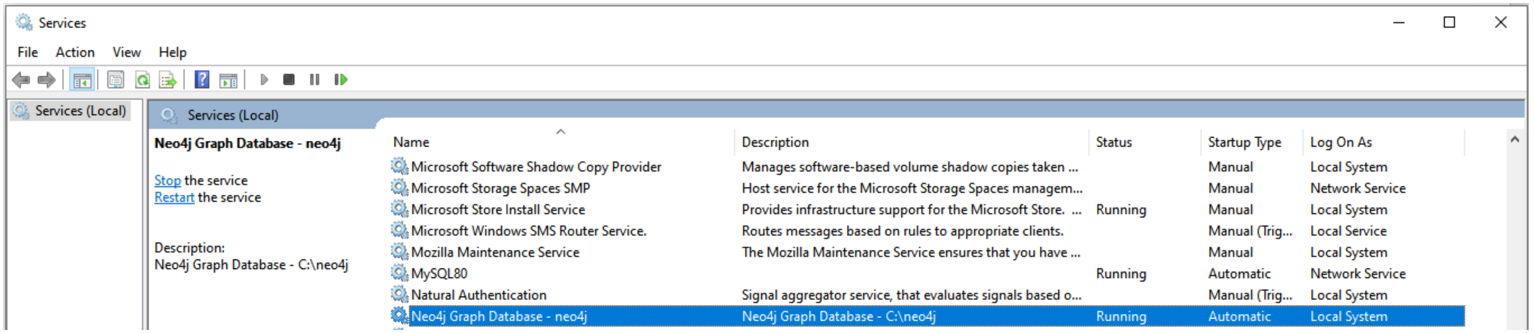
Neo4j Repositories

Debian/Ubuntu	Neo4j on Debian and Ubuntu Cypher Shell
Linux Yum	Neo4j Stable Yum Repo
Docker	Neo4j Docker Image



Windows (zip)

1. If it is not already installed, get OpenJDK 8 or Oracle Java 8, recommended for Neo4j 3.0.x Version 7 is recommended for releases prior to 2.3.0.
2. Find the zip file you just downloaded and right-click, extract all.
3. Place the extracted files in a permanent home on your server, for example `D:\neo4j\`. The top level directory is referred to as NEO4J_HOME.
   - To run Neo4j as a console application, use: `<NEO4J_HOME>\bin\neo4j console`
   - To install Neo4j as a service use: `<NEO4J_HOME>\bin\neo4j install-service`.
   - For additional commands and to learn about the Windows PowerShell module included in the Zip file, see the Windows installation documentation.
4. Visit http://localhost:7474 in your web browser.
5. Connect using the username 'neo4j' with default password 'neo4j'. You'll then be prompted to change the password.



Neo4j Graph Database - neo4j

Stop the service
Restart the service

Description:
Neo4j Graph Database - C:\neo4j

Name	Description	Status	Startup Type	Log On As
Microsoft Software Shadow Copy Provider	Manages software-based volume shadow copies taken ...		Manual	Local System
Microsoft Storage Spaces SMP	Host service for the Microsoft Storage Spaces managem...		Manual	Network Service
Microsoft Store Install Service	Provides infrastructure support for the Microsoft Store. ...	Running	Manual	Local System
Microsoft Windows SMS Router Service.	Routes messages based on rules to appropriate clients.		Manual (Trig...	Local Service
Mozilla Maintenance Service	The Mozilla Maintenance Service ensures that you have ...		Manual	Local System
MySQL80		Running	Automatic	Network Service
Natural Authentication	Signal aggregator service, that evaluates signals based o...		Manual (Trig...	Local System
Neo4j Graph Database - neo4j	Neo4j Graph Database - C:\neo4j	Running	Automatic	Local System

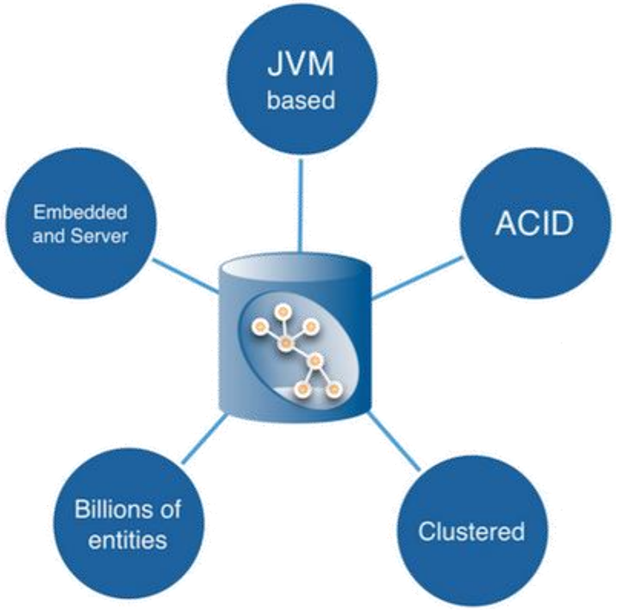
Graph Databases

- Use graph structures for semantic queries with nodes, edges, and properties to represent and store data
- Use the Property Graph Model:
  - Connected entities (nodes) can hold any number of attributes (key-value-pairs) and can be tagged with labels representing their different roles in your domain
  - Relationships provide directed, named connections between two node-entities. A relationship always has a direction, a type, a start node, and an end node.
- Well suited for semi-structured and highly connected data
- Require a new query language

Relational vs. Graph Databases

- Relational
  - Store highly structured data in tables with predetermined columns of certain types and many rows of the same type of information
  - Require developers and applications to strictly structure the data used in their applications
  - References to other rows and tables are indicated by referring to their (primary-)key attributes via foreign-key columns
  - In case of many-to-many relationships, you have to introduce a JOIN table (or junction table) that holds foreign keys of both participating tables which further increases join operation costs
- Graph
  - Relationships are first-class citizens of the graph data model
  - Each node (entity or attribute) directly and physically contains a list of relationship-records that represent its relationships to other nodes
  - The ability to pre-materialize relationships into database structures provides performances of several orders of magnitude advantage

Neo4j Graph Database



- NoSQL Graph Database
- Implemented in Java and Scala
- Open source
- Free and open-source Community edition and Enterprise editions which provide all of the functionality of the Community edition in addition to scalable clustering, fail-over, high-availability, live backups, and comprehensive monitoring.
- Full database characteristics including ACID transaction compliance, cluster support, and runtime failover
- Constant time traversals for relationships in the graph both in depth and in breadth

Cypher Query Language

- SQL-inspired language for describing patterns in graphs visually using an ASCII-art syntax
- Declarative – allows us to state **what** we want to select, insert, update or delete from our graph data without requiring us to describe exactly **how** to do it
- Contains clauses for searching for patterns, writing, updating, and deleting data
- Queries are built up using various clauses. Clauses are chained together, and they feed intermediate result sets between each other
- Cypher query gets compiled to an execution plan that can run and produce the desired result
- Statistical information about the database is kept up to date to optimize the execution plan
- Indexes on Node or Relationships properties are supported to improve the performance of the application

//TODO before next lecture:

- Homework 4 due on 4/20 at 11:59 pm EDT. Must be submitted on Submitty.